Dehazing Notebook

Benjamin Poulin

Table of contents

Preface

Notebook on dehazing progress.

1 Bibliography

Dehazing is a wide class of image processing problems, with several categories of solution:

- **Model-based** dehazing uses a model of light propagation into the camera sensor (an Image Formation Model, see [1] and [2]) in conjunction with a statistical prior about the distribution of intensities in images without haze. The success of model-based methods depends significantly on the quality of their priors, which are therefore designed to apply to as restricted a class of images as possible.
- **Model-free** dehazing uses more traditional image processing techniques to improve the quality of hazy images (histogram equalization, color correction, etc.).
- **Machine learning** solutions usually run CNNs on hazy images to predict the ground truth haze-free scene.
- Finally, some systems use **specialized hardware** like polarizing filters, lasers, or multiple camera sensors to gain extra information about a scene before dehazing. I will not consider these.

Thus far, I believe these algorithms represent good starting points to test dehazing performance:

Algorithm	Year	Technique	Fluid	Media
DCP [3]	2009	Model-based	Air	Image
UDCP [4]	2013	Model-based	Water	Image
MIP [5]	2010	Model-based	Water	Image
HL [6]	2016	Model-based	Air	Image
HL (underwater) [7]	2021	Model-based	Water	Image
ULAP [8]	2018	Model-based	Water	Image
DCP + TF [9]	2016	Model-based	Air	Video
UWCNN [10]	2020	Learning	Water	Video
WaterNet [11]	2020	Learning	Water	Image
UIEC^2-Net [12]	2021	Learning	Water	Image

Some of these models were selected for their high performance and recent publication, but some were selected because they represent baseline algorithms against which to compare other

models. I still need to identify pipelines that fit these criteria in the model-free domain, and to find more recent algorithms which target video dehazing.

Part I

Progress

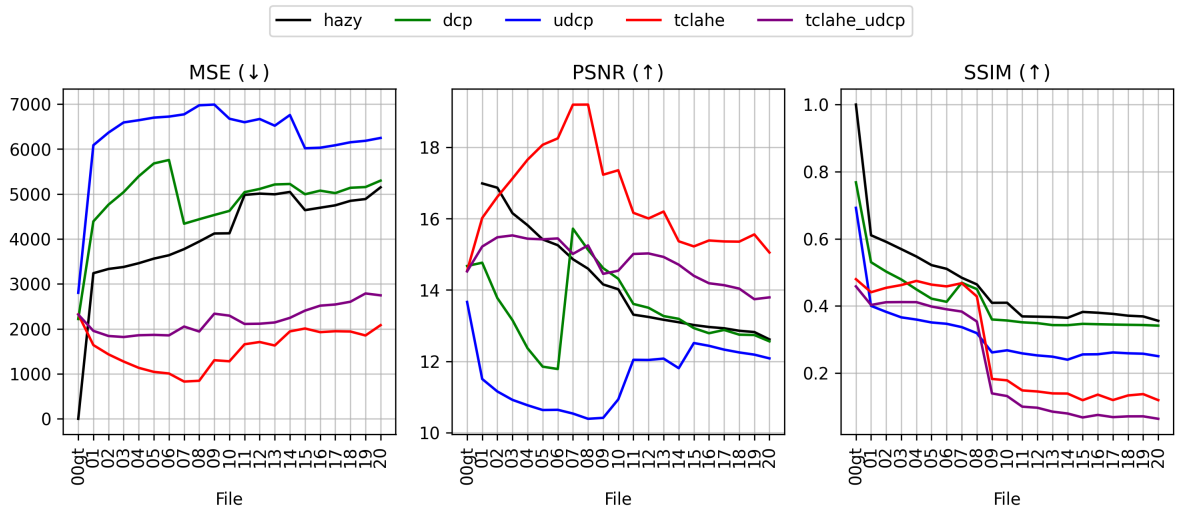
2 Basic metrics

I evaluated output quality on the DeepBlue reference dataset for TCLAHE, DCP, UDCP, and TCLAHE followed by UDCP. I used reference metrics MSE, PSNR, and SSIM, and reference-less metrics entropy and UCIQE. See Figure 2.1.

