BLIND IMAGE DEHAZING USING INTERNAL PATCH RECURRENCE

A Project Report

Submitted by

Purva M. Tendulkar 111403049

in partial fulfillment for the award of the degree

of

B.Tech Computer Engineering

DEPARTMENT OF COMPUTER ENGINEERING

AND

INFORMATION TECHNOLOGY,

COLLEGE OF ENGINEERING, PUNE-5

April, 2018

DEPARTMENT OF COMPUTER ENGINEERING

AND

INFORMATION TECHNOLOGY,

COLLEGE OF ENGINEERING, PUNE

CERTIFICATE

Certified that this project, titled "BLIND IMAGE DEHAZING USING INTERNAL PATCH RECURRENCE" has been successfully completed by

Purva M. Tendulkar   111403049

and is approved for the partial fulfillment of the requirements for the degree of "B.Tech. Computer Engineering".

SIGNATURE	SIGNATURE
DR. VANDANA INAMDAR	**DR. VANDANA INAMDAR**
Project Guide	**Head**
Department of Computer Engineering	**Department of Computer Engineering**
and Information Technology,	**and Information Technology,**
College of Engineering Pune,	**College of Engineering Pune,**
Shivajinagar, Pune - 5.	**Shivajinagar, Pune - 5.**

Abstract

Images of outdoor scenes are often degraded by haze, fog and other scattering phenomena. Dehazing such images to obtain a haze-free image can be useful in several real-world applications such as road visibility for autonomous cars and improving the clarity of images in order to extract useful information from them (e.g. forensic reports, surveillance cameras). While several algorithms already exist for image dehazing, speed is a serious concern which has not yet been completely solved. For real-time applications such as self-driving cars, dehazing images in real-time becomes very important. If it cannot achieve the desirable speed, the application is useless.

The aim of this project is to implement the paper - “Image Dehazing Using Internal Patch Recurrence” - authored by renowned researcher Michal Irani and make the code deployable as a software. We also intend to improve on the existing algorithm by finding ways to optimize it and improve the running speed. Besides this, the novel idea of using patch recurrence for Image Dehazing opens up several avenues for research such as clustering images by similarity and composition, extracting information from videos, etc. Through this project, we hope to investigate the patch recurrence property in these fields and understand the real-world constraints for application-level deployment.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The Image Dehazing Problem

Images of outdoor scenes are usually degraded by a scattering medium (e.g. airlight, dust particles, water droplets, mist, fog and smoke). The irradiance received by the camera from the scene point is attenuated along the line of sight. Furthermore, the incoming light is blended with the airlight and this becomes even more complex when other light sources such as streetlights, lamps or car lights interfere with the ambient airlight. Since the amount of scattering depends on the distance of the scene points from the camera, the degradation is spatially variant. We can generalize this degradation to be attributed to two main factors : (i) the irradiance L(x) emitted from scene points getting attenuated due to scattering caused by haze particles along the line of sight, and (ii) ambient airlight A getting scattered by haze particles.

Haze removal has become very popular in computational photography and mobile applications these days. For example, in case of road visibility for regular and autonomous cars, having a system which can take a take a snapshot of the road and dehaze it can help the driver or system get better information about distant objects or obstacles on the road, steep slopes, sharp edges or large potholes which are not visible due to the changing weather, insufficient lighting or dense foggy conditions.

Mathematically this can be modeled as :

$$I(x) = t(x)L(x) + (1 - t(x))A \tag{1.1}$$

where t(x) is the corresponding attenuation factor, known as the transmission

$$t(x) = e^{-\beta Z(x)} \tag{1.2}$$

where β is a scattering coefficient and Z(x) is the distance to the scene point. t(x) is typically assumed to be the same for all three color channels (R,G,B). Blind image dehazing, namely, recovering the haze parameters A and t(x) and inverting Eq. (1.1) to recover a haze-free image, is an under-constrained problem.

1.2 Background

While several algorithms already exist for image dehazing, speed is a serious problem which has not yet been completely solved. For real-time applications such as self-driving cars, dehazing images in real-time becomes very important. If it cannot achieve the desirable speed, the application is useless.

Another problem with many existing implementations is that they make certain assumptions about images which often leads to messy and non-intuitive solutions. The current state-of-the art paper [5] also has certain assumptions such as uniform global airlight for the entire image, visibility of scene points situated at infinity and that the airlight is the brightest color in the image. These assumptions result in problems for correct airlight detection in images such as Fig 1.1.

Figure 1.1: Underwater Water Tank

As a part of this project, we have implemented the current state-of-the-art paper [5] as a baseline but have mainly targetted the recent paper by Bahat and Irani. This paper exploits the patch recurrence property of natural images to recover the haze parameters of Eq. (1.1) and eventually recover the dehazed output. The implementation of this paper had three main issues (i) hacky implementation of core principle, (ii) unavailability of the code for deployment and (iii) extremely slow speed. We have attempted to resolve each of these issues through this project by modifying the core algorithm to get a speed improvement as also develop a more elegant implementation than the authors had achieved.

The rest of this report is organized as follows. Chapter 2 gives the Literature Survey in this field. Chapter 3 gives an overview of the original algorithm. Chapter 4 gives details about the modified part of the algorithm for the purpose of speed improvement. Chapter 5 contains results and Chapter 6 talks about the next steps for implementation and challenges in this field.

Chapter 2

Literature Survey

Blind image dehazing, namely, recovering the haze parameters A and t(x) and inverting Eq. (1.1) to recover a haze-free image, is an under-constrained problem. Different dehazing methods have suggested different ways to cope with this problem. Some assume having multiple images of the same scene taken under different conditions, e.g. different polarizations [[10], [11]] or under different weather conditions [[7], [8], [9]]). More recently, methods based on a single image have been proposed, which tackle the lack of constraints by incorporating various priors. Tan assumed maximal local contrast in the dehazed image. However this often leads to an exaggerated contrast. He et al. suggested t-map recovery by using the Dark Channel Prior, and Tang et al. combined these notions and others into a learning framework. Zhu et al. performed image dehazing by learning a linear color attenuation model. All four methods [[5], [13], [14], [15]] estimate A by implicitly assuming that regions at infinity (e.g., the sky) are visible in the image, and that the airlight color A is the brightest color among these regions. Fattal assumed that the scene irradiance L(x) and transmission map t(x) are locally uncorrelated, and used a user assisted approach to estimate A. In his later works [[4], [12]], he recovered haze parameters by assuming that the distribution of pixels inside constant albedo patches can be described as lines in RGB space.

