ISTA 421 / INFO 521 – Final Project OPTION A: Metropolis-Hastings MCMC

Due: Monday, December 13, 5pm

Total 20% of final grade

Arun Ravishankar

Graduate

Introduction

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

The instructions start with 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 a noisy 2D image using Metropolis Hastings

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

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

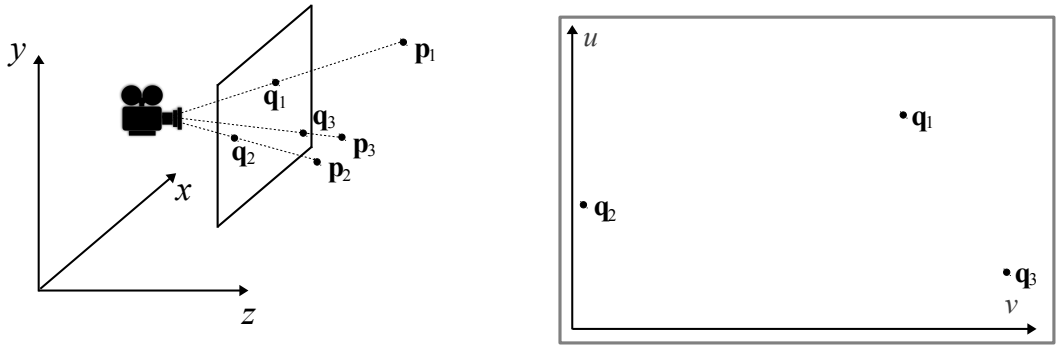
where $\widetilde{u}$, $\widetilde{v}$, and $\widetilde{w}$ are defined by

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

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

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 become 2D image pixel points by Equations 1 and 2.

[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) Graphical 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.

Throughout this exercise we will assume that we have a camera 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 $(0, 0, 0)$, pointing down the positive z-axis, with unit focal length. What follows is the problem setup.

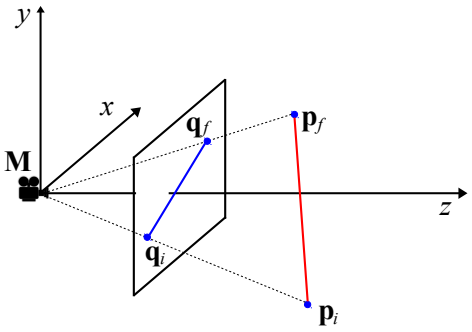
Let $\boldsymbol{\ell} = (\mathbf{p}_i, \mathbf{p}_f)$ be a 3D line segment with end-points $\mathbf{p}_i, \mathbf{p}_f \in \mathbb{R}^3$. (The subscripts on the end-points stand for "initial" and "final", respectively.) Now, consider the following generative process. We "take a picture" of $\boldsymbol{\ell}$ 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 $\mathbf{q}_i$ 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 be half way between $\mathbf{q}_i$ to $\mathbf{q}_f$, so for this point $t_m = 0.5$).

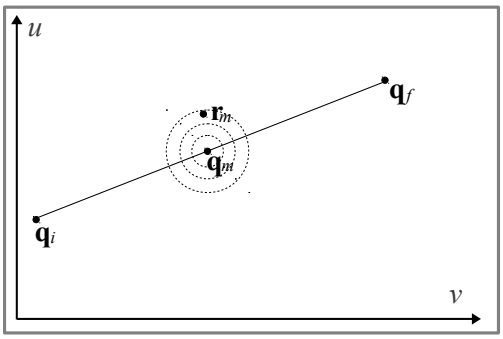
Since the imaging and rendering processes are inherently noisy, we add Gaussian noise to the $\mathbf{q}_s$, which gives us $\mathbf{r}_s = \mathbf{q}_s + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$, with $\boldsymbol{\Sigma}$ a 2×2 covariance 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:

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



(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 2D line $\mathbf{l}_{2D} = (\mathbf{q}_i, \mathbf{q}_f)$ (in blue).



(b) Image $\boldsymbol{\ell}_{2D} = (\mathbf{q}_i, \mathbf{q}_f)$ of the projected 3D line; $\mathbf{q}_m$ 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

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 $t_1, ..., t_S$
   - 2D points $\mathbf{r}_1, ..., \mathbf{r}_S$
   - values for $\boldsymbol{\Sigma}$, $\boldsymbol{\mu}_{\boldsymbol{\ell}}$, and $\boldsymbol{\Sigma}_{\boldsymbol{\ell}}$,

   uses the Metroplis-Hastings (MH) 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

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

   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 MH 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 requires that you describe how to run your script for the following 4 tasks (tasks 2 through 5). You will complete the writing of these instructions as you complete the following tasks.

