ENGI 9804 – Image processing & Applications

Instructor: Dr. Stephen Czarnuch

Low-Light Image Enhancement

COMPLETED BY:

HADI MOHAMMADPOUR

Problem Statement

Given an input image which has been **captured in low-light conditions** and **suffers from low visibility**, I can generate an output image with an **improved illumination**.

Besides degrading the visual aesthetics of images, the low quality of the images taken in low-light conditions can affect the performance of different machine vision methods. Therefore, it is necessary that we use image enhancement techniques to solve this issue.

- Formation of a low-light image: $\mathbf{L} = \mathbf{R} \circ \mathbf{T}$ $\mathbf{R} = \mathbf{L}/\mathbf{T}$

My goal is to estimate the desired illumination map(T).

Methodology

- **First step:** Calculating the initial illumination map:

$$\hat{\mathbf{T}}(x) \leftarrow \max_{c \in \{R,G,B\}} \mathbf{L}^c(x),$$

- **Second step:** Calculating the refined illumination map using Algorithm 1 (see next slide)

- **Third step:** Applying Gamma correction on the refined illumination map

Algorithm 1

Algorithm 1 Exact Solver to Problem (8)

Input: The positive coefficient α, and the initial illumination map $\hat{\mathbf{T}} \in \mathbb{R}^{m \times n}$.

Initialization: $\mathbf{T}^{(0)} = \mathbf{0} \in \mathbb{R}^{m \times n}$, $\mathbf{G}^{(0)} = \mathbf{Z}^{(0)} = \mathbf{0} \in \mathbb{R}^{2m \times n}$, $t = 0$, $\mu^{(0)} > 0$, $\rho > 1$.

while *not converged* **do**

  Update $\mathbf{T}^{(t+1)}$ via Eq. (13);

  Update $\mathbf{G}^{(t+1)}$ via Eq. (15));

  Update $\mathbf{Z}^{(t+1)}$ and $\mu^{(t+1)}$ via Eq. (16);

  $t = t + 1$;

end

Output: Optimal solution $\mathbf{T}^* = \mathbf{T}^{(t)}$.

$$\hat{\mathbf{T}}(x) \leftarrow \max_{c \in \{R,G,B\}} \mathbf{L}^c(x), \quad \nabla \mathbf{T} = \mathbf{G}$$

$\mathbf{Z}$ is the Lagrangian multiplier.

$$(13)\quad \mathbf{T}^{t+1} \leftarrow \mathcal{F}^{-1}\left(\frac{\mathcal{F}\left(2\hat{\mathbf{T}} + \mu^{(t)}\mathbf{D}^T(\mathbf{G}^{(t)} - \frac{\mathbf{Z}^{(t)}}{\mu^{(t)}})\right)}{\mathbf{2} + \mu^{(t)} \sum_{d \in \{h,v\}} \overline{\mathcal{F}(\mathbf{D}_d)} \circ \mathcal{F}(\mathbf{D}_d)}\right)$$

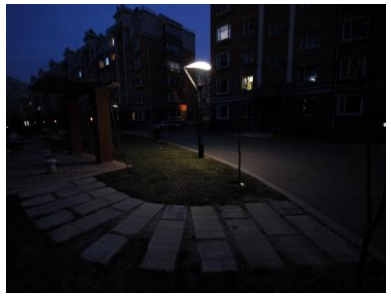
$$(15)\quad \mathbf{G}^{(t+1)} = \mathcal{S}_{\frac{\alpha \mathbf{w}}{\mu^{(t)}}}\left[\nabla \mathbf{T}^{(t+1)} + \frac{\mathbf{Z}^{(t)}}{\mu^{(t)}}\right]$$

$$\mathcal{S}_\varepsilon[x] = \operatorname{sgn}(x) \max(|x| - \varepsilon, 0)$$

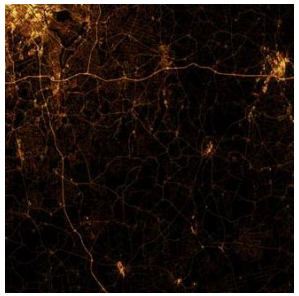
$$(16)\quad \begin{aligned} &\mathbf{Z}^{(t+1)} \leftarrow \mathbf{Z}^{(t)} + \mu^{(t)}(\nabla \mathbf{T}^{(t+1)} - \mathbf{G}^{(t+1)}); \\ &\mu^{(t+1)} \leftarrow \mu^{(t)}\rho, \rho > 1. \end{aligned}$$

Input images







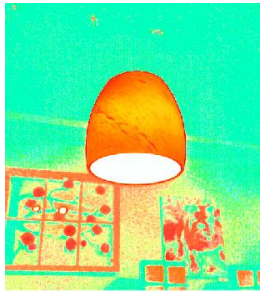
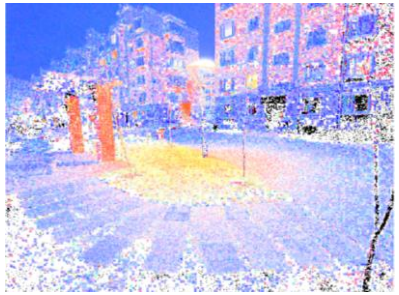
Step 1: Estimating the initial illumination map

```
function Initial_map = calc_init_map(img)

    [M, N, ~] = size(img);
    Initial_map = zeros(M,N);

    for i = 1:M
        for j = 1:N
            % find the channel with highest intensity
            Initial_map(i,j) = max([img(i,j,1), img(i,j,2), img(i,j,3)]);

            if Initial_map(i,j) == 0
                Initial_map(i,j) = 1e-7;
            end
        end
    end

end
```

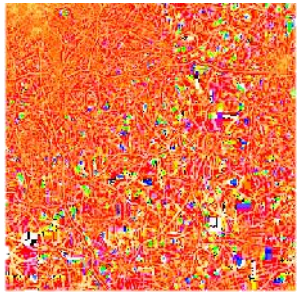
- **Input:** low-light image
- **Functionality:** Finding the channel with highest value in an RGB image
- **Explanation:** This function examines each of the pixels in the image using the for loops, and in case the maximum value was zero, it replaces that with a minimal value (1e-7) to avoid saturation. This is based on the idea that the illumination for each point is at least the maximal value of R, G, and B channels.