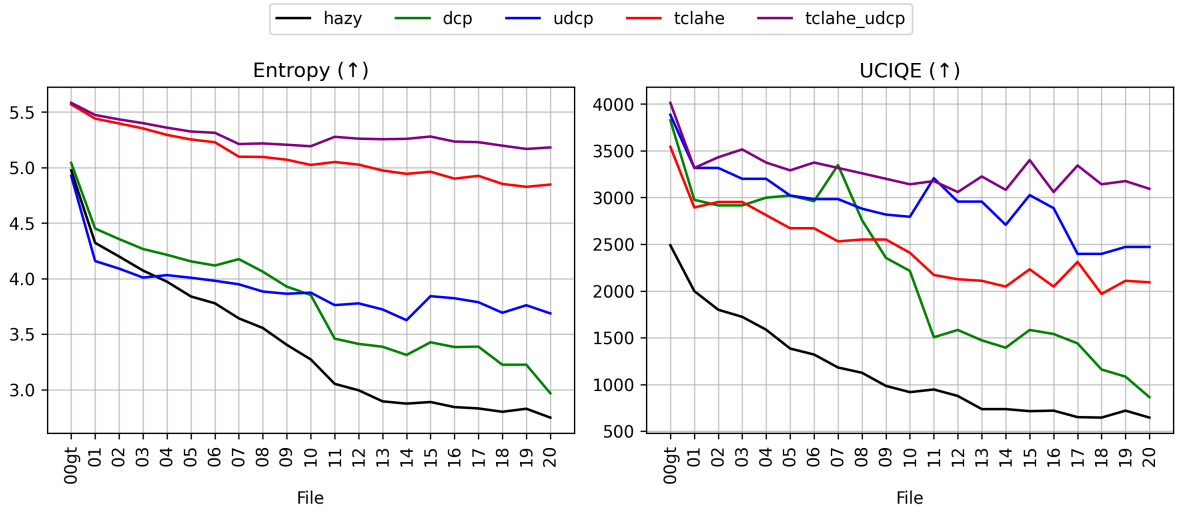
```
import json

PRETTY_METRIC_NAMES = {
  'mse': 'MSE (↓)',
  'psnr': 'PSNR (↑)',
  'ssim': 'SSIM (↑)',
  'entropy': 'Entropy (↑)',
  'uciqe': 'UCIQE (↑)'
}
REF_METRICS = ('mse', 'psnr', 'ssim')
REFLESS_METRICS = ('entropy', 'uciqe')
PIPELINES = ('hazy', 'dcp', 'udcp', 'tclahe', 'tclahe_udcp')
PIPELINE_COLORS = {
  'dcp': 'green',
  'udcp': 'blue',
  'tclahe': 'red',
  'tclahe_udcp': 'purple',
  'hazy': 'black'
}
FILENAMES = ('00gt.png', '01.png', '02.png', '03.png', '04.png', '05.png', '06.png', '07.png
FILENAMES_NO_EXTENSION = [filename[:-4] for filename in FILENAMES]

ref_data = {}
refless_data = {}

def make_metric_data(metrics_json, metric_name: str):
  data = []
  for pipeline in PIPELINES:
    data.append([])
    for filename in FILENAMES:
      data[-1].append(metrics_json[pipeline][filename][metric])
```

```
  return data

with open('../data/metrics.json', 'r') as metrics:
  metrics_json = json.load(metrics)
  for metric in REF_METRICS:
    ref_data[metric] = make_metric_data(metrics_json, metric)
  for metric in REFLESS_METRICS:
    refless_data[metric] = make_metric_data(metrics_json, metric)
```

```
import matplotlib.pyplot as plt

for metrics, data in ((REF_METRICS, ref_data), (REFLESS_METRICS, refless_data)):
  fig, axs = plt.subplots(nrows=1, ncols=len(metrics))
  axs = axs.flatten()
  for metric, ax in zip(metrics, axs, strict=True):
    ax.grid(alpha=0.7)
    ax.tick_params(axis='x', labelrotation=90)
    ax.set(
      xlabel='File',
      # ylabel=PRETTY_METRIC_NAMES[metric],
      title=PRETTY_METRIC_NAMES[metric]
    )
    for series, label in zip(data[metric], PIPELINES, strict=True):
      ax.plot(
        FILENAMES_NO_EXTENSION, series,
        label=label, color=PIPELINE_COLORS[label]
      )

  handles, labels = axs[0].get_legend_handles_labels()
  fig.legend(handles, labels, loc='upper center', ncol=len(PIPELINES), bbox_to_anchor=(0.5,
  fig.set_size_inches(10, 4)
  fig.tight_layout()
  plt.show()
```



(a) Reference metrics



(b) Referenceless metrics

Figure 2.1: Reference and referenceless Metrics for TCLAHE, DCP, UDCP, and TCLAHE -> UDCP on the DeepBlue dataset. Higher filenames are more turbid. `00gt.png` has no turbidity.

