

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

Due: Monday, December 13, 5pm

Total 20% of final grade

Ti-Tai Wang

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 you 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

$$\mathbf{q} = \begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{\tilde{w}} \begin{bmatrix} \tilde{u} \\ \tilde{v} \end{bmatrix} \quad (1)$$

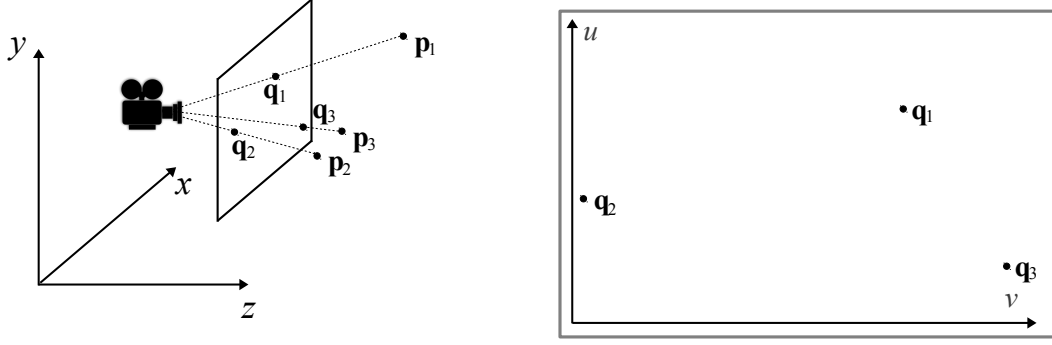
where \tilde{u} , \tilde{v} , and \tilde{w} are defined by

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \mathbf{M} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} = \mathbf{M} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{bmatrix} \quad (2)$$

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.

¹For 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 $\ell = (\mathbf{p}_i, \mathbf{p}_f)$ 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 ℓ using camera \mathbf{M} , producing $\ell_{2D} = (\mathbf{q}_i, \mathbf{q}_f)$ (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 ℓ_{2D} . This is accomplished by taking the ratios t_1, \dots, 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 $0 \leq t_s \leq 1$, $s = 1, \dots, S$. (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 $\mathbf{r}_s = \mathbf{q}_s + \epsilon$, where $\epsilon \sim \mathcal{N}(\mathbf{0}, \Sigma)$, with Σ a 2×2 covariance matrix (since this noise is in the 2D image plane). Naturally, this implies that $\mathbf{r}_s | \mathbf{p}_i, \mathbf{p}_f, t_s, \Sigma \sim \mathcal{N}(\mathbf{q}_s, \Sigma)$. 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 ℓ (this is a *different* Gaussian than the image Gaussian noise ϵ). These prior parameters are subscripted by the bold- ℓ 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:

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

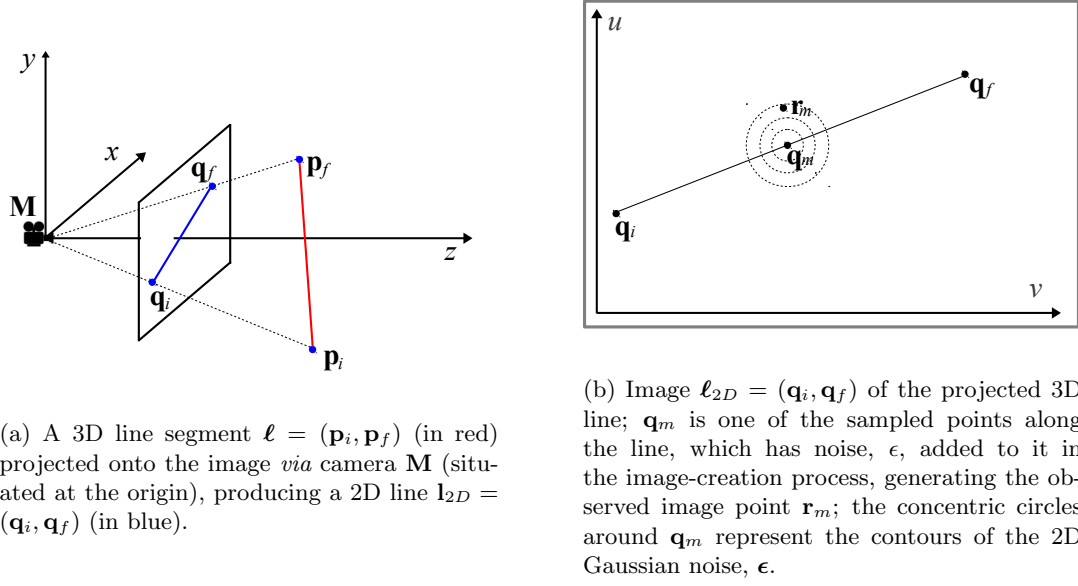


Figure 2: The noisy generative process projecting end-points along the line to the image

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, \dots, t_S
- 2D points $\mathbf{r}_1, \dots, \mathbf{r}_S$
- values for Σ , μ_ℓ , and Σ_ℓ ,

uses the Metropolis-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

$$p(\mathbf{p}_i, \mathbf{p}_f | \mathbf{r}, \mathbf{t}, \Sigma, \mu_\ell, \Sigma_\ell) \propto p(\mathbf{r} | \mathbf{p}_i, \mathbf{p}_f, \mathbf{t}, \Sigma) p(\mathbf{p}_i, \mathbf{p}_f | \mu_\ell, \Sigma_\ell),$$

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

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.

