Input image at "blue hour" (just after sunset)

A database of time-lapse videos

Hallucinate at night

Data-driven Hallucination of Different Times of Day from a Single Outdoor Photo

- By Patanjali

Team Members :

Aryan Sakaria(20171123)

Prajwal Krishna Maitin(20171086)

Faizan Farooq Khan(20171209)

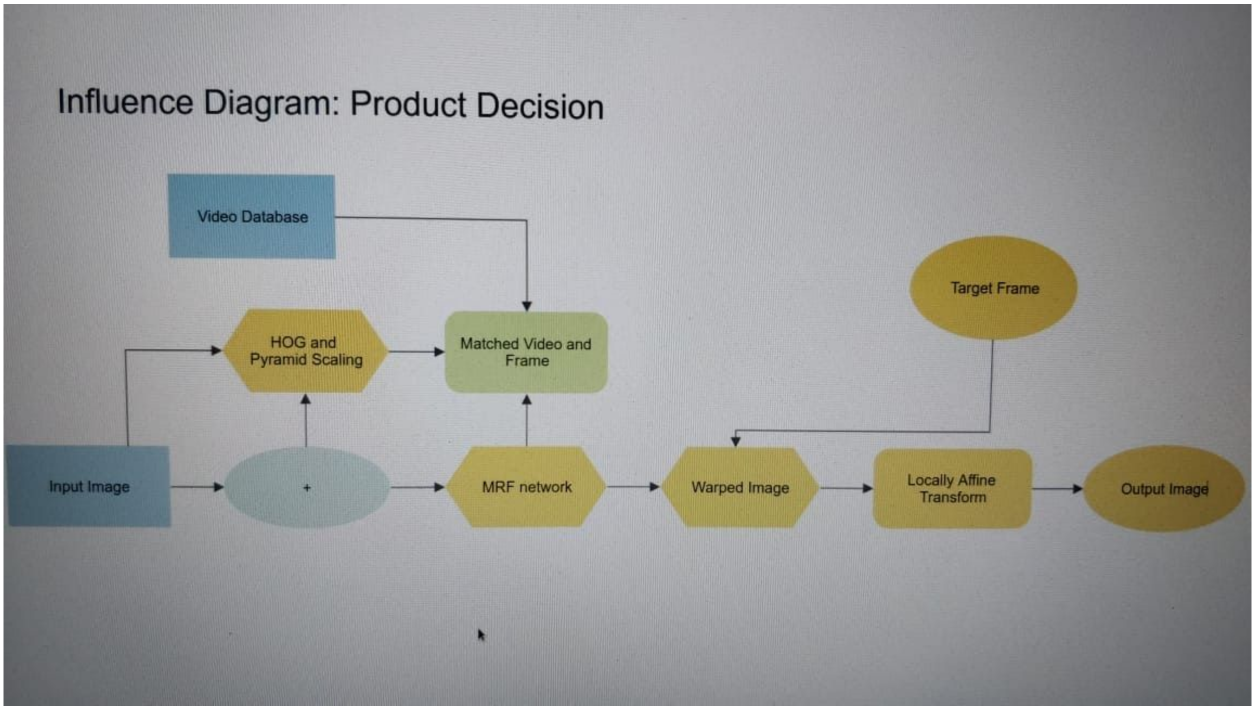
T.A : Pranay Gupta

Problem Statement :

Given a single image as input, we want to automatically create a plausible-looking photo that appears as though it was taken at a different time of the day. This should be done using a fixed database of time lapse videos, of which the input image may not be a part of. Also we want to make the image look realistic, while preserving the colour schema.

We want to work from a single input photograph and allow the user to request a different time of day

Overview :



Database :

Our database contains around 450 time-lapse videos, which covers a wide range of videos with different backgrounds and foregrounds, such as different landscapes and cityscapes, including city skyline, lake, and mountain view. This database is given a priori and independent of the user input, in particular, it does not need to contain a video of the same location as the input image.

Link for the dataset:
https://drive.google.com/open?id=169AH1CZ9jSlBNGalyiKwh9ZuIOxCf0MT

Procedure :

- Global Matching
- Selection Of Frames
- Local Matching
- Locally Affine Color Transfer
- Handling Noisy Inputs

Global Matching :

Given the scene in the input image, we would like to identify the time lapse video from our dataset, which shows a scene similar to the given input.

We sample 10 regularly spaced frames from each video, and then compare the input to all these sampled frames. To assign a score to each time-lapse video,we use the highest similarity score in feature space of its sampled frames. For this we would use a standard scene matching algorithm, using Histogram of Oriented Gradients (HOG) and Spatial Pyramid Matching.