Chapter 3

Algorithm Overview

3.1 Algorithm

Algorithm 1 Overview of the Algorithm

```
Input: Hazy image I(x)
Output: Airlight $\hat{A}$, tmap $t(\hat{x})$, dehazed image $L(\hat{x})$
1: Detect “co-occurring pairs”:
    (a) Extract structured (high-variance) patches from image I(x).
    (b) Search for matching patches (with high normalized correlation).
2: Extract Pairwise haze parameters for each pair:
    (a) Estimate relative t-values $t_2/t_1$.
    (b) Estimate their shared airlight correlation.
3: Estimate Global haze parameters:
    (a) Recover global airlight $\hat{A}$ airlight estimates.
    (b) Recover dense t-map $t(\hat{x})$ which: (i) is smooth, and (ii) satisfies the sparse pairwise
        constraints.
4: Recover haze-free $L(\hat{x})$: $\frac{I(x) - \hat{A}}{t(\hat{x})} + \hat{A}$
```

3.2 Brief Overview

Small image patches tend to repeat abundantly inside a natural image, both within the same scale, as well as across different scales. However, as the conditions become unideal, in the case where images are taken under bad weather conditions (haze, fog, underwater scattering, etc.), this property diminishes to the same degree. This can be seen in Fig 3.1. The given algorithm exploits the deviations from the ideal patch recurrence for "Blind Dehazing" - namely, recovering the unknown haze parameters and reconstructing a haze-free image. We seek the haze parameters that, when used for dehazing the input image, will maximize the patch recurrence in the dehazed output image. The algorithm thus attempts to obtain information about the haze in the image by checking the degree to which the image is geting distorted. It thus looks for similar (or co-occurring) patches for obtaining this information. This will then be used to recover the original image.

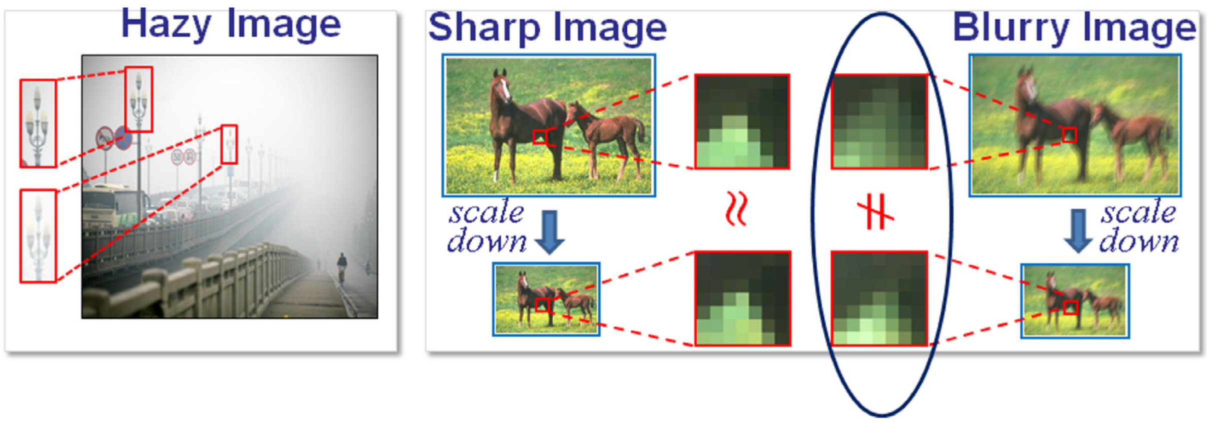


Figure 3.1: Deviations from Ideal Behavior

3.3 The Patch Recurrence Property

Step 1 of the algorithm has been implemented by using a KDTree to search for kNNs (k Nearest Neighbors). However, before we begin to search for the pairs of patches, there are some pre-processing operations that need to be performed, namely (i) removal or airlight and (ii) normalization of patches. These are explained in the following subsections.

3.3.1 Airlight Removal

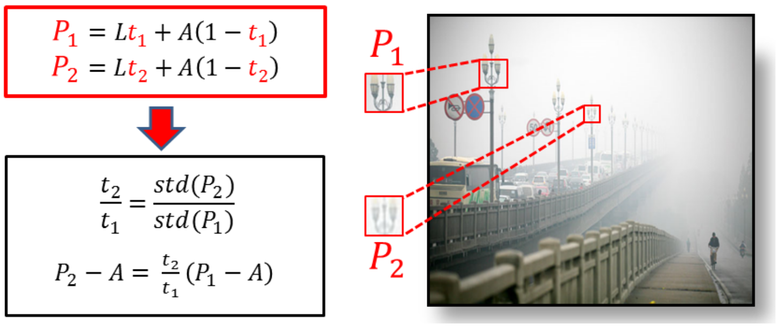
Let $P_1[x]$ and $P_2[x]$ denote a pair of small co-occurring patches (7 x 7) of the same patch L[x] in the haze-free image, but situated at different depths, $Z_1[x]$ and $Z_2[x]$. Since these patches are very small, we can assume constant depths in each patch, $Z_1[x] = Z_1$ and $Z_2[x] = Z_1$. We thus also assume constant transmission values, t_1 and t_2 (since $t = e^{-\beta Z}$). We further assume that the airlight is locally uniform within each patch ($A_1[x] = A_1$ and $A_1[2] = A_2$), even if not uniform in the entire image. According to Eq. (1.1):

$$P_1[x] = L[x]t_1 + A_1(1 - t_1) \tag{3.1}$$

$$P_2[x] = L[x]t_2 + A_2(1 - t_2) \tag{3.2}$$

Note that ideally, the patches P_1 and P_2 should be perfectly identical due to their co-occurrence property and the fact that they are checked across different scales. However, due to the haze and their different depths, they appear to be quite different (See Fig. 3.2).

