BM3D_filter

June 7, 2022

0.1 BM3D

Extensive experiments where BM3D is utilized - Exact Transform-Domain Noise Variance for Collaborative Filtering of Stationary Correlated Noise

```
Y. Mäkinen, L. Azzari and A. Foi, "Exact Transform-Domain Noise Variance for
Collaborative Filtering of Stationary Correlated Noise," 2019 IEEE International
Conference on Image Processing (ICIP), 2019, pp. 185-189, doi: 10.1109/ICIP. 2019.8802964.
```

Detailed Analysis of BM3D Image Denoising Method

- An Analysis and Implementation of the BM3D Image Denoising Method

```
Lebrun, M. (2012). An Analysis and Implementation of the BM3D Image.
```

0.1.1 Simplified implementation of Block matching and 3D (BM3D) filtering for Image Denoising.

Step 1. When using Google Colab, mount your Google Drive first to access files in it.

[1]:
```
from google.colab import drive
drive.mount ('/content/drive/')
```

```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call
drive.mount("/content/drive/", force_remount=True).
```

Step 2. Install and import required libraries.

[2]:
```
pip install bm3d
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: bm3d in /usr/local/lib/python3.7/dist-packages
(3.0.9)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages
(from bm3d) (1.4.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
(from bm3d) (1.21.6)
Requirement already satisfied: PyWavelets in /usr/local/lib/python3.7/dist-
packages (from bm3d) (1.3.0)
```

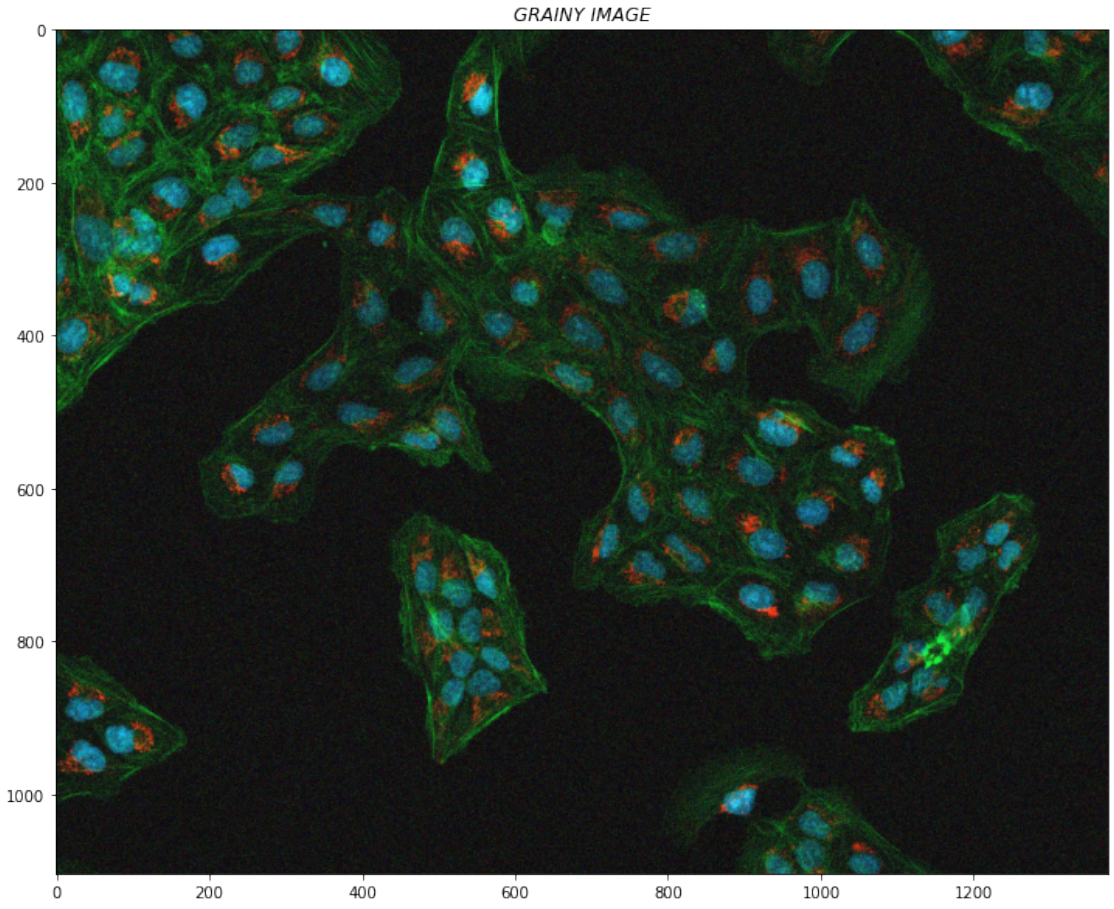
[3]:
```
from skimage import io, img_as_float     # img_as_ float - converts image to␣
 ↪floating point (integer types become 64-bit floats)
from matplotlib.pyplot import figure     # figure - sets figure size of the plot
import matplotlib.pyplot as plt          # plt - plots images
import cv2, bm3d                         # cv2 for accessing files and bm3d to␣
 ↪use the filter
```

Step 3. Load and display the grainy image.

