Predicting stronghold locations with Bayesian statistics

Discord user Ninjabrain#9740

November 2020

1 Introduction

The traditional triangulation technique involves throwing two eyes of ender and intersecting the lines they trace. This traditional model will work perfectly every time if the user is able to read the angle of the ender eye perfectly. However, this is not possible since the user has to point their crosshair manually at the eye, adding a measurement error. In this paper I will present a model that accounts for the error, and has multiple benefits over the traditional method:

- More reliable results by incorporating the '8,8 strat'.
- Gives a certainty value (probability) of the predicted stronghold chunk location, so the player can know whether or not the prediction should be trusted.
- Any number of measurements can be included to increase the precision indefinetly (as opposed to 2 with the traditional triangulation).
- Uses statistical stronghold generation mechanics to give more accurate predictions.

2 Model

2.1 Closest stronghold

An ender eye always points to the nearest stronghold, meaning that closer chunks in the direction of the ender eye generally have a higher probability of having a stronghold than farther chunks. In this section the math that is used to model this is presented.

Let n_k denote the number of strongholds in the k:th ring. The positions of the strongholds pre-snapping is given by

$$(-r_i \sin\phi_i, r_i \cos\phi_i)\,, \quad \phi \sim U(0, 2\pi), \phi_i = \phi + \frac{2i\pi}{n_k}, r_i \sim U(a_k, b_k), i = 0, \ldots, n_k - 1.$$

Then two snapping steps are performed, first the points are snapped to the nearest chunk origin ($(0,0)$ in the chunk), then the points are snapped up to 7 chunks away in both dimensions, approximately uniformly distributed. Let ϕ_i' denote the polar angle of the $(8,8)$ position of stronghold i post-snapping, and r_i its distance from $(0,0)$. The difference between the stronghold positions pre and post snapping is at most $8 + 7 \times 16 + 8$ blocks in both dimensions, which means the maximum distance the stronghold can move during snapping is $128\sqrt{2}$ blocks. In turn, this means that $|\phi + \frac{2i\pi}{n_k} - \phi'_i| \leq \arctan\big(128\sqrt{2}/a_k\big) < 128\sqrt{2}/a_k$, and

$$\phi'_i = \phi_i + \delta_i,$$

where all δ_i are independent and less than $128\sqrt{2}/a_k$. It is also made the assumption that all δ_i are identically distributed continuous random variables, which isn't true but is a practical approximation.

When an ender eye is thrown let j denote the index of the stronghold it points towards. The probability that $j = i$ given ϕ'_i, r'_i satisfies

$$p(j = i|\phi'_i, r'_i) \propto f(\phi'_i, r'_i|j = i)p(j = i) \propto f(\phi'_i, r'_i|j = i),$$

since $p(j = i)$ is constant due to symmetry. Let $\overline{p} = (\phi_p, r_p)$ denote the position of the player when they throw the eye, let $\overline{r}_i$ denote the position of stronghold i, and let $d_i = |\overline{r}_i - \overline{p}|$. Since stronghold j is closer than any other stronghold it also holds that

$$d_j \leq d_i, \quad \forall i \neq j,$$

which implies

$$p(j = i|\phi'_i, r'_i) = \prod_{l \neq i} p(d_i \leq d_l|\phi'_i, r'_i).$$

We obtain

$$p(d_i \leq d_l|\phi'_i, r'_i) = 1 - p(d_i < d_l|\phi'_i, r'_i) = 1 - \int_0^{2\pi} \int_{R_0}^{R_1} f(\phi'_l, r'_l|\phi'_i, r'_i)\mathrm{d}r'_l \mathrm{d}\phi'_l,$$

where the integration bounds R_0 and R_1 (not including the derivation here) are given by

$$\begin{aligned}
R_m &= d_i \frac{\sin\alpha_k}{\sin(\phi_p - \phi'_l)}, \quad m = 0, 1 \\
\alpha_0 &= \beta - (\phi_p - \phi'_l) \\
\alpha_1 &= \pi - (\phi_p - \phi'_l) - \beta \\
\beta &= \arcsin\left(\frac{r_p}{d_i}\sin(\phi_p - \phi'_l)\right).
\end{aligned}$$