When the transmission t and airlight A in Eq. (1.1) are locally uniform, subtracting the local mean-color eliminates the airlight component. In order to eliminate the airlight component, we subtract the mean value from each side of Eq. (3.1) and (3.2):



$P_1 = Lt_1 + A(1 - t_1)$
$P_2 = Lt_2 + A(1 - t_2)$
$\frac{t_2}{t_1} = \frac{std(P_2)}{std(P_1)}$
$P_2 - A = \frac{t_2}{t_1}(P_1 - A)$
P_1
P_2

Figure 3.2: Deviations from Ideal Behavior

$$\widetilde{P_1}[x] = \widetilde{L}[x]t_1 \tag{3.3}$$

$$\widetilde{P_2}[x] = \widetilde{L}[x]t_2 \tag{3.4}$$

where $\widetilde{P_i}[x] = P_i[x] - mean(P_i)$ and $\widetilde{L}[x]$ = L[x] - mean(L) (the mean is computed separately for each color channel within each patch).

3.3.2 Normalization of Patches

We have now removed the effect of airlight in the mean free patches $\widetilde{P_1}$ and $\widetilde{P_2}$ (Eq. (3.3) and (3.4)). However, their different transmissions still obscure their similarity. We therefore normalize the mean-free patches, $\widetilde{P_1}$ and $\widetilde{P_2}$, by dividing by their l_2 norm):

$$\frac{\widetilde{P_1}[x]}{\|\widetilde{P_1}\|} = \frac{t_1\widetilde{L}[x]}{t_1\|\widetilde{L}\|} = \frac{\widetilde{L}[x]}{\|\widetilde{L}\|} \tag{3.5}$$

$$\frac{\widetilde{P_2}[x]}{\|\widetilde{P_2}\|} = \frac{t_2\widetilde{L}[x]}{t_2\|\widetilde{L}\|} = \frac{\widetilde{L}[x]}{\|\widetilde{L}\|} \tag{3.6}$$

$$\frac{\widetilde{P_1}}{\|\widetilde{P_1}\|} = \frac{\widetilde{P_2}}{\|\widetilde{P_2}\|} \tag{3.7}$$

This shows the recurrence property of hazy patches which is obtained by normalizing their mean-free versions. We can now apply Nearest-Neighbors (NN) search on the normalized patches to find the pairs of co-occurring patches.

3.3.3 Nearest Neighbor Search

A search is now applied to obtain the k NNs (Nearest Neighbours) on the normalized co-occurring patches detected above. This search is done across multiple scales of the image to account for the fact that same patches when farther away in the image, become smaller in size. We can do this since scaling down the hazy image does not change the physical parameters of the scene (the haze scattering parameters or β). It will simply magnify or demagnify the image, affecting the display of the transmissions map, but preserving its absolute range of values.

Given a hazy input image I(x), its multi-scale versions are generated: I^{sc}, i.e., sc = 1, 0.75, 0.5, 0.25. NN-search is applied only to patches which have high std (above 25 gray-scale levels) and discard pairs of patches whose normalized-correlation is lower than 0.7. This was considered in order to only include distinct patches with definite features. It was thought that such patches will give maximum information about the dehazed image. This however is incorrect since we require hazy (low std) patches as well, in order to get information about the haze parameters. Our goal is to get information about the haze parameters; not the dehazed image. This is discussed later in the paper.

3.3.4 Importance of Outlier Removal

We can obtain a lower-Bound $t_{LB}(x)$ on t(x) as:

$$t(x) \geq \max_{c \in R,G,B} \{1 - \frac{I_c(x)}{A_c}\} \triangleq t_{LB}(x) \tag{3.8}$$

This t_{LB} preserves t-ratios, i.e. a false outlier pair does not satisfy-

$$\frac{t_{LB_2}[x]}{t_{LB_1}[x]} = \frac{t_2[x]}{t_1[x]} \tag{3.9}$$

Thus, for each pixel in the patch, we compute the difference

$$|\frac{t_{LB_2}[x]}{t_{LB_1}[x]} - \frac{\hat{t_2}}{t_1}| \tag{3.10}$$

between the estimated t-ratio $\frac{\hat{t_2}}{t_1}$ and the t_{LB} ratio. We then take the average over all pixels in the pairs of patches. Patches for which this average difference exceeds a certain threshold (0.08 in the current implementation) are detected as outliers. This can be seen in Fig. 3.3.