Global Matching : First Step -

First, we calculate a dense HOG descriptor, which was given by Dalal and Triggs(2005). Here, we take 16 * 16 windows with stride of 8 on image and calculate gradient magnitude and direction over each points in four 8*8 subwindows, take these 64 gradients and put them in 9 bins (undirected edges) depending on the angle, (we allow sharing of magnitude between bins) to get a histogram of gradients which we normalize. To handle color images we select strongest gradient in RGB.Then we put the results of each subwindow one after another, which results in a 36 * 1 descriptor per window.

Global Matching : Second Step -

Now since we are doing dense correspondence with overlapping window, we can compress our descriptor without losing much information, PCA would work but will make it computationally inefficient. So we instead approximate this 36 values using 4 normalizing factors and 9 orientation factors thus a total of 13 values representing the 4*9 original vectors. Also, we can add directed edges which will take 18 orientations. Keeping both takes 31 dimensional features. This is described in detail here. Also we combine these for 2*2 neighbourhood to get a 124 dimensional feature.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan.

Global Matching : Third Step -

We use the concept of bag-of-words and use this to quantize these descriptors into 300 visual words found using K-Means on the initial frames from timelapse dataset. This gives a small descriptor(1024 for 256*256) for images. We repeat above for 3 levels of spatial histograms and use histogram intersection, at each level which combined, where weight of each level is 1/(2^L); this gives us a similarity between images.

Selection Of Frames :

Now that we have a video corresponding to an input image, we want to pick the frame, best corresponding to the input image. Once we find this, we pick the output frame by using the time, at which we want to show the hallucinated image.

We use the color histogram and L2 norm to pick the matched frame.

Local Matching :

- We seek to pair each pixel in the input image I with a pixel in the match frame M.
- Methods like SIFT are not used because they are designed to match with a single image and are not designed for videos.
- For this, we formulate the problem as a Markov random field (MRF) using a data term and pairwise term.
- MRF formulation allows us to exploit structure found in time-lapse video
- Scene geometry remains constant over the frames as the camera does not move in time-lapse videos.

Energy Function :

- ### Data Term :

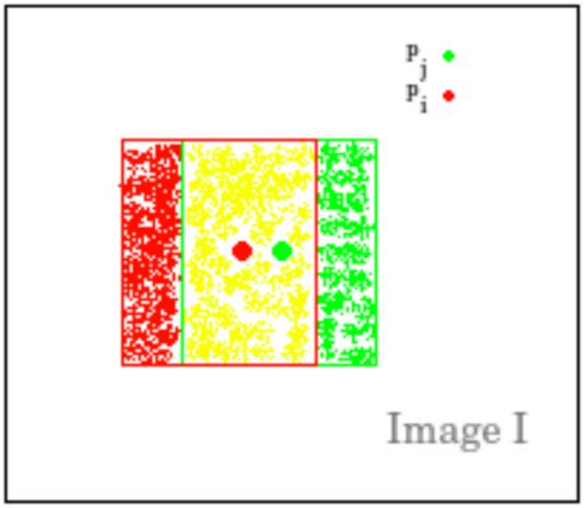
    - For each patch in I(Input images), we seek a patch in M(best matched frame) that looks similar to it. We use the L2 norm over square patches of side length 2r + 1. For pixel p ∈ I and the corresponding pixel q ∈ M , we calculate pixel wise difference in intensities for the patch centered around the pixel.

$$E_1 = \sum_{i=-r}^{+r} \sum_{j=-r}^{+r} \left\| I(x_p + i, y_p + j) - M(x_q + i, y_q + j) \right\|^2$$

- ### Pairwise Term :

    - A pairwise MRF term was introduced to gain additional knowledge from the video, so that the matches should remain consistent throughout the video.
    - For two adjacent pixels pi and pj in I, we name Ω the set of the overlapping pixels between the two patches centered at pi and pj.
    - For two adjacent pixels pi and pj in I, we name Ω the set of the overlapping pixels between the two patches centered at pi and pj.

    - Yellow region represents Ω, the set of overlapping pixels.

