ISTA 421 / INFO 521 – Final Project OPTION A: Metropolis-Hastings MCMC Inference of 3D Line

Due: Monday, December 13, 5pm

Total 20% of final grade

Emilio Popovits

Undergraduate

Introduction

The following are the instructions for Option A of the final project. If you are doing option B, you do NOT do this option.

The instructions start with a general background description of the model and problem scenario, followed by a set of Tasks. You must commit and push your executable code along with a PDF of written answers to your final_project repository. The name of your written PDF must be `final_project_OptionA_answers.pdf`. The first task has you implement the core code to your Metropolis-Hastings sampler. Tasks 2 through 5 then have your run that code on the provided data in various configurations, generate plots, report values, and describe the outcomes.

Option A Background: Inferring parameters of a 3D line from noisy 2D images using Metropolis-Hastings MCMC

In vision problems, we often represent the process by how an image get produced by considering a parameterized camera model (in a sense, this defines the "perspective" toward the scene) and the scene being imaged.

In the pinhole camera model, cameras can be represented by matrices that contain parameters for the camera, such as its position in space and its focal length.[1] This representation is useful because it allows us to obtain images of 3D points in a very simple way.

Let $\mathbf{M} \in \mathbb{R}^{3\times 4}$ be a camera matrix. Then, if $\mathbf{p} \in \mathbb{R}^3$ is a 3D point (i.e., $\mathbf{p} = [p_1, p_2, p_3]^\top, p_i \in \mathbb{R}$), its 2D image $\mathbf{q} \in \mathbb{R}^2$ is given by

$$\mathbf{q} = \begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{\widetilde{w}} \begin{bmatrix} \widetilde{u} \\ \widetilde{v} \end{bmatrix} \tag{1}$$

where $\widetilde{u}$, $\widetilde{v}$, and $\widetilde{w}$ are defined by

$$\begin{bmatrix} \widetilde{u} \\ \widetilde{v} \\ \widetilde{w} \end{bmatrix} = \mathbf{M} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} = \mathbf{M} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{bmatrix} \tag{2}$$

Notice the number 1 that is appended to $\mathbf{p}$ when multiplying by $\mathbf{M}$. This is called the *homogeneous* coordinate, and it is necessary for posing camera projections as matrix multiplication. This projection principle is illustrated in Figure 1

[1] For those interested (*not* required for this project), details can be found in Hartley and Zisserman (2004) *Multiple View Geometry in Computer Vision*; you can also internet search for "camera matrix."

(a) A depiction of the projection of three points $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3 \in \mathbb{R}^3$ onto the image plane of a camera using Eqs 1 and 2.

(b) Graphical depiction of the 2D image plane with image points $\mathbf{q}_1$, $\mathbf{q}_2$, $\mathbf{q}_3 \in \mathbb{R}^2$ projected from points $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$.

Figure 1: Two views of projection.

(a) A 3D line segment $\boldsymbol{\ell} = (\mathbf{p}_i, \mathbf{p}_f)$ (in red) projected onto the image *via* camera $\mathbf{M}$ (situated at the origin), producing a (non-noisy) 2D line $\mathbf{l}_{2D} = (\mathbf{q}_i, \mathbf{q}_f)$ (in blue).

(b) Image of $\boldsymbol{\ell}_{2D} = (\mathbf{q}_i, \mathbf{q}_f)$ of the projected 3D line segment in the camera u,v plane; $\mathbf{q}_m$ is one of the sampled points along the line, which has noise, ϵ, added to it in the image-creation process, generating the observed image point $\mathbf{r}_m$; the concentric circles around $\mathbf{q}_m$ represent the contours of the 2D Gaussian noise, $\boldsymbol{\epsilon}$.

Figure 2: The noisy generative process projecting end-points along the line to the image

NOTE: A complete understanding of this material, while very valuable in itself (!), is not necessary to complete this problem. The only thing you are required to understand is that 3D points (in coordinates x, y and z of Figure 1a) become 2D image pixel points (in coordinates u, v of Figure 1b) by Equations 1 and 2.

We will start by defining a "Camera 1" whose matrix is

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

which corresponds to a camera located at world coordinates $(0, 0, 0)$ and pointing down the positive z-axis, with unit focal length. What follows is the problem setup.

Let $\boldsymbol{\ell} = (\mathbf{p}_i, \mathbf{p}_f)$ be a 3D line segment with end-points $\mathbf{p}_i, \mathbf{p}_f \in \mathbb{R}^3$. (The subscripts on the end-points stand for "initial" and "final", respectively.) Now, consider the following generative process. We "take a picture" of $\boldsymbol{\ell}$ using camera $\mathbf{M}$, producing $\boldsymbol{\ell}_{2D} = (\mathbf{q}_i, \mathbf{q}_f)$ (that is, the $\mathbf{q}_i$ and $\mathbf{q}_f$ are obtained using Equations 1 and 2).