See Figure 2.2 for examples of dehazed images.

```
from pathlib import Path
import cv2

BASE_PATH = Path('/home/mantis/Documents/ms/underwater-dehazing-toolkit/data/deepblue')
```

```
FILES = ('02.png', '10.png', '20.png')

fig, axs = plt.subplots(nrows=len(FILES), ncols=len(PIPELINES), sharex=True, sharey=True)
for file, axs_row in zip(FILES, axs, strict=True):
  for pipeline, ax in zip(PIPELINES, axs_row, strict=True):
    image = cv2.imread(str(BASE_PATH / pipeline / file))
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image_shape = image.shape
    ax.imshow(image)
    ax.set_xticks([])
    ax.set_yticks([])
    ax.set_xticklabels([])
    ax.set_yticklabels([])
    ax.set(
      xlabel=pipeline,
      ylabel=file
    )
    ax.label_outer()
fig.subplots_adjust(wspace=0, hspace=0, left=0, right=1, top=1, bottom=0)
subplot_width = 2
subplot_height = (subplot_width / image_shape[1] * image_shape[0])
fig.set_size_inches(
  subplot_width * 5,
  subplot_height * 3
)
plt.show()
```

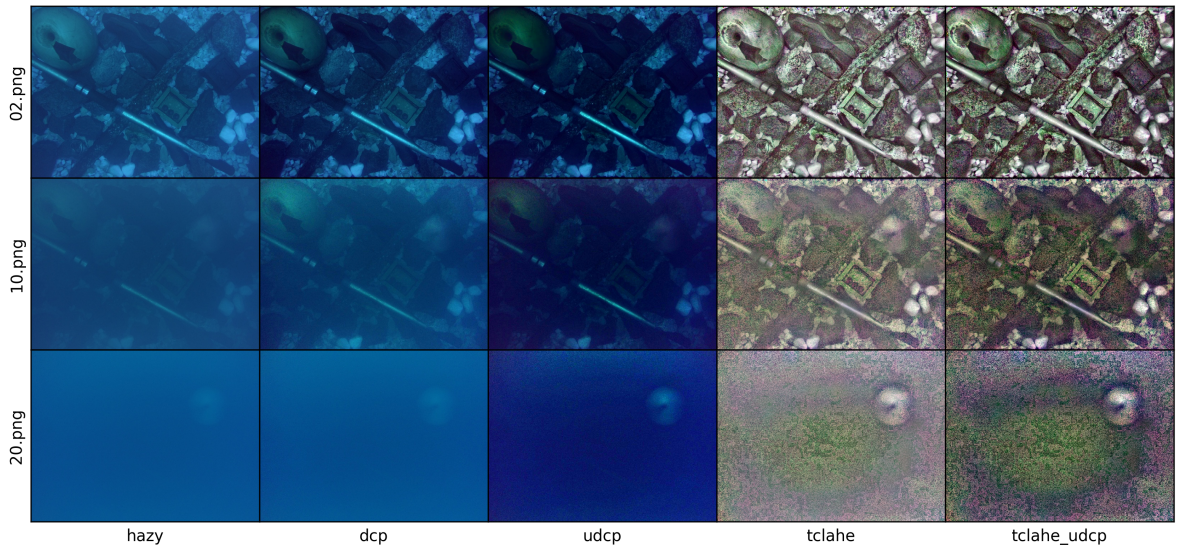


Figure 2.2: Low (`02.png`), medium (`10.png`), and high (`20.png`) turbidity images from the DeepBlue dataset, processed by TCLAHE, DCP, UDCP, and TCLAHE -> UDCP. The `hazy` column are the original images.

TCLAHE and TCLAHE -> UDCP are consistantly the best performing pipelines (barring their low SSIM scores). DCP seems to modify images the least.