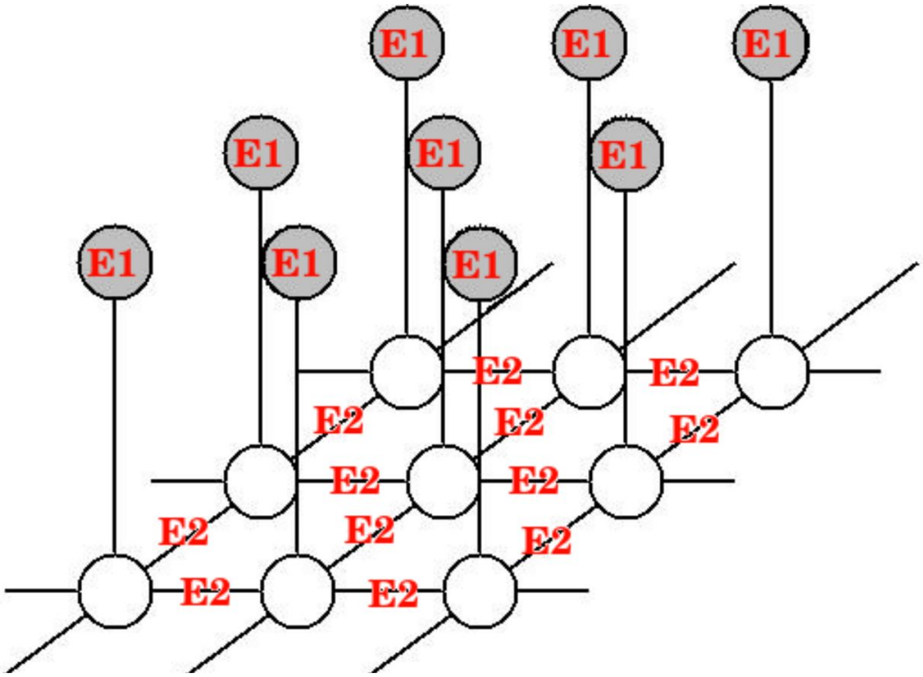


- For each pixel o∈Ω, we define the offsets δi=o−pi and δj=o−pj.
- For the energy we use L2 norm within each frame
- Then we take the L∞ across the frames of the video Yellow r

$$E_2(q_i, q_j) = \max_t \sum_{o\in\Omega} \left\| V_t(q_i + \delta_i) - V_t(q_j + \delta_j) \right\|^2$$

MRF Graphs :

MRF graph for one image:



MRF Formulation :

Final equation for energy:
$$\sum_{i \in I} E_1(p_i, q_i) \quad + \quad \lambda \sum_{i \in I, j \in N_i} E_2(q_i, q_j)$$

Denoting λ parameter controlling the importance of the pairwise term compared to the data term, Ni the neighboring pixels of i,we find q to minimize the energy function, we plan to use Belief Propagation to do so.Considering all possible pairings between a pixel in I with a pixel in M would be impractical because of the number of possible assignments. We pick the top n(=30) samples by randomly sampling the candidates for each patch according to

the probability: $\frac{1}{Z} \exp\left(-\frac{E_1}{2\sigma^2}\right)$.

where Z is a normalization factor and σ controls how diverse the sampled patches are.

Locally Affine Color Transform :

Given a densely-aligned pair of time-lapse frames obtained from our first strategy, we still need to address remaining discrepancies with our input, both because the distribution of object colors is never exactly the same.So we address to transfer the variation of color rather than the output color itself. The intuition being, if a red building turns dark red over time, transferring this time of day to a blue building should result in a dark blue.

For this we have four variables input image(I), output image(O), matched frame(M) and target frame(T) for construct {Ak}.

We design the transfer to meet two goals -

1. We want it to explain the color variations observed in the time-lapse video. We seek a series of affine models {Ak} that locally describe the color variations between T and M.
2. We want a result that has the same structure as the input and that exhibits the same color change as seen in the time lapse video. We seek an output O that is locally affine to I, and explained by the same affine models {Ak}.

A naive solution would be to compute each affine model A k as a regression between the k th patch of M̃ and its counterpart in T̃ , and then independently apply A k to the k th patch of I for each k.

However, the boundary between any two patches of O would not be locally affine with respect to I, and would make O have a different structure from I, e.g., allows for spurious discontinuities to appear at patch boundaries. Instead of this naive approach, we formulate this problem as a least-squares optimization that seeks local affinity everywhere between O and I

L2 Optimal Locally Affine Model :

We use vk to denote the kth patch of an image given in argument. For a patch containing N pixels, vk is a 3 × N matrix, each column representing the color of a pixel as (r, g, b) . We use v¯k to denote the patch augmented by ones, i.e., 4 × N matrix where each column is (r, g, b, 1) T . The local affine functions are represented by 3 × 4 matrices, Ak . With this notation, the first term in our energy models the need for the Ak matrices to transform M into T . With a least-squares formulation using the Frobenius norm this gives:

$$\sum_k \left\| \mathbf{v}_k(\tilde{T}) - \mathbf{A}_k \, \bar{\mathbf{v}}_k(\tilde{M}) \right\|_{\mathsf{F}}^2$$

We also want the output patches to be well explained by the input patches transformed by the A k matrices:

$$\sum_k \left\| \mathbf{v}_k(O) - \mathbf{A}_k \, \bar{\mathbf{v}}_k(I) \right\|_{\mathsf{F}}^2$$

.

