Homework 1

Release: 10/06/2023 Due: Sun. 10/23/2023, 11:59 PM

- You are **allowed** to consult any external resources but you must cite them. You are also **allowed** to discuss with each other but you need to acknowledge them. However, your submission must be your own work; specifically, you must **not** share your code or proof.
- Your submission should be a single PDF file, containing proof, code, and results. We recommend using Jupyter Notebook and export as PDF. We have provided a solution template as **HW1.ipynb**. This notebook contains setup code and some convenient functions. You do not have to use them, but using them can probably save you some time.
- A display is required to visualize with Open3D. Since this homework does not require much computational resource, It is recommended that you finish the homework on a local machine.
- This homework is worth 30/100 of your final grade.

Part I: Rotation [9pt]

Problem 1 (conversion of rotation representations). Let $p := (1+i)/\sqrt{2}$ and $q := (1+j)/\sqrt{2}$ denote the unit-norm quaternions. Recall that the rotation $M(p)$ is a 90-degree rotation about the X axis, while $M(q)$ is a 90-degree rotation about the Y axis. In the notes, we composed the two rotations $M(p)$ and $M(q)$. Here, we instead investigate the rotation that lies halfway between $M(p)$ and $M(q)$.
The quaternion that lies halfway between p and q is simply

$$\frac{p+q}{2} = \frac{1}{\sqrt{2}} + \frac{i}{2\sqrt{2}} + \frac{j}{2\sqrt{2}}$$

Questions:

1. Calculate the norm $|(p+q)/2|$ of that quaternion, and note that it is not 1. Find a quaternion r that is a scalar multiple of $(p+q)/2$ and that has unit norm, $|r| = 1$, and calculate the rotation matrix $M(r)$. Around what axis does $M(r)$ rotate, and through what angle (say, to the nearest tenth of a degree)? [1pt]
2. What are the exponential coordinates of p and q? [1pt]
3. (skew-symmetric representation of rotation) In this problem, we use $[\omega] \in \mathbb{R}^{3\times 3}$ to represent a skew-symmetric matrix constructed from $\omega \in \mathbb{R}^3$ as instructed in class:
   (a) Build the skew-symmetric matrix $[\omega_p]$ of p and $[\omega_q]$ of q, and derive their rotation matrices.[1pt]
   (b) Using what you have above to verify that the following $\exp([\omega_1]+[\omega_2]) = \exp([\omega_1])\exp([\omega_2])$ relationship does *not* hold for exponential map in general Therefore, composing rotations in skew-symmetric representation should not be done in the above way. [1pt]
   (c) Given two point clouds $X, Y \in \mathbb{R}^{3\times n}$ sampled from the surfaces of two shapes (two teapots, provided in **HW1_P1.npz**), we aim to estimate the rigid transformation to best align them. For simplicity, assume that we have found the correspondence between the two point clouds, so that the j-th column of X and Y are matched points. We also assume that the two point

clouds are already aligned by translation. Then, the point cloud alignment problem can be formulated as the following Stiefel manifold optimization problem:

$$\begin{aligned} \underset{R}{\text{minimize}} \quad & \|RX - Y\|_F^2 \\ \text{subject to} \quad & R^T R = I \\ & \det(R) = 1 \end{aligned}$$

We can solve it using our knowledge of the exponential map. Note that, given a rotation matrix R_1, rotation matrices in its local neighborhood R_2 can be parameterized as $R_2 = R_1 \exp([\Delta\omega]) \approx R_1(I + [\Delta\omega])$ for $\Delta\omega \approx 0$.

i. Use the above knowledge to build an optimization algorithm by filling in the following routine:

Step 1: Initiate $R := I$

Step 2: *Describe your routine to find a* $\Delta\omega$ [1pt]
Hint: Convert the objective to a linear least square problem and solve by your solver in HW0:

$$\begin{aligned} \underset{\Delta\omega}{\text{minimize}} \quad & \|A\Delta\omega - B\|_F^2 \\ \text{subject to} \quad & \|\Delta\omega\|^2 \leq \epsilon \end{aligned}$$

Step 3: Update R by $R := R\exp([\Delta\omega])$

Step 4: Go to Step 2

ii. [Programming Assignment] Use your algorithm to solve the point cloud data alignment problem with our provided data. [2pt]

Note: For the point cloud alignment problem under rigid transformation, there exists a closed form solution to be covered in our next homework. While the closed form solution is more efficient, this iterative solver has better flexibility (e.g, modified derivation may work with different objective and/or constraint set), and is often used in solving non-rigid alignment problems.