We then sample S points $\mathbf{q}_s$ from the line $\boldsymbol{\ell}_{2D}$. This is accomplished by taking the ratios $t_1, ..., t_S$ along the straight-line path from $\mathbf{q}_i$ to $\mathbf{q}_f$ so that $\mathbf{q}_s = \mathbf{q}_i + (\mathbf{q}_f - \mathbf{q}_i)t_s$, with $0 \leq t_s \leq 1$, $s = 1, ..., S$. (Figure 2(b) shows one of these sampled points, $\mathbf{q}_m$, which happens to exactly be half way between $\mathbf{q}_i$ to $\mathbf{q}_f$, so for this point $t_m = 0.5$).

Since the imaging and rendering processes are inherently noisy, we add Gaussian noise to the $\mathbf{q}_s$, which gives us $\mathbf{r}_s = \mathbf{q}_s + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$, with $\boldsymbol{\Sigma}$ a 2×2 covariance matrix (since this noise is in the 2D image plane). Naturally, this implies that $\mathbf{r}_s | \mathbf{p}_i, \mathbf{p}_f, t_s, \boldsymbol{\Sigma} \sim \mathcal{N}(\mathbf{q}_s, \boldsymbol{\Sigma})$. See Figure 2 for a summary illustration of the generative process.

We will additionally place a Gaussian prior on the possible end-points, $\mathbf{p}_i, \mathbf{p}_f$, of the line $\boldsymbol{\ell}$ (this is a *different* Gaussian than the image Gaussian noise $\boldsymbol{\epsilon}$). These prior parameters are subscripted by the bold-$\boldsymbol{\ell}$ to indicate they refer to the properties of the 3D line. This represents the prior belief about where the line, as defined by end-points $\mathbf{p}_i$ and $\mathbf{p}_f$, might be in 3D space:

$$\mathbf{p}_i, \mathbf{p}_f \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\ell}}, \boldsymbol{\Sigma}_{\boldsymbol{\ell}}).$$

Tasks [30 points total]

NOTE: Some of the details of the tasks will be made more concrete once the data is released on Monday, November 20.

All data files needed for this task will be provided in `projectA-3D-line-MH/data_2018/`

For this project option, you will need to complete the following five tasks:

1. **[10 points]** Write a python script that, given

   - inputs $t_1, ..., t_S$ (the relative offset ratios of points $\mathbf{q}_s$ along the line segment between the $\mathbf{q}_i$ and $\mathbf{q}_f$ endpoints),
   - 2D points $\mathbf{r}_1, ..., \mathbf{r}_S$ (the observed points in the camera u,v image plane),
   - $\boldsymbol{\Sigma}$ (the image projection noise covariance),
   - $\boldsymbol{\mu_\ell}$ and $\boldsymbol{\Sigma_\ell}$ (the priors over the 3D line segment endpoints),

   uses the Metroplis-Hastings (M-H) algorithm to sample from the posterior of $\mathbf{p}_i$ and $\mathbf{p}_f$. In other words, the script should generate samples of $\mathbf{p}_i$ and $\mathbf{p}_f$ according to the posterior distribution

   $$p(\mathbf{p}_i, \mathbf{p}_f | \mathbf{r}, \mathbf{t}, \boldsymbol{\Sigma}, \boldsymbol{\mu_l}, \boldsymbol{\Sigma_l}) \propto p(\mathbf{r} | \mathbf{p}_i, \mathbf{p}_f, \mathbf{t}, \boldsymbol{\Sigma}) p(\mathbf{p}_i, \mathbf{p}_f | \boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_\ell),$$

   with $\mathbf{r} = \{\mathbf{r}_1, ..., \mathbf{r}_S\}$ and $\mathbf{t} = \{t_1, ..., t_S\}$.

   NOTE: The prior and likelihood probabilities can be *extremely* small, to the point that their product can result in floating-point number underflow errors – i.e., the computer cannot represent the small values and so treats them as though they were zero. This particularly becomes a hazard when you take the product of two very small numbers, which can happen when computing the posteriors in the M-H acceptance ratio, which in turn can lead to "division by zero" errors (when the denominator of the acceptance ratio is treated as zero). A work-around is to take the log (remember, we follow the machine learning convention that this is really the *natural log*) of the acceptance ratio (be sure to carry the log completely through the ratio and products), do your numeric computations to get the log-ratio α; you can transform the final value back into the original scale by computing e^α, or keep it in the log scale... as long as you also ensure your random sample is also log scale! If you use the log scale, the python function `scipy.stats.multivariate_normal.logpdf` will be handy.

   The written answer for this task (task 1) requires that you describe how to run your script for the following 4 tasks (tasks 2 through 5). As you complete the following tasks, you should update you description here.

   Solution: How to run the script: Run this script by getting into the `code` directory and running `python index.py`.

2. **[6 points]** Use your script to draw samples from the posterior distribution of the 3D line segment end-points.

