Background on spot normalization

A point cloud $\mathcal{P} \in \mathbb{R}^{N \times 3}$ from our lidar data has some weird sweeping patterns. If I were to plot the local density of points from the sky it would look like this:

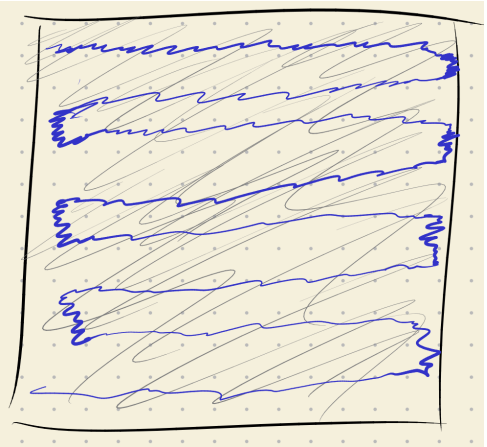
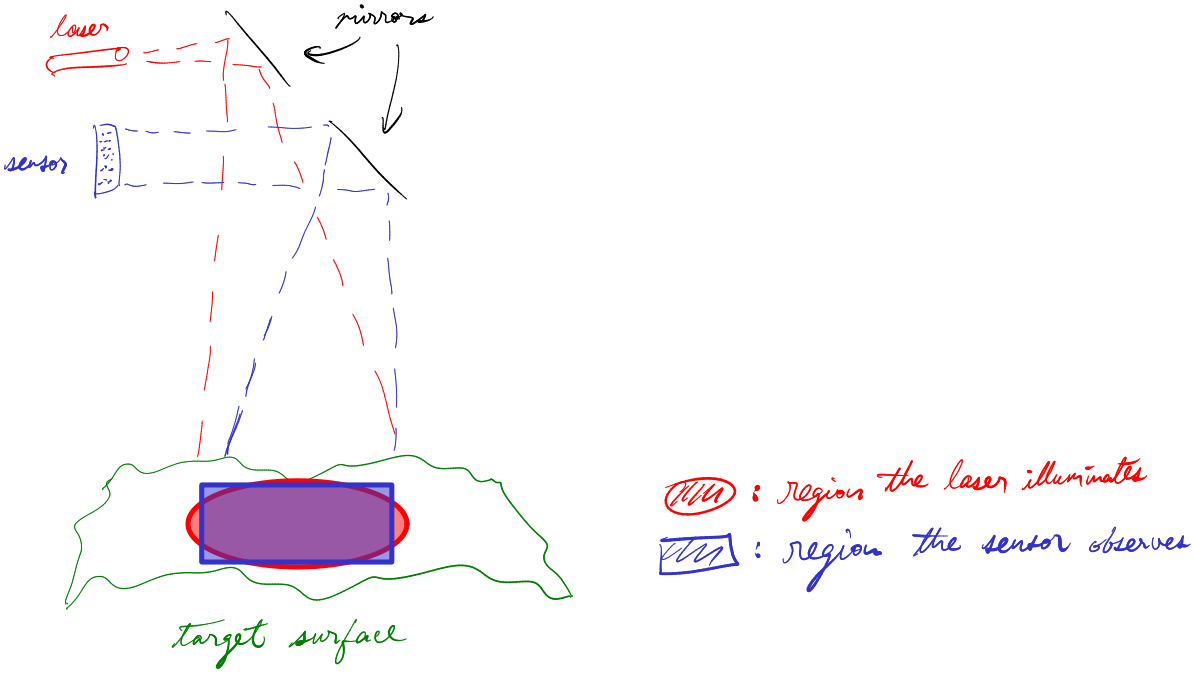