Solution

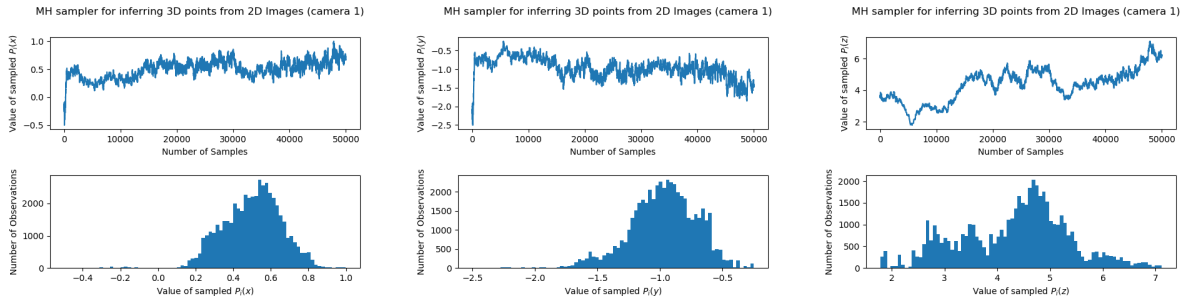
The Metropolis-Hastings sampler was implemented to sample from the posterior of $\mathbf{p}_i$ and $\mathbf{p}_f$ given the inputs $t_1, ..., t_s$ which are the ratios that dictate where the points of the line between q_i and q_f are sampled from the formula $q_s = q_i + (q_f - q_i)t_s$. The sampled points which have a Gaussian spread $\mathbf{r}_1, ..., \mathbf{r}_s$ and the values for the parameters for the variances and means $\mathbf{\Sigma}$, μ_ℓ and $\mathbf{\Sigma}_\ell$ are also given. In order to avoid numerical errors that happen when dealing with very small numbers, we use the log of the acceptance ratio in the MH sampler.

2. **[6 points]** Use your script to draw samples from the posterior distribution of the 3D line segment end-points using the inputs and data contained in the files `inputs.csv` and `points_2d_camera_1.csv`, with $\mathbf{\Sigma_\ell} = 6 \cdot \mathbf{I}_{3\times3}$ (where here $\mathbf{I}_{3\times3}$ is the 3×3 identity matrix), $\boldsymbol{\mu_\ell} = [0, 0, 4]^\top$, and $\mathbf{\Sigma} = (0.05)^2 \cdot \mathbf{I}_{2\times2}$ (where here $\mathbf{I}_{2\times2}$ 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. 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, 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

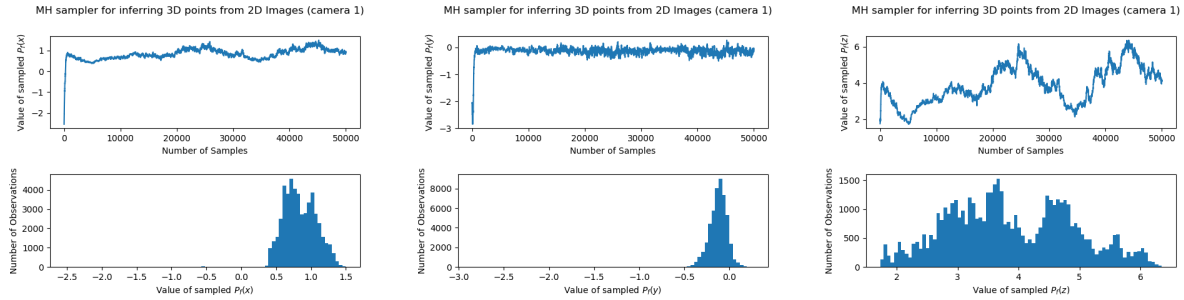


(a) This figure shows the x coordinate of the sampled initial point $\mathbf{p}_i$ as a function of the number of samples in the MH sampler

(b) This figure shows the y coordinate of the sampled initial point $\mathbf{p}_i$ as a function of the number of samples in the MH sampler

(c) This figure shows the z coordinate of the sampled initial point $\mathbf{p}_i$ as a function of the number of samples in the MH sampler

Figure 3: These figures show how the Metropolis-Hastings sampler works as a function of the sample number read from camera 1. We see that the x and y coordinates are normally distributed but the z coordinate is not and has a large variance with multiple local peaks. This is probably because of the direction the camera is facing.



(a) This figure shows the x coordinate of the sampled initial point $\mathbf{p}_f$ as a function of the number of samples in the MH sampler

(b) This figure shows the y coordinate of the sampled initial point $\mathbf{p}_f$ as a function of the number of samples in the MH sampler

(c) This figure shows the z coordinate of the sampled initial point $\mathbf{p}_f$ as a function of the number of samples in the MH sampler

Figure 4: These figures show how the Metropolis-Hastings sampler works as a function of the sample number read from camera 1. We see that the x and y coordinates are well distributed but the z coordinates is not. This is probably because of the direction the camera is facing.