   The observations you will use in this task are taken from the perspective of Camera 1. As described above, the observations consist of points $\mathbf{r}_s$ in the u,v image plane of the camera that result from a noisy projection process originating from points $\mathbf{q}_s$ along the 3D line segment. Each $\mathbf{q}_s$ was itself sampled some relative offset t_s along the 3D line segment (relative to the segment endpoints, $\mathbf{q}_i$ and $\mathbf{q}_f$). The data file `inputs.csv` contains the relative offset ratios for each $\mathbf{q}_s$. The projection of point $\mathbf{q}_s$, however, is not directly observed, but instead appears in the u,v image plane as the result of a noisy process (Figure 2(b)) that produces the observed point $\mathbf{r}_s$ in the u,v image plane coordinates. The u,v coordinates of each of these observations are contained in the data file `points_2d_camera_1.csv`. (The row indices of the offset ratios in `inputs.csv` corresponds to the row indices of the points in `points_2d_camera_1.csv`.)

Assume the following prior parameters: $\boldsymbol{\Sigma}_{\boldsymbol{\ell}} = 6 \cdot \mathbf{I}_{3\times 3}$ (where here $\mathbf{I}_{3\times 3}$ is the 3×3 identity matrix), $\boldsymbol{\mu}_{\boldsymbol{\ell}} = [0, 0, 4]^\top$.

Also assume that $\boldsymbol{\Sigma} = (0.05)^2 \cdot \mathbf{I}_{2\times 2}$ (where here $\mathbf{I}_{2\times 2}$ is the 2×2 identity matrix and $(0.05)^2$ is the scalar value of the square of 0.05).

It is recommended that you sample your initial end-points from the prior. Sample for $50,000$ iterations.

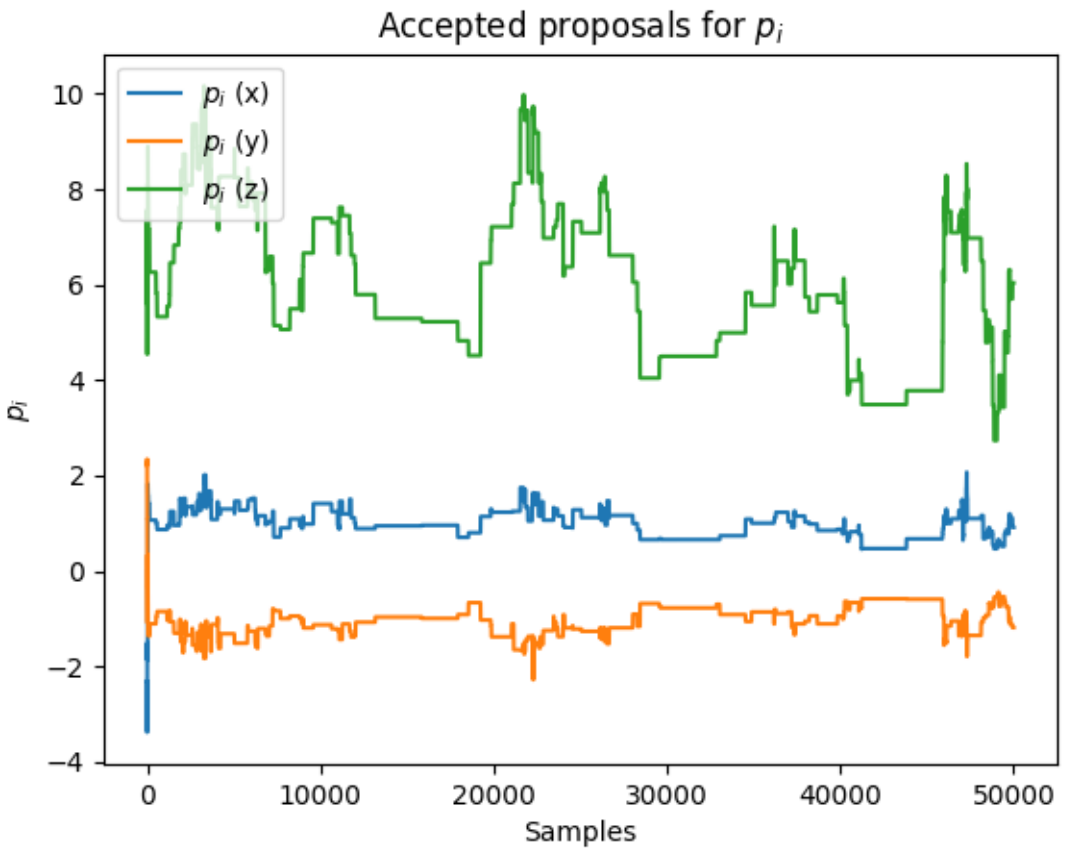
It is up to you to choose your proposal distribution! You may need to try some different parameterizations to get a good one.

