Computer Vision
2^{nd} *Assignment Report*

Tea Pula

Overview

In this assignment, the objective is to solve an image inpainting problem, which is part of the family of inverse problems. The available input contains limited information (masked cat image), and the goal is to generate an output with enhanced details (restore the missing information) compared to the original input (sharper and inpainted cat image). To achieve this, our approach involves using the Bayesian Framework for formulating the energy minimization model as an optimization problem. To incorporate additional knowledge, a new positive term is introduced during the regularization step, which guides the optimization process towards a more realistic and desirable outcome. Then the problem is discretized into finite quantities, to make it computationally feasible. The other step is calculating the Gradient Descent, to refine the solution by moving towards the steepest decrease in the energy function, with the objective of minimizing it. The last step is using the Gauss-Seidel method to refine the solution even more. All of these steps work together in synergy in order to get a good result in the end.

Report Questions

1. Finite difference approximation of the objective function E.

Following the application of the Bayesian Framework, specifically emphasizing the Maximum Posteriori function, and subsequently conducting the required computations, we get the energy function as below, where the first component is the data term and the second component is the regularization term with a non-negative λ:

$$E[u_c] = |u_c - g_c|_\Omega^2 + \lambda |\nabla u_c|_{2,1}^2.$$

The problem is discretized from continuum into finite quantities in order to be able to compute them. We use the approach where we discretize the energy from the start and then compute the exact solution. Between forward difference, backward difference, and central difference, I have used forward difference since it is commonly used in numerical methods for discretizing partial derivatives because it is relatively straightforward to implement and computationally efficient. Central differences provide a more accurate approximation of derivatives and stability compared to forward or backward differences because information on both sides of a point is used, but it is computationally expensive. The total variation regularization term is given by:

$$\sum_{i=1}^{N-1} \sum_{j=1}^{M} |\nabla u_c[i,j]|_{2,1}^2$$

The 2D discretization of the energy function using forward differences and the provided formula for total variation is given by:

$$\begin{aligned} E[u_c] = & \sum_{i=1}^{N} \sum_{j=1}^{M} \Omega[i,j](u[i,j] - g[i,j])^2 \\ & + \lambda \sum_{i=1}^{N-1} \sum_{j=1}^{M} \sqrt{(u[i+1,j] - u[i,j])^2 + (u[i,j+1] - u[i,j])^2 + \delta}. \end{aligned}$$

The whole total variation can be replaced by $\tau^2[i,j]$. In cases where $\tau^2[i,j] = 0$, the gradient cannot be computed because we are dividing by 0, which is undefined, and that is why the small constant δ is added.

2. Calculation of the exact gradient of the discretized E

To find the exact gradient of the objective function E, we need to calculate the partial derivatives of E with respect to each pixel $u[i,j]$.

1. **First Term:**

$$\frac{\partial}{\partial u[i,j]} \left(\Omega[i,j](u[i,j] - g[i,j])^2\right) = 2\Omega[i,j](u[i,j] - g[i,j])$$

2. **Second Term:** We consider the gradient of discretized energy away from boundaries.

$$\frac{\partial \|\nabla u_c\|}{\partial u[i,j]} = \frac{\partial}{\partial u[i,j]} \left(\tau[i,j] + \tau[i-1,j] + \tau[i,j-1]\right)$$

Using the chain rule:

$$= \frac{\partial \tau[i,j]}{\partial u[i,j]} + \frac{\partial \tau[i-1,j]}{\partial u[i,j]} + \frac{\partial \tau[i,j-1]}{\partial u[i,j]}$$