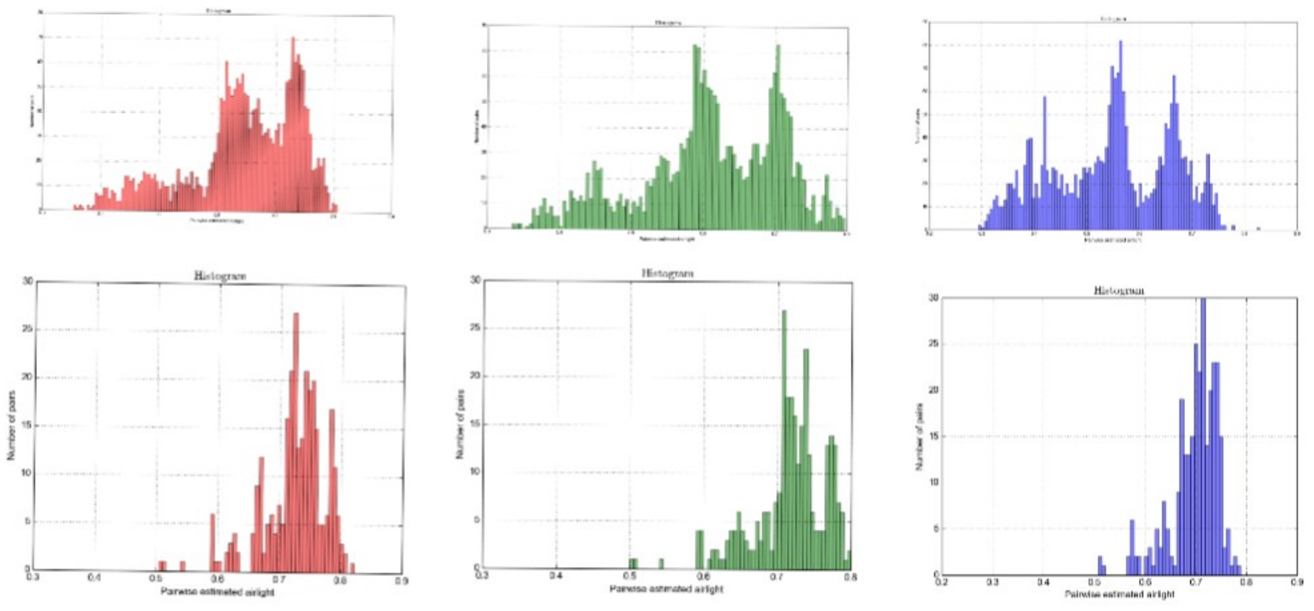


Figure 3.3: Histogram Comparison before and After Outlier Removal for R, G, B values respectively

3.4 Airlight Estimation

3.4.1 Obtaining Pairwise Relative Transmission

Given a pair of co-occurring patches, we now obtain their relative transmission parameters t_2/t_1. Eq. (3.3) and (3.4) entail:

$$\|\widetilde{P_1}\| = \|\widetilde{L}t_1\| = \|\widetilde{L}\|t_1 \tag{3.11}$$

$$\|\widetilde{P_2}\| = \|\widetilde{L}t_2\| = \|\widetilde{L}\|t_2 \tag{3.12}$$

Now t_1 and t_2 are scalars between 0 and 1. Hence,

$$\frac{t_2}{t_1} = \frac{\|\widetilde{P_2}\|}{\|\widetilde{P_1}\|} \tag{3.13}$$

Assuming l_2 norm, we get the simplified ratio between the tmaps of two co-occurring patches, P_1 and P_2 as:

$$\frac{t_2}{t_1} = \frac{std(P_2)}{std(P_1)} \tag{3.14}$$

3.4.2 Obtaining Pairwise Shared Airlight

Assuming that a pair of co-occurring patches share the same airlight color, $A_1 = A_2 =$ A, we now calculate this shared airlight. Eq. (3.1) and (3.2) can be rewritten as:

$$P_1[x] - A = (L[x] - A)t_1 \tag{3.15}$$

$$P_2[x] - A = (L[x] - A)t_2 \tag{3.16}$$

$$(P_2[x] - A) = \frac{t_2}{t_1}(P_1[x] - A) \tag{3.17}$$

Combining Eq. (3.3) and (3.4) with Eq. (3.17) yields:

$$(P_2[x] - A)\widetilde{P_1}[x] - (P_1[x] - A)\widetilde{P_2}[x] = 0 \tag{3.18}$$

The shared airlight A can thus be estimated using Least-Squares as:

$$A = \frac{[\widetilde{P_2} - \widetilde{P_1}][P_1 \circ \widetilde{P_2} - P_2 \circ \widetilde{P_1}]}{\|\widetilde{P_2} - \widetilde{P_1}\|^2} \tag{3.19}$$

where $\circ$ denotes the element-wise vector multiplication (also known as the Hadamard product).

3.4.3 Estimating Global Airlight

The global airlight estimate is simply a weighted average of the above obtained pairwise estimates.

$$\hat{A} = \frac{\sum_k w_k \hat{A}_k}{\sum_k w_k} \tag{3.20}$$

where w_k is determined heuristically.

3.5 Tmap Estimation

We would like to obtain a dense tmap such that, when we substitute these parameters in Eq (1.1), resulting haze free image L(x) will have maximal internal patch recurrence, within and across different scales of L(x).

Specifically, we want the t-map t(x) that will minimize:

$$\underset{x}{\operatorname{argmax}}\{\rho_l(t(x)) + \lambda\rho_s(t(x))\} \qquad s.t.\ t_{LB} \leq t(x) \leq 1 \tag{3.21}$$

where $\rho_l(t)$ is the patch recurrence constraint on the resulting dehazed image L(x), $\rho_s(t)$ is a smoothness term on the recovered t-map, and λ is empirically set to 0.5.

The patch-recurrence data term is given as-

$$\rho_l(t(x)) = \sum\nolimits_{k \in pairs} \|(P_{k_1} - \hat{A})t_{k_2}(x) - (P_{k_2} - \hat{A})t_{k_1}(x)\|^2 \tag{3.22}$$

where $t_{k_1}(x)$ and $t_{k_2}(x)$ are the transmission values corresponding to the center pixels of patches $P_{k_1}(x)$ and $P_{k_2}(x)$, respectively. These are pairs of patches that have low SSD without normalization. Such patches, not only correspond to co-occurring patches in the real world, but also underwent the same "hazing" and therefore we assume they have the same t-value. The implementation includes running a simple, fast NN-search directly on the image, without first normalizing each patch (e.g. Using the PatchMatch algorithm). For such patches, we minimize this modified version of the term in Eq. (3.22)-

$$\rho_l(t(x)) = \sum\nolimits_{k \in low-SSDpairs} \|t_{k_2}(x) - t_{k_1}(x)\|^2 \tag{3.23}$$

Combining the two sets of pairs increases the spatial density of the patch recurrence condition.

The smoothness term is given as-

$$\rho_s(t(x)) = \sum_{allpixelsx} w(x) \|\nabla log(t(x))\|^2 \tag{3.24}$$

where w(x) is a decreasing sigmoid function of ∇ L(x). Now from Eq. (1.2) we have log(t(x)) $\propto$ Z(x). Thus, the "depth map", log(t(x)), is penalized for having large depth discontinuities in the dehazed image L(x).