There are two python scripts under the folder called code. No hyper-parameter is needed for running these scripts.

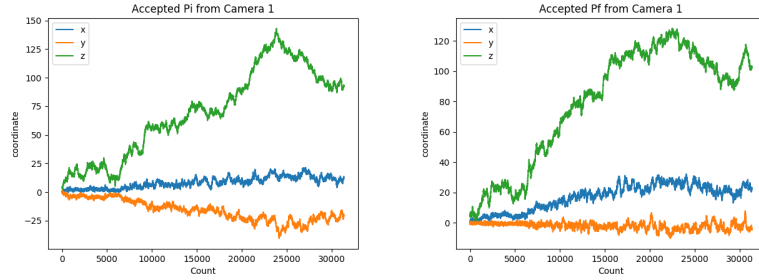
MH_camera1.py would show the plots from task 1 to 4, including the camera 1 training on camera 2 perspective. MH_both.py use

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 $\Sigma_{\ell} = 6 \cdot \mathbf{I}_{3 \times 3}$ (where here $\mathbf{I}_{3 \times 3}$ is the 3×3 identity matrix), $\mu_{\ell} = [0, 0, 4]^T$, and $\Sigma = (0.05)^2 \cdot \mathbf{I}_{2 \times 2}$ (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. 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) The coordinates of the accepted Pi. (b) The coordinates of the accepted Pf.

Figure 3: Two plots both pointed out that Z axis is much more unstable than the other two dimensions, which is fairly reasonable, given that the camera 1 is set on z-axis and points down to the positive z-axis. Therefore, the only information my model had was that Pi and Pf on x-axis and y-axis projection, but lost the z coordinates (one dimension), because camera 1 transited 3d to 2d.

It was expected that x and y coordinates were closer to each other, since the likelihood I had (r points provided by camera 1) was helping me correct these two dimensions every time I sampled Pi and Pf. Also, I think that if the camera 1 is the only perspective my model has, it will never find the right locations of

P_i and P_f , as there could be infinite possibilities for it. Thus, in order to determine the real P_i and P_f , z coordinate information is needed, which camera 1 was not able to provide.

3. [2 points] Using the samples you generated, find the maximum a posterior (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.

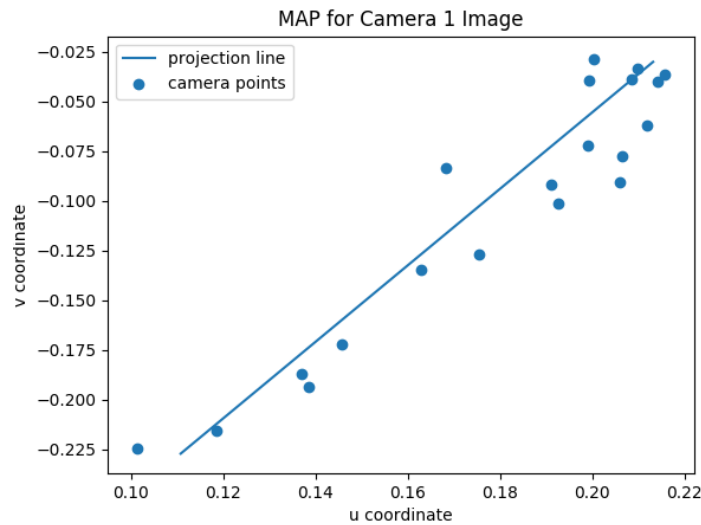


Figure 4: The projection line from camera 1 fits perfectly well with the given \mathbf{r} points. The line was chosen by the MAP among all accepted \mathbf{P}_i , \mathbf{P}_f samplings.

MAP: 83.4710498555834

The acceptance rate : 0.627

\mathbf{p}_i [13.75080193 -28.19282769 124.13856477]

\mathbf{p}_f [26.58615807 -3.76105075 124.68538517]

\mathbf{q}_i [0.11076978 -0.22710773]

\mathbf{q}_f [0.21322594 -0.03016433]

Figure 4 proves that MH (Metropolis-Hastings) model is working perfectly with the information given from camera 1. The points are supposed to be generated by the line with noises of Gaussian distribution.

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 $\mathbf{r}'_1, \dots, \mathbf{r}'_S$, 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.