4. Double-covering of quaternions

(a) What are the exponential coordinates of $p' = -p$ and $q' = -q$? What do you observe by comparing the exponential coordinates of $(p, -p)$ and $(q, -q)$? Does this relation hold for any quaternion pair $(r, -r)$? If it does, write down the statement and prove it. [1pt]

(b) Note that the above is called *double-covering* of quaternions. Based on this property of quaternions, answer the following question. When designing a neural network to regress a quaternion output, can you use the L2 distance between the ground truth quaternion and the predicted quaternion? Why and why not? [1pt]

Part II: Differential Geometry[14pt]

Problem 2. In this problem, you'll do a bit of calculus to see how the operators we talk about in class work on a simple manifold. Consider the map $f : \mathbb{R}^2 \to \mathbb{R}^3$ defined by

$$f(u, v) = \begin{bmatrix} a\cos u \sin v \\ b \sin u \sin v \\ c \cos v \end{bmatrix}, \text{where } -\pi < u < \pi,\ 0 < v < \pi$$

.

Warm-up: Verify that for all (u_0, v_0) be a point in $-\pi < u < \pi, 0 \le v \le \pi$, $f(u,v)$ lies on an ellipsoid with semi-axes length a, b, and c.

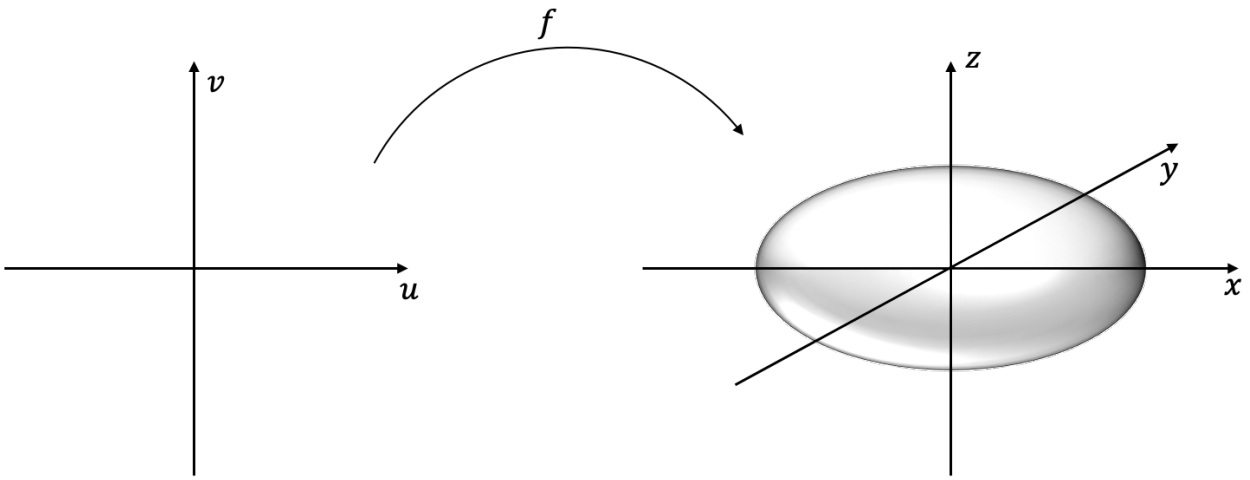

Figure 1: Illustration of P2.2

Let $a = 1, b = 1, c = \frac{1}{2}$. Let $\mathbf{p} = (u, v)$ be a point in the domain of f, and let $\gamma : (-1, 1) \to \mathbb{R}^2$ be a curve with $\gamma(0) = \mathbf{p}$ and $\gamma'(t) = \mathbf{v}$.

