Draft version April 9, 2020
Typeset using LaTeX default style in AASTeX63

PHYS 321 Final Project: Bayesian Shoot-out

William Frost[1]

[1] *Department of Physics, McGill University*
3600 Rue Université, Montréeal, QC, H3A 2T8, Canada

ABSTRACT

The time is midnight, November 5$^{\text{th}}$, 2099. A relentless game of cat and mouse between 2 mercenaries (Red and Blue) has been raging for hours within a deserted slum. Having sustained severe injuries, both of them end up taking refuge inside identical adjacent buildings. The injuries are such that once they stop to rest, they are unable to move again.

However, the action does not stop there. Both are equipped with heavy-duty, armor-piercing rail-guns capable of penetrating any surface. The fight will obviously end once someone is able to hit the other through the building walls, but none knows where the other is located. All seems fair, except there's a catch. Red carries effectively unlimited ammo, while Blue is down to his last shot and a couple of reconnaissance drones. Soon realizing that he has the upper hand, the red mercenary starts spewing a barrage of random gunfire. For Blue, the odds seem bleak.

But Blue has been trained in the potent art of Bayesian Inference$^{\text{TM}}$. He figures that Red's shots will likely be uniformly distributed across the horizontal and vertical firing angles in his general direction. If he can survive enough random shots piercing through the building walls, his knowledge of Bayesian statistics will allow him to estimate Red's location and hopefully end this once and for all. Using his remaining drones, Blue obtains a top and side view of the situation and is able to detect exit points of enemy fire from Red's building. With no time to waste, he begins his calculations...

Keywords: Bayesian Statistics — Markov Chain Monte Carlo

1. INTRODUCTION

Extravagant scenarios aside, the problem this paper solves consists of detecting projectiles on a 2D surface in order to infer the 3D location of the stationary projectile emitter. The analysis is performed using Bayesian Inference and a Markov Chain Monte Carlo (MCMC) to traverse a 3-parameter space (α, β, γ) in search of the most likely emitter location, given the projectiles detected on our 2D surface. The goal is to sufficiently constrain the likely location of the emitter and disable it with a projectile of our own.

For our purposes, the 2D detection surface happens to be exactly at the outer wall of Blue's building which faces the one Red is in. We suppose that Blue's drones have 1D lines of sight, each corresponding to one of the 2 dimensions of the detection surface, thus allowing horizontal and vertical positions to be measured. Both mercenaries correspond to a 3D "hit-box" inside their respective buildings. Fig. 1 shows the map layout of the situation.

1.1. *Likelihood function of the shots fired*

To approach the problem using Bayesian Inference, a few assumptions are made as to the trajectories of the random shots. First, we suppose that the firing angles θ and ϕ obey a uniform distribution and that all shots performed will be directed towards the opposing building along the β-axis. The simplest probability density function (pdf) for the likelihood of θ and ϕ that satisfies these constraints is

$$p(\theta, \phi|\alpha, \beta, \gamma) = \begin{cases} \frac{1}{\pi^2} & \text{for } \frac{-\pi}{2} \leq \theta, \phi \leq \frac{\pi}{2} \\ 0 & \text{otherwise} \end{cases} . \tag{1}$$

However, these bounds for θ and ϕ are not representative of the actual ranges detectable by the drones. Nor are they likely valid constraints for where the red mercenary might decide to target. For example, shots going straight up or