[4]:
```
# convert image to float
image = img_as_float(io.imread("/content/drive/My Drive/Colab Notebooks/images/
 ↪osteosarcoma.tif"))

# plot the image
fig = figure(figsize=(20, 10))
plt.imshow(image)
plt.title("GRAINY IMAGE", style="oblique")
```

[4]:
```
Text(0.5, 1.0, 'GRAINY IMAGE')
```



Step 4. Denoise the above image using BM3D filter.

[5]:
```
# denoising the image using sigma = 0.2
BM3D_denoised_image = bm3d.bm3d(image, sigma_psd = 0.2, stage_arg = bm3d.
↪BM3DStages.HARD_THRESHOLDING)
```

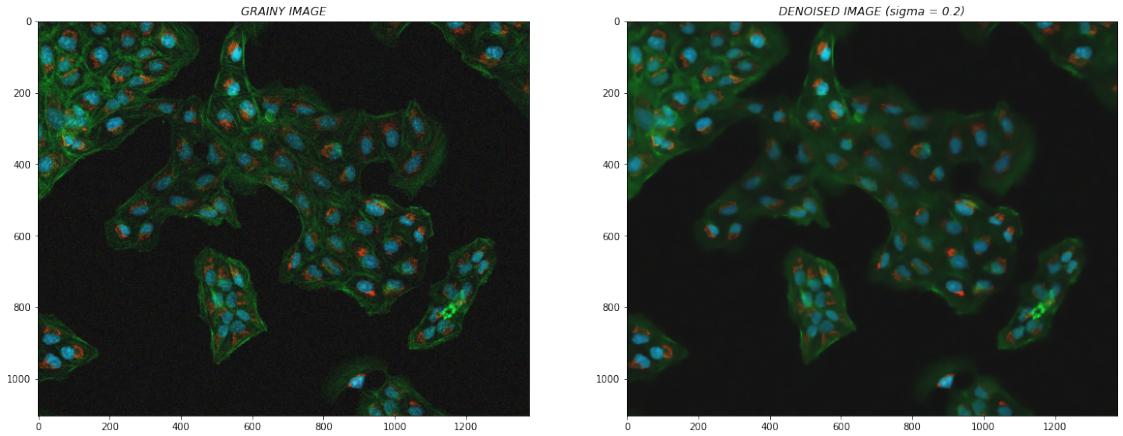
Parameters: - `sigma_psd` - noise standard deviation - `stage_arg` - determines whether to perform hard-thresholding or Wiener filtering. - `stage_arg` - BM3DStages.HARD_THRESHOLDING or BM3DStages.ALL_STAGES (slow but powerful)

Step 5. Plot the original image vs the denoised image side by side for comparison.

[6]:
```
# plot the two images to see the difference in quality
images = [image, BM3D_denoised_image]
labels = ['GRAINY IMAGE', 'DENOISED IMAGE (sigma = 0.2)']

# set the figure size to 20x10 (depends on the preference)
fig = figure(figsize=(20, 10))

# loops the subplot. this is used when multiple images are plotted.
for i in range(0, 1*2):
  fig.add_subplot(1, 2, i+1)
  plt.title(labels[i], style="oblique")
  plt.imshow(images[i])
plt.show()
```



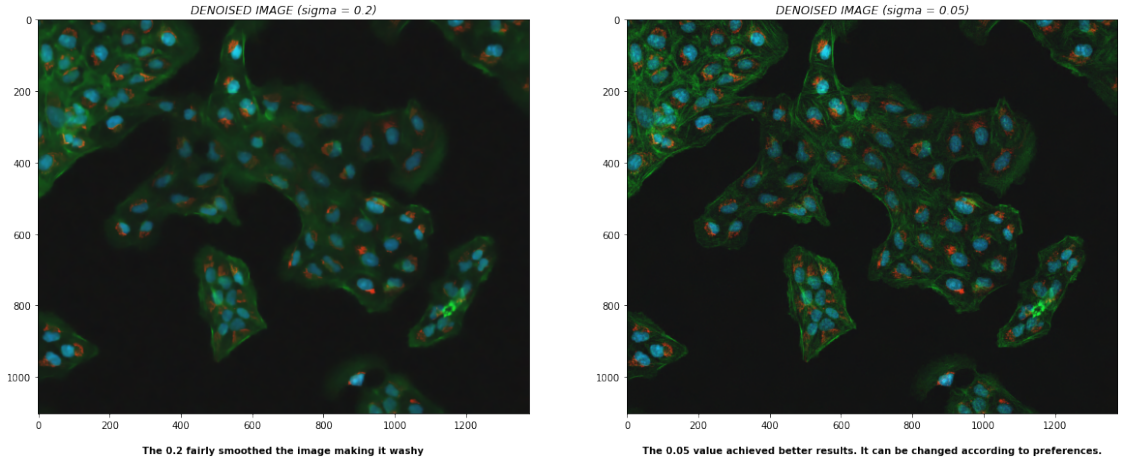
Step 6. Calibrate the value on the `sigma_pds` to achieve better results and plot the image together with the previous denoised image to see the difference in changing the values.