If strongholds i and l are in different rings ϕ'_i and ϕ'_l are independent and we obtain

$$p(d_i \leq d_l|\phi'_i, r'_i) = 1 - \int_0^{2\pi} \int_{R_0}^{R_1} f(\phi'_l, r'_l)\mathrm{d}r'_l \mathrm{d}\phi'_l = 1 - \int_0^{2\pi} \frac{1}{2\pi}(F_r(R_1) - F_r(R_0))\mathrm{d}\phi'_l.$$

If strongholds i and l are in the same ring $\phi'_l = \phi'_i + \frac{(i-l)2\pi}{n_k} - \delta_i + \delta_l = \phi'_i + \frac{(i-l)2\pi}{n_k} + \Delta_{il}$ and by a change of variables we get

$$\begin{aligned}
p(d_i \leq d_l|\phi'_i, r'_i) &= 1 - \int_0^{2\pi} \int_{R_0}^{R_1} f_\Delta(\phi'_i + \frac{(i-l)2\pi}{n_k} + \Delta_{il}, r'_l)\mathrm{d}r'_l \mathrm{d}\Delta_{il} \\
&= 1 - \int_0^{2\pi} f_\Delta(\phi'_i + \frac{(i-l)2\pi}{n_k} + \Delta_{il})(F_r(R_1) - F_r(R_0))\mathrm{d}\Delta_{il}.
\end{aligned}$$

Putting it all together:

$$f(\phi_i', r_i'|j=i) \propto \prod_{l\neq i} p(d_i \leq d_l|\phi_i', r_i').$$

To be able to numerically estimate the integral above we need to know the distribution of Δ. Let r_ϕ^i denote the distance in the ϕ direction that stronghold i is moved due to snapping. Then, using small angle approximation, we get $\Delta_{il} \approx \frac{r_\phi^l}{r_l} - \frac{r_\phi^i}{r_i} \approx \frac{r_\phi^l - r_\phi^i}{a_k}$. Let $q = r_\phi^l - r_\phi^i$, values from q's distribution is sampled, with the assumption that snapping is uniform on the 15×15 grid centered on $(0, 0)$. In reality the $(0, 0)$ offset has a slightly higher probability (around 0.06, because of ocean strongholds)[1], and biome snapping favors offsets towards the edges, but these effects are not that significant and assuming a uniform distribution is a good approximation. The samples from q's distribution can then be used to fit a distribution, since q is bounded we choose to fit a β distribution. The fit reveals that q is approximately $\beta(5.5, 5.5)$-distributed, see Figure 1. Note that $q \in [-7.5 \times \sqrt{2} \times 2, 7.5 \times \sqrt{2} \times 2]$, since the maximum snapping distance along an axis is 7.5 chunks, 0.5 when the closest chunk is chosen, and 7 when the "biome snapping" step occurs. Thus, the pdf of Δ in ring k is approximately proportional to

$$f_\Delta(\Delta) \propto \left(1 + \frac{a_k\Delta}{15\sqrt{2}}\right)^{4.5} \left(1 - \frac{a_k\Delta}{15\sqrt{2}}\right)^{4.5}, \quad |\Delta| < \frac{15\sqrt{2}}{a_k}.$$

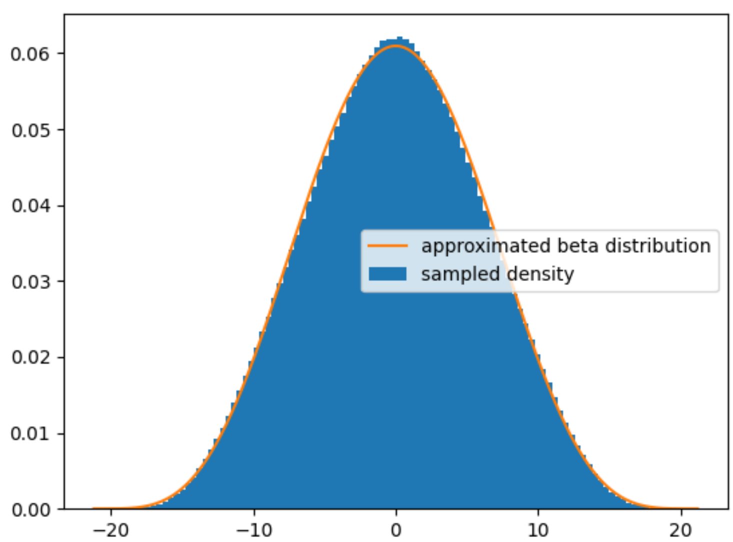

Figure 1: The sampled density of q (10M samples) compared to the pdf of a $\beta(5.5, 5.5)$ distribution.