3 Next Steps

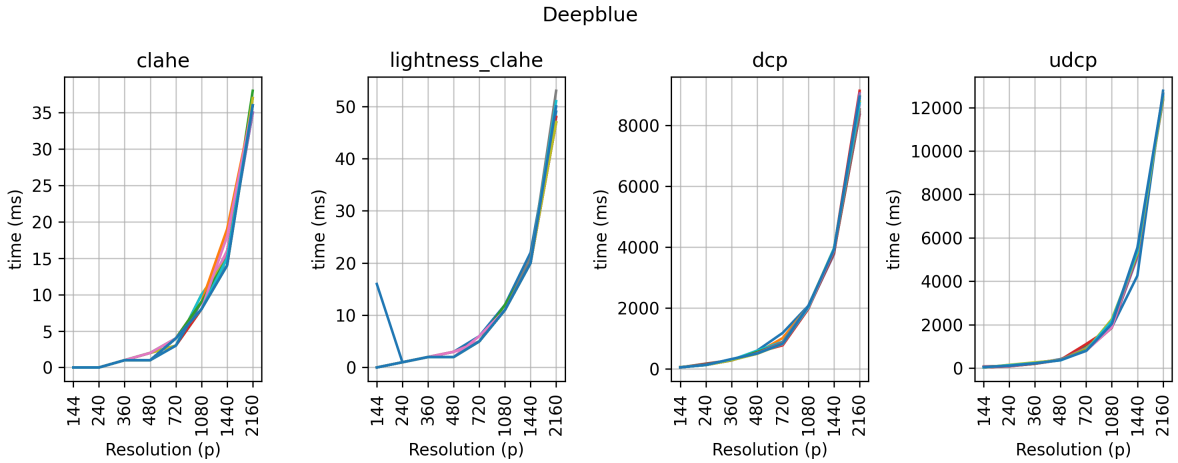
- Measure runtimes
- Add learning pipelines
- Choose which Retinex/CLAHE/etc. model-free pipelines to use

4 Runtime Trials

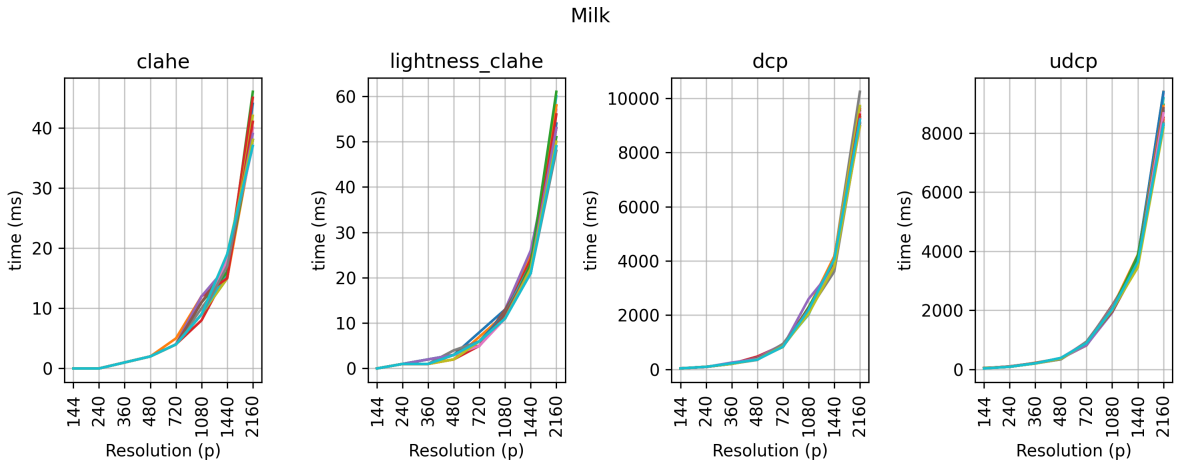
I measured runtimes for DCP, UDCP, CLAHE, and CIE L*a*b* luminance-channel CLAHE. See Figure 4.1.

```
import matplotlib.pyplot as plt
import json

DATASET_NAMES = (
  {
    'key': 'deepblue_prog',
    'pretty': 'Deepblue'
  },
  {
    'key': 'milk_prog',
    'pretty': 'Milk'
  }
)

PIPELINES = ('clahe', 'lightness_clahe', 'dcp', 'udcp')
RESOLUTIONS = ('144', '240', '360', '480', '720', '1080', '1440', '2160')

with open('../data/timing.json', 'r') as runtimes_file:
  runtimes_json = json.load(runtimes_file)

  for names in DATASET_NAMES:
    dataset = runtimes_json[names['key']]
    fig, axs = plt.subplots(nrows=1, ncols=len(PIPELINES))
    axs = axs.flatten()
    for pipeline, ax in zip(PIPELINES, axs, strict=True):
      seriess = [[] for _ in dataset[pipeline][RESOLUTIONS[0]]]
      for resolution in RESOLUTIONS:
        for i, time in enumerate(dataset[pipeline][resolution]):
          seriess[i].append(time // 1000)
      ax.grid(alpha=0.7)
      ax.tick_params(axis='x', labelrotation=90)
      ax.set(
```

```
      title=pipeline,
      xlabel='Resolution (p)',
      ylabel='time (ms)'
    )
    for series in seriess:
      ax.plot(RESOLUTIONS, series)
  fig.set_size_inches(10, 4)
  fig.suptitle(names['pretty'])
  fig.tight_layout()
  plt.show()
```