Note that the pairs in Eq. (3.23) are raw pairs which are obtained by running a PatchMatch search [2] on the raw input image. These pairs, along with those used for the airlight estimation step should attempt to cover all patches in the images, in order for dense recovery of the final image.

3.6 Final Dehazed Image Recovery

The final dehazed image is obtained by substituting the obtained parameter values of $T(\hat{x})$ and $\hat{A}$ in Eq. (1.1).

$$L(x) = \frac{I(x) - A}{t(x)} + A \tag{3.25}$$

3.7 Issues with this implementation

There are several issues with this implementation which are explained below-

3.7.1 Consideration of only High Standard Patches

As mentioned above, the algorithm mentions strictly using patches with high standard deviation values and bluntly discards all others as less-informative. However, it is very important to include the low standard patches in our KDTree searches as well, as we need to account for the depth effect which in turn causes the image hazing.

3.7.2 KDTree Search for Nearest Neighbors

The original algorithm for generating pairs involved extracting 7 x 7 patches from each of the scaled images and then building a different KDTrees for each scaled image iteration. This means that for the first iteration, the query patches will be all the patches from the first scaled image and the candidate patches will be all the patches across all scaled images made to form the first KDTree. The second iteration will consist of a KDTree made up of from the second scaled image onwards and so on.

Building a KDTree has worst case complexity $\mathcal{O}(nlog^2n)$. Querying an axis-parallel range in a balanced KD tree takes $\mathcal{O}(n^{1-1/k} + m)$ time, where m is the number of the reported points, and k the dimension of the KD tree. Thus, due to the high dimension of KD Tree as a result of the large number of patches per scaled image, the querying step becomes very expensive. Besides this, it is observed that the above method produces a number of pairs of which the patches are in close proximity to each other. This is obvious because of the small size of patches and existence of scaled versions of the image. Thus, these do not provide us with a lot of useful information about the image haze parameters, and leads to a waste of time. Another drawback is that since the query set and candidate set contain overlapping patches, there is an explicit need to remove duplicates.

We therefore need an optimal method to select better pairs without having to iterate through all possible patches for every single query patch. In fact, it is not even required that we find nearest neighbors for every single query patch. We simply require pairs which will provide us with enough information to closely estimate the airlight. This requirement can be characterized such that patches in a pair must-

1. have high correlation (necessary requirement)
2. be sufficiently far away from each other to provide information about depth
3. have sufficiently large difference in their standard deviation values

3.7.3 Optimization step for T-map estimation

The tmap optimization step involves minimizing Eq. (3.21). The original implementation of this is MATLAB required close to 40 mins for the completion of just this step. This implementation is too slow and is of no good use for post applications.

All these problems have been tackled in the next section to improve on the speed and efficiency of the algorithm.

Chapter 4

Algorithm Modification

4.1 Detailed Algorithm for Generating Pairs

Algorithm 2 Generate Pairs

```
Input: Hazy image I(x)
Output: Pairs of co-occurring patches
1: Extracting patches
    (a) Obtain scaled versions of the image $I^{sc}$, with scale factors, sc = 1, 0.75, 0.5, 0.25.
    (b) Extract alternate patches from these scaled images
2: Smoothen the standard deviations
    (a) Apply Gaussian filter across images to smoothen the std devs.
    (b) Assign bucket numbers to every patch based on its new std dev value
    (c) Perform Histogram Equalization across each scaled image.
3: Search operation
    (a) Divide the patches lying in buckets 1-5 as query patches and those from 7-10 as
        candidate patches.
    (b) Build a single global KD Tree of the candidate patches
    (c) Randomly select a some query patches from the query set. This selection is heuristic
        and is dependent on number and std dev ranges of patches (e.g. 100 for cityscape).
4: Return the obtained pairs
```

4.2 Extracting patches

Just like in the original algorithm, scaled versions of the image are first obtained. However, we observe that since these patches are all overlapping, not all of them are necessarily required for estimating the dehazing parameters. Hence to save time required for extracting patches, we only extract alternate patches.

4.3 Smoothening standard deviations

We wish to obtain pairs such that the patches are spatially at a distance from each other and have sufficiently distinct std dev values. We achieve both these conditions by applying Gaussian Filtering followed by Gaussian Filtering followed by bucketing. Applying a filter to smoothen the std dev values ensures that large regions of smooth std deviation values will be created. Bucketing means dividing the patches into 10 buckets depending on their std values.

$$bucketDiff = \frac{maxStd - minStd}{numBuckets} \tag{4.1}$$

Thus,

$$bucket_{patch_i} = i$$
$$if(i-1)minStd \leq std_{patch_i} < (i-1)minStd + bucketDiff$$

where $bucket_{patch_i}$ is the bucket number of $patch_i$, and maxStd and minStd is the maximum and minimum std dev values among all patches in all scaled images. This can be seen in Fig 4.1.

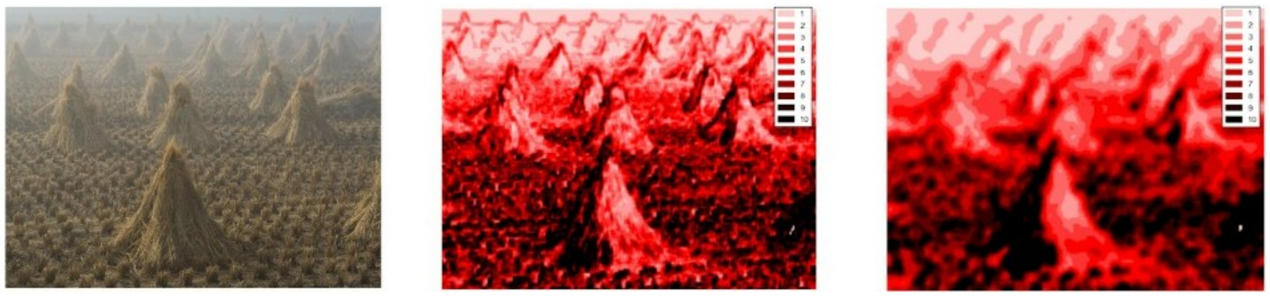


Figure 4.1: Buckets before and after smoothening

4.3.1 Histogram Equalization

In order to ensure that there are equal number of patches in every bucket, we apply a technique called Histogram Equalization. Intuitively this means that smoothened std dev values of all patches are arranged in increasing order and are then divided into 10 buckets each. This step is necessary in order to ensure equal distribution of patches across all buckets since we will be separating these patches into query and candidate patches on the basis of their bucket numbers in the search operation. A complete example with this done across all scaled images is shown in Fig. 4.2.

