Exposure Correction

Team Zion

Our Method

Dual Illumination Estimation for Robust Exposure Correction

- A Retinex based method which uses illumination map estimation without considering a reflectance component.
- Given an input image, we first invert the image.
- On each, we perform illumination estimation to obtain the forward and reverse illuminations, from which we recover the intermediate under- and over-exposure corrected images.
- The two intermediate exposure correction images together with the input image are fused into the desired image.

Method Overview

Original

Create Illumination Map

Underexposed Correction

Fusion

Final Corrected Image

Inverted

Create Illumination Map

Overexposed Correction

Algorithmic Overview

The equation we need to calculate:

$$I = I' \times L$$

I – Original Normalised image

I' - Desired Enhanced Image

L – Single channel illumination map

x - Pixel wise multiplication

The initial illumination map is calculated as follows:

$$L'_p = \max I^c_p, \quad \forall c \in \{r, g, b\}$$

Optimization Problem to be solved:

$$\arg\min_L \sum_p \left(\left(L_p - L'_p\right)^2 + \lambda \left(w_{x,p} \left(\partial_x L\right)^2_p + w_{y,p} \left(\partial_y L\right)^2_p \right) \right)$$

∂_x and ∂_y : Horizontal and vertical spatial derivatives

$w_{x,p}$ and $w_{y,p}$: Spatially varying smoothness weights

Algorithmic Overview

The same needs to be done for inverted image which is obtained by: $I_{inv} = 1 - I$

We obtain I' and I'_{inv}.

Do exposure fusion on original image, enhanced original image and enhanced inverted image.

Then, fuse the images together using the following formula:

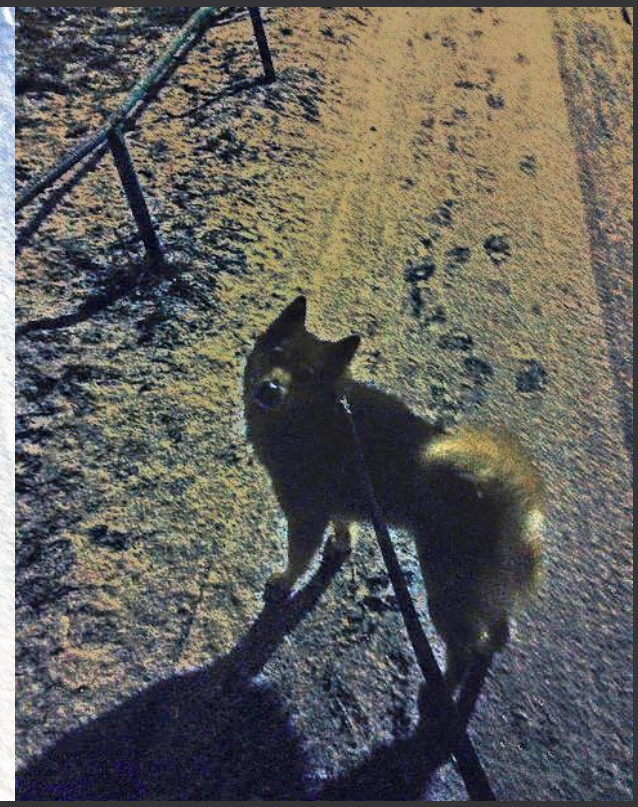
$$\hat{V}_p^k = \begin{cases} 1, & \text{if } k = \arg\max_j V_p^j,\ \forall j \in [1,3] \\ 0, & \text{otherwise} \end{cases}$$