We plug the derivatives of the first and second term into the gradient expression:

$$\nabla_u E[i,j] = 2\Omega[i,j](u[i,j] - g[i,j]) + 2\lambda\tau[i,j] \left(\frac{\partial \tau[i,j]}{\partial u[i,j]} + \frac{\partial \tau[i-1,j]}{\partial u[i,j]} + \frac{\partial \tau[i,j-1]}{\partial u[i,j]}\right)$$

$$\begin{aligned}\nabla_u E[i,j] =& 2\Omega[i,j](u[i,j] - g[i,j]) \\ &+ 2\lambda\tau[i,j] \left(\frac{2u[i,j] - u[i+1,j] - u[i,j+1]}{\tau[i,j]} + \frac{u[i,j] - u[i-1,j]}{\tau[i,j]} + \frac{u[i,j] - u[i,j-1]}{\tau[i,j]}\right)\end{aligned}$$

$$\begin{aligned}\nabla_u E[i,j] =& 2\Omega[i,j](u[i,j] - g[i,j]) \\ &+ 2\lambda \left(\frac{\tau[i,j](2u[i,j] - u[i+1,j] - u[i,j+1] + u[i,j] - u[i-1,j] + u[i,j] - u[i,j-1])}{\tau[i,j]}\right)\end{aligned}$$

$$\nabla_u E[i,j] = 2\Omega[i,j](u[i,j] - g[i,j]) + 2\lambda(4u[i,j] - u[i+1,j] - u[i,j+1] - u[i-1,j] - u[i,j-1])$$

The formula that we have is for internal pixels [i, j] not on the boundaries. When applying the energy function, because of forward-difference now we have to handle all cases of boundaries and corners. The eight special cases are applied to three color channels $C \in \{R, G, B\}$. Let's take a matrix of size 9x9 just for illustration purposes so we get to see each special case clearly:

$$\begin{bmatrix}
[i,j] & [i,j+1] & c & [i,j-1] & [i,j] & [i,j+1] & g & [i,j-1] & [i,j] \\
[i+1,j] & k & l & m & [i+1,j] & o & p & q & [i+1,j] \\
s & t & u & v & w & x & y & z & 0 \\
[i-1,j] & 2 & 3 & 4 & [i-1,j] & & 7 & 8 & [i-1,j] \\
[i,j] & [i,j+1] & 12 & [i,j-1] & [i,j] & [i,j+1] & 16 & [i,j-1] & [i,j] \\
[i+1,j] & 20 & 21 & 22 & [i+1,j] & 24 & 25 & 26 & [i+1,j] \\
28 & 29 & 30 & 31 & 32 & 33 & 34 & 35 & 36 \\
[i-1,j] & 38 & 39 & 40 & [i-1,j] & 42 & 43 & 44 & [i-1,j] \\
[i,j] & [i,j+1] & 48 & [i,j-1] & [i,j] & [i,j+1] & 52 & [i,j-1] & [i,j]
\end{bmatrix}$$

Corners special cases: top-left, top-right, bottom-left, bottom-right

$$\nabla_u E[1,1] = 2\Omega[1,1](u[1,1] - g[1,1]) + 2\lambda(2u[1,1] - u[2,1] - u[1,2])$$
$$\nabla_u E[1,M] = 2\Omega[1,M](u[1,M] - g[1,M]) + 2\lambda(2u[1,M] - u[1,M-1] - u[2,M])$$
$$\nabla_u E[N,1] = 2\Omega[N,1](u[N,1] - g[N,1]) + 2\lambda(2u[N,1] - u[N-1,1] - u[N,2])$$
$$\nabla_u E[N,M] = 2\Omega[N,M](u[N,M] - g[N,M]) + 2\lambda(2u[N,M] - u[N,M-1] - u[N-1,M])$$

Boundaries special cases: top, left, right, bottom

$$\nabla_u E[1,j] = 2\Omega[1,j](u[1,j] - g[1,j]) + 2\lambda(3u[1,j] - u[1,j-1] - u[2,j] - u[1,j+1])$$
$$\nabla_u E[i,1] = 2\Omega[i,1](u[i,1] - g[i,1]) + 2\lambda(3u[i,1] - u[i-1,1] - u[i,2] - u[i+1,1])$$
$$\nabla_u E[N,j] = 2\Omega[N,j](u[N,j] - g[N,j]) + 2\lambda(3u[N,j] - u[N,j-1] - u[N-1,j] - u[N,j+1])$$
$$\nabla_u E[i,M] = 2\Omega[i,M](u[i,M] - g[i,M]) + 2\lambda(3u[i,M] - u[i-1,M] - u[i,M-1] - u[i+1,M])$$