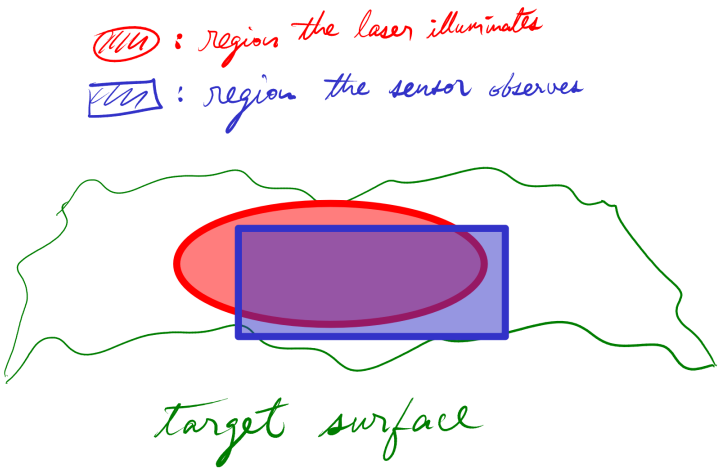
Figure 1: Snaking pattern

where blue represents regions of higher density.

This is because of the path our lidar sensor scans in, and the way it illuminates a given area.



The blue and red spots move more or less together in a snaking pattern over the target, as in Figure 1. At times, they may not be perfectly aligned, as depicted below:



Due to the snaking pattern and misalignment, certain spots on the target surface might have higher probability than others of being observed. Given the sensor observation region, the laser illumination region, and the snaking pattern all over time, we would like to model the probability that a given spot on the ground is observed. Then we can normalize our data's density by that probability, yielding a more meaningful density measurement.

Approach

Setup

Assumptions

We assume the region that the laser illuminates is a spot with a Gaussian distribution (Wikipedia, Gaussian beam). Our samples of this region come through the camera, but the region the camera can observe is limited, so what we observe is a 2-dimensional truncated Gaussian distribution. We already know the bounds of truncation since we know how many pixels are in the sensor grid. We want to learn the mean and covariance of the illumination spot throughout the scan.

In the scan, we think the intensity of the laser doesn't change, but due to misalignment, the position of the mean will change. The covariance might change too.

Hitmaps

At each frame, 20 to 30 percent of the pixels in the sensor record a hit. A *hitmap* is a matrix where each element represents a pixel in the sensor grid, with 1 representing a hit and 0 representing no hit in a frame. By aggregating hitmaps from consecutive frames together we can estimate the illumination region within an interval of time. To perform such modeling I used a window size of 100. In other words, to model the illumination spot at frame 300, I used data from frames 250 through 349.

GMMis

A common tool for estimating the mean and covariance of a Gaussian given some samples is expectation maximization via Gaussian mixture models (GMMs). However, since our data is truncated, we saw that using GMMs naively gave our estimated means a bias towards the center and the estimated covariances shrunk too small. Some astronomers faced a similar problem with their data and created a modified expectation maximization procedure to solve this and other problems, and released their code under the MIT licence as pygmmis (Filling the gaps: Gaussian mixture models from noisy, truncated or incomplete samples). See Figure 2.