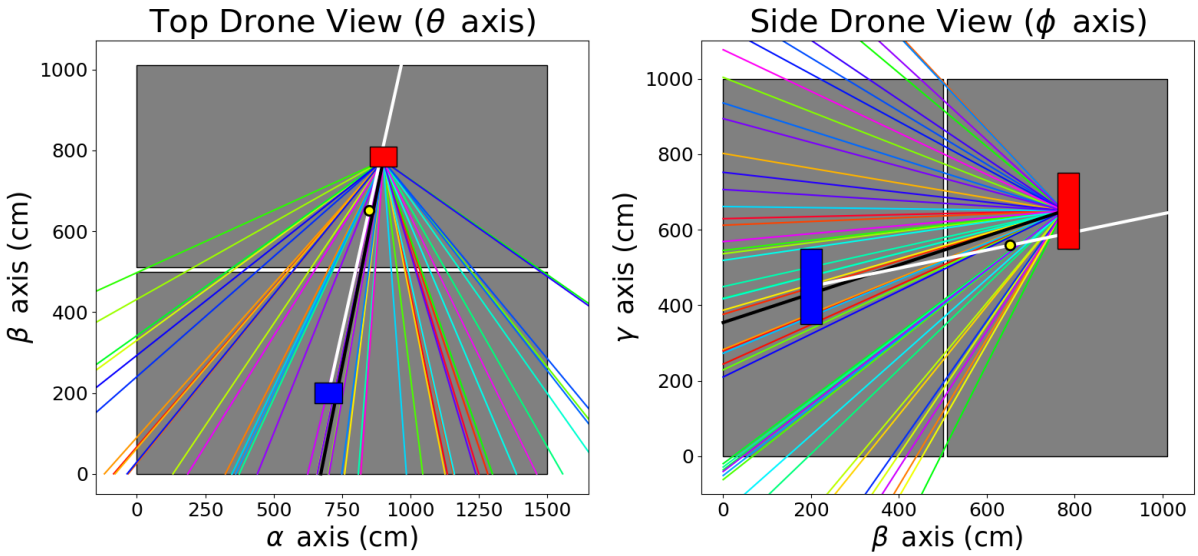


Figure 1: Example of a randomly generated layout and subsequent shoot-out. Axis dimensions are in centimeters. Red and Blue's position are shown as colored rectangles in the top and side views. Note that what Blue's drones actually detect are the positions in 2D space at which the shots (represented by the multi-colored lines coming out of the red figure) pass through the outer wall of Red's building. This 2D detection plane is shown here as a small gap at $\beta = 500$ for the case where there is no building separation. Thick black traces represent shots that hit Blue, while a thick white trace will be used to represent Blue's attempted shot. A yellow circle will indicate the inferred location found by Blue's MCMC.

down would not pass through the 2D detection of the drones, and would also mean Red chooses not to target his opponent's building. Due to this problematic, the author attempted working with tighter expected angle ranges based on the (unknown) location parameters of the red mercenary, by including them as additional factors in the likelihood function. such that now

$$p(\theta, \phi|\alpha, \beta, \gamma) = \begin{cases} \frac{1}{(\theta_2-\theta_1)(\phi_2-\phi_1)} & \text{for } \theta_1 \leq \theta \leq \theta_2 \text{ and } \phi_1 \leq \phi \leq \phi_2 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $\theta_1, \theta_2, \phi_1, \phi_2$ are functions of α, β, γ and the building dimensions, such that these angle ranges always ensure the red mercenary lands a shot on the opposing building. Working with this type of likelihood lead to undesired and infeasible computational outcomes. A compromise was thus made to keep the *expected* range of firing angles constant, regardless of the red mercenary's possible positioning, but to constrain the *actual* angle ranges that Red will take, such that all his shots would always hit the blue mercenary's building. This way, not only are the shots being fired more realistic (avoids shots to the ground and straight up, not directed towards building), but also all shots fired will always be visible to the drones on the outside.

The downside to this choice is that the MCMC will most likely lose some accuracy in its parameter-space tracking. This is because the Bayesian approach used to infer the position of an emitter relies on projectiles emitted at uniform angle ranges having larger probability of detection directly under the emitter. By not detecting projectiles emitted at wide angles, we effectively make the MCMC parameter-space walk underestimate the distance between the emitter and the detection plane. This will be the case if the red mercenary's separation between Blue's building increases. However, since Blue requires only a valid direction to a coordinate rather than the coordinate itself, the trade-off might not affect the desired result too much.

1.2. *Changing variables in the likelihood*

The change in variables for a 2D pdf, given some parameters Θ, is defined as

$$p(y_1, y_2|\Theta) = \frac{p(x_1, x_2|\Theta)}{\left|\frac{\partial(y_1,y_2)}{\partial(x_1,x_2)}\right|} \tag{3}$$

The likelihood pdf for θ and ϕ is constant and the detected points are along the X-Z axis.
Since (x or z) = (α or γ) + $\beta tan(\theta$ or $\phi)$, the Jacobian in the denominator of Eq. 3 evaluates to $\beta^2 sec^2\theta sec^2\phi$. This allows for a change of variables

$$p(x, z|\alpha, \beta, \gamma) \propto \frac{cos^2\theta cos^2\phi}{\beta^2} = \frac{\beta^2}{(\beta^2 + (x-\alpha)^2)(\beta^2 + (z-\gamma)^2)} \tag{4}$$

And thus appears our likelihood pdf as a function of X, Z. It can now be used in our MCMC procedure to deduce a position above this X-Z plane given some x,z coordinates.