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 MH 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 The MAP estimate of the line end-points using the data from the first camera are

   $$(\mathbf{p}_i, \mathbf{p}_f) = ([0.2822, -0.6644, 2.6599], [0.6016, -0.0347, 2.6536])$$

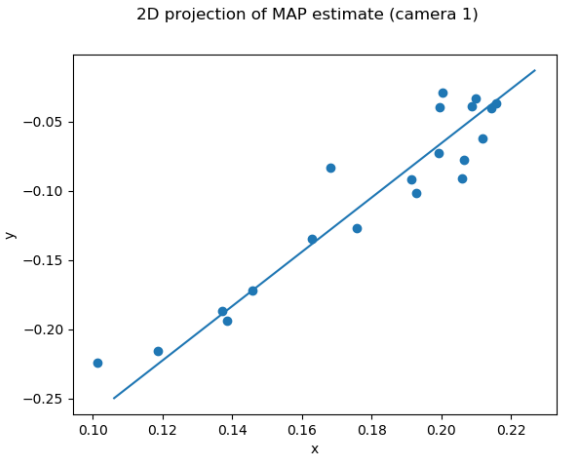
   .

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!



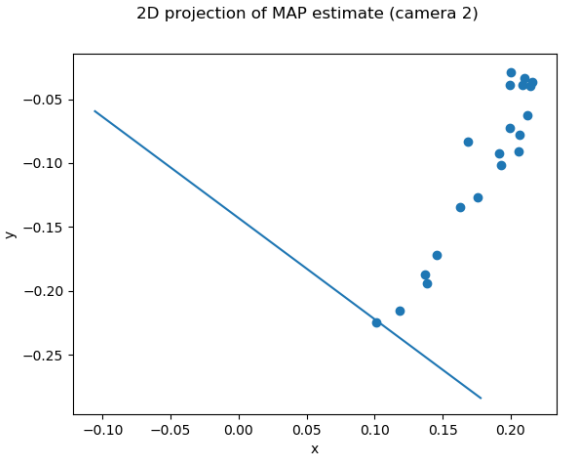
(a) This figure shows the 2D projection on the first camera of the given points and the MAP estimate of the line inferred from the MH sampling algorithm

Solution

The MAP estimate of the line end-points using the second camera are

$$(\mathbf{p}_i, \mathbf{p}_f) = ([-0.9973, -1.7013, 6.0668], [-1.5392, -0.3888, 4.3097])$$

.



(a) This figure shows the 2D projection on the second camera of the given points and the MAP estimate of the line inferred from the MH sampling algorithm. We see that the fit is very bad. This is probably because the line is placed in the normal direction to the face of the camera. This way, it is not possible to infer the coordinates of the line.

5. **[10 points]** Now, perform MH 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 MH

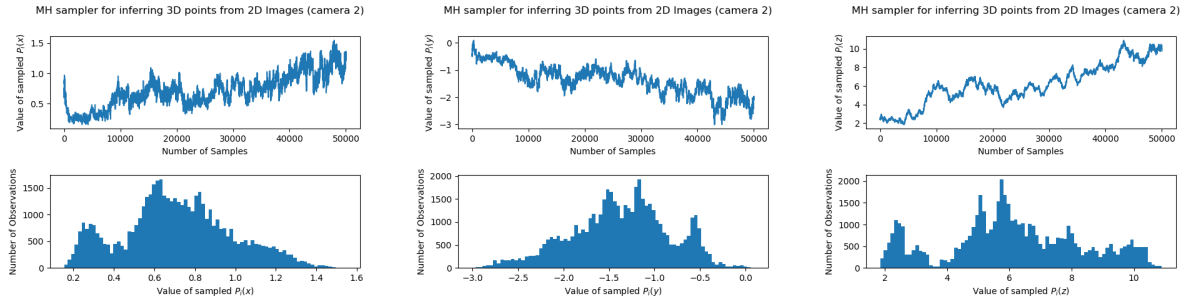
stays the same! Again, 50,000 iterations in MH should be sufficient to converge and to see longer-term the trend. Report the following:

- 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? What might explain any 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?

Using the likelihood from both the cameras, the new MAP estimate of the two end points of the line is

$$(\mathbf{p}_i, \mathbf{p}_f) = ([0.1992, -0.7977, 2.6905], [0.6987, -0.0189, 3.0519])$$

.