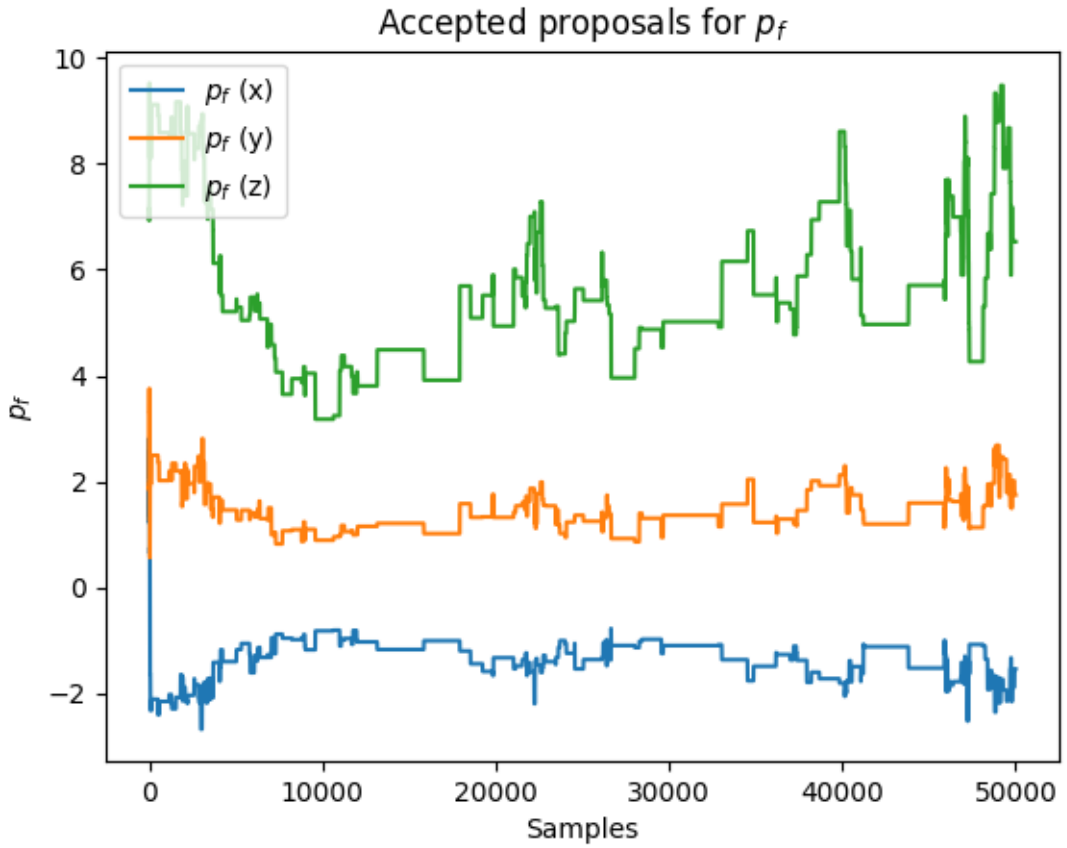
Note: you will likely find your best (MAP) fit points within the first 3,000-5,000 samples, but you should sample for $50,000$ to see what the longer-term trend in your estimates looks like.

In your written answers for this task, including the following:

- Generate a plot like that of Figure 4.12(c), page 161 of FCML (2nd edition), for each parameter you are estimating, to view progress during sampling. Remember, only plot the *accepted* samples, not the rejected samples.
- Describe the trends you see in these plots, and include an explanation of what they are telling you. Expect to see some differences between the plots you generate and what you see in FCML Figure 4.12(c).

Solution:



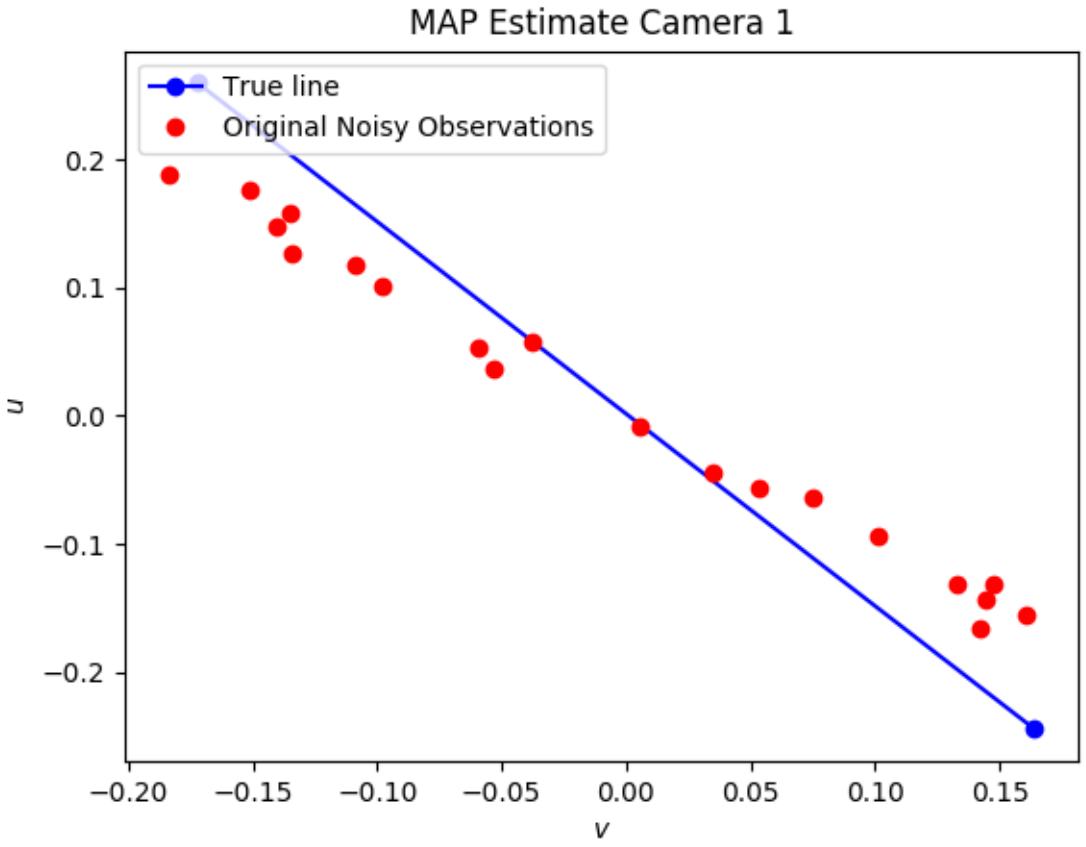


What I see in these plots is that p_i is converging on about $[1, -1, 6]$ as more samples are taken, and that p_f is converging on about $[-2, 2, 7]$ as more samples are taken. The differences between these plots and the one in FCML is that this plot accounts for each axis, and FCML's plot was just performing MH on one number.

3. **[2 points]** Using the samples you generated, find the maximum a posteriori (MAP) estimate of the line end-points. Report your results, including:

   - Report your MAP estimate of the line end-points (the 3d points of each endpoint).
   - Plot the MAP estimate of the line (that is, the MAP for each endpoint with a line connecting them) as projected onto the camera image plane (using the camera matrix). Include in the plot the original noisy observations, $\mathbf{r}$, of points sampled along the true line. You should see that your inferred line fits fairly well among the points. If not, try running your M-H procedure again with a different starting point for your line end-point hypothesis (it is generally a good idea to sample your starting end-points from the prior; update your plots in task 2 if you re-run, so that what you report there corresponds to the same data that you report here).

   Solution:



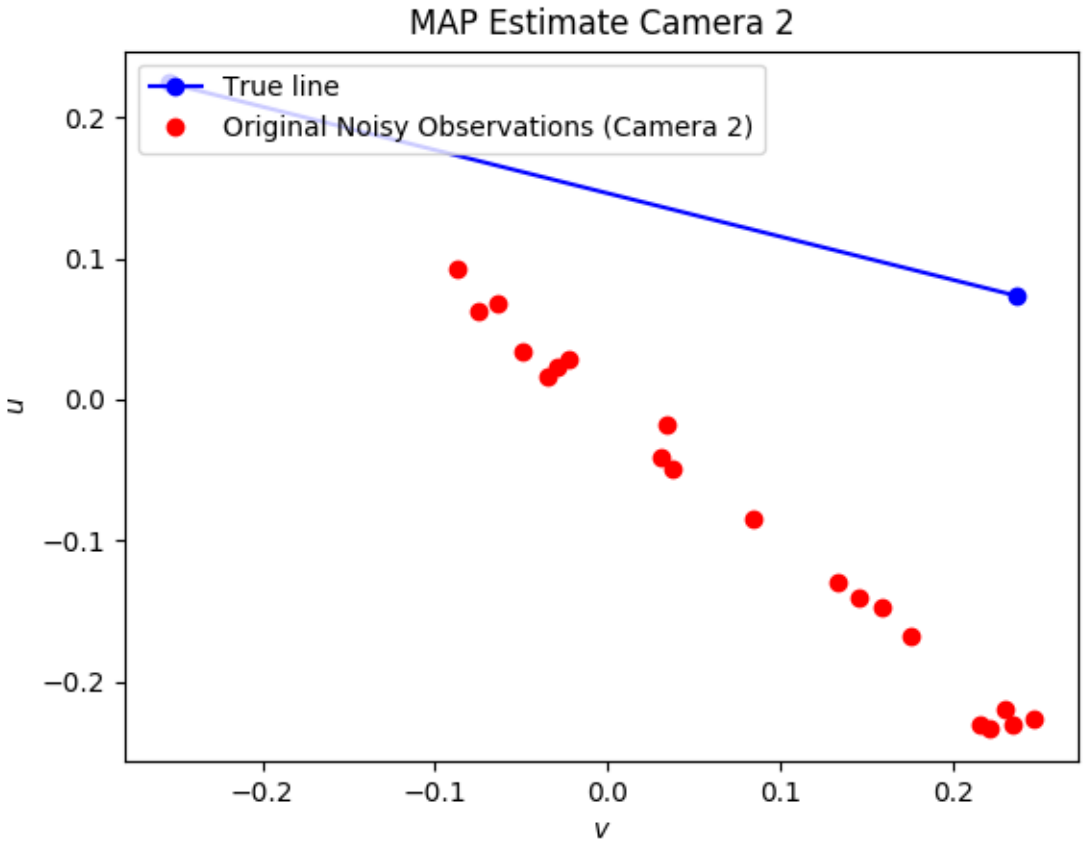
4. **[2 points]** Suppose we have a second camera

$$\mathbf{M}' = \begin{bmatrix} 0 & 0 & 1 & -5 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 5 \end{bmatrix},$$