1. Illustrate $\mathbf{p}$, $\mathbf{v}$, γ, $f \circ \gamma$, and $(f \circ \gamma)'(0)$ by drawing them on Figure 1. (You can choose any $\mathbf{p}, \mathbf{v}$ you want. You just need to illustrate the concept.) [1pt]

2. [Programming Assignment] Let $\mathbf{p} = (\frac{\pi}{4}, \frac{\pi}{6})$ and $\mathbf{v} = (1, 0)$. Draw the curve of $(f \circ \gamma)(t)$ on the surface of the ellipsoid. [1pt]

3. (differntial map) The differential of a function f at a point $\mathbf{p}$ is a linear map. Equivalent to how we define the differential in class, we can also define by the gradient of the curve w.r.t. the curve parameter: $Df_{\mathbf{p}}(\mathbf{v}) = (f \circ \gamma)'(t)|_{t=0}$.

   (a) What is $Df_{\mathbf{p}}$? Express it as a matrix. [1pt]

   (b) Describe the geometric meaning of $Df_{\mathbf{p}}$.[1pt]

   (c) [Programming Assignment] Draw $Df_{\mathbf{p}}(\mathbf{v})$ on the ellipsoid when $\mathbf{p} = (\frac{\pi}{4}, \frac{\pi}{6})$ and $\mathbf{v} = (1, 0)$.[1pt]

   (d) What is the normal vector of the tangent plane at $\mathbf{p}$? Evaluate at $\mathbf{p} = (\frac{\pi}{4}, \frac{\pi}{6})$.[1pt]

   (e) [Programming Assignment] Give an orthonormal bases of the tangent space at $f(\mathbf{p})$ when $\mathbf{p} = (\frac{\pi}{4}, \frac{\pi}{6})$. Draw the orthonormal bases on the ellipsoid.[1pt]

4. (normal) Given $\mathbf{p} = (\frac{\pi}{4}, \frac{\pi}{6})$ and $\mathbf{v} = (1, 0)$. For simplicity, let $g_{\mathbf{v}}(t) = (f \circ \gamma)(t)$ denote the curve which passes through $f(\mathbf{p})$ at $t = 0$.

   (a) What is the arc length $s(t)$ as the point moves from $g_{\mathbf{v}}(0)$ to $g_{\mathbf{v}}(t)$?[1pt]

   (b) Give the arc-length parameterization $h_{\mathbf{v}}(s)$ of the curve $f \circ \gamma$. [1pt]

   (c) What is the normal vector of the curve at a point $h_{\mathbf{v}}(s)$? *Hint:* Use $h_{\mathbf{v}}(s)$ to derive the normal.[1pt]

Note: Note that $h_{\mathbf{v}}(0)$ corresponds to the point $\mathbf{p}$. Compare the curve normal you get in 4(c) with surface normal in 3(d) at $s = 0$, and you can see they are different.

5. (curvature) In 3(d), you have computed the normal at $\mathbf{p}$. Denote this normal as as $N_{\mathbf{p}}$.

   (a) Compute the differential of the normal $DN_{\mathbf{p}}$, and express it as a matrix. Hint 1: the form is quite complicated. Hint 2: you can use `wolframalpha.com` to compute complicated derivatives that you do not want to compute by hand; they support Latex input.[1pt]

   (b) Find the eigenvectors of the shape operator at $\mathbf{p}$. Hint: you can show the shape operator is diagonal (what does it tell you about the eigenvectors?).[1pt]

   (c) [Programming Assignment] Draw the two principal curvature directions in the tangent plane of the ellipsoid at $\mathbf{p} = (\frac{\pi}{4}, \frac{\pi}{6})$.[1pt]

   (d) Briefly describe the direction of the principal curvatures. Hint: you can describe them with 2 English words, or describe them mathematically.[1pt]

Part III: Mesh Processing [7pt]

Problem 3 Define the Taubin matrix $M_{\mathbf{p}}$ for point p as

$$M_{\mathbf{p}} = \frac{1}{2\pi} \int_{-\pi}^{\pi} \kappa_{\mathbf{p}}(t_\theta)\, t_\theta\, t_\theta^T \, \mathrm{d}\theta$$