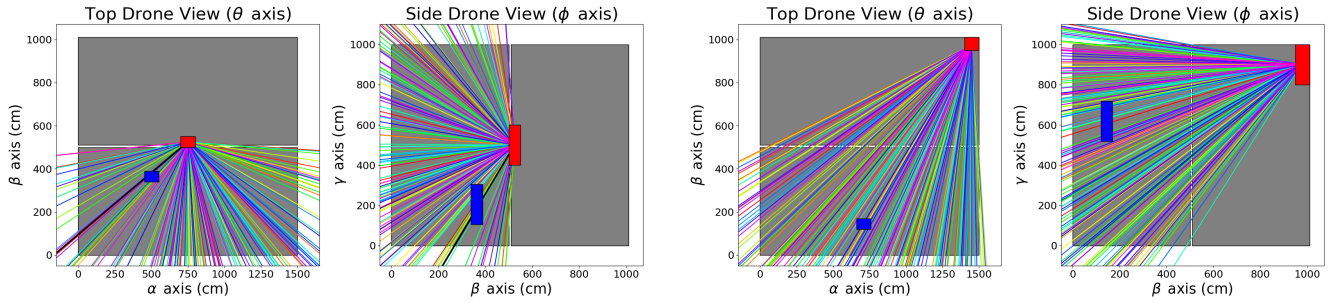
2. IMPLEMENTATION AND RESULTS

The first thing to do was instantiate the environment. Because Python's matplotlib.pyplot module does not render 3D figures exceptionally well, it was best to represent the various elements of our shootout in 2D form. This was illustrated in Fig. 1, where (α, β, γ) represent the 3 dimensions of our universe in units of cm. Mercenary locations can be user-generated or randomized in each building. The same is true for the dimensions or "hit boxes" representing their bodies. When running the MCMC, attributes such as after how many shots taken does the MCMC start and after what amount of shots collected to we decide to run it again can be varied to the user's discretion. The default setting makes it start after 20 shots received and calculates a new inferred location every 10 additional shots.

Once the environment variables are set, the battle commences. There are 3 outcomes. Either the red mercenary is able to strike within range of Blue's hit box using random firing, or Blue is able to correctly line up a shot towards Red's hit box, or he misses and is inevitably doomed to perish. Although this could be simply played as a game, the goal here will be to analyse Blue's MCMC algorithm and its performance given different mercenary locations and allowed firing angles. It will therefore be possible to comment on where both mercenaries should hide inside their respective buildings to maximize either hitting their opponent or avoiding being shot/detected.

The first test will be to assess the locating capabilities of Blue's algorithm when facing the best-case scenario. This can serve as a proof of concept that an opposing shooter can be adequately found under ideal circumstances. This best case is when both firing angles have maximum spread which still allow Blue's drones to detect them. This would correspond to a shooter location shown in Fig. 2(a). The counterpart to the easy case would be something like 2(b), where the shot spread is constrained to due to θ and ϕ ranges almost being cut in half.

Angle Ranges For Two Different Shooting Locations



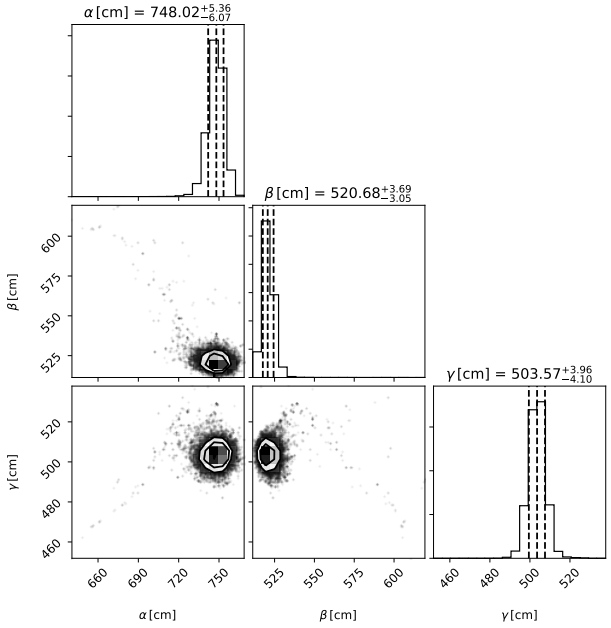
(a) (b)

Figure 2: The widest possible θ and ϕ range is approximately $-\pi/2 < \theta, \phi < \pi/2$ and $\phi_\phi \leq \pi/2$. This inequality holds only if the mercenary is at his minimum β position and if there is no separation between buildings. This spread is seen in (a). A worst case spread can be thought of as all angles that allow the opposing building to be hit, as seen in (b).