(a camera at $[5, 0, 5]^\top$ looking in the negative x-axis and unit focal length), and, consequently, a second set of observed data points $\mathbf{r}'_1, ..., \mathbf{r}'_S$, where the indices of these points correspond to the same points viewed from the first camera (those were provided in `points_2d_camera_1.csv`), but now we have what those points look like through the second camera, as found in `points_2d_camera_2.csv`. First, let's see how your estimate from task 2 looks from this second camera:

- Create another plot like you did in task 3, where you plotted the MAP hypothesis of the end-points sampled in task 2, but instead use the second camera matrix to project your MAP hypothesis of the end-points (still connect them by a line), and also plot the new views of the points along the line from the perspective of the second camera, as found in `points_2d_camera_2.csv`. Don't expect the fit to be very good!

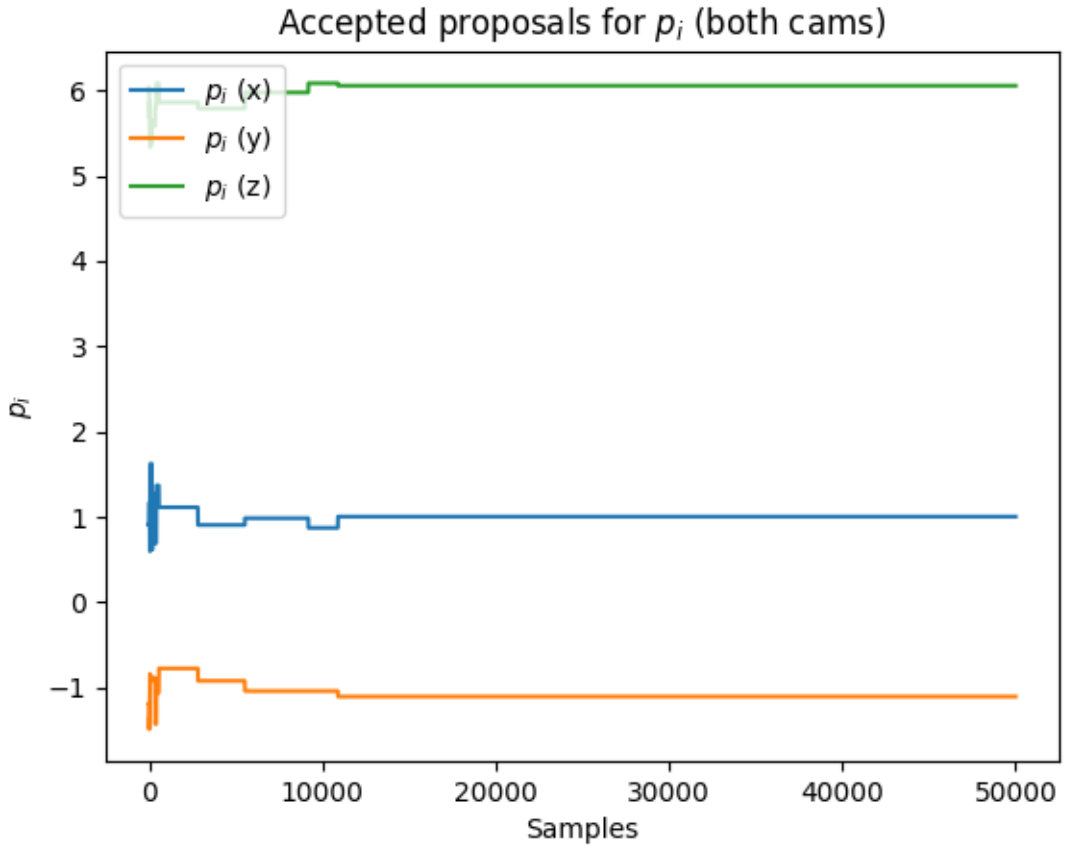
Solution:

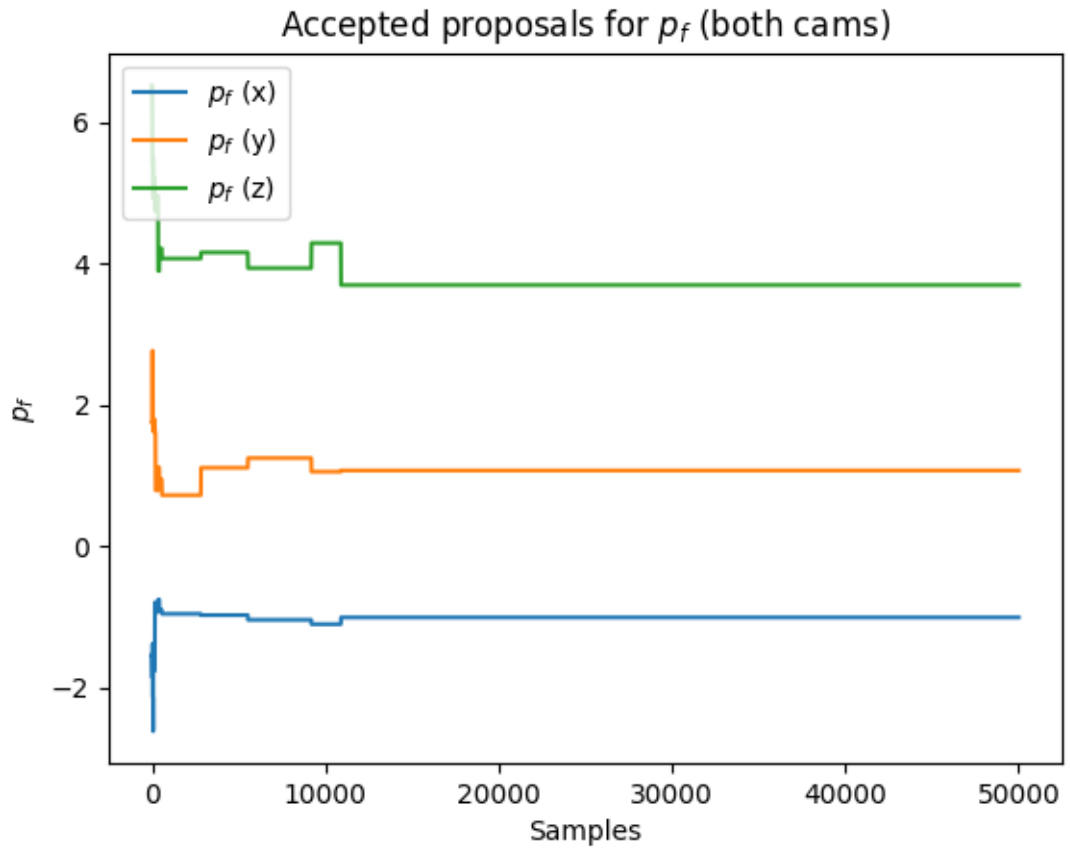


5. **[10 points]** Now, perform M-H sampling again, but this time use the data from **both** cameras. You will need to adapt your likelihood to now consider the two camera data sources, but the rest of M-H stays the same! Use the same observation noise parameterization (a diagonal matrix with $(0.05)^2$ down the diagonal), although note that now that you are combining the observations from both cameras, the dimension of the likelihood function is not the same as with a single camera! Again, $50,000$ iterations in M-H should be sufficient to converge and to see longer-term the trend. Report the following:

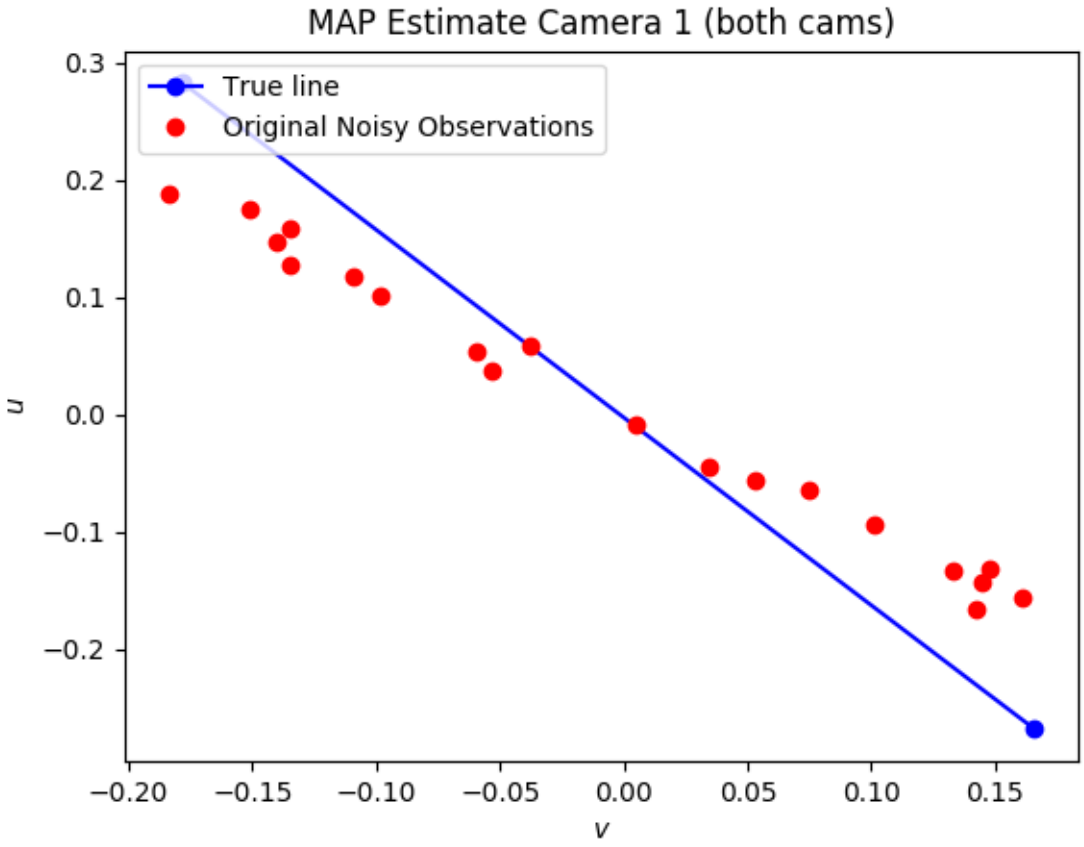
   - Find and report the MAP estimate of the line using the same setup as in task 2 using both the observations in `points_2d_camera_1.csv` from the first camera and the points `points_2d_camera_2.csv` viewed by the second camera.
   - Make the same plots that you did in task 2. Do they qualitatively look the same as those from task 2? If they're different, describe how and offer an explanation as to why we see those differences.
   - Plot the projection of your **new** MAP end-points estimate for the combined cameras; make one plot each for the projection of your MAP end-points connected by a line for the first camera (along with the noisy data for that camera), and also for the new camera. How well do you fit the points in both camera projections?

