DIP Final Report

Team4
r11943018 林子軒、r11943022 范詠為

Division of the work
r11943018 林子軒：paper survey(50%), NN method reproduce(100%), report(60%)
r11943022 范詠為：paper survey(50%), DCP algorithm code(100%), report(40%)

About the code
github link: https://github.com/BrianEE07/2023_DIP_Final
To run our code, please refer to README.md
For the input argument, please refer to the line 7~14 in src/dehaze.py

Paper title
Single Image Haze Removal Using Dark Channel Prior

Motivation

When traveling in the mountains and forests, the photos often appear hazy and unclear. By using image dehazing, we can restore the pictures to their original clarity as much as possible.

Single Image Haze Removal is a computer vision technique that eliminates haze, enhancing image quality for surveillance, aerial photography, and computer vision. The Dark Channel Prior approach is popular, effective, and widely used. Implementing it enables leveraging existing research and codebase while gaining practical experience in image enhancement and optimization.

Problem definition

The problem of image haze removal refers to the task of improving the visibility and appearance of a hazy or foggy image by eliminating the atmospheric haze or fog present in the image. The goal is to enhance the details, colors, and overall quality of the image while producing a visually pleasing and natural-looking haze-free result.

Algorithm

In computer vision and computer graphics, the model widely used to describe the formation of a hazy image is:

$$\mathbf{I}(\mathbf{x}) = \mathbf{J}(\mathbf{x})t(\mathbf{x}) + \mathbf{A}(1 - t(\mathbf{x})),$$

which is often called the haze image equation, where I is the observed intensity, J is the scene radiance, A is the global atmospheric light, and t is the medium transmission describing the portion of the light that is not scattered and reaches the camera.

The dark channel prior is a technique that relies on a common feature of outdoor images without haze. Specifically, in most areas of the image that are not sky, at least one color channel will contain some pixels with very low intensity values that are nearly zero. This means that the minimum intensity in these areas is also very low.

haze-free image dark channel

So, based on the observation and the statistics, we first normalize the haze imaging equation by A.

$$\frac{I^c(\mathbf{x})}{A^c} = t(\mathbf{x})\frac{J^c(\mathbf{x})}{A^c} + 1 - t(\mathbf{x}).$$

Then, we calculate the dark channel on both sides.

$$\min_{\mathbf{y}\in\Omega(\mathbf{x})}\left(\min_c \frac{I^c(\mathbf{y})}{A^c}\right) = \tilde{t}(\mathbf{x}) \min_{\mathbf{y}\in\Omega(\mathbf{x})}\left(\min_c \frac{J^c(\mathbf{y})}{A^c}\right) + 1 - \tilde{t}(\mathbf{x}).$$

As the scene radiance J is a haze-free image, the dark channel of J is close to zero due to the dark channel prior

$$J^{\text{dark}}(\mathbf{x}) = \min_{\mathbf{y}\in\Omega(\mathbf{x})}\left(\min_c J^c(\mathbf{y})\right) = 0.$$

Therefore, we can estimate the transmission t simply by

$$\tilde{t}(\mathbf{x}) = 1 - \min_{\mathbf{y}\in\Omega(\mathbf{x})}\left(\min_c \frac{I^c(\mathbf{y})}{A^c}\right).$$

With the atmospheric light and the transmission map, we can recover the scene radiance.

$$\mathbf{J}(\mathbf{x}) = \frac{\mathbf{I}(\mathbf{x}) - \mathbf{A}}{\max(t(\mathbf{x}), t_0)} + \mathbf{A}.$$

The implementation of the DCP algorithm pipeline including following steps:

1. Preprocessing: Scales pixel values to 0~1.

2. Dark channel prior: Calculates the dark channel prior using the input image.

    We first pad the image with a border and then compute the minimum channel value within each patch of the padded image. The resulting dark channel represents the minimum intensity value across these patches and is returned as the output.

```python
def dark_channel_prior(img, patch_size=15):
    H, W, C = img.shape
    # pad image with border value
    img_pad = cv2.copyMakeBorder(img, patch_size//2, patch_size//2, patch_size//2, patch_size//2, cv2.BORDER_REPLICATE)
    # get min channel value of padded image
    img_pad = np.min(img_pad, axis=2)
    # dark channel
    dark_channel = np.zeros((H, W), dtype=np.float32)
    for i in range(H):
        for j in range(W):
            dark_channel[i, j] = np.min(img_pad[i:i + patch_size, j:j + patch_size])

    return dark_channel
```

3. Atmospheric light estimation: Estimates the atmospheric light using the dark channel prior and the input image.

    We calculate the number of pixels in the dark channel and determine the top "n" pixels based on a given percentage (default is 0.001). The dark channel and image are flattened, and the dark channel values are sorted in descending order. The atmospheric light is then obtained by selecting the maximum pixel values from the top "n" pixels in each RGB channel. The estimated atmospheric light is returned as the output.

```python
def atmospheric_light_estimation(dark_channel, img, percent=0.001):
    H, W = dark_channel.shape
    # get number of pixels
    num_pixels = H * W
    # get top n pixels
    top_n = int(max(num_pixels * percent, 1))
    # sort dark channel in descending order
    img_flat = img.reshape(-1, 3)
    dark_channel_flat = dark_channel.reshape(-1)
    dark_channel_flat_sorted_indices = np.argsort(dark_channel_flat)[::-1]
    # get atmospheric light from top n pixels in each RGB channel
    atmospheric_light = np.max(img_flat[dark_channel_flat_sorted_indices[:top_n]], axis=0, keepdims=True)

    return atmospheric_light
```

4. Transmission map estimation: Estimates the transmission map using the input image, atmospheric light, and a specified omega parameter.

    We first divide the image by the atmospheric light in each RGB channel. Then, we calculate the dark channel using the "dark_channel_prior" function with a specified patch size. Finally, the transmission map is obtained by subtracting the product of the dark channel and a parameter "omega" from 1 (omega is set to 0.95 referring to the paper). The resulting transmission map is returned as the output.

```python
def transmission_map_estimation(img, atmospheric_light, omega=0.95, patch_size=15):
    # divide image by atmospheric light in each RGB channel
    img = img / atmospheric_light
    # dark channel prior
    dark_channel = dark_channel_prior(img, patch_size)
    # transmission map
    transmission_map = 1 - omega * dark_channel

    return transmission_map
```

5. Refine transmission map: Refines the transmission map using either a guided filter or soft matting technique.

Soft_matting:

We start by creating a binary map of constraints based on the image shape. Prior confidence is then computed using a specified lambda value (default is 0.0001). The "closed_form_matting_with_prior" (obtained from open source code) function is called to obtain a refined transmission map, considering the image, transmission map, prior confidence, and constraints map. The resulting refined transmission map is further improved using bilateral filtering with specified parameters. Finally, the refined transmission map is returned as the output.

```
def soft_matting(img, transmission_map, lambda_=1e-4):
    # soft-matting
    consts_map = np.zeros((img.shape[0], img.shape[1]), dtype='bool')
    prior_confidence = lambda_ * np.ones((img.shape[0], img.shape[1]), dtype=np.float32)
    refined_transmission_map = closed_form_matting_with_prior(image=img, prior=transmission_map, prior_confidence=prior_confidence,
    # bilateral filtering
    refined_transmission_map = cv2.bilateralFilter(refined_transmission_map, 20, 50, 50)

    return refined_transmission_map
```

Guided_filter:

Based on the standard guided filter algorithm. We prepare the guidance and input images, calculating mean, covariance, and variance values, determining filter coefficients, and applying the coefficients to obtain the filtered output.

```
def guided_filter(img, p, r=60, eps=0.0001):
    mean_I = cv2.boxFilter(img,cv2.CV_64F,(r,r))
    mean_p = cv2.boxFilter(p, cv2.CV_64F,(r,r))
    mean_Ip = cv2.boxFilter(img*p,cv2.CV_64F,(r,r))
    cov_Ip = mean_Ip - mean_I*mean_p

    mean_II = cv2.boxFilter(img*img,cv2.CV_64F,(r,r))
    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,(r,r))
    mean_b = cv2.boxFilter(b,cv2.CV_64F,(r,r))

    q = mean_a*img + mean_b

    return q
```

6. Scene radiance reconstruction: Reconstructs the scene radiance using the input image, refined transmission map, and atmospheric light.

   Based on the last equation in above part, we calculated the scene_radiance as below.

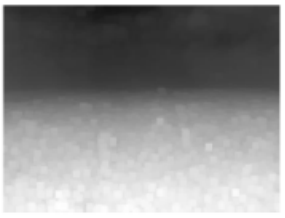
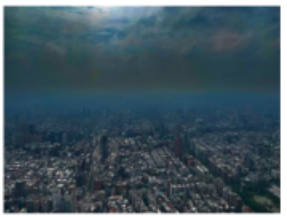
```
def scene_radiance_reconstruction(img, transmission_map, atmospheric_light):
    H, W, C = img.shape
    # scene radiance
    scene_radiance = np.zeros_like(img, dtype=np.float32)
    t = cv2.max(transmission_map,0.1)
    for i in range(C):
        scene_radiance[:, :, i] = (img[:, :, i] - atmospheric_light[0, i]) / t + atmospheric_light[0, i]

    return scene_radiance
```

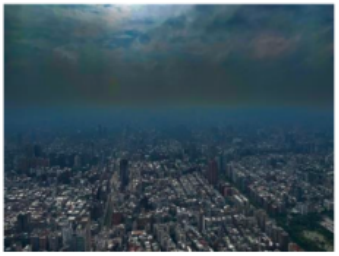
Experimental results

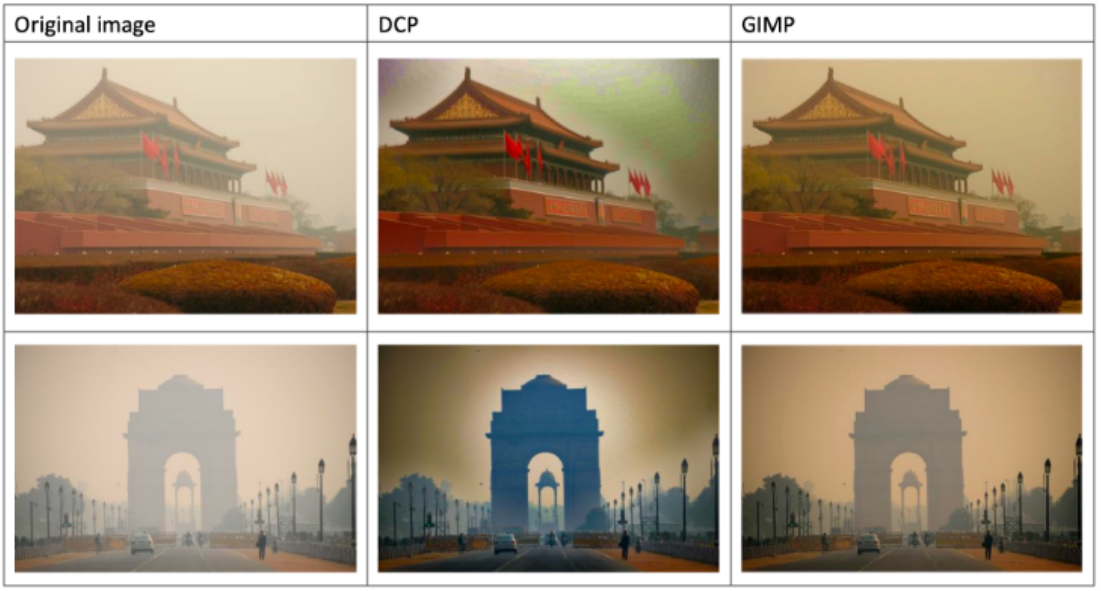
1. Qualitative result

image	dark channel	transmission map	result
			
		refine	result
			

image	dark channel	transmission map	result
			
		refine	result
			

Apart from DCP, we also use an online NN dehazing method called GMIP, both results are shown below.

