CPSC 542g Assignment 3 *PDEs and more: 15%* **Due Before the exam**

(1) Camera registration is a standard problem in computer vision and graphics. We'll consider a simple case: given an orthogonal projection of a set of known 3D points into a 2D image, find the matching camera position and orientation. Say the set of 3D points is $\{x_i\}_{i=1}^n$ and their measured 2d projections are $\{p_i\}_{i=1}^n$. We can model the camera using a 3D position $(\alpha_1, \alpha_2, \alpha_3)$ and three Euler angles $\alpha_4, \alpha_5, \alpha_6$, which projects a 3D point x to a 2D point p as:

$$\begin{pmatrix} p_1 \\ p_2 \\ \cdot \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha_6 & -\sin\alpha_6 \\ 0 & \sin\alpha_6 & \cos\alpha_6 \end{pmatrix} \begin{pmatrix} \cos\alpha_5 & 0 & \sin\alpha_5 \\ 0 & 1 & 0 \\ -\sin\alpha_5 & 0 & \cos\alpha_5 \end{pmatrix} \begin{pmatrix} \cos\alpha_4 & -\sin\alpha_4 & 0 \\ \sin\alpha_4 & \cos\alpha_4 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 - \alpha_1 \\ x_2 - \alpha_2 \\ x_3 - \alpha_3 \end{pmatrix}$$

Here the '$\cdot$' indicates we ignore the last coordinate. For a given set of camera parameters stored in a 6D vector α, the projection of all n 3D points can be represented as the $2n$ vector $f(\alpha)$. Explain in your write-up how to find α to match up to a given set of projected positions, and implement code that does it (MATLAB or Python is fine). Your solution should be reasonably robust even if small errors are made in measuring the 2D positions.

(2) For the Gaussian kernel function $e^{-\|x-x_i\|^2}$ derive a similar approximation to the one we used for the gravity potential in Barnes-Hut: how does the error involved in replacing a cluster with a single point behave?

(3) Approximate the smallest eigenvalue of the Laplacian with Dirichlet boundary conditions on the unit square $[0,1] \times [0,1]$, i.e. try to numerically find the smallest magnitude λ satisfying:

$$\begin{aligned} \nabla \cdot \nabla u(x,y) &= \lambda u(x,y) \quad \text{in } (0,1) \times (0,1) \\ u(x,y) &= 0 \quad x \in \{0,1\} \text{ or } y \in \{0,1\} \end{aligned}$$

Discretize this on a regular grid, deriving a matrix eigenproblem $Au = \lambda u$, which you can solve with inverse iteration; solve the sequence of sparse systems using Successive Over-Relaxation (SOR). Using a compiled language to do this should be fairly easy and much faster to execute than MATLAB. Try refining the grid to get close to the exact answer of $\lambda = -2\pi^2$ (with eigenfunction $u = \sin(\pi x)\sin(\pi y)$).

(4) Repeat question 3, but instead on an L-shaped domain, the unit square without the top right corner $[1/2, 1] \times [1/2, 1]$. The eigenvalue and eigenfunction are somewhat more complicated in this case.