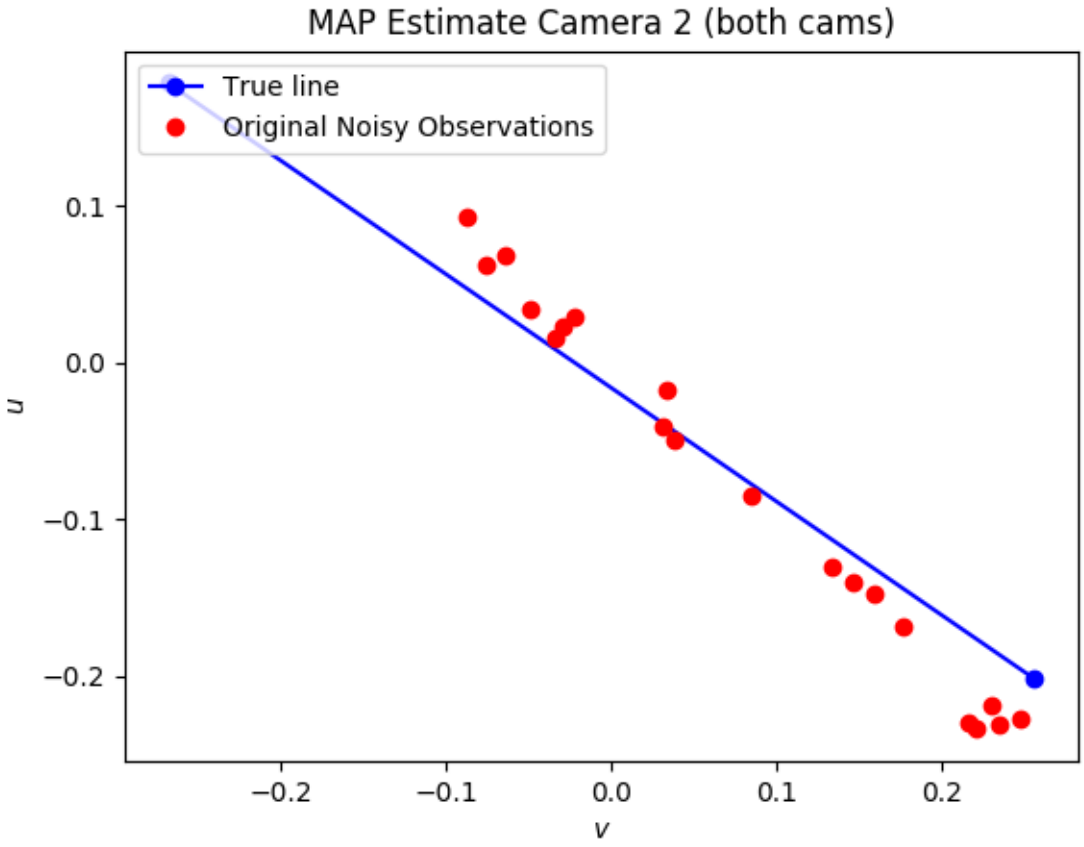
   Solution:


Accepted proposals for p_i (both cams)
p_i (x)
p_i (y)
p_i (z)
p_i
6
5
4
3
2
1
0
−1
0
10000
20000
30000
40000
50000
Samples



The plots that i made in task 2 do not look qualitatively the same as the ones I made in this task because, in this task, we can see that all axis coordinates for the points converge much faster (in less samples) and much better around a certain point than the axes in task 2. I believe that this is because we now have more information (data) on the line we're trying to find, and we have it from a different perspective. This means that the predictions will be more accurate and convergence around a point will be faster.


MAP Estimate Camera 1 (both cams)
True line
Original Noisy Observations
0.3
0.2
0.1
0.0
−0.1
−0.2
u
−0.20
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
v



In the case of the MAP estimate, the first plot (for camera 1) looks qualitatively the same as before, but not the second plot (for camera 2). In this case, the points in this camera's projection fit much better as we are now accounting for camera 2's data. In both plots, in general the noise points fit the MAP created pretty well.