Original image	DCP	GIMP
		
		



2. Quantitative result (averaged on 20 images sampled from RESIDE dataset)

	Org PSNR	PSNR(↑)	Org SSIM	SSIM(↑)
DCP	14.8835	**20.0845**	0.8382	**0.9275**
GIMP		17.8781		0.9044

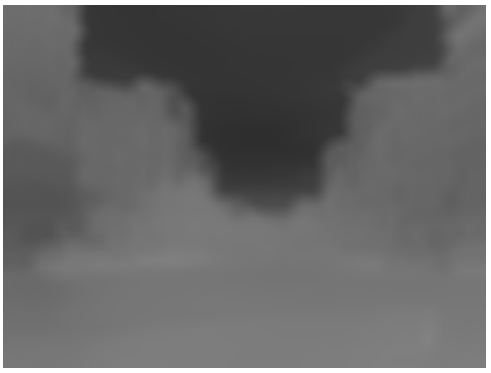
Discussions

1. DCP algorithm
    a. transmission map refinement (use guided filter)

city_0	original	refined
transmission map		
result dehazed image		

We can observe that without refining the transmission map, the dehazed image may exhibit a halo effect, which appears unnatural.

b. refining method

0025_heavy	guided filter	soft matting
original image		
refined transmission map		
result dehazed image		
PSNR(↑)	19.8546	20.0480
SSIM(↑)	0.9395	0.9423
speed(s)	1.079	24.808

We found that using the soft-matting approach for refinement, although the transmission map may be blurrier, the dehazed image in the sky region appears more natural. Additionally, it performs slightly better in terms of PSNR and SSIM values compared to using the guided filter. However, soft-matting approach requires much more computational time than guided filter, so we prefer the latter without sacrificing too much performance.

c. heavy vs. light haze (use guided filter)

	0025_light.jpg	0025_heavy.jpg
original image		
refined transmission map		
result dehazed image		
PSNR(↑)	18.9198	19.8546
SSIM(↑)	0.9311	0.9395

Regarding the comparison of the haze removal results between dense fog and light fog using the DCP algorithm, we were surprised to find that despite the slight overexposure in the dehazed images of dense fog, they achieved higher PSNR and SSIM values compared to the results obtained from light fog. In conclusion, removing haze too aggressively may not necessarily improve the PSNR and SSIM values.

d. speed (s) (on 2018 macbook pro)

guided filter	(400, 600)	(800, 1200)	(1200, 1800)
1.preprocess	0.006	0.024	0.096
2.dcp	**1.358**	**6.700**	**12.345**
3.estimate A	0.006	0.030	0.078
4.estimate tmap	**1.345**	**5.361**	**12.236**
5.refine tmap	0.021	0.080	0.180
6.scene reconstruction	0.007	0.023	0.095
total	2.743	12.219	25.032

soft matting	(400, 600)	(800, 1200)	(1200, 1800)
1.preprocess	0.035	0.047	0.060
2.dcp	1.628	5.501	11.832
3.estimate A	0.010	0.034	0.076
4.estimate tmap	1.521	5.492	11.884
5.refine tmap	**30.066**	**337.704**	**1122.611**
6.scene reconstruction	0.012	0.148	0.446
total	33.273	348.926	1146.909

Regarding the computational speed, the guided filter performs refinement calculations very quickly. Therefore, the bottleneck lies in the computation of the DCP algorithm, estimated to have a complexity of O(h × w), where h and w represent the height and width of the image, respectively. On the other hand, soft-matting refinement becomes the bottleneck for the entire pipeline. As the image size increases, the computational time of the pipeline with soft-matting refinement increases faster compared to the pipeline that uses the guided filter.