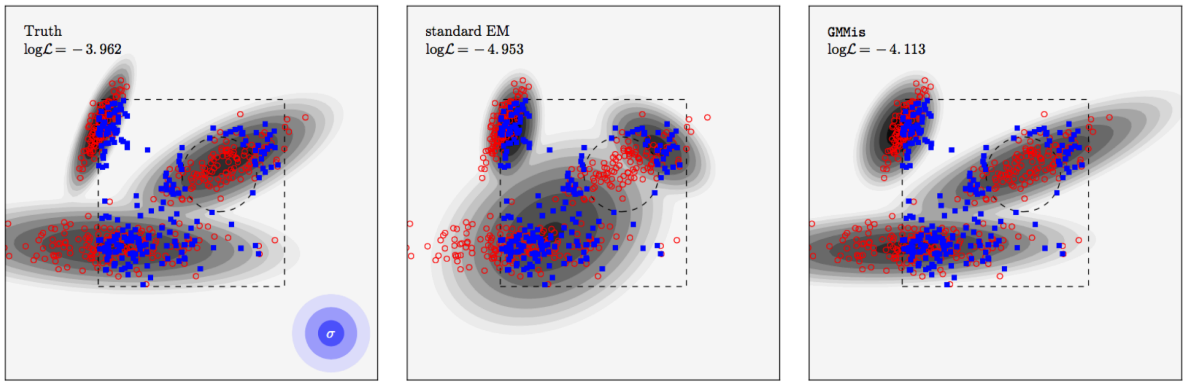


Figure 2: From pygmmis project on GitHub: "In the example above, the true distribution is shown as contours in the left panel. We then draw 400 samples from it (red), add Gaussian noise to them (1,2,3 sigma contours shown in blue), and select only samples within the box but outside of the circle (blue)."

It solved the issues we were facing with biased means and covariances, as shown in Figure 3.

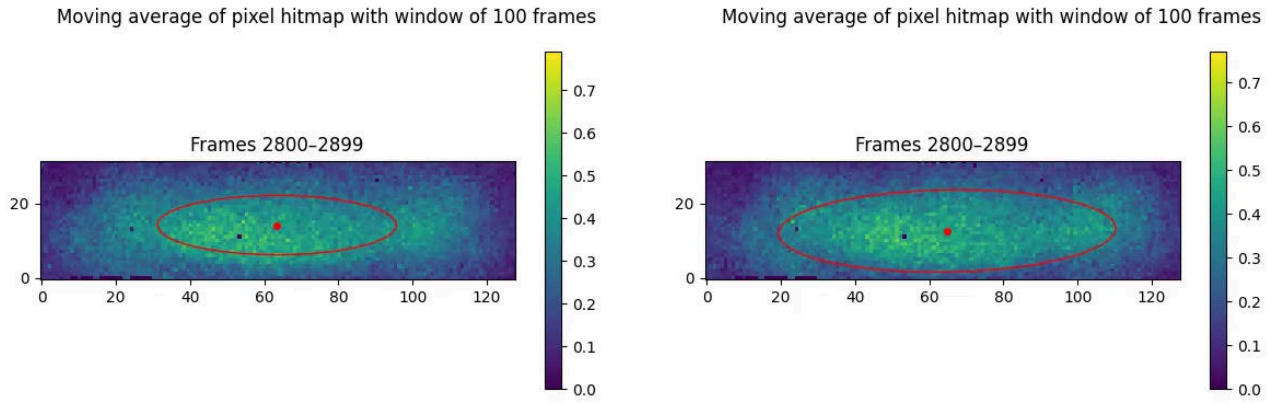


Figure 3: In both images above, the underlying heatmap, which contains 100 hitmaps averaged together, is the same. The red dot is the estimated mean, and the ellipse marks one standard deviation. **Left**: Gaussian spot modeled using standard GMM with one component. **Right**: Gaussian spot modeled with pygmmis. The mean is lower, where one would expect, and the covariance is much larger.

Results

To apply GMMis to estimate means and covariances over a whole scan, we do the following:

Algorithm 1

- Fix a window width $w \in \mathbb{N}$ to be the number of hitmaps we aggregate to infer the spot distribution at a given frame. I used $w = 100$.
- Throw out 95% of the data at random (for speed).
- For each frame:
  - Collect together all pixel hits in the window around this frame.
  - Fit a GMMi model to them.
    - For each model, clarify in the settings that for a hit point from pixel (x, y), we know that $0 \leq x < n_x$ and $0 \leq y < n_y$ where the pixel grid is $n_x \times n_y \in \mathbb{N} \times \mathbb{N}$ pixels.
  - Extract the estimated mean and covariance from the model.

Applying this method over an entire scan, we saw that the mean of the illumination spot tended to move up and down with perfect periodicity. See a brief animation here, and a plot of this phenomenon in Figure 4.