Answer questions 1 and 2.
Hint 1: Suppose that $\mathbf{T}_1$ and $\mathbf{T}_2$ are principal curvature directions, then $t_\theta = \cos\theta \mathbf{T}_1 + \sin\theta \mathbf{T}_2$.
Hint 2: $\kappa_{\mathbf{p}}(t_\theta) = \kappa_{\mathbf{p}}^1 \cos^2(\theta) + \kappa_{\mathbf{p}}^2 \sin^2(\theta)$, where $\kappa_{\mathbf{p}}^1$ and $\kappa_{\mathbf{p}}^2$ are principal curvatures.

1. Prove that the surface normal at $\mathbf{p}$ is an eigenvector of $M_{\mathbf{p}}$. What is the corresponding eigenvalue?[1pt]

2. Prove that the other two eigenvectors are the principal curvature directions. (The corresponding eigenvalues are $\frac{3}{8}\kappa_{\mathbf{p}}^1 + \frac{1}{8}\kappa_{\mathbf{p}}^2$ and $\frac{1}{8}\kappa_{\mathbf{p}}^1 + \frac{3}{8}\kappa_{\mathbf{p}}^2$. You do not have to prove this part).[2pt]

3. [Programming Assignment:] Use Rusinkiewicz's method described in lecture 4, compute the principal curvatures for each face in **icosphere.obj** and **sievert.obj**. You may use a library (e.g. **trimesh / Open3D**) to load the mesh and compute normals. However, you should not use any function that computes curvatures (in fact, the functions in trimesh do not compute the curvatures correctly, so you also should not check your answer with them).[2pt]

   Hint: You can use the following Open3D utility to compute normals:

   ```
   pcd = open3d.geometry.PointCloud()
   pcd.points = open3d.utility.Vector3dVector(mesh.vertices)
   pcd.estimate_normals(open3d.geometry.KDTreeSearchParamKNN(knn=50))
   ```

   The mesh in **icosphere.obj** has points at the same coordinates but different indices, so using `trimesh.mean_vertex_normals` to compute normals will not result in correct normals. Therefore, it is highly recommended to use open3d to compute normals.

4. [Programming Assignment:] In fact, the surface (**Sievert's surface**) in **sievert.obj** is locally isometric to a region on a sphere! Verify this fact by computing the Gaussian curvature for both the sphere and Sievert's surface. You can understand the distribution of Gaussian curvatures by creating histograms of Gaussian curvatures for each model. Next, similarly compare the mean curvatures of the models.[2pt]

Part IV: Point Cloud Processing [4+2 pt]

Problem 4 (point cloud sampling). In this exercise, we will practice common point cloud processing routines. All problems are programming assignments.

1. Given mesh **saddle.obj**, sample $100K$ points uniformly on the surface. You may use a library (such as trimesh or open3d) to do it. Note internally, they eventually use the triangle sampling method described in HW0.[1pt]

2. Use iterative farthest point sampling method to sample $4K$ points from the $100K$ uniform samples, and visualize the sampled points. You need to implement this algorithm. You may only use computation libraries such as Numpy and Scipy. *Hint:* To accelerate computation, use Numpy matrix operations whenever possible (they can be 100x times faster than pure python). You may also want to use a progress bar library since the computation may take a minute.[1pt]

3. At each point of the $4K$ points, estimate the normal vector by Principal Component Analysis using 50 nearest neighbors from the $100K$ uniform points (You can use `sklearn.decomposition.PCA`). Since making the direction of normals consistent is a non-trivial task, for this assignment, you can orient the normals so that they roughly points in the Y direction. [2pt]

4. [extra credit] Principal curvature estimation for point cloud: modify the Rusinkiewicz's method in problem 3.3 and apply it to this point cloud. Describe your algorithm and plot the Gaussian curvature for the shape. [2pt extra]

Part V: Course Feedback [1 pt]

1. How many hours did you spend on this homework?

2. How many hours did you spend on the course each week?

3. Do you have any course related feedback?