e. parameter - dcp window size

w	dcp	refined tmap	dehazed image
5			
15			
25			

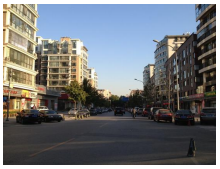
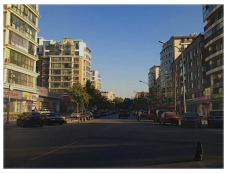
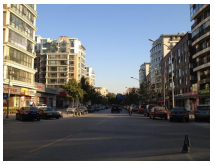
It can be observed that as the window size of the DCP algorithm increases, although the refined transmission map becomes less clear, the dehazed images do not suffer from excessive haze removal and overly strong contrast. Therefore, we ultimately **chose a window size of w=25** and deemed it to produce the best results.

f. parameter - guided filter window size

w	dcp	refined tmap	dehazed image
5			
20			
60			

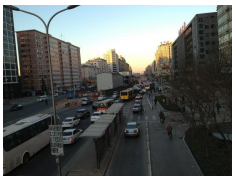
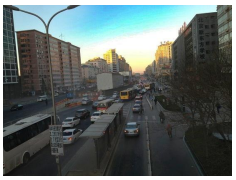
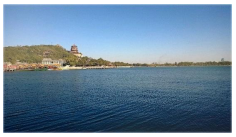
It can be observed that as the window size of the guided filter decreases, the guidance effect from the grayscale image of the original image becomes poorer. This results in the presence of halo artifacts in the final results. Therefore, we ultimately **chose a window size of w=60** for the guided filter.

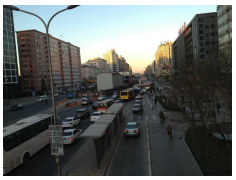
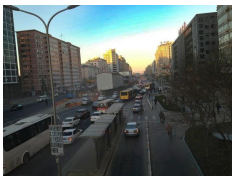
2. Compare between DCP and GIMP
    a. DCP vs. GIMP

	original image	ground truth	DCP		GIMP	
0025 light				psnr **18.920** ssim **0.931**		psnr 18.411 ssim 0.918
0025 heavy				psnr **19.855** ssim **0.940**		psnr 15.417 ssim 0.889

In the experimental results section, it can be observed that despite considering DCP as cleaner in other areas, GIMP yields a more natural effect in the sky region compared to DCP. DCP, on the other hand, exhibits color deviations. A drawback of GIMP, however, is its poorer performance in handling dense fog. In terms of the analysis of PSNR and SSIM, averaging multiple values from images with ground truth, as shown in the above example, the PSNR and SSIM values of GIMP are significantly lower than those of DCP. Nevertheless, we still believe that there exists a better neural network model that can surpass the traditional DCP algorithm.

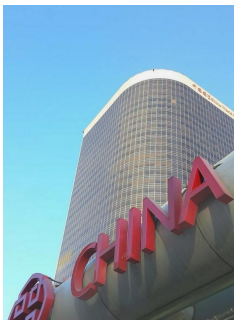
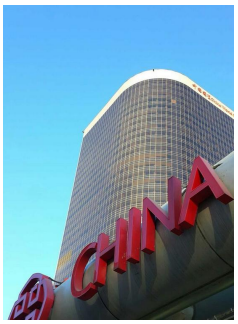
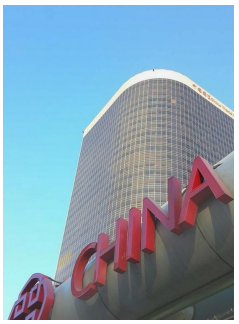
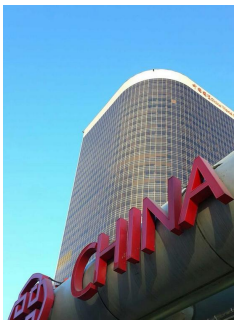
b. Most improved result