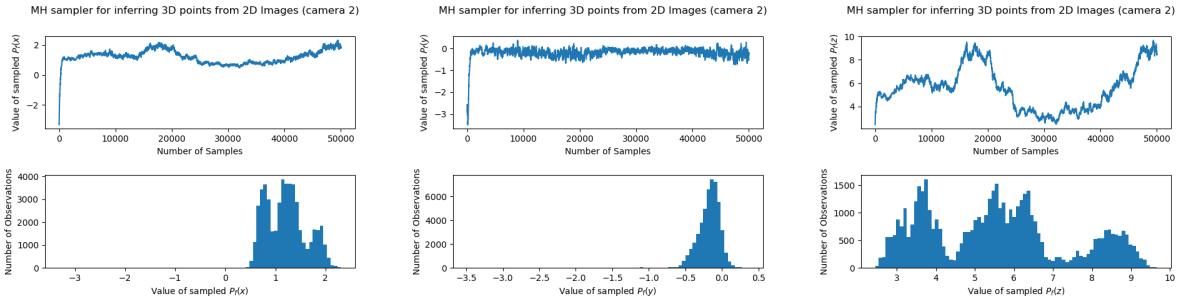


(a) This figure shows the x coordinate of the sampled initial point $\mathbf{p}_i$ as a function of the number of samples in the MH sampler using the likelihood from both cameras.

(b) This figure shows the y coordinate of the sampled initial point $\mathbf{p}_i$ as a function of the number of samples in the MH sampler using the likelihood from both cameras.

(c) This figure shows the z coordinate of the sampled initial point $\mathbf{p}_i$ as a function of the number of samples in the MH sampler using the likelihood from both cameras.

Figure 7: These figures show how the Metropolis-Hastings sampler works as a function of the sample number read from the data from both camera 1 and 2. We see that now, the distributions are more spread out in all the coordinates for the point $\mathbf{p}_i$.

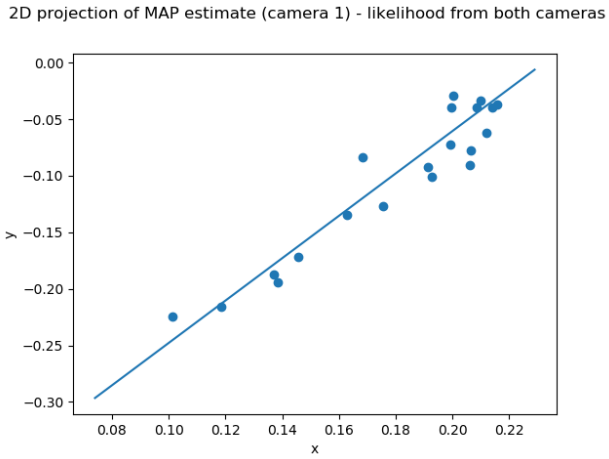


(a) This figure shows the x coordinate of the sampled initial point $\mathbf{p}_i$ as a function of the number of samples in the MH sampler using the likelihood from both cameras.

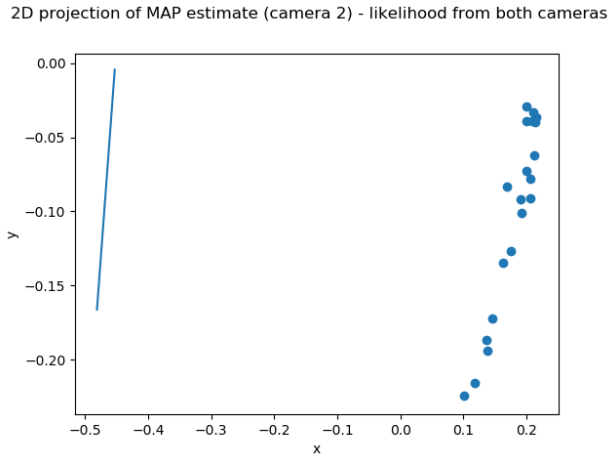
(b) This figure shows the y coordinate of the sampled initial point $\mathbf{p}_i$ as a function of the number of samples in the MH sampler using the likelihood from both cameras.

(c) This figure shows the z coordinate of the sampled initial point $\mathbf{p}_i$ as a function of the number of samples in the MH sampler using the likelihood from both cameras.

Figure 8: These figures show how the Metropolis-Hastings sampler works as a function of the sample number read from the data from both camera 1 and 2. We see that now for the point $\mathbf{p}_f$, the distributions for the x and y coordinates are still normally distributed but the z coordinate is spread out but has more significant peaks that arises from the data from both the cameras together.



(a) This figure shows the 2D projection on the first camera of the given points and the MAP estimate of the line inferred from the MH sampling algorithm also using the likelihood from the second camera.



(a) This figure shows the 2D projection on the second camera of the given points and the MAP estimate of the line inferred from the MH sampling algorithm also using the likelihood from the first and second camera. We see that the fit is still bad even though the slope of the fit seems to align with the data and has improved only slightly. It is unclear as to why this is the case.