Digital Image Processing: Homework 2 Report

Task1: Low-luminosity Enhancement

I simply add an intensity value to the r,g,b channel for each pixel. Note that the value should be between [0, 255].

```
/*Do Low-luminosity Enhancement on images*/
unsigned char increase_intensity = 20;
if (enhance_degree == 2) {
    increase_intensity = 40;
}
for(int i = 0; i < data.size(); i+=3){
    // double max_channel = max(max(data[i], data[i + 1]), data[i + 2]);
    data[i] = min(255, data[i] + increase_intensity);
    data[i + 1] = min(255, data[i + 1] + increase_intensity);
    data[i + 2] = min(255, data[i + 2] + increase_intensity);
}

string output_filename = "output1_" + to_string(enhance_degree) + ".bmp";
ofstream output(output_filename, ios::out | ios::binary);
if (!output.is_open()) {
    std::cerr << "Error creating the output file" << std::endl;
    return -1;
}
```

degree 1 (increase_intensity = 20) v.s. degree 2 (increase_intensity = 40)

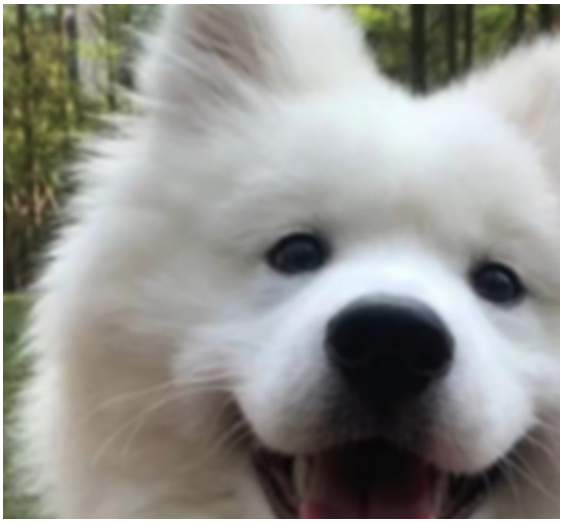

Task2: Sharpness Enhancement

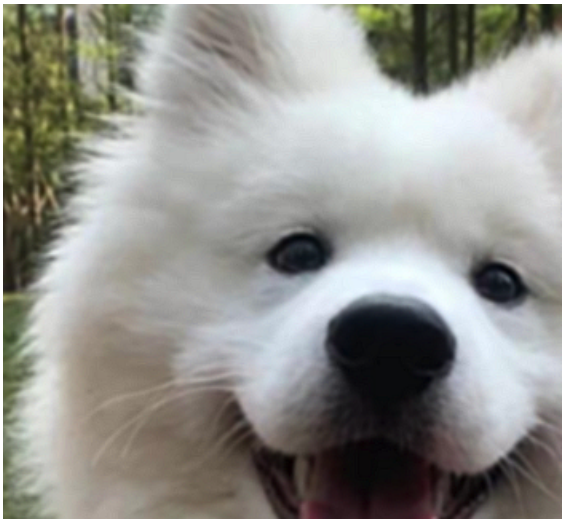
Apply a composite laplacian filter for the raw image. Note that the final value of each pixel should be between [0, 255].

```cpp
int channels = 3; // Assuming RGB image (3 channels)

for (int y = 1; y < (height - 1); y++){
    for (int x = 1; x < (width - 1); x++){
        for(int c = 0; c < channels; c++){
            int sum = 0;
            for(int j = -1; j <= 1; j++){
                for(int i = -1; i <= 1; i++){
                    sum += data[(y + j) * (width * channels) + (x + i) * channels + c] * kernel[j + 1][i + 1];
                }
            }
            if (sum < 0) sum = 0;
            if (sum > 255) sum = 255;
            resultData[y * (width * channels) + x * channels + c] = static_cast<unsigned char>(sum);
        }
    }
}

data = resultData; // Update the data with the sharpened result
```

degree 1 v.s. degree 2 (different composite laplacian kernel)

```cpp
int kernel[3][3] = {
        {0, -1, 0},
        {-1,  5, -1},
        {0, -1, 0}
    };

if (enhance_degree == 2) /* Composite Laplacian kernell 2 (sharper) */
{
    for(int i = 0; i < 3; i++) {
        for(int j = 0; j < 3; j++) {
            kernel[i][j] = -1;
            if ((i == 1) && (j == 1))
            {
                kernel[i][j] = 9;
            }
        }
    }
}
```





Task3: Denoise

Apply Gaussian blur to denoise the image, where I adjust blur radius for more or less blurring.

```cpp
/* Apply Gaussian blur to denoise the image */
int blurRadius = 3; // Adjust the blur radius for more or less blurring
if(enhance_degree == 2){
    blurRadius = 5;
}
for (int y = blurRadius; y < height - blurRadius; y++) {
    for (int x = num_channel * blurRadius; x < (width - blurRadius) * num_channel; x += num_channel) {
        for (int c = 0; c < num_channel; c++) {
            int sum = 0;
            for (int j = -blurRadius; j <= blurRadius; j++) {
                for (int i = -blurRadius; i <= blurRadius; i++) {
                    sum += data[(y + j) * (width * num_channel) + (x + i * num_channel) + c];
                }
            }
            int blurredValue = sum / ((2 * blurRadius + 1) * (2 * blurRadius + 1));
            data[y * (width * num_channel) + x + c] = static_cast<unsigned char>(blurredValue);
        }
    }
}
```

degree 1 (blur radius 3) v.s. degree 2 (blur radius 5)