Resulted images from step 1







Not the results we are looking for, but a starting point for the project!

Step 2: Refining the illumination map

```
function Refined_map = calc_refined_map(Initial_map, Alpha, Rho, Mu, Num_iter)

    [M, N] = size(Initial_map);
    T = zeros(M,N);

    Z = zeros(2*M,N);
    G = Z;
    W = ones(2*M,N);

    for k = 0:Num_iter
        B = Alpha * W / Mu;

        %% Updating T
        T_numerator = T_num(Initial_map, G, Z, Mu);
        T_denominator = T_denom(M, N, Mu);
        T_F = T_numerator./T_denominator;
        T = ifft2(ifftshift(T_F));

        gradient_T = multiplyd(T);

        %% Updating G using shrinkage operator
        G = sign(gradient_T +  Z / Mu) .* max(abs(gradient_T +  Z / Mu)- B, zeros(size(B)));

        %% Updating Z
        Z = Z + Mu * (gradient_T - G); %Z and mu sub-problem

        %% Updating Mu
        Mu = Mu * Rho;

    end

    Refined_map = T;

end
```

- Inputs:
  - **Initial_map:** This has been calculated in the previous step.
  - **Alpha:** A coefficient used to balance the involved terms in the optimization problem in next slide. (Alpha = 3)
  - **Mu:** A positive penalty scalar for the optimization problem. (Mu = 0.05)
  - **Rho:** A coefficient used to update the value of Mu parameter at each iteration.(Rho = 1.15)
  - **Num_iter:** Number of iterations for the optimization solution to converge (Num_iter = 50)

Note: Alpha, Mu, Rho, and Num_iter has been optimized by testing different values to obtain the best results.

Step 2: Refining the illumination map

- **Functionality:** Refining the initial illumination map to be able to preserve the overall structure

- **Explanation:** To reach a structure-aware illumination map, the reference paper has proposed the following optimization problem:

$$\min_{\mathbf{T}} \|\hat{\mathbf{T}} - \mathbf{T}\|_F^2 + \alpha \|\mathbf{W} \circ \nabla \mathbf{T}\|_1$$

Where T is the desired illumination map, W is the weight matrix, and ∇T is the first order derivative filter. In the objective above, the first term keeps the refined map related to the initial map and the second term preserves the structure-aware smoothness. By setting the weight matrix as a matrix of ones, I turned the optimization problem to a l2 loss variation minimzation problem.

The **Calc_refined_map** function implements an iterative ato solve the optimization problem and refine the illumination map. In this function, an auxiliary variable G is introduced to replace ∇T for making the problem separable and thus easy to solve, and Z is the Lagrangian multiplier. At each step of the for loop, the illumination map is updated based on the updated G, Z, and Mu. Here, I have also utilized some other functions such as **T_num**, **T_denom**, and **multiplyd**.

- **T_num** and **T_denom** : As the formula for updating the illumination map is complicated and lengthy, I have defined two separate functions for calculating the numerator and the denominator for it. The inputs for these functions are just the variables used in the formula.
  - **multiplydtrans**: This function is called inside T_num and is used for creating the D matrix which operates horizontal and vertical gradience and multiplying the transpose of the D matrix with (G – Z/Mu) matrix.

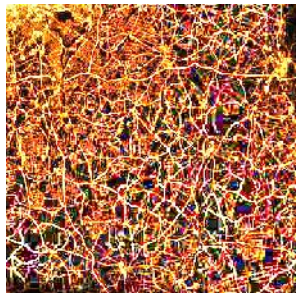
- **multiplyd:** This function is used to calculate the gradient of the illumination map. Given the illumination map (T), this function multiplies that with a matrix D that results in the gradient of T. The D matrix consists of a horizontal gradient and another vertical gradient operators with forward difference. (* The codes used for multiplying with the D matrix were not written by me.)

Resulted images from refined illumination map
(Step 2)







Significant improvement in details of the images in the darker regions, while preserving the structure of the image.

Step 3: Gamma correction

- **Input :** Gamma value (Gamma = 0.8)

- **Functionality:** Improving the illumination map while preserving the overall structure, smoothing the textural details and enhancing the quality of image.

- **Explanation:** For this step, the elements of the refined illumination map were taken to a power of gamma. The main disadvantage of this method is that the relationship of each pixel with its neighbors is not considered, which may cause inconsistency in the results. However, when combined with the refined illumination map obtained in the previous step, it results in a robust tool for enhancing low-light images.

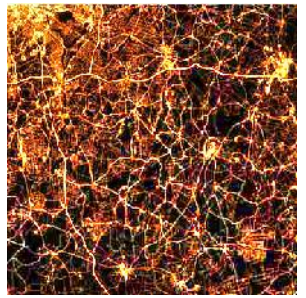
Final illuminated Images

Using refined and Gamma corrected illumination map (Step 3)







Final results! Improving the visibility and illumination of the images, preserving the overall structure, and smoothing the textural details!