Maxime Plumejeau
ID : 92454759
2024/05/01

Exercise : Single image dehazing

Single image dehazing in computer vision is a process aiming at reducing or removing the haze of a hazed picture without any other modification. This complex task can be solved using several algorithms. In this report, an implementation of the dark channel prior technique is presented.

In the first part, the goal of the exercise is presented, then, the experimental method is explained. The obtained results are presented in a third part. After that, a discussion is conducted about the results, and a conclusion ends this report.

I - Goal

The primary objective of single image dehazing is to remove haze from a single input image. Haze, often caused by particles like dust, smoke, and water droplets in the air, reduces visibility and degrades the quality of images. Effective haze removal enhances the clarity and visibility of images, which is crucial for various applications, including computer vision tasks, remote sensing, and photography.

The dark channel prior (DCP) is a widely used technique for single image dehazing in computer vision. The theory behind DCP is based on a statistical observation of outdoor haze-free images. The dark channel of an image is calculated by taking the minimum intensity value across the RGB channels in a local patch for each pixel. In most haze-free images, some pixels in these patches (excluding sky regions) have very low intensity in at least one color channel. These pixels are referred to as the "dark pixels".

In hazy images, these dark pixels become brighter due to the scattering of light by atmospheric particles. By leveraging this observation, the dark channel prior helps in estimating the haze transmission map and the atmospheric light, which are then used to recover the haze-free image.

II - Method

The dehazing process using dark channel prior involves several key steps :

- calculating the dark channel of the input image
- estimating the atmospheric light
- estimating the transmission map
- refining the transmission map using guided filtering
- recovering the haze-free image

Let’s detail the different steps.

a) Dark channel prior calculation

The first step is to compute the dark channel of the image. This is done by taking the minimum value across the RGB channels in a local patch around each pixel using appropriate functions from OpenCV library.

```python
# Calculate the dark channel of the image
def dark_channel_prior(image, size=15):
    min_img = cv2.min(cv2.min(image[:, :, 0], image[:, :, 1]), image[:, :, 2])
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (size, size))
    dark_channel = cv2.erode(min_img, kernel)
    return dark_channel
```

Figure 1 : dark_channel_prior function

The *dark_channel_prior* function takes the RGB image and the size of the local patch as inputs and returns the dark channel of the image, allowing to accurately estimate the haze quantity in the original image.

b) Atmospheric light estimation

Next, the script estimates the atmospheric light, which is the light scattered by atmospheric particles. This is done by selecting the brightest pixels in the dark channel.

```python
# Estimate the atmospheric light in the image
def atmospheric_light(image, dark_channel):
    flat_image = image.reshape(-1, image.shape[2])
    flat_dark = dark_channel.ravel()
    search_idx = (-flat_dark).argsort()[:int(0.001 * len(flat_dark))]
    atmospheric_light = np.mean(flat_image[search_idx], axis=0)
    return atmospheric_light
```

Figure 2 : atmospheric_light function

The *atmospheric_light* function takes the original RGB image and the dark channel image as inputs and returns a 3-element vector representing the brightness of the atmospheric light, respectively for the 3 components of the RGB image.

c) Transmission map estimation

The transmission map represents the portion of the light that is not scattered and reaches the camera. It is estimated using the normalized image and the dark channel prior.

```python
# Estimate the transmission map of the image
def transmission_estimate(image, atmospheric_light, omega=0.95, size=15):
    norm_image = image / atmospheric_light
    transmission = 1 - omega * dark_channel_prior(norm_image, size)
    return transmission
```

Figure 3 : transmission_estimate function

The *transmission_estimate* function takes the original RGB image, the vector representing the atmospheric light previously computed and 2 parameters, omega and size, as inputs. The parameter omega controls the quantity of estimated haze while the parameter

size represents the size of the local patch used for computing the dark channel. The function returns an image representing the estimated light transmission in the original image.