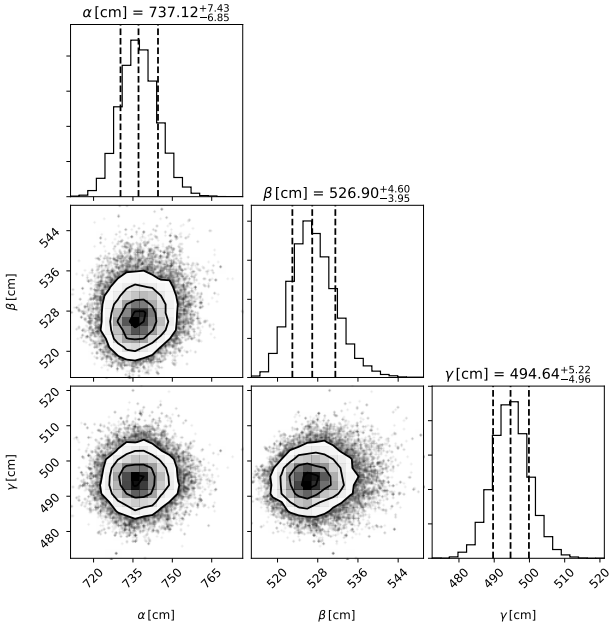
Running the MCMC two times on both situations shown in Fig.2, but with only 40 shots fired, the inferred emitter locations are shown in corner plots.

For the best case, Figs. 3a and 3b tell us that estimated values of (α, β, γ) all fall within 1 or 2 standard deviations of the actual values, corresponding to differences on the order of $10^1 cm$. Given that the opponent is not the size of a

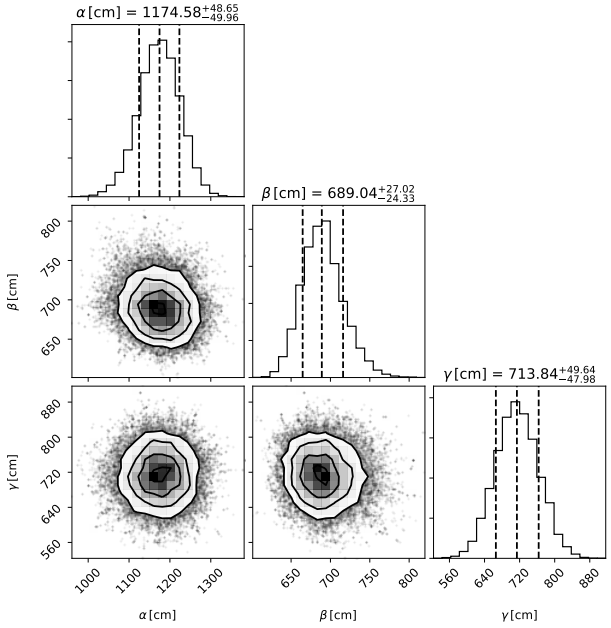
Corner Plots of the (α, β, γ) coordinates found by the MCMC