[7]:
```
# changing sigma_psd to 0.05 for experimental results.
BM3D_denoised_image1 = bm3d.bm3d(image, sigma_psd = 0.05, stage_arg = bm3d.
↪BM3DStages.HARD_THRESHOLDING)
```

[8]:
```
# plot the images to see the changes.
images = [BM3D_denoised_image, BM3D_denoised_image1]
labels = ['DENOISED IMAGE (sigma = 0.2)', 'DENOISED IMAGE (sigma = 0.05)']
caption = ['\nThe 0.2 fairly smoothed the image making it washy', '\nThe 0.05␣
 ↪value achieved better results. It can be changed according to preferences.']
fig = figure(figsize=(20, 10))

# loops through the subplot.
for i in range(0, 1*2):
  fig.add_subplot(1, 2, i+1)
  plt.title(labels[i], style="oblique")
  plt.imshow(images[i])
  ax = fig.add_subplot(1, 2, i+1)
  ax.set_xlabel(caption[i], fontsize='medium', fontweight='bold')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12:
MatplotlibDeprecationWarning: Adding an axes using the same arguments as a
previous axes currently reuses the earlier instance.  In a future version, a new
instance will always be created and returned.  Meanwhile, this warning can be
suppressed, and the future behavior ensured, by passing a unique label to each
axes instance.
  if sys.path[0] == '':
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
```



0.1.2 Performance Metrics.

Step 1. Install dependencies.

[9]:
```
pip install sewar
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: sewar in /usr/local/lib/python3.7/dist-packages
(0.4.5)
Requirement already satisfied: Pillow in /usr/local/lib/python3.7/dist-packages
(from sewar) (7.1.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages
(from sewar) (1.4.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
(from sewar) (1.21.6)
```

Step 2. Import required packages.

[10]:
```
from numpy.fft import fft2, ifft2
from sewar import full_ref
from skimage import measure, metrics
from skimage.metrics import structural_similarity as ssim
```

Step 3. Create an instance of the original image and the denoised image (0.05 sigma)

[11]:
```
# original image
orig = image
# denoised image with 0.05 sigma (the cleaner version)
denoised = BM3D_denoised_image1
```

Step 4. Measure the RSME.

[12]:
```
rmse_skimg = metrics.normalized_root_mse(orig, denoised)
print("RMSE: based on scikit-image = ", rmse_skimg)
```

```
RMSE: based on scikit-image =  0.34605233742383995
```

Step 5. Measure the MSE.

[13]:
```
mse_skimg = metrics.mean_squared_error(orig, denoised)
print("MSE: based on scikit-image = ", mse_skimg)
```

```
MSE: based on scikit-image =  0.0029487014106375628
```

Step 6. Measure the PSNR.

[14]:
```
psnr_skimg = metrics.peak_signal_noise_ratio(orig, denoised, data_range=None)
print("PSNR: based on scikit-image = ", psnr_skimg)
```

```
PSNR: based on scikit-image =  25.303692024491532
```

Step 7. Measure the SSIM.

[15]:
```
ssim_skimg = ssim(orig, denoised, data_range = orig.max() - orig.min(),␣
 ↪multichannel = True)
print("SSIM: based on scikit-image = ", ssim_skimg)
```

```
SSIM: based on scikit-image =  0.4567581485658551
```

Dataframe for the Results Above.

[16]:
```
import pandas as pd

metrics = {'Metrics':['RMSE','MSE','PSNR','SSIM'],
      'Score':[rmse_skimg, mse_skimg, psnr_skimg, ssim_skimg]}

df = pd.DataFrame(metrics)
```

[17]:
```
print(df)
```

```
  Metrics      Score
0    RMSE   0.346052
1     MSE   0.002949
2    PSNR  25.303692
3    SSIM   0.456758
```

The Root Mean Square Error (RMSE) got a 34% result which is not bad. While the Mean Squared Error (MSE) measures how close fitted line is to data points which means the higher the