d) Transmission map refinement

The initial estimated light transmission map is then refined using a guided filter to preserve edges and details.

```
# Apply a guided filter to refine the transmission map
def guided_filter(image, p, radius=60, eps=1e-3):

    mean_I = cv2.boxFilter(image, cv2.CV_64F, (radius, radius))
    mean_p = cv2.boxFilter(p, cv2.CV_64F, (radius, radius))
    mean_Ip = cv2.boxFilter(image * p, cv2.CV_64F, (radius, radius))
    cov_Ip = mean_Ip - mean_I * mean_p

    mean_II = cv2.boxFilter(image * image, cv2.CV_64F, (radius, radius))
    var_I = mean_II - mean_I * mean_I

    a = cov_Ip / (var_I + eps)
    b = mean_p - a * mean_I

    mean_a = cv2.boxFilter(a, cv2.CV_64F, (radius, radius))
    mean_b = cv2.boxFilter(b, cv2.CV_64F, (radius, radius))

    q = mean_a * image + mean_b
    return q
```

Figure 4 : guided_filter function

The *guided_filter* function takes the original image converted into grayscale, the estimated light transmission map previously computed, and 2 parameters allowing to control the smoothing effect of the filter, radius and eps, as inputs. The function returns a refined map of the light transmission estimation in the original image.

e) Haze-free image recovery

Finally, the haze-free image is recovered using the refined transmission map and the estimated atmospheric light.

```
# Recover the image by removing haze
def recover(image, t, A, t0=0.1):
    t = np.maximum(t, t0)
    t = t[:, :, np.newaxis]  # Add a new axis to match image dimensions
    J = (image - A) / t + A
    J = np.clip(J, 0, 1)
    return J
```

Figure 5 : recover function

The *recover* function takes the original image, the refined transmission map, the estimated atmospheric light and the parameter t0 as inputs. The parameter t0 controls the limitation of noise amplification in the reconstructed image. The function returns a map allowing to reconstruct the dehaze image.

f) Dehaze function

The *dehaze* function integrates all the steps to dehaze the input image.

```python
# Main function to remove haze from an image
def dehaze(image, size, omega, t0):

    image = image.astype('float64') / 255
    dark_channel = dark_channel_prior(image, size)
    A = atmospheric_light(image, dark_channel)
    t_estim = transmission_estimate(image, A, omega, size)

    # Convert image to uint8 with cv2.cvtColor
    gray_image = cv2.cvtColor((image * 255).astype('uint8'), cv2.COLOR_BGR2GRAY)
    t_guided = guided_filter(gray_image, t_estim)

    J = recover(image, t_guided, A, t0)
    return (J * 255).astype('uint8'), dark_channel, A, t_estim, t_guided
```

Figure 6 : dehaze function

g) Parameters variation

Since the optimal values of size, omega and t0 parameters differ from one haze image to another, the script implements the possibility to try the dark channel algorithm for 3 different values of each 3 parameters, around the usual values used for them. Hence, 27 images are obtained, allowing to slightly refine the optimal dehaze image. Once it's not the optimal solution, it is complicated to refine more due to the slight variation in result image for a parameter variation and the great number of parameters used by the algorithm.

```python
# compute and store 27 dehazed images with variations in size, omega and t0 parameters

for size in range(5, 20, 5):

    for omega in range(85, 100, 5):

        for t0 in range(10, 25, 5):

            dehazed_image, dark_channel, atmo_light, t_estim, t_guided = dehaze(image, size, omega/100, t0/100)

            output_path_real = output_path + '/' + str(size) + '_' + str(omega) + '_' + str(t0) + '.jpg'

            cv2.imwrite(output_path_real, dehazed_image)
```

Figure 7 : parameters size, omega and t0 variation

III - Results

This section presents the results of the different parts of the script, following the order of the functions presented previously.

a) Partial results

In this first part, let's see the partial results of the different functions of the script.

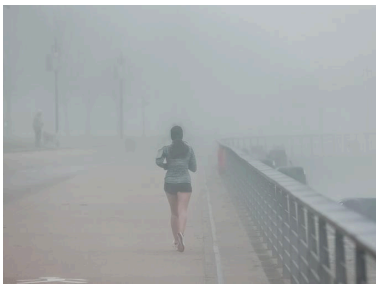
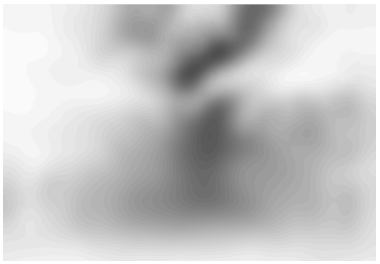
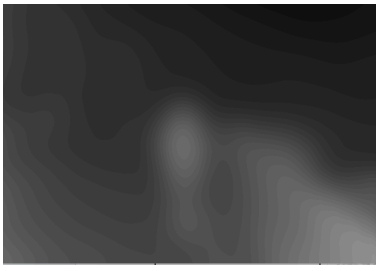
original image		
dark_channel_prior		
transmission_estimate		
guided_filter		