(a) Deepblue dataset



(b) Milk dataset

Figure 4.1: Timing trials. All images are 16:9; the dimension given is the vertical.

5 Next Steps

- Measure penetration improvement for new Erie data
- Possibly investigate speed improvements for DCP/UDCP (already using OpenMP, but seems unexpectedly slow)
- Add learning pipelines
- Choose which Retinex/CLAHE/etc. model-free pipelines to use
- Parameter searching

References

[1] J. S. Jaffe, "Computer modeling and the design of optimal underwater imaging systems," *IEEE Journal of Oceanic Engineering*, vol. 15, no. 2, pp. 101–111, Apr. 1990, doi: 10.1109/48.50695.

[2] Y. Y. Schechner and N. Karpel, "Clear underwater vision," in *Proceedings of the 2004 IEEE computer society conference on computer vision and pattern recognition, 2004. CVPR 2004.*, Jun. 2004, pp. I–I. doi: 10.1109/CVPR.2004.1315078.

[3] K. He, J. Sun, and X. Tang, "Single image haze removal using dark channel prior," in *2009 IEEE conference on computer vision and pattern recognition*, Jun. 2009, pp. 1956–1963. doi: 10.1109/CVPR.2009.5206515.

[4] P. Drews, E. R. Nascimento, F. Moraes, S. S. C. Botelho, and M. F. Montenegro Campos, "Transmission estimation in underwater single images," in *2013 IEEE international conference on computer vision workshops*, Dec. 2013, pp. 825–830. doi: 10.1109/ICCVW.2013.113.

[5] N. Carlevaris-Bianco, A. Mohan, and R. M. Eustice, "Initial results in underwater single image dehazing," in *OCEANS 2010 MTS/IEEE SEATTLE*, Sep. 2010, pp. 1–8. doi: 10.1109/OCEANS.2010.5664428.

[6] D. Berman, T. Treibitz, and S. Avidan, "Non-local image dehazing," in *2016 IEEE conference on computer vision and pattern recognition (CVPR)*, Jun. 2016, pp. 1674–1682. doi: 10.1109/CVPR.2016.185.

[7] D. Berman, D. Levy, S. Avidan, and T. Treibitz, "Underwater single image color restoration using haze-lines and a new quantitative dataset," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 8, pp. 2822–2837, Aug. 2021, doi: 10.1109/TPAMI.2020.2977624.

[8] W. Song, Y. Wang, D. Huang, and D. Tjondronegoro, "A rapid scene depth estimation model based on underwater light attenuation prior for underwater image restoration," in *Advances in multimedia information processing – PCM 2018*, R. Hong, W.-H. Cheng, T. Yamasaki, M. Wang, and C.-W. Ngo, Eds., Cham: Springer International Publishing, 2018, pp. 678–688.

[9] C. Qing, F. Yu, X. Xu, W. Huang, and J. Jin, "Underwater video dehazing based on spatial–temporal information fusion," *Multidim Syst Sign Process*, vol. 27, no. 4, pp. 909–924, Oct. 2016, doi: 10.1007/s11045-016-0407-2.

[10] C. Li, S. Anwar, and F. Porikli, "Underwater scene prior inspired deep underwater image and video enhancement," *Pattern Recognition*, vol. 98, p. 107038, Feb. 2020, doi: 10.1016/j.patcog.2019.107038.

[11] C. Li *et al.*, "An underwater image enhancement benchmark dataset and beyond," *IEEE Transactions on Image Processing*, vol. 29, pp. 4376–4389, 2020, doi: 10.1109/TIP.2019.2955241.

[12] Y. Wang, J. Guo, H. Gao, and H. Yue, "UIEC^2-net: CNN-based underwater image enhancement using two color space," *Signal Processing: Image Communication*, vol. 96, p. 116250, Aug. 2021, doi: 10.1016/j.image.2021.116250.