Finally, we add a regularization term on the Ak matrices to make sure that they are not wildly different. For this we regularize Ak using a global affine model G, the regression by the entire picture of M and T , with the Frobenius norm. Formally, we solve

$$\begin{aligned} O = \arg\min\nolimits_{O,\{\mathbf{A}_k\}} & \sum_k \left\| \mathbf{v}_k(O) - \mathbf{A}_k \, \bar{\mathbf{v}}_k(I) \right\|^2 \\ & + \epsilon \sum_k \left\| \mathbf{v}_k(\tilde{T}) - \mathbf{A}_k \, \bar{\mathbf{v}}_k(\tilde{M}) \right\|^2 + \gamma \sum_k \left\| \mathbf{A}_k - \mathbf{G} \right\|_{\mathsf{F}}^2 \end{aligned}$$

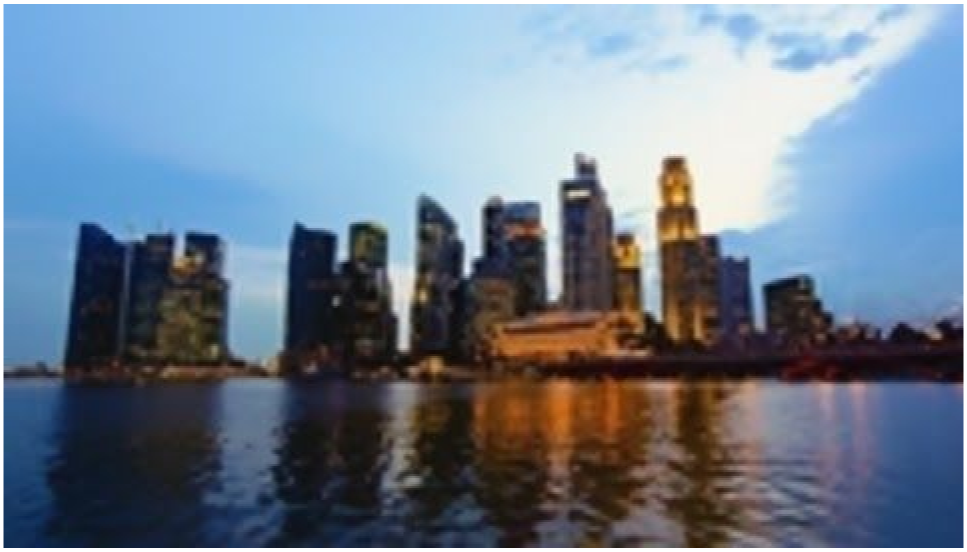
Denoising Image :

One of the side effects of using the affine mapping is that it may magnify the noise in the input image (if any). To avoid the noise magnification, we use the bilateral filtering to decompose the input image into a base layer and a detail layer. The base layer is mostly noise-free, so we apply our locally affine transfer to the base layer instead of the input image. Then finally, we obtain the final output image by adding the detail layer to the transferred base layer.

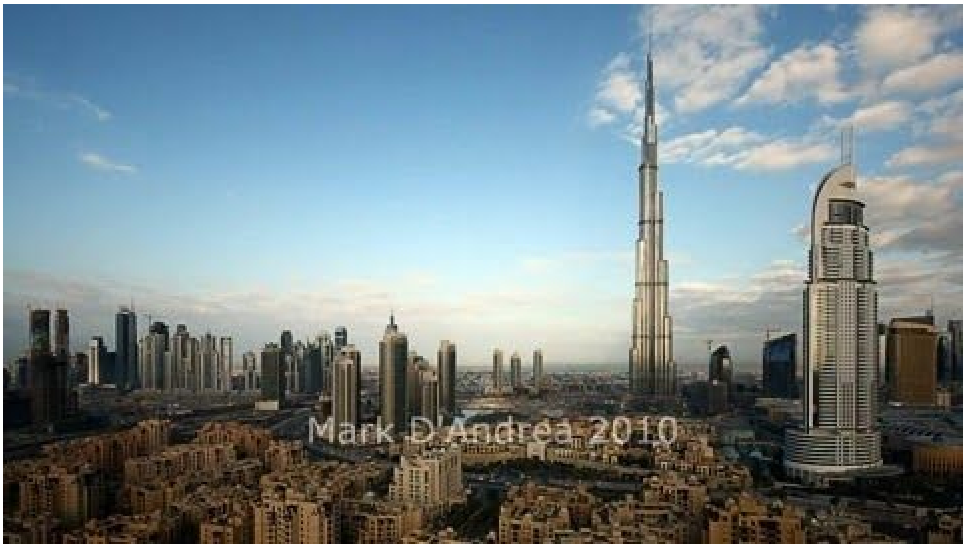
Results After Each Step :