Figure 8 : Original images and results of dark_channel_prior, transmission_estimate and guided_filter functions for 2 haze images

b) Without parameters variations

In this second part, let's analyze the results of the dehaze algorithm for several images with the usual values for the parameters :

- size = 15
- omega = 0.9
- t0 = 0.15

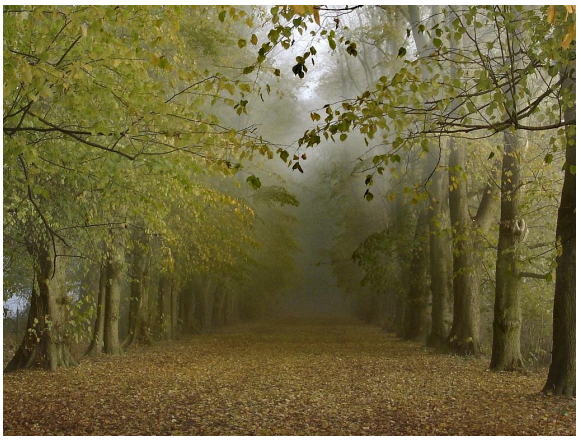
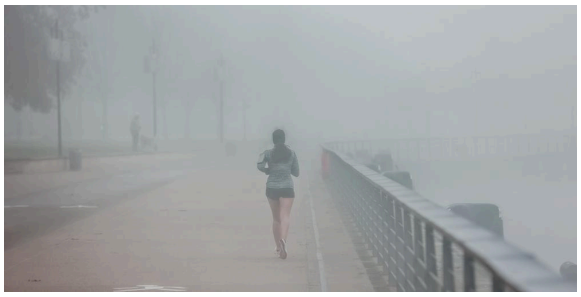
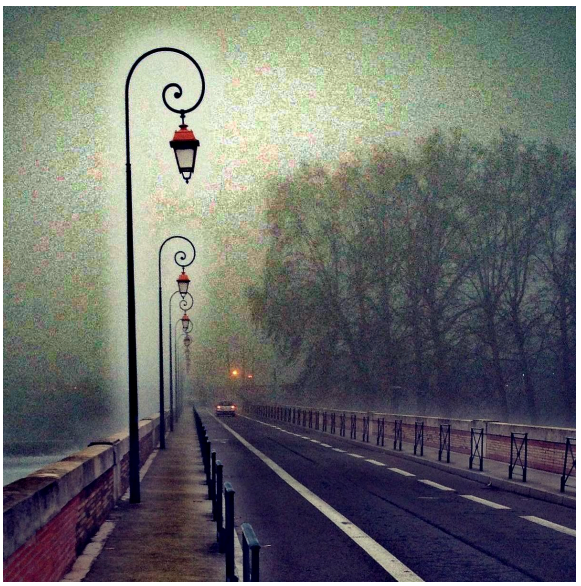
Original haze image	Reconstruct dehaze image
	
	
	

Figure 8 : Results of the script for several haze images

In the following, these images will be respectively referred to as *forest*, *runner*, *bridge* and *road*. These 4 dehazed result images are representative of the different outputs obtained using the script. In some cases (*forest*), the dehazing is successful and the resulting image is in the same visual quality as the original one. However, in other cases, the dehazing implies a noticeable reduction of the visual quality of the image (every image except *forest*) with the apparition of small square blocks, mostly in the sky. Moreover, in some cases, the colors of the dehaze image seem strange, supernatural (*runner* and *road*).

c) With parameters variations

In order to correct the defects presented in the previous part (reduction in image quality and strange colorization), a possibility of variations in 3 parameters (size, omega and t0) has been implemented, as explained in II-g). Hence, the script returns 27 images with slight differences and it is possible to pick the best one for a human eye.

Let's analyze the effect of the 3 parameters using the *road* image.