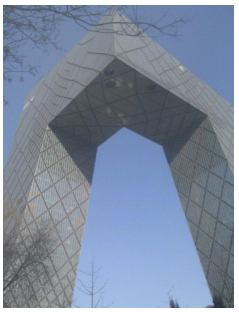
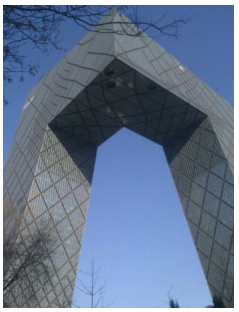
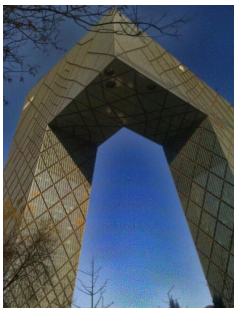
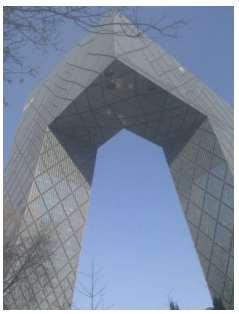
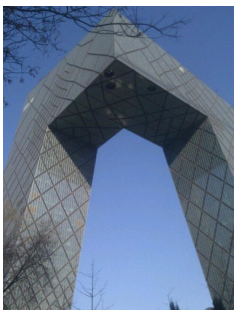
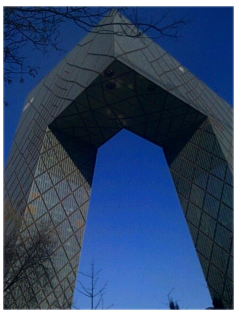
PSNR	original image	ground truth	dehazed image	improved PSNR
DCP (1048_heavy)				**14.4046**
GIMP (0352_heavy)				11.5833

SSIM	original image	ground truth	dehazed image	improved SSIM
DCP (1048_heavy)				**0.3113**
GIMP (1048_heavy)				0.2475

It can be observed that in terms of the most significant improvements in PSNR and SSIM individually, the 1048_heavy image shows considerable progress for both methods. However, we believe that DCP performs better in terms of its cleaner fog removal effect.

c. Most degraded

PSNR	original image	ground truth	dehazed image	degraded PSNR
DCP (0930_light)				**-6.7463**
GIMP (0930_light)				-8.4130

SSIM	original image	ground truth	dehazed image	degraded SSIM
DCP (0801_light)				**-0.0800**
GIMP (0801_light)				-0.1387

In terms of the aspects with the most significant degradation, the images from both methods are the same. We believe that if there are color deviations from the ground truth after removing fog from large-scale buildings, it can lead to a decrease in the PSNR and SSIM scores. Overall, the images that perform well and those that perform poorly are quite similar, and there is a relatively larger improvement in dense fog conditions (the scores in

light fog may be higher than in dense fog but possibly lower than the scores before fog removal).

Reference

main algorithm

He, Kaiming, Jian Sun, and Xiaoou Tang. "Single image haze removal using dark channel prior." *IEEE transactions on pattern analysis and machine intelligence* 33.12 (2010): 2341-2353.

survey-type

Agrawal, Subhash Chand, and Anand Singh Jalal. "A comprehensive review on analysis and implementation of recent image dehazing methods." *Archives of Computational Methods in Engineering* 29.7 (2022): 4799-4850.

RESIDE dataset

Li, Boyi, et al. "Benchmarking single-image dehazing and beyond." *IEEE Transactions on Image Processing* 28.1 (2018): 492-505.

GIMP

Soman, Kritik. "GIMP-ML: python plugins for using computer vision models in GIMP." *arXiv preprint arXiv:2004.13060* (2020).
https://github.com/kritiksoman/GIMP-ML/blob/master/docs/REFERENCES.md