We chose low resolution images because of low computational cost, which our laptops are able to handle.

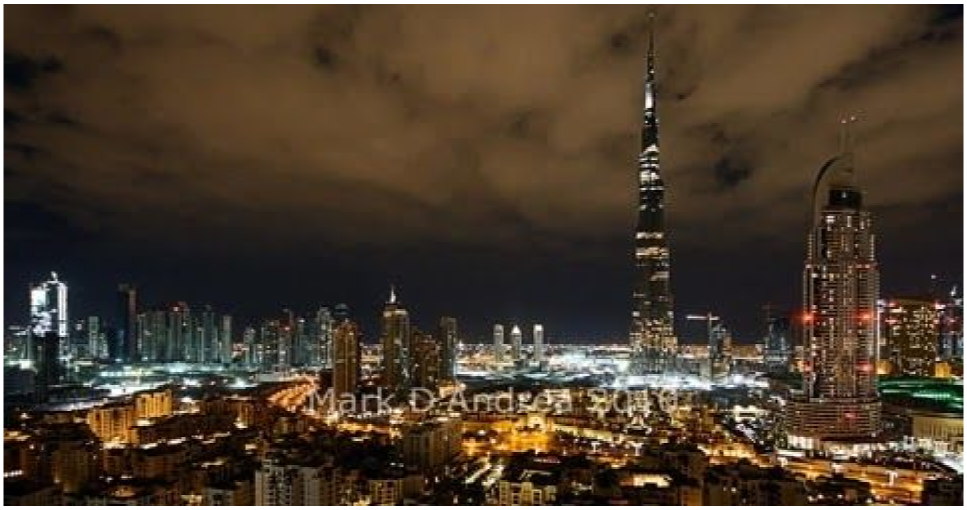
Input Image(Singa[pre) :



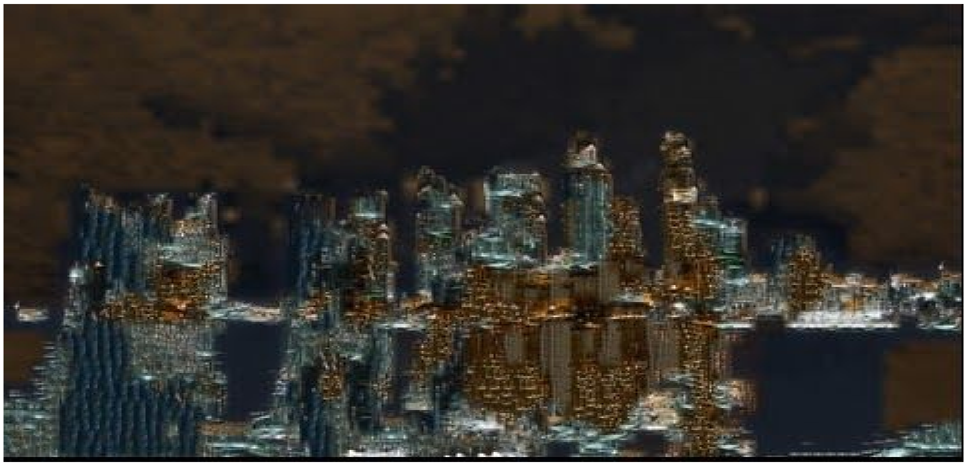
Best Frame Selected From The Database :