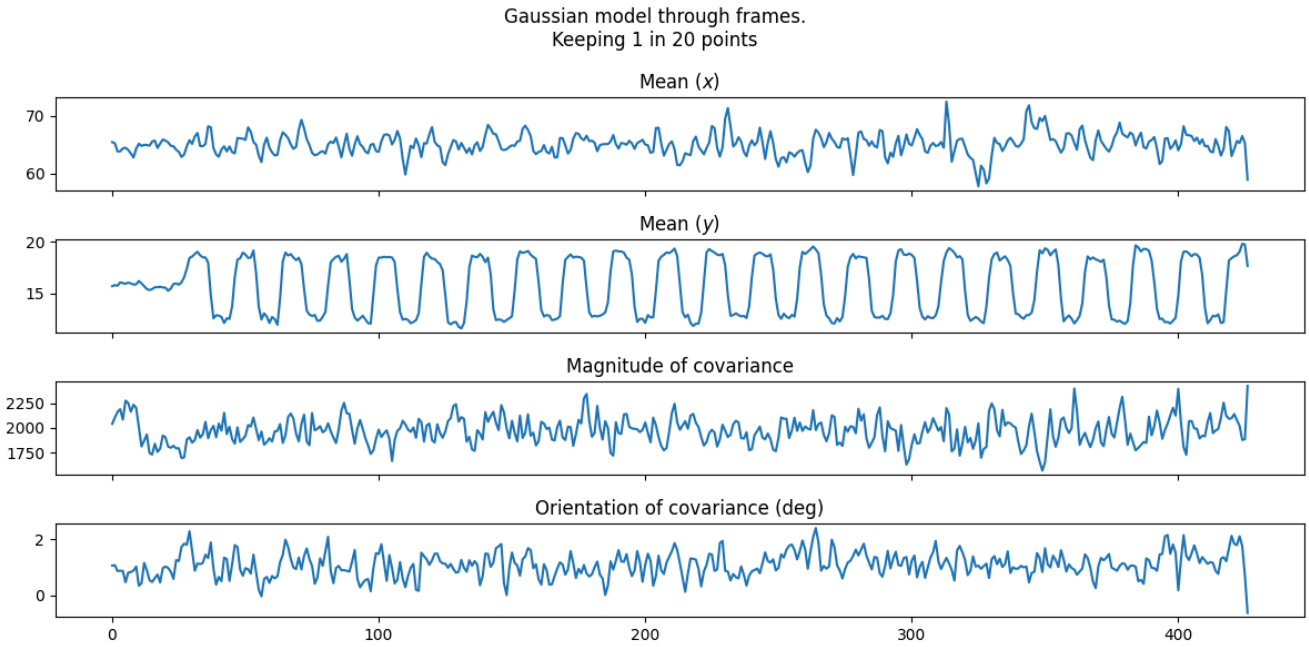


Figure 4: The y position of the illumination spot's mean moves periodically.

It's too slow, though. Even after throwing out 95% of hits, fitting the GMMi models takes around 5 minutes for one scan on my laptop with 32 gb RAM and an Intel Ultra 9.

Modifying method for efficiency

One can see in Figure 4 that the mean oscillates back and forth, while the covariance stays more or less the same (with noise). Each time we fit a GMMi model, we are recalculating the full distribution parameters, but in reality, many of the parameters stay the same. (I think pygmmis has a way I can encode this… maybe I should try doing that too.)

A modified algorithm vastly sped up the modeling process:

Algorithm 2

- Fix a window width $w \in \mathbb{N}$ to be the number of hitmaps we aggregate to infer the spot distribution at a given frame. I used $w = 100$.
- (Unlike before, keep all the data.)
- Estimate constant illumination spot covariance.
  - Randomly sample k window positions. I used $k = 3$.
  - In each of those window positions, fit a GMMi model to the data in the window, and save the resulting covariance.
  - Estimate the actual covariance by averaging together the k covariances just calculated.
- Calculate the spot's mean at each frame. (Actually, every n frames. We can interpolate to find the values in between. I used $n = 50$.) For each frame:
  - Average together hitmaps for all frames in this frame's window, and call it the *average hitmap*.

