# IN3200/IN4200 Spring 2024
# Feedback about oblig 2

May 13, 2024

# Outline

- General feedback about oblig 2 (in the form of questions & discussions)

# About allocate_image

```
void allocate_image (image *u, int m, int n) {
    u->image_data = (float**) malloc(m * sizeof(float*));

    for (int i = 0; i < m; i++) {
        u->image_data[i] = (float*) malloc(n * sizeof(float));
    }

    u->m = m;
    u->n = n;
}
```

Anything to improve?

# About `deallocate_image`

How many calls to the `free` function are needed?

# About iso_diffusion_denoising

Computing interior points and handling boundary points

```
int m = u->m, n = u->n;
for(int i = 0; i < m; i++)
    for(int j = 0; j < n; j++)
       if (i==0) {
         // ...
       } else if (i==m-1) {
         // ...
       } else if (j==0) {
         // ...
       } else if (i==n-1) {
         // ...
       } else {
         // ...
       }
```

Any inefficiency?

# About iso_diffusion_denoising

Computing interior points and handling boundary points

```
int m = u->m, n = u->n;
for(int i = 1; i < m-1; i++)
    for(int j = 1; j < n-1; j++)
        u_bar->image_data[i][j] = u->image_data[i][j]
          + kappa * ( u->image_data[i-1][j]
          + u->image_data[i][j-1] - 4 * u->image_data[i][j]
          + u->image_data[i][j+1] + u->image_data[i+1][j] );

for(int i = 0; i < m; i++){
    u_bar->image_data[i][0] = u->image_data[i][0];
    u_bar->image_data[i][n-1] = u->image_data[i][n-1];
}
for(int j = 0; j < n; j++){
    u_bar->image_data[0][j] = u->image_data[0][j];
    u_bar->image_data[m-1][j] = u->image_data[m-1][j];
}
```

Any inefficiency?

# About iso_diffusion_denoising

```
for(int it = 0; it < iters; it++) {
   // computing interior points and handling boundary points
   // ....

   // before the next iteration
   for (i=0; i<m; i++)
      for (j=0; j<n; j++)
         u->image_data[i][j] = u_bar->image_data[i][j];
}
```

Any inefficiency?

# About `iso_diffusion_denoising`

How to implement pointer swap?

# About `iso_diffusion_denoising`

What happens if the number of iterations is an even number?

(What does `u_bar` in the main program contain when we return from `iso_diffusion_denoising`?)

# About parallelization

How to divide the $m$ rows evenly amoung $P$ processes?
(How to compute `my_m` on each process?)

Remember that each process will call

```
allocate_image (&u, my_m, n);
allocate_image (&u_bar, my_m, n);
```

# About parallelization

How to distribute the initial JPEG picture among $P$ processes?

# About parallelization

Is it efficient to ask process 0 to call `MPI_Send` toward each of the other processes?

# About parallelization

Why and how to use `MPI_Scatterv`?

```
int MPI_Scatterv(const void *sendbuf, const int sendcounts[],
                 const int displs[], MPI_Datatype sendtype, void *recvbuf,
                 int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)
```

# About `iso_diffusion_denoising_parallel`

In each iteration, communication before the computation, or after the computation?

# About iso_diffusion_denoising_parallel

```
// "downward" communication
if (my_rank>0)
   MPI_Send (...)   // send to my_rank-1

if (my_rank<P-1)
   MPI_Recv (...)   // receive from my_rank+1

// "upward" communication
if (my_rank<P-1)
   MPI_Send (...)   // send to my_rank+1

if (my_rank>0)
   MPI_Recv (...)   // receive from my_rank+1
```

Any efficiency concerns?

# About iso_diffusion_denoising_parallel

```
// "downward" communication
if (my_rank>0)
   MPI_Ssend (...)   // send to my_rank-1

if (my_rank<P-1)
   MPI_Recv (...)   // receive from my_rank+1

// "upward" communication
if (my_rank<P-1)
   MPI_Ssend (...)   // send to my_rank+1

if (my_rank>0)
   MPI_Recv (...)   // receive from my_rank+1
```

Will it deadlock?

# About `iso_diffusion_denoising_parallel`

How to use `MPI_Isend` and `MPI_Irecv`?

# About iso_diffusion_denoising_parallel

How to use MPI_Sendrecv?

```
int MPI_Sendrecv(const void *sendbuf, int sendcount, MPI_Datatype sendtype,
                 int dest, int sendtag, void *recvbuf, int recvcount,
                 MPI_Datatype recvtype, int source, int recvtag, MPI_Comm comm,
                 MPI_Status *status)
```