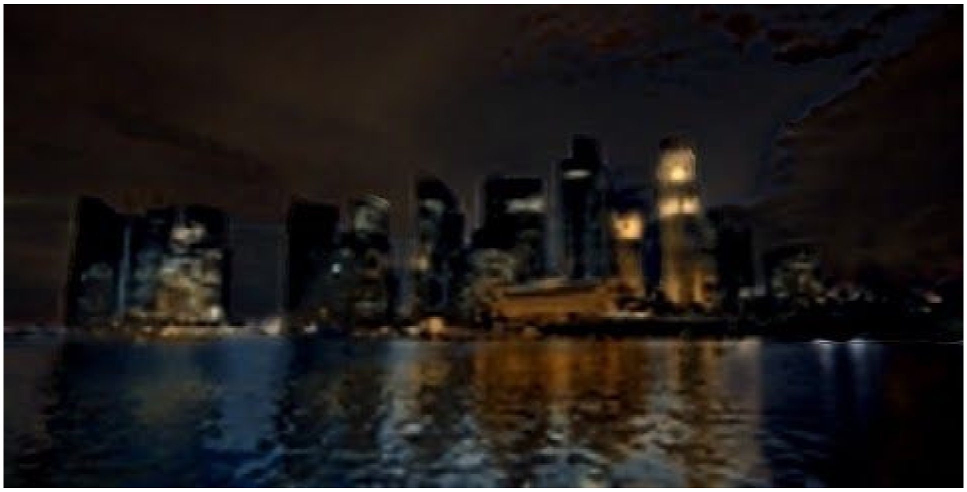
Reference Image of the time at which we want the output :



Local Matching :

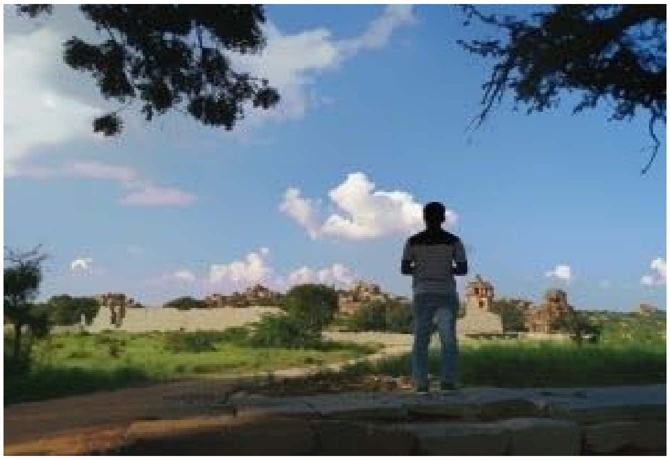


Output :

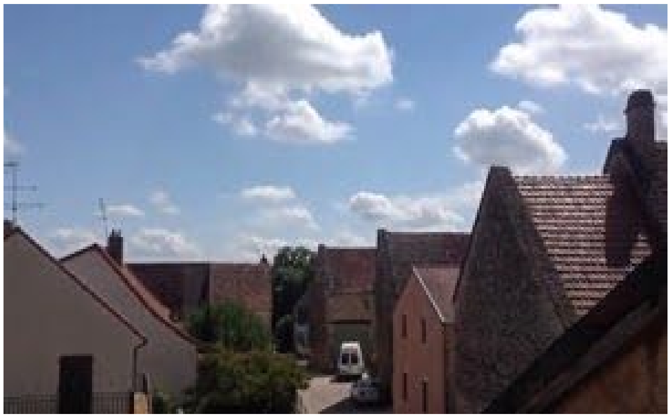


Result On Our Own Images :

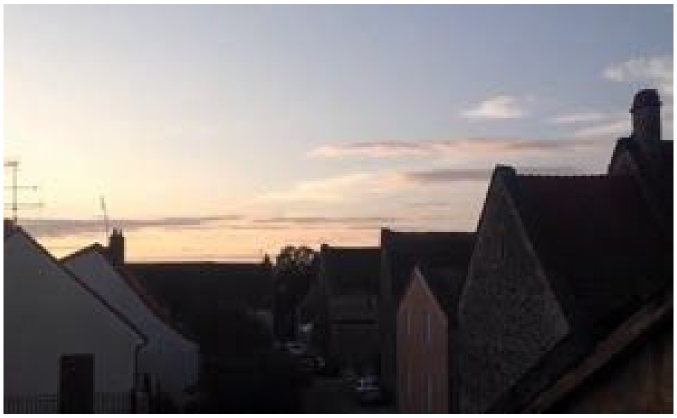
Input Image(The Image Used If Of One Of My Friend’s) :