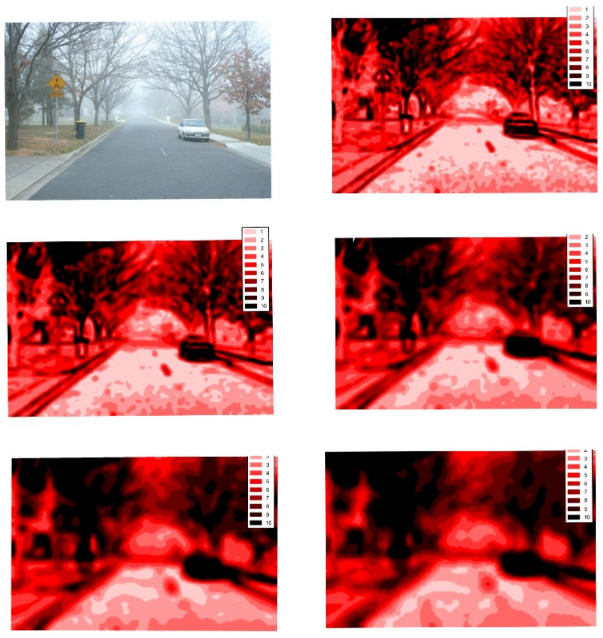


Figure 4.2: A complete example

4.4 Search Operation

After smoothening and bucketing, the search operation is performed by creating a single global KD Tree made out of all patches with bucket numbers from 1 to 5. The candidate patches are randomly chosen from patches with bucket numbers from 7 to 10. Patches with bucket number 6 are intentionally chosen to ensure distinct standard deviations and correspondingly, distances. Due to this separation of query and candidate patches, we do not require the additional step of removing duplicates and achieve a good speedup without much compromise on the correctness of the values.

Note that there is no “correct” value as such for airlight estimation or tmap. The ground truth values of airlight were obtained by manually marking regions of “sky” in the images, and averaging the RGB values over all pixels in the region. Thus there being no “absolute” truth, we simply seek the values which will satisfy Eq. (1.1) and lead to dense recovery of the haze-free image.

4.5 Tmap estimation implementation

The estimation steps given in Section 3.5 involve minimization of the Eq. (3.21). We have implemented this step by using Iterative Minimization where each iteration is a constrained non-convex optimization problem. We start with an initial guess t(x) = t_{LB}(x). Note that t_{LB}(x) already satisfies the constrained range of t(x), as well as the t-ratio constraints, but is usually not smooth. At each iteration we approximate L(x) by using Eq. (3.25) with t(x) from the previous iteration. We re-estimate the weights w(x) = sigmoid(∇ L(x)) of the smoothness term using the current L(x), and re-estimate t(x) by minimizing Eq. (3.21). This process is iterated until there is no significant change in the error term (typically around 20 iterations). The above iterative process results in our final t-map t(x) and our dehazed image L(x). Examples of recovered t-maps are shown in Fig. (4.3), (4.4), (4.5). Since small amount of haze is present even on clear days, totally haze-free images tend to look unnatural [5]. Therefore, we allow for a small amount of haze to be left in our final dehazed output images L(x), namely: $L(x) = \alpha\ L(x) + (1\ \ \alpha)\ I(x)$. We used $\alpha = 0.85$ in our displayed images.

For solving the non-convex optimization problem, we make use of Pytorch library which automatically builds the computational graph when the operations are specified as Torch variables belonging to the nn.Module. Pytorch is typically used for Deep Learning problems but we utilize it effectively for optimization problems which leads to a major speedup to a couple of minutes. Moreover, Pytorch allows GPU portability which results in further speedup.

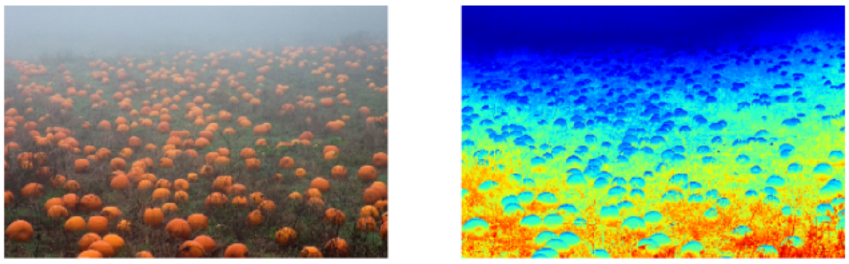


Figure 4.3: Tmap for pumpkins

Figs. (4.3), (4.4), (4.5) shows some recovered depth maps before performing the optimization step. This itself gives a very good indication about distant scene points and image depth. The warm colors indicate that the patches are closer to the camera while the cooler colors indicate that they are farther away.