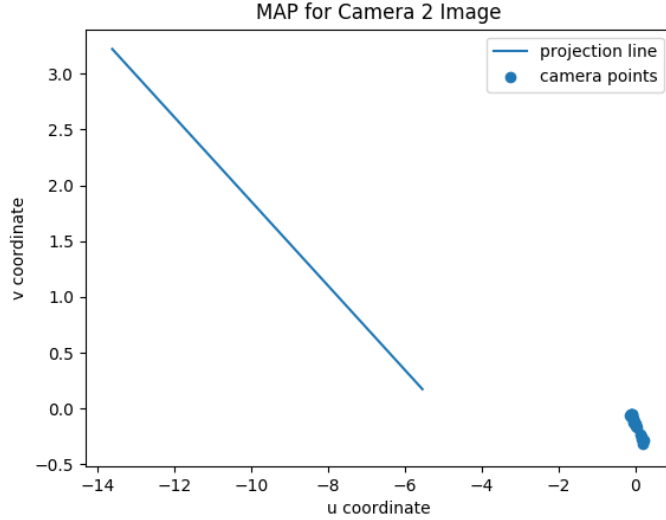


Figure 5: The projection line from the perspective of camera 2 fits poorly to \mathbf{r} data points, since the model was trained by camera 1 projection.

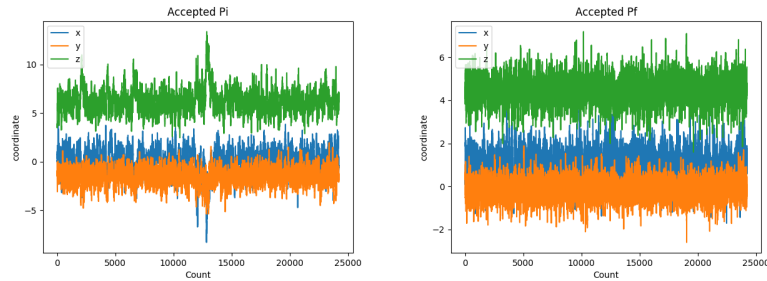
qi $[-13.61458821 \ 3.22174218]$ and qf $[-5.54454317 \ 0.17423437]$

Figure 5 further confirms that only using camera 1 can't help us locate the real coordinate of \mathbf{P}_i and \mathbf{P}_f . As we can see from camera 2 perspective, the line is way off from the data points. And we know that camera 2 is set on x -axis, which indicates that it shows the projection of y and z plane. But from figure 3, we already knew that the model doesn't have the right z coordinate of \mathbf{P}_i and \mathbf{P}_f , so it pretty much explains why I have figure 5 like this.

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:

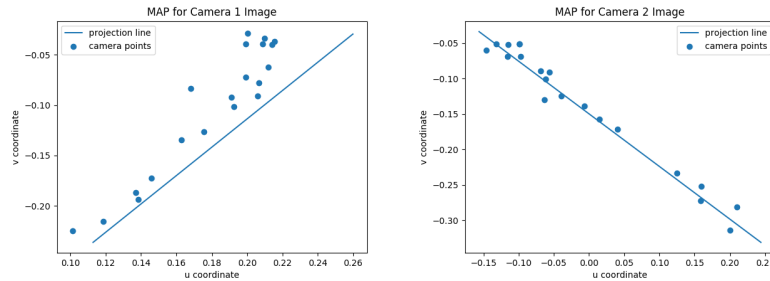
- Find and report the MAP estimate of the line using the same setup as in task 2 using both the observations in `points_2d_camera_1.csv` from the first camera and the points `points_2d_camera_2.csv` viewed by the second camera.
- Make the same plots that you did in task 2. Do they qualitatively look 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?

Solution.



(a) The accepted Pi from both cameras. (b) The accepted Pf from both cameras.

Figure 6: Comparing to figure 3, we can see that three coordinates are all stable now.



(a) The projection line and data points from camera 1 perspective. (b) The projection line and data points from camera 2 perspective.

Figure 7: Graph (b) now shows better fit than we had in task 4, as we consider two cameras perspective for training.

MAP: 160.2944952672417

The acceptance rate : 0.48442

Pi [0.68322628 -1.42920144 6.05427681]

Pf [1.14291834 -0.12982082 4.39758304]


```
qi [ 0.11285019 -0.23606477] from camera 1
qf [ 0.25989693 -0.02952095] from camera 1
qi [ 0.24422795 -0.33108093] from camera 2
qf [-0.15618465 -0.03365778] from camera 2
```

As figure 6 shown, I don't get the same plots as in task 2. I think that the main reason is in task 2, camera 1 only provides x and y coordinates, but z diverged. And now we can have all three dimensions information given by both cameras. Thus, x, y, and z are all stable now.

Figure 7(b) shows a much better fit than figure 5, since now we have included camera 2 into model training, the true location of Pi and Pf can be approached. Noted that camera 1 giving a different plot from figure 4, because the effect from camera 2 can affect both y and z coordinates.