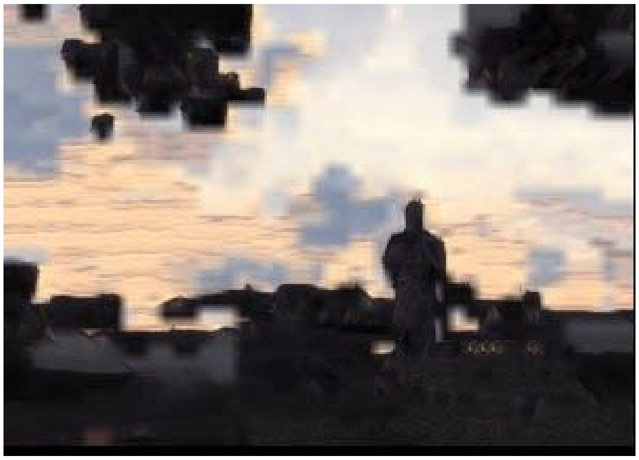
Reference Image :



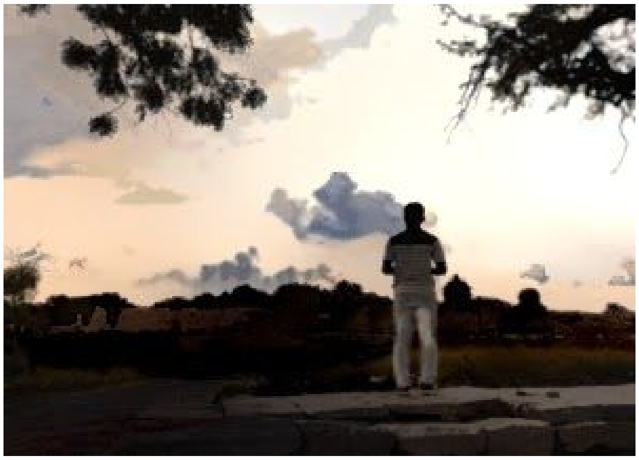
Reference Output Image :



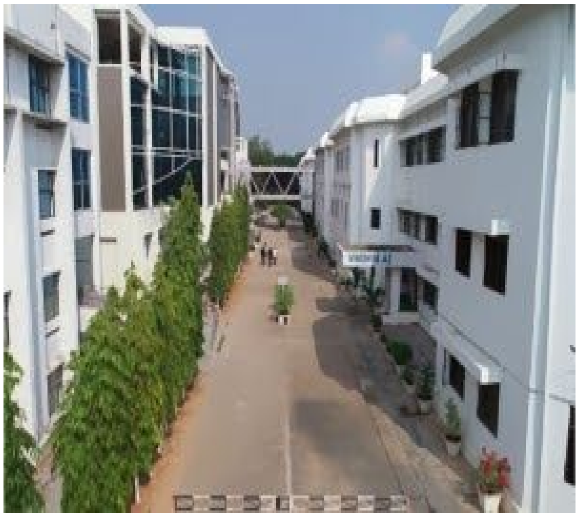
Local Matching :



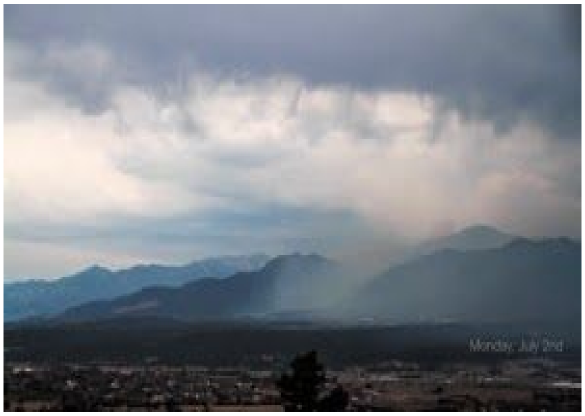
Output Image :



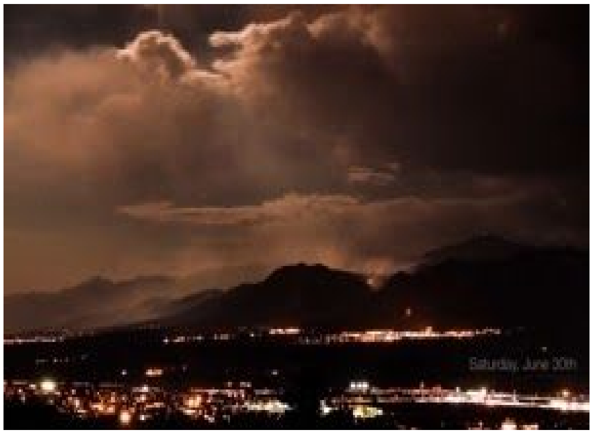
Input Image From College :



