CV1 : project #2 (total 10 points)
Restoration and Inpainting
Due October 26th (Thur), 2023, 11:59pm

1 Background

This project is based on Section 3.2.1, Section 3.2.2, and Section 3.2.3 of the textbook. You shall read the corresponding parts and understand the underlying logistics before writing your code.

1.1 Python Library

Please install the latest cv2, PIL, numpy, scipy, matplotlib, tqdm, torch (including torch-vision), and the cython (if you want) packages. You are also welcome to utilize any libraries of your choice, **but please report them in your report (for autograder)! Again, report any customized library in the report (do not go too crazy as this will add a significant burden to TAs).**

1.2 What to hand in?

Please submit both a formal report and the accompanying code. For the report, kindly provide a PDF version. You may opt to compose a new report or complete the designated sections within this document, as can be started by simply loading the tex file to Overleaf. Your score will be based on the quality of **your results**, **the analysis** (diagnostics of issues and comparisons) of these results in your report, and your **code implementation**.

Notice

1. There is no immediate feedback autograder to help with the debugging this time. The autograder will be run after the end of the homework!
2. Do not modify the function names in the given code, unless explicitly specified in this document.

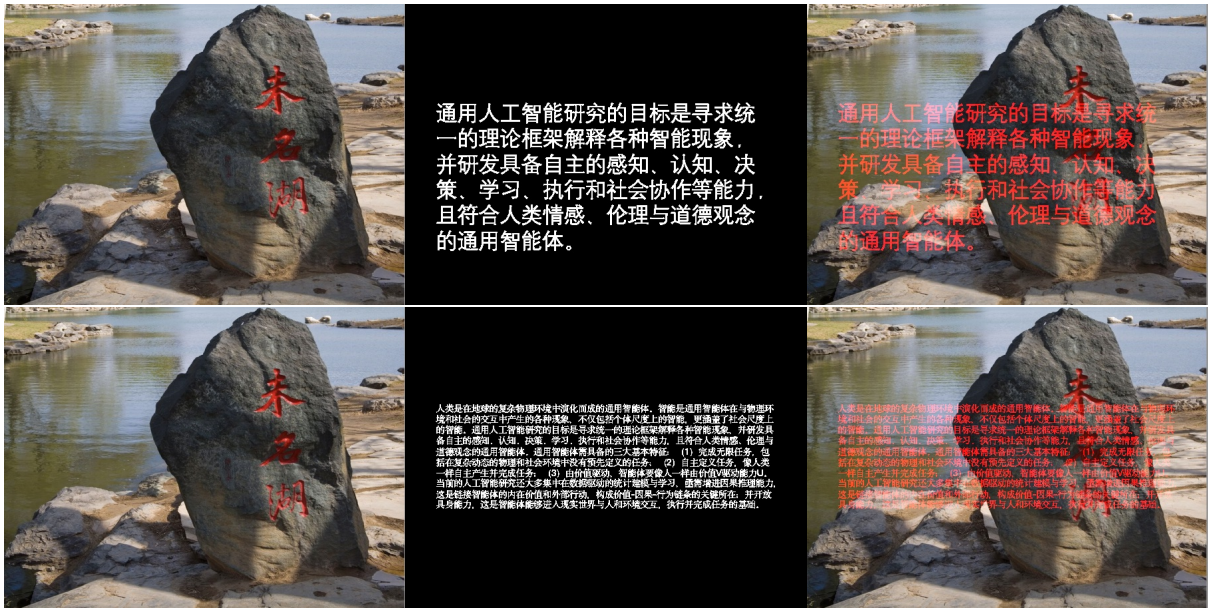
1.3 Help

Make a diligent effort to independently address any encountered issues, and in cases where challenges exceed your capabilities, do not hesitate to seek assistance! Collaboration with your peers is permitted, but it is crucial that you refrain from directly examining or copying one another's code. Please be aware that you'll fail the course if our **code similarity checker**, which has found some prohibited behaviors for Project 1, detects these violations.

For details, please refer to https://yzhu.io/s/teaching/plagiarism/

2 Objective

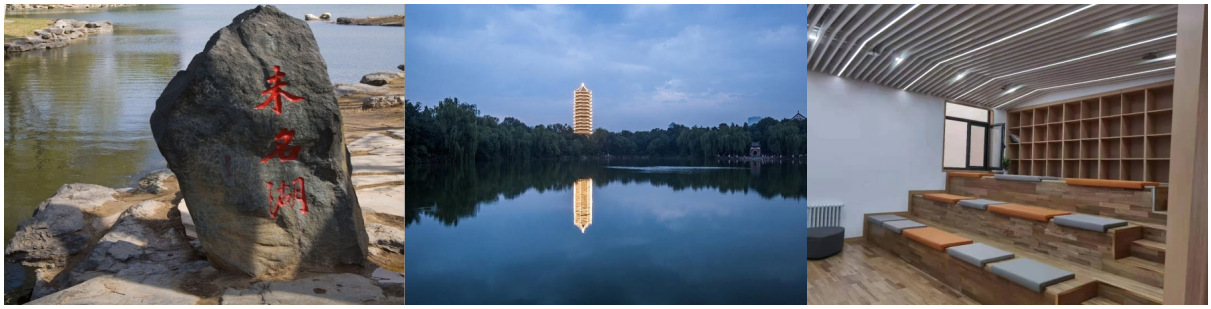
This project serves as a preliminary exercise so that you can get familiar with Gibbs/MRF models, the fundamental principles of the Gibbs sampler, and the application of Partial Differential Equations (PDE) in the context of image restoration and inpainting. Three kinds of images are featured in the illustration presented below, all enclosed within the compressed file provided. The original image comprises distinct color bands, specifically Red, Green, and Blue. At the same time, the distorted counterpart is the result of superimposing a mask image onto the **Red band** of the original image. It is worth noting that two image sets are provided, one featuring a small font size, and the other, a big font size.