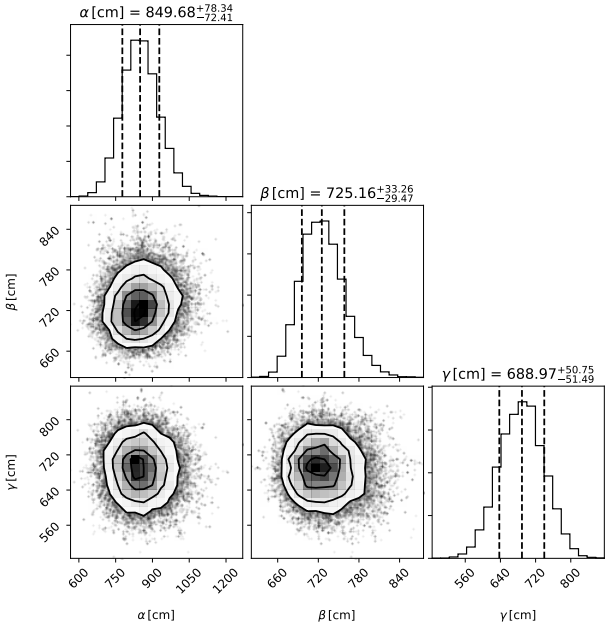
3a



3b



3c



3d

Figure 3: Corner plots showing the estimated positions of a shooter. Figs. 3a and 3b show results for when the shooter was at $(\alpha, \beta, \gamma) = (750.0, 525.0, 500.0)$ cm, as shown in 2a. Figs. 3c and 3d show results for when the shooter was at $(\alpha, \beta, \gamma) = (1450.0, 975.0, 900.0)$ cm, as shown in Fig. 2b. All dotted lines and reported values correspond to the median, 16th and 84th quantiles of each parameter. Circular contours inddicate no covariance in the parameters.

grapefruit, this shows that their locations can be resolved accurately if firing angles are minimally constrained.
For the worst case, Figs. 3c and 3d show estimated (α, β, γ) values that stray away from their true values differences

of $\sim 10^2 cm$. The estimated coordinates also have their standard deviations approximately increased by a factor of 10. This undoubtedly shows our MCMC has difficulty estimating the location of opponents that limit their firing angles, as the shots have uniform angular distribution over a smaller range which leads to underestimates in the parameters of interest.

The outputs of the MCMC can be best seen as a scatter plot of the various locations it finds for a constant emitter location. Figs. 4,5 show the output of 10 MCMC runs, processed with the same amount of 40 random points for each run. The best case location for an emitter has impeccable accuracy. The worst case positioning of an emitter leads to an underestimate of all parameters as well as a larger spread than the values found in the best case shooter location.

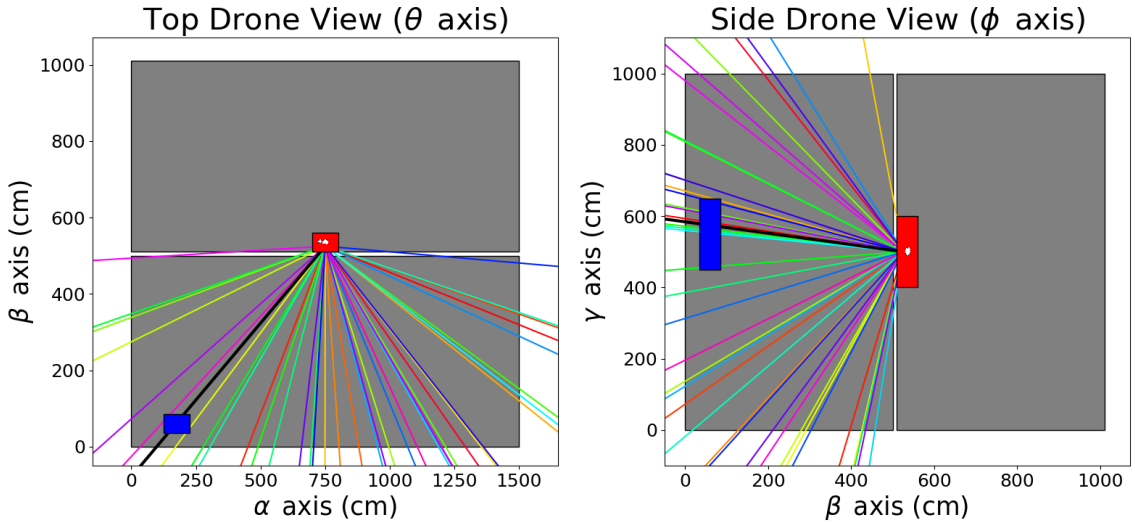


Figure 4: 10 MCMCs executed to locate a best-case emitter. What should be white crosses representing 1σ error bars laid over a black rectangle are in fact visible as tiny white dots at the center of the red mercenary's hit-box. This indicates all MCMC executions were accurate enough to correctly locate the projectile emitter.