Output

Original

Underexposed Correction

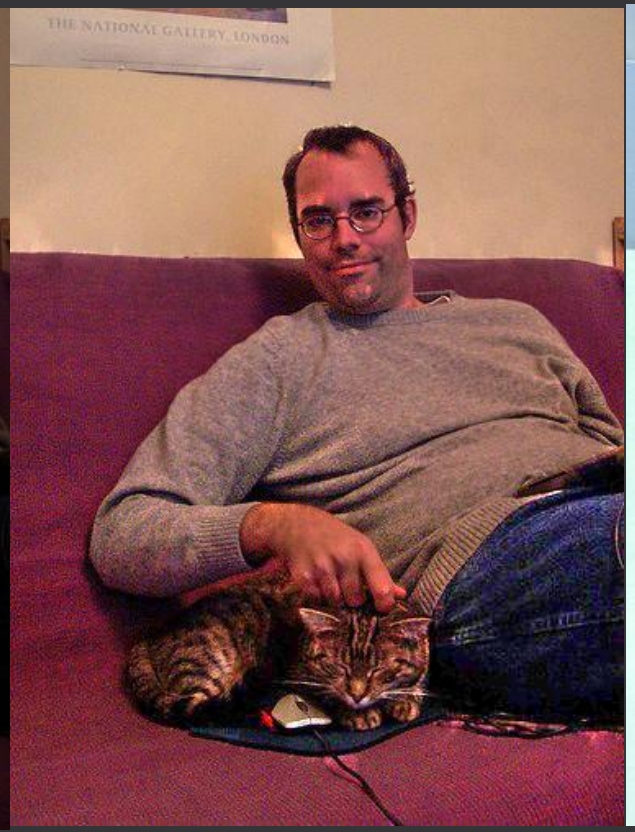
Overexposed Correction



Final Correction

Output

Original



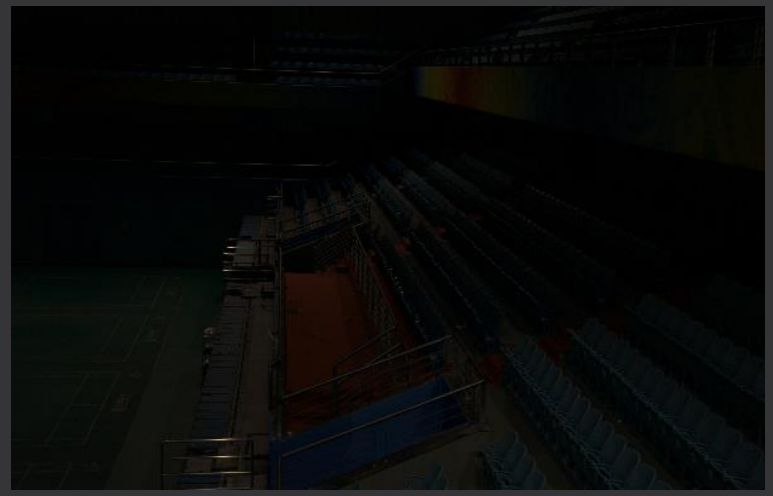
Underexposed Correction

Overexposed Correction

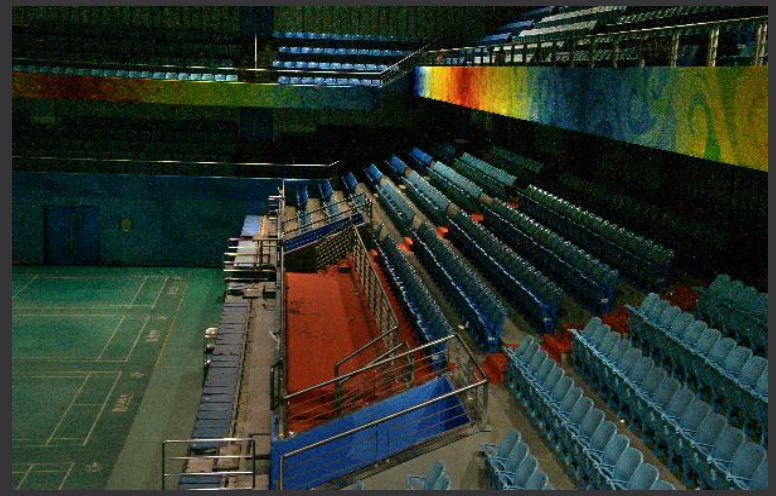
Final Correction

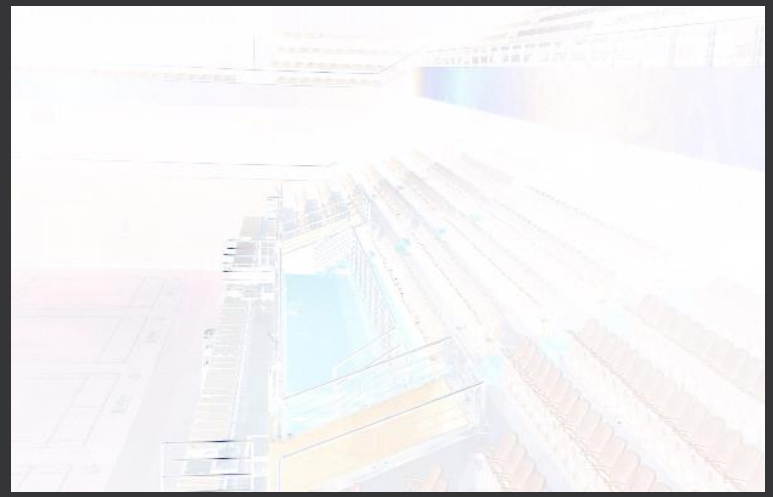
Output

Original

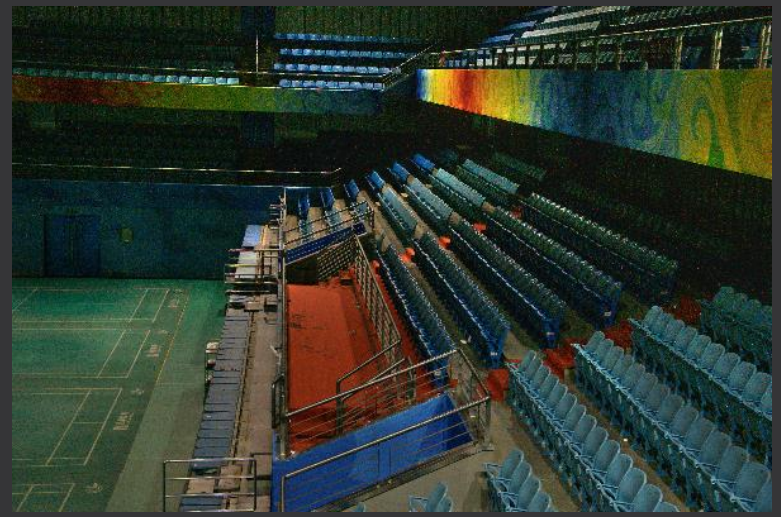


Underexposed Correction





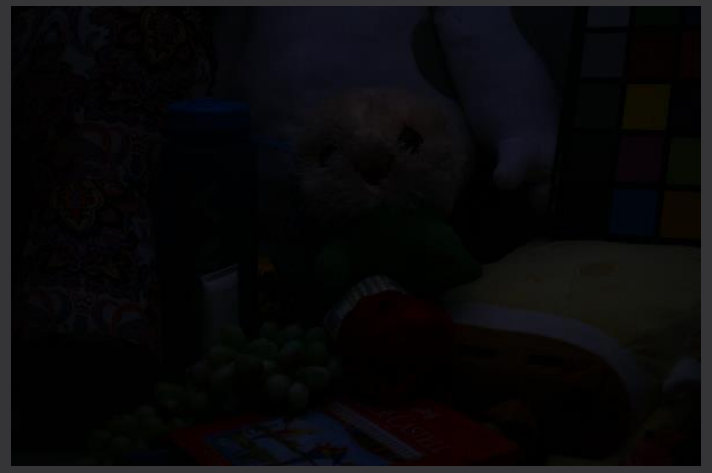
Overexposed Correction



Final Correction

Output

Original



Underexposed Correction

Overexposed Correction

Final Correction

Shot boundary detection

1. Calculate all the differences between frames.
2. Detect the possible frame by creating a window frame and finding the largest difference in that window and that frame should be n time more than average difference between the other frame in that window.
3. Then shift the window by some frames and repeat Step 2.
4. Optimize the possible frames by checking whether the difference of the possible frame is no less than threshold and the difference is twice the average difference between the last n frame and subsequent n frame.

Output

Attempt on videos :

https://www.youtube.com/watch?v=fOIYhHrjDFI

Accomplishments

Met objective: False

Learnt things: True

- Objective 1: Use GPU to make things fast
  - Flawed because image operations weren't the bottleneck. LU decomposition of a large sparse matrix is.
  - Data dependency in LU decomposition solving makes it almost impossible for parallelization.
- Objective 2: Extend for videos
  - Frame-by-frame enhancement would take days for a video that is only a minute long.
  - Shot-by-shot enhancement would take hours and doesn't look good. Illumination map sticks out in the shot.

Team members

Ch N V B Dattatreya (2020201011)

Penukonda Sai Navya Sree (2020201033)

Vaishali Singh (2020201070)

Rahul Valluri (2020202007)

Thank You