[1]Source: Matthew Bolan

2.2 Measurement error

Let the set of grid points

$$G = \{g_k \in \mathbb{R}^2 : g_k \text{ is in one of the 8 stronghold rings}, g_k = (16i+8, 16j+8), i, j \in \mathbb{Z}\}$$

denote the set of all chunk centers (8,8) where strongholds can spawn (after "biome snapping" occurs). Let $s_i \in G$ denote the location of the i:th stronghold. What we ultimately want is to assign a value to the probability $P(s_i = g_k)$ for each $k = 1, \ldots, K$ based on eye throws, and choose the g_k that maximizes $P(s_i = g_k)$ as our prediction. At the time of throw n, let γ_{n,g_k} denote the true angle (in degrees) to chunk center g_k from the player, and let α_n denote the angle that is measured by the player. It is assumed that the measurement error is normally distributed with mean 0 and variance σ^2, which is standard practice for measurement errors and testing has confirmed it to be a fitting distribution. This gives us:

$$\alpha_n = \gamma_{n,s} + \epsilon_n, \quad \epsilon_n \sim \mathrm{N}(0, \sigma^2).$$

Thus, the conditional probability density of α_n given that $s_i = g_k$ is

$$p(\alpha_n | s = g_k) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(\alpha_n - \gamma_{n,g_k})^2/2\sigma^2}.$$

Let $\alpha = (\alpha_1, \ldots, \alpha_N)$ denote the vector containing all α_n (all throws by the player). If we assume that the measurement errors ϵ_n are independent we get the joint conditional probability density

$$p(\alpha | s = g_k) = \prod_{n=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} e^{-(\alpha_n - \gamma_{n,g_k})^2/2\sigma^2}.$$

Using Bayes' theorem we get the following expression for the probability of chunk location g_k containing the stronghold, given the eye throws:

$$P(s_i = g_k | \alpha) = \frac{p(\alpha | s = g_k) P(s_i = g_k)}{p(\alpha)} \propto P(s_i = g_k) \prod_{n=1}^{N} e^{-(\alpha_n - \gamma_{n,g_k})^2/2\sigma^2}$$

where $P(s_i = g_k)$ is the prior probability mass funciton of the strongholds, and $p(\alpha)$ is the prior distribution of α. More specifically, $P(s_i = g_k)$ is the probability that that chunk g_k has a stronghold, so $\sum_{g \in G} P(s_i = g) = 128$ because there are 128 strongholds. I wont go into detail as to exactly how $P(s_i = g_k)$ is calculated because it is not important, but basically it is calculated by first integrating the pre-snapping stronghold density over the chunk region, and then performing a convolution step which models biome snapping. Finally, we obtain the posterior distribution, where we also take into account that s_i has to be the closest stronghold:

$$P(s_i = g_k | \alpha, i = j) \propto P(s_i = g_k | \alpha) f(\phi_i', r_i' | j = i)$$

The posterior distribution is used to give the player the probability of each chunk containing the stronghold.

3 Notes

In practice the probabilities $p(\alpha|s_i = g_i)$ are not calculated for all chunks in G, doing so is too computationally intensive. Also, the prior $P(s_i = g_k)$ is not calculated exactly in practice, mainly because the convolution step is very expensive. Instead, a post-snapping density is approximated and the convolution can be skipped entirely. Also, it is possible that the assumption that $\mathbb{E}[\epsilon_n] = 0$ is false if the player doesnt know where to aim on the eye, but this can be corrected by a guide, for example. The parameter σ is set depending on how accurate the player is. For players that are just introduced to the tool I have found $\sigma = 0.1$ to be a good value. For a user that is experienced at measuring eyes at 30 FOV $\sigma = 0.03$ is a good value. For someone who measures subpixels flawlessly the value of σ can be as low as $0.010 - 0.005$. In practice, the smaller σ is, the more 'certain' the algorithm will be that its prediction is correct.

4 Examples

The model has been tested in creative mode, and the model has been better or equally good at predicting the correct chunk than other triangulation methods. An example of the calculator working in a real run can be seen at `https://youtu.be/zK96gjkLTGc?t=871`.