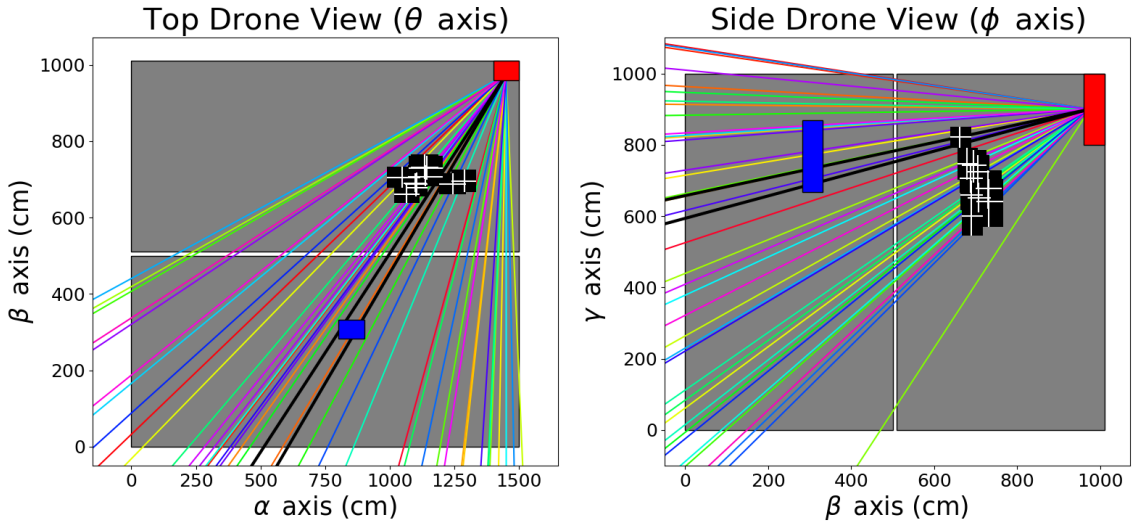


Figure 5: 10 MCMCs executed to locate a worse-case emitter. Compared to Fig. 4, the black regions corresponding to 1σ errors are more pronounced as well as being more dispersed. The inferred locations are inaccurate while maintaining reasonable precision.

3. DISCUSSION

With what has been presented thus far, what could the blue mercenary deduce from the data he receives from his drones? The first thing he can look for in an inferred location is whether the parameter's 68% confidence intervals are smaller than or equal to the opposing mercenary's hit box. Given that 1σ uncertainties for an emitter's worst-case location seen in Fig. 5 and a human's size are both along the order of $10^2 cm$, choosing to retaliate when the 68% confidence thresholds become lower than the opponent's hit box dimensions is a reasonable approach.

We have seen that for a best-case positioning of an emitter, meaning that it is very close to the 2D detection plane, the MCMC tends to output a precise position. The 68% confidence interval is on the order of $10^1 cm$. Upon receiving such a precise answer, the blue mercenary should not hesitate to return fire in that direction since it has also been observed that precise outputs also tend to be very accurate.

If his MCMC outputs a location value with 68% confidence intervals of order $10^2 cm$, he could assume that the inferred coordinates are likely not accurate, despite reasonable precision. However, from Fig. 5 we observe that the MCMC outputs seem distributed along an X-Z plane at relatively constant β. Furthermore, from Fig. 6, a greedy observer could deduce that the shooter's location, his inferred location and the center of the building he's shooting at seem aligned. The blue mercenary, if desperate enough, could extrapolate the shooters actual location in 3D given his building's center coordinate and his inferred shooter location. But that would not be a very rigorous statistical approach. Alas, Blue decides he is going to live and die by Bayes Theorem. However, if he does decide to be in the center, the line of sight between him and the inferred location could still intersect with his opponent, at which point contact could still be made by a retaliatory shot.

The performance of Blue's MCMC algorithm teaches us a few things about where each person should hide inside their respective buildings. For Red, it is obvious that the more he can constrain his uniform firing angles, the more Blue's MCMC algorithm loses precision and accuracy. This means he should always find refuge in the back corners of his building, since this is where his firing angles are more limited. For Blue, he has seen that in the best case where there is minimal separation between his opponent and his building, the precise outputs of his algorithm tend to be accurate. He has also seen that worst-case opponent locations still lead to satisfactory precision in inferred coordinates, but often lack accuracy. However, there is still an ideal location for him to be in. Since Red's shots will always be directed towards Blue's building, and that the actual and inferred positions of Red seem aligned with the Blue's building center, the ideal location for Blue would be exactly in the middle of his building. This will allow his retaliatory shots to still be in the general direction of his opponent. Figs. 7 and 8 show that for the same opponent placements and MCMC parameters, his location in 7 does not allow him to connect with his opponent, while his placement in 8 does.