Figure 9 : Results of the script for variations in parameters size, omega and t0

Each 3x3 block of images share the same value of size parameter, respectively 5, 10 and 15 from up to down. In a 3x3 block, the images in the same line share the same value of omega, respectively 0.85, 0.9 and 0.95 from up to down, and the images in the same column share the same value of t0, respectively 0.1, 0.15 and 0.2 from left to right.

Hence, it is possible to roughly conclude about the influence of each parameter :

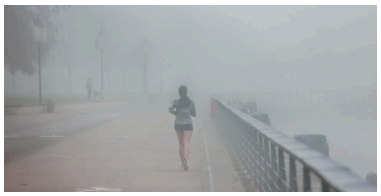
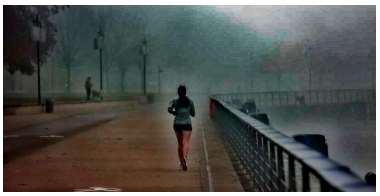
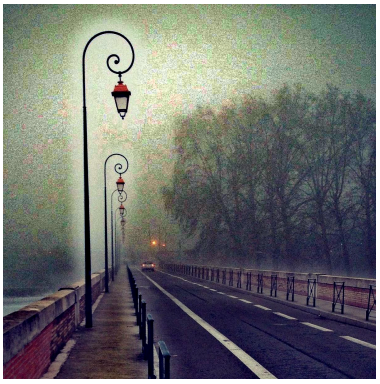
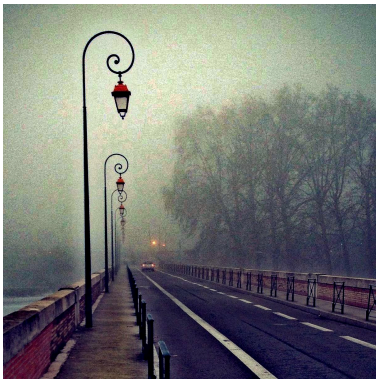
- size : an increase of size involve a augmentation of the brightness of the light transmission of the dehaze image
- omega : an increase of omega involve an augmentation of the contrast of the dehaze image
- t0 : an increase of t0 smooth the dehaze image, decreasing the noise but lowering the details

The selection the best solution depends on the purpose of the output dehaze image, however, according to the human eye criteria, it is possible to say that the optimal parameters for this image are :

- size = 5 or 10
- omega = 0.85
- t0 = 0.2

These parameters seem to be the best option allowing a dehaze image without too many reduction in quality or details and realistic colors.

It is possible to apply this variation and eye-selection technique to the *runner* and *bridge* images as well to improve the quality of the dehaze images.

Original haze image	Reconstruct dehaze image with usual parameters	Reconstruct dehaze image with selected parameters
		
		
		

Although the result is better using the optimal parameters for each image, there are still problems. The visual quality is still worse than the original image, and some images are not completely dehaze (*bridge* and *road*).

IV - Discussion

In this part, let's discuss the results obtained using the dark channel prior algorithm and analyze why it works better for some images.

According to the results, the dark channel prior method works well for images with significant haze and distinct dark regions (such as *forest*). The method is effective because it leverages the statistical observation that haze-free outdoor images have at least one low-intensity pixel in most local patches. However, the performance of the script may vary based on certain factors :

- Presence of sky regions : sky regions may not have dark pixels, leading to overestimation of transmission and incorrect dehazing in those areas
- Homogeneous regions : in regions with similar intensity values across patches, the dark channel prior might not perform well, resulting in artifacts
- Complex atmospheric light : when the atmospheric light is not uniformly distributed, the estimation might be less accurate, affecting the overall dehazing quality
- Artifacts and halo effects : the guided filter helps mitigate artifacts and halo effects but might not eliminate them completely, especially around edges

V - Conclusion

Single image dehazing using the dark channel prior is a powerful technique that significantly improves the visibility and clarity of hazy images. The method leverages the statistical properties of natural haze-free images to estimate the haze and recover the original image. While the technique is effective for many outdoor scenes, it has limitations in handling sky regions, homogeneous areas, and complex atmospheric light distributions. Future improvements could focus on enhancing the accuracy of atmospheric light estimation and refining the transmission map to further reduce artifacts and improve image quality.