3. Solvers

3.a Gradient Descent

We use gradient descent to find the best possible values for our pixels. The basic Gradient function with step size ϵ is

$$u_{t+1} = u_t - \epsilon \nabla_u E(u_t)$$

The necessary condition that should be satisfied is

$$\nabla_u E(u) = 0$$

which is achieved by iteratively updating our pixel values little by little so we do not overstep, and when it is small enough (converges to 0) then we stop.
To approximate the function, we use Taylor Expansion and approximate the difference in gradients

$$\nabla_u E[u^{t+1}] \simeq \nabla_u E[u^t] + \mathbf{H}(E[u^t])(u^{t+1} - u^t)$$

$$\nabla_u E[u^{t+1}] \simeq \nabla_u E[u^t] + \frac{1}{\epsilon}(u^{t+1} - u^t).$$

In order to find the gradient update rule, we set it to the negative value of the gradient in the opposite direction and solve it for u^{t+1}:

$$\nabla_u E[u^t] + \frac{1}{\epsilon}(u^{t+1} - u^t) = -\nabla_u E[u^{t+1}]$$

$$\nabla_u E[u^t] = -\epsilon \nabla_u E[u^{t+1}] - \frac{1}{\epsilon}(u^{t+1} - u^t)$$
$$\epsilon \nabla_u E[u^t] = -\epsilon \nabla_u E[u^{t+1}] - u^{t+1} + u^t$$
$$u^{t+1} = u^t - \epsilon \nabla_u E[u^t] - \epsilon \nabla_u E[u^{t+1}]$$

3.b Linearization and Gauss-Seidel

We linearize the gradient by writing its Taylor Expansion and then removing components of order higher than one, then we have:

$$\nabla_u E[u^{t+1}] \simeq \nabla_u E[u^t] + \mathbf{H}(E[u^t])(u^{t+1} - u^t) = 0$$

$$\nabla_u E[u^t] \simeq \nabla_u E[u^t] + \mathbf{H}(E[u^t])[u^{t+1}] - \mathbf{H}(E[u^t])[u^t]$$
$$\mathbf{H}(E[u^t])[u^{t+1}] \simeq \mathbf{H}(E[u^t])[u^t] - \nabla_u E[u^t]$$

Where the left component is matrix Au^{t+1} and the right part is matrix b. The next step is to solve

the linear system by using the Gauss-Seidel method. This method is an iterative numerical method for solving linear systems while guaranteeing convergence if the matrix is diagonally dominant $|a[i,i]| \geq \sum_{j \neq 1} |a[i,j]|$ for all i, and it is also easily computable. It repeatedly updates the values of the variables in a system of linear equations. At each iteration, the method considers one equation at a time, utilizing the most recent values for the other variables.
We LU decompose the system $Au = b$ as $(L + U)u = b$ and solve it for u as follows:

$$\begin{aligned} Au &= b \\ (L+U)u &= b \\ Lu^{t+1} + Uu^t &= b \\ Lu^{t+1} &= b - Uu^t \\ u^{t+1} &= L^{-1}(b - Uu^t) \end{aligned}$$

Now, we compute the value of $u^{t+1}[i]$ using the lower triangular matrix L and the upper triangular matrix U.

$$u^{t+1}[i] = \left(\frac{b[i] - \sum_{j=1}^{i-1} A[i,j]u^{t+1}[j] - \sum_{N}^{j=i+1} A[i,j]u^t[j]}{A[i,i]} \right)$$

The process is iterative until the difference between the left-hand side of independent equations is similar to the right side of the independent equations.