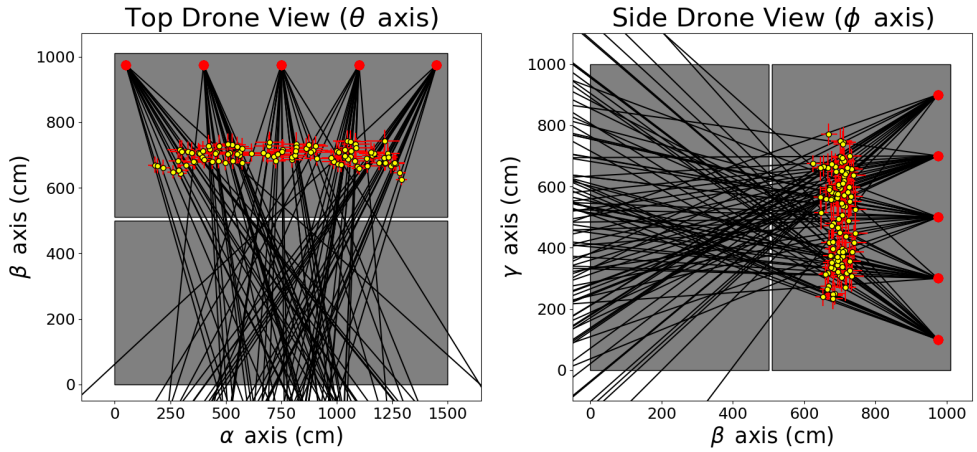


Figure 6: Plot of a total of 25 different emitter positions across an X-Z plane which lies at $\beta = 975$. The yellow dots represent an inferred position by the MCMC, of which 4 were performed for each of the 25 emitter positions. 40 shots were used in the MCMC for all $25 * 4 = 100$ inferred positions. We observe that the results are relatively spread out along $\beta \approx 700$. The black lines connect the actual and inferred positions and show that these points line up vaguely towards the opposing buildings center coordinate.

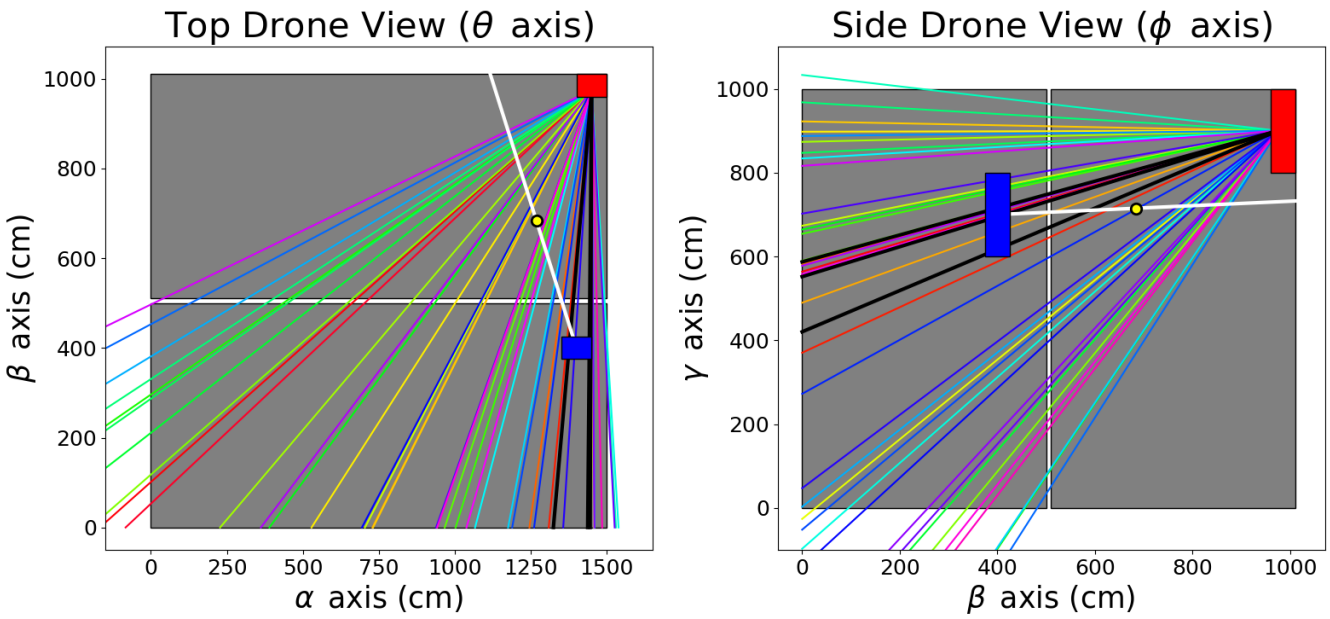


Figure 7: An example outcome of an inaccurate inferred location that leads to a missed shot by Blue.