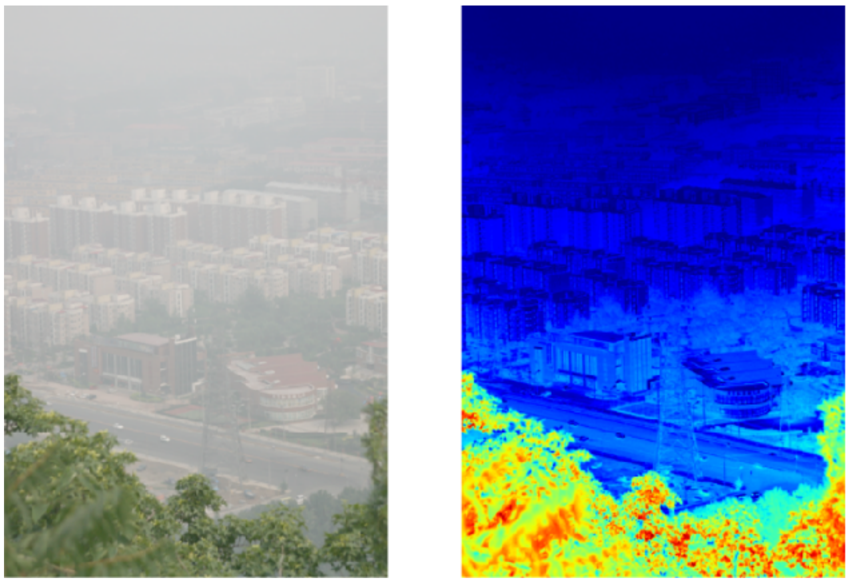


Figure 4.4: Tmap for cityscape

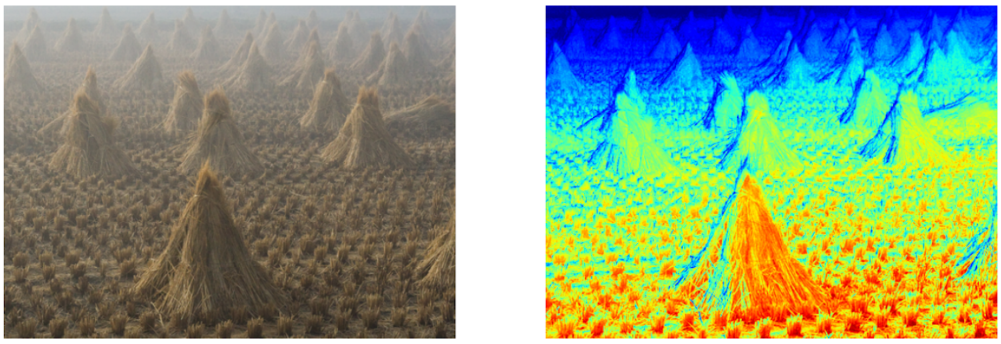


Figure 4.5: Tmap for cones

Chapter 5

Results

5.1 Estimated Airlight Value Comparison

Table 1: Airlight values for some images		
Image	**Our Results**[1]	**Deviation from GT**[2]
cones	[0.71976981 0.72745643 0.71854034]	0.0234
road	[0.93264639 0.87870285 0.81200055]	0.2336
tiananmen	[0.7886754 0.86859117 0.9039509]	0.01088
cityscape	[0.80868984 0.80752552 0.8102263]	0.3298
pumpkins	[0.76308387 0.73434754 0.6916499]	0.1437
mountain	[0.81163521 0.69354138 0.6299637]	0.2106
underwaterWaterTank	[0.46073113 0.35306381 -0.00569156]	0.0079878

Table 5.1: Airlight values for some images

5.2 Some Dehazed Outputs

Following are some dehazed output images. The first is the hazy input image, the second is our current non-optimized result, and the third is our baseline obtained from our Dark Channel Prior paper implementation.