- Apply a strong Gaussian blur to the average hitmap. TODO: refine this explanation
  - Blurring too much would bias the position of the maximum, and not blurring enough means there are too many local maxima.
  - Blurring evenly in all directions caused too much blurring in one direction while there still wasn't enough in the other.
  - The Gaussian spot had much greater variance in one axis than the other, as seen in Figure 3. To account for this, when applying Gaussian smoothing, I kept the ratio of blurring in one axis to the other the same as the ratio of the diagonal elements of the covariance matrix.
- Since the mean of a Gaussian is where the distribution's density is maximized, estimate the mean as the position where the blurred average hitmap is maximized.

We found that Algorithm 2, while much faster than Algorithm 1, yielded means with a bit more fluctuation and bias. The means did not fluctuate so much as to cause serious issues for our purposes, though. See Figure 5.

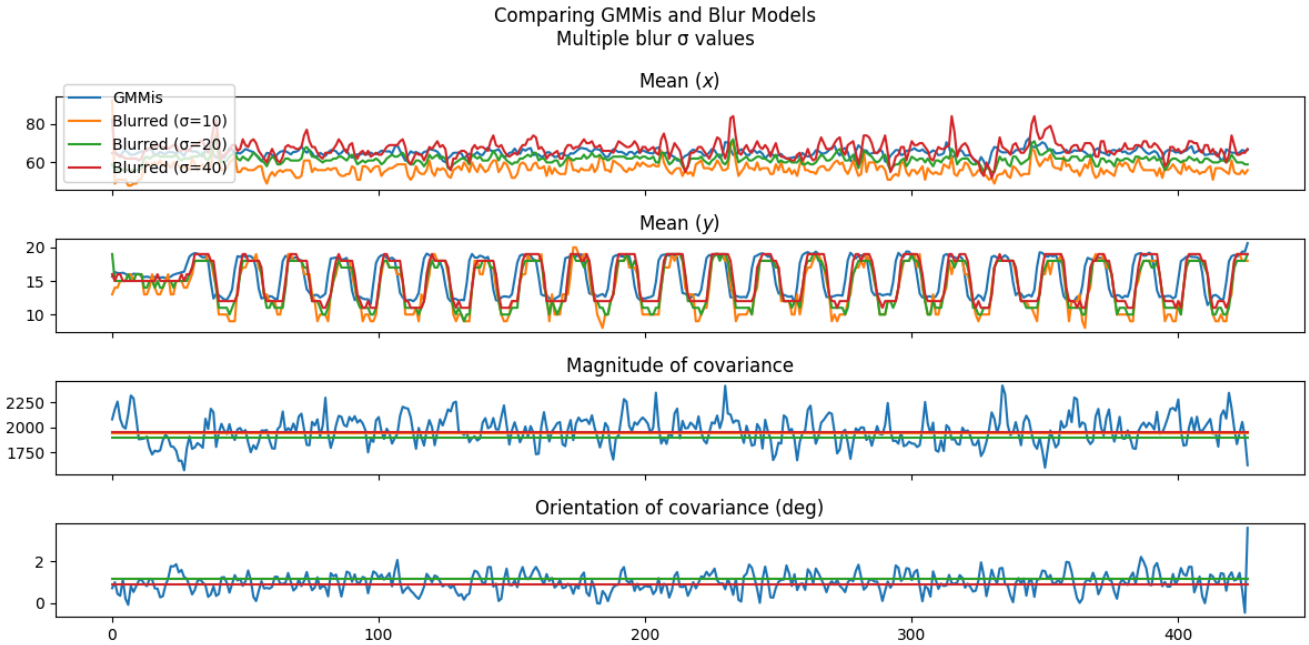


Figure 5: Comparing various spot illumination models. The lines labeled "GMMis" were calculated with Algorithm 1, and the rest were calculated with Algorithm 2, with various σ settings controlling the degree of blurring. The blue "GMMis" lines can be considered the truth, and the other "Blurred" lines an approximation of that truth. One can see that different σ values had generally similar results, with higher σ values performing a bit better since they have less bias in the mean. TODO: Investigate off-by-one issue