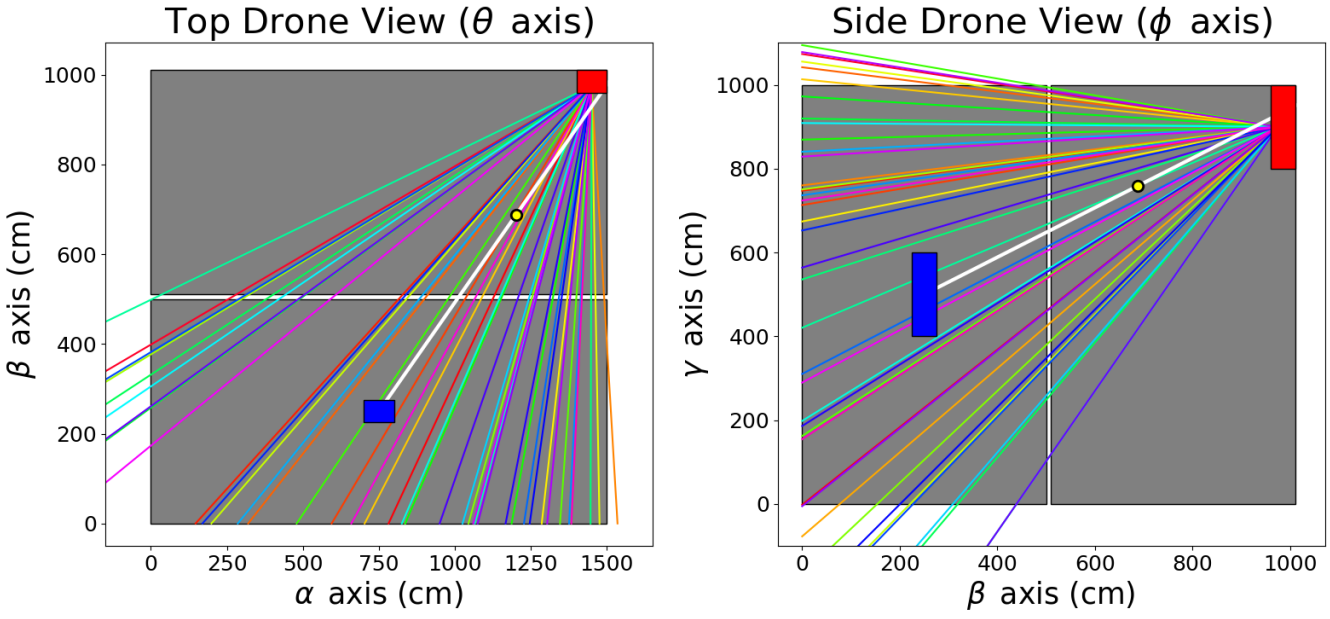


Figure 8: An example outcome of an inaccurate inferred location that leads to a successful shot by Blue. By positioning himself in the center of the building, the trajectory of his shot based of the MCMC result had better probability of lining up with the actual location of his opponent.

4. CONCLUSIONS

The performance of Blue's MCMC algorithm teaches us a few things about how an emitter's location in 3D space can be can be inferred by detecting its projectiles targeted towards another object. The key constraint here is that the emitter is deliberately targeting a flat surface over some range of uniform angles, which is different from having uniform distribution over all angles. By having a 2D detection plane to measure impact positions of these projectiles, in the best case we can expect a precise and accurate inferred 3D location. In the worst-case, we can still expect satisfactory precision but poor accuracy in the inferred location. However, if an observer would be positioned at the center of the object being targeted, we can guess that the inferred location from our point of view is in the general direction of the emitter location. In the situation introduced at the beginning of this report, where we desire to target the emitter, a good general direction might be all that is required for a successful shot.

I encourage you to bring the story shown in this paper to life by running the python code BayesianShootout.py found in the github repo of this project!