Reference Image :



Reference Output Image :

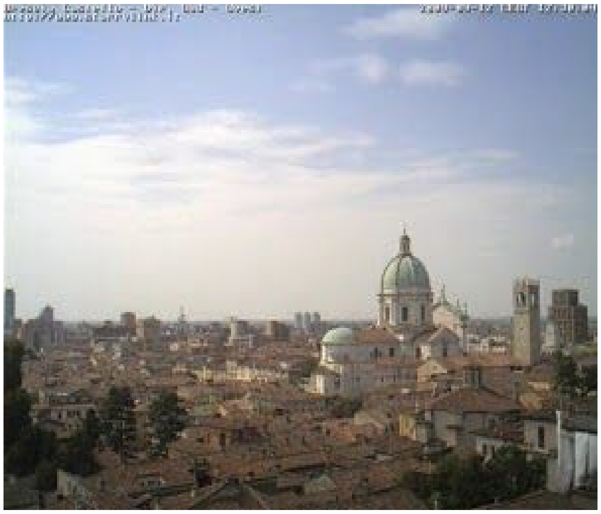


Local Matching :

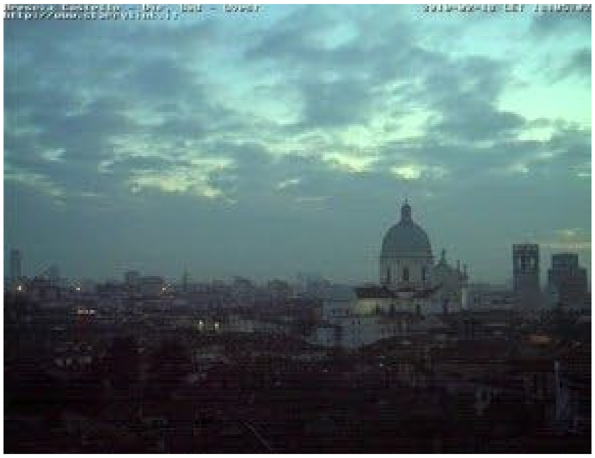
Output Image :

Input Image From Felicity Ground :

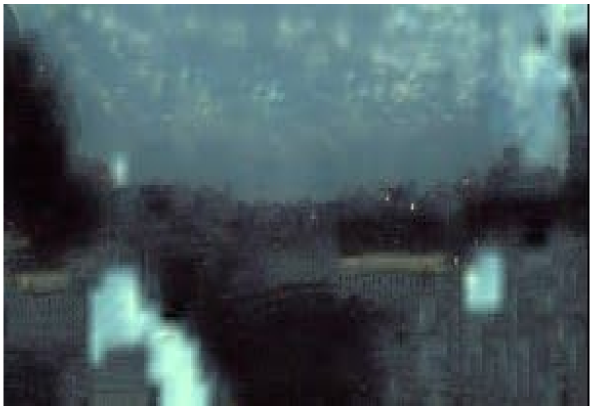
Reference Image :



Reference Output Image :



Local Matching :



Output Image :

Instruction To Run The Code :

- git clone https://github.com/PK1210/Image-Hallucination
- Install mexOpenCV for matlab
- Do pip install -r requirements.txt
- Add path of mexOpenCV in addpath in libs/Timelapse/SearchCandidates.m
- Download timelpaseHOG.pickle from the same drive link and put it in src.
- Download the dataset from this link: https://drive.google.com/open?id=169AH1CZ9jSlBNGalyiKwh9ZuIOxCf0MT in the main folder of the repository
- First run the jupyter notebook src/"selecting best videos.ipynb" <video path> <frame number>
- Do matlab -nojvm
- Edit src/config.m , Set there desired image path, video path, approximate output frame number
- Do run_exp config
- Check the results in results folder

Note: The development was done on a linux environment

References :

1.http://people.csail.mit.edu/yichangshih/time_lapse/time_lapse.pdf

2. XIAO , J., HAYS , J., EHINGER , K., OLIVA , A., AND TORRALBA , A. 2010. Sun database: Large-scale scene recognition from abbey to zoo. In Computer vision and pattern recognition (CVPR), 2010 IEEE conference on, IEEE, 3485–3492

3. https://papers.nips.cc/paper/1832-generalized-belief-propagation.pdf

4. https://inc.ucsd.edu/~marni/Igert/Lazebnik_06.pdf

5. http://cs.brown.edu/people/pfelzens/papers/lsvm-pami.pdf