[1]Values in B, G, R

[2]GT = Ground Truth. Difference calculated as Euclidean sum

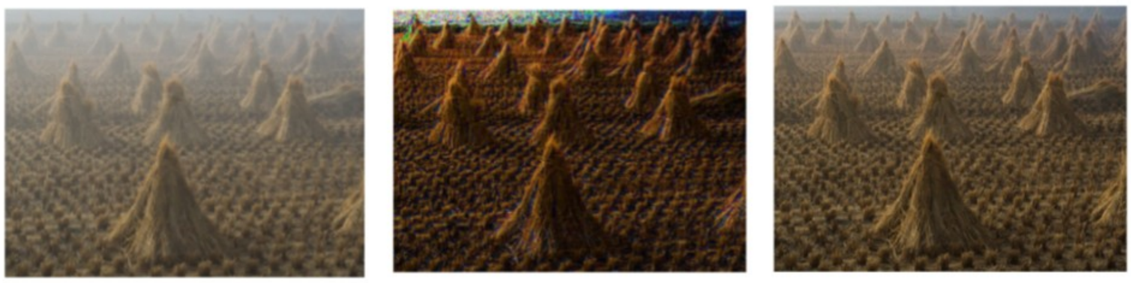


Figure 5.1: Dehazed Cones

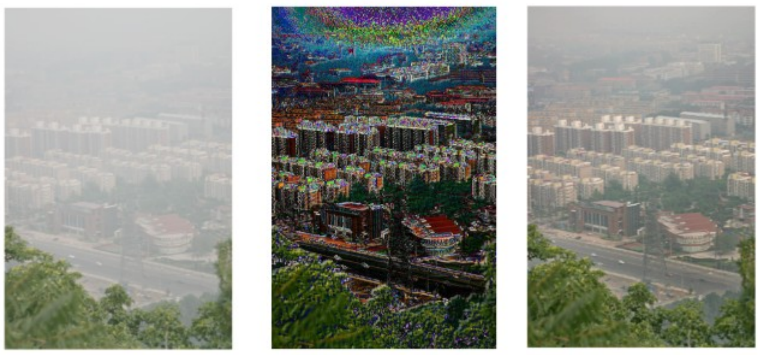


Figure 5.2: Dehazed Cityscape

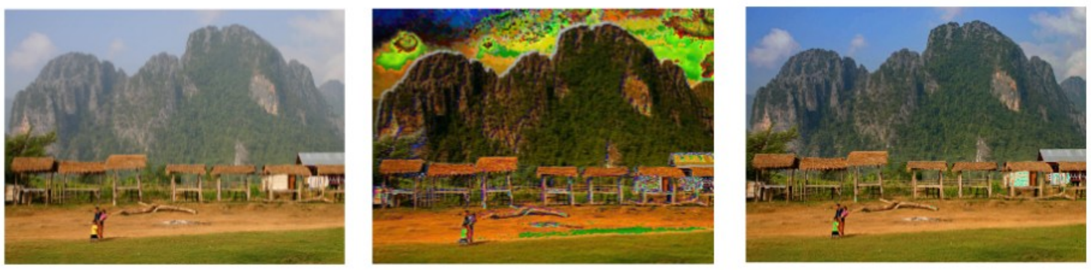


Figure 5.3: Dehazed Mountain

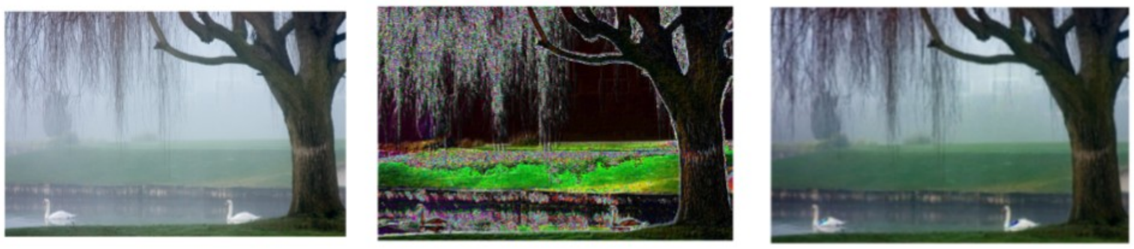


Figure 5.4: Dehazed Swan

Chapter 6

Conclusion and Future Work

6.1 Outcome & Challenges

At the current stage of implementation of the project we have been successful at achieving the following:

1. Speed improvement from approx 40 mins to 2 mins
2. Visibility of distant objects with well-defined boundaries
3. Restoration of original colors of the image

The fast speed along with good color restoration are important goals of our project that we have been able to achieve. There are some more issues that are yet to be addressed. In the algorithm for generate pairs, we are currently randomly selecting some number query patches from the set. This number varies from image to image and is dependent on a number of factors such as total number of patches per scaled image, standard deviation ranges of these patches, their distribution, etc. Thus, we would like to come up with a simple way to determine this number by simply looking at the image.

While we have significantly improved on the original speed, in order for this to become a real-time application, we still need to speed it up for practical use. One approach is to make use of GPUs which results in the code running in a few seconds. However, GPUs are not always a practical solution for real-applications (e.g. on mobile phones). A second approach would thus be to compromise on the exact image restoration, but perfectly determine limited useful information, such as colors or shapes. For all these approaches, it is important that we determine the patches providing maximum information about the haze parameters. We are still searching for ways to achieve this efficiently in minimum time.

Another issue in this solution is that we have hard-coded the number of buckets to be used in the step for generating pairs. Since this is a significantly important part of our algorithm, where we are determining which patches will be queried and which will make up the KD Tree, it is important that we understand exactly which patches are chosen. This selection may vary from image to image. It thus seems important to come up with a solution which tackles this problem of determining the optimal number of regions an image should be divided into, based on certain properties of the image (degree of haze, percentage of visible entities, effect of airlight, depth map, etc.)

6.2 Future Work

The novel idea of using the internal patch recurrence property of natural images can be extended to encompass a number of problems in Computer Vision and Computational Photography. This is currently being explored for different areas such as follows [6]-

6.2.1 Similarity by Composition

Similarity by Composition measures the similarity between 2 images or image patches. This is defined as the ease with which an image (or image patch) can be converted to another. The smaller the patches are, the easier it is to obtain one from the other. Therefore, we look for pairs of images with greater regions of similarity, i.e. larger matching regions or patches.

6.2.2 Videos

This involves detecting the deviation in the patch recurrence property temporally (in time) rather than spatially. It is through these deviations that the action is identified. This is called Video Clustering.

All these methods make use of patch recurrence properties of images and the deviation from their normal behaviors. We look forward to further exploring this field and extending this idea of Patch Recurrence to other domains of Computer Vision.

Bibliography

[1] Yuval Bahat and Michal Irani. Blind dehazing using internal patch recurrence. In *Computational Photography (ICCP), 2016 IEEE International Conference on*, pages 1–9. IEEE, 2016.

[2] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics-TOG*, 28(3):24, 2009.

[3] Raanan Fattal. Single image dehazing. *ACM transactions on graphics (TOG)*, 27(3):72, 2008.

[4] Raanan Fattal. Dehazing using color-lines. *ACM transactions on graphics (TOG)*, 34(1):13, 2014.

[5] Kaiming He, Jian Sun, and Xiaoou Tang. Single image haze removal using dark channel prior. *IEEE transactions on pattern analysis and machine intelligence*, 33(12):2341–2353, 2011.

[6] Michal Irani. blind visual inference by composition. *Pattern Recognition Letters*, 2017.

[7] Srinivasa G Narasimhan and Shree K Nayar. Chromatic framework for vision in bad weather. In *Computer Vision and Pattern Recognition,*

2000. Proceedings. IEEE Conference on, volume 1, pages 598–605. IEEE, 2000.

[8] Srinivasa G. Narasimhan and Shree K. Nayar. Contrast restoration of weather degraded images. *IEEE transactions on pattern analysis and machine intelligence*, 25(6):713–724, 2003.

[9] Shree K Nayar and Srinivasa G Narasimhan. Vision in bad weather. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 820–827. IEEE, 1999.

[10] Yoav Y Schechner, Srinivasa G Narasimhan, and Shree K Nayar. Instant dehazing of images using polarization. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001.

[11] Sarit Shwartz, Einav Namer, and Yoav Y Schechner. Blind haze separation. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1984–1991. IEEE, 2006.

[12] Matan Sulami, Itamar Glatzer, Raanan Fattal, and Mike Werman. Automatic recovery of the atmospheric light in hazy images. In *Computational Photography (ICCP), 2014 IEEE International Conference on*, pages 1–11. IEEE, 2014.

[13] Robby T Tan. Visibility in bad weather from a single image. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[14] Ketan Tang, Jianchao Yang, and Jue Wang. Investigating haze-relevant features in a learning framework for image dehazing. In *Proceedings*

of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2995–3000, 2014.

[15] Qingsong Zhu, Jiaming Mai, and Ling Shao. A fast single image haze removal algorithm using color attenuation prior. *IEEE Transactions on Image Processing*, 24(11):3522–3533, 2015.