(a) Original Image (b) Mask Image (c) Distorted Image

In this experimental setting, pretend that you are provided solely with the distorted image denoted as **I** in subfigure (c) and the corresponding mask image **M** featured in subfigure (b). The primary objective of this experiment is to reconstruct the original image, represented as **O**, by filling in the masked pixels exclusively within the Red-band. It is essential to underscore that no restorative action is required for the Green and Blue bands. Given that information within the masked pixels has been irreversibly compromised, our task is to approximate the original image as **X**, which serves as an estimable substitute for **O**. To evaluate the efficacy of this restoration process, a per-pixel error assessment must be undertaken, explicitly concerning all the pixels concealed by the mask, comparing the reconstructed **X** with the ground truth image **O**.

As demonstrated below, to make the project more interesting, we offer three original images, and their corresponding distorted version. Note that all of the images are of the same size, and they are masked at the same place. **You should conduct experiments on all of them.**



(a) stone (b) sce (c) room

3 Method 1: Gibbs Sampler

3.1 Overview

Let Λ be the white pixels in the mask image **M** (distorted in I), and $\partial\Lambda$ the boundary condition (i.e., the undistorted pixels), which will stay unchanged. We compute

$$X_\Lambda | X_{\partial\Lambda} \sim p(X_\Lambda | X_{\partial\Lambda})$$

by sampling from a Gibbs or MRF model of the following form

$$p(X_\Lambda | X_{\partial\Lambda}) = \frac{1}{Z} \exp\{-\beta \sum_{(x,y)\in\Lambda} E(\nabla_x X(x,y)) + E(\nabla_y X(x,y))\},$$

where E() is a potential energy. We need to try two choices of functions:

- L_1 norm: $E(\nabla_x X(x,y)) = |\nabla_x X(x,y)|$
- L_2 norm: $E(\nabla_x X(x,y)) = (\nabla_x X(x,y))^2$.

From this Gibbs model, we can derive the local characteristics, given

$$X_s \sim p(X_s | X_{\partial s}), \forall s \in \Lambda.$$

By drawing from this 1D probability (using the inverse CDF method mentioned in Project 1), we can iteratively compute the values of the distorted pixels. Visiting all distorted pixels once is called 1-sweep (in whatever order, it does not matter). You need to experiment with an annealing scheme by slowly increasing the parameter β from 0.1 to 1 (or more).

3.2 Instructions

1. Please read through the main file and understand the whole logic.
2. Please implement the "conv()" function, which calculates the x direction and y direction gradients. Please be careful that the function's return should hold the same shape as the input, and the boundary condition selected is **periodic boundary condition**.
3. Please implement the "gibbs_sampler()" function designed to execute Gibbs sampling for an individual pixel. (Hints: Consider optimizations to enhance computational efficiency by avoiding superfluous calculations.)
4. Implement the main function and tune the parameters for better effect.

4 Method 2: PDE

4.1 Overview

For L_2-norm energy functions, you can minimize the energy by the heat-diffusion equation.

4.2 Instructions

1. Implement the "pde()" function, which performs pde update for a specific pixel.
2. Implement the main function and tune the parameters for better effect.
3. Please run more sweeps to see the marvelous effect of this method.

5 Speed Up (Optional)

This is not required in this project, and there is no extra bonus for implementing it. If you have time, you can have a try.

1. If your implementation is done by using Torch, it'll be much faster to change to numpy.
2. Use the **cython** library! Please first read the basic tutorial, and understand some basic logistics of **cython**. Please be cautious that the **"pyximport" method** is the **preferred approach** as opposed to the **"setup.py" method**. The latter may generate different files on different operating systems, and it's important to be aware that the autograder utilizes the Linux operating system. Now, you are almost ready to speed up your Gibbs sampling, please read the working with numpy section of the document. You have done a great job, and please speed up your Gibbs sampling process with Cython!

6 Hand-in

1. Plot the per pixel error $\frac{1}{|\Lambda|}\sum(X(x,y)-O(x,y))^2$ over the number of sweeps t in for both methods.
2. Show the restored image sequence corresponding to the number of sweeps.
3. **(Bonus, Optional)** Where are the original images from? Try to find them in PKU, take nice photos of them at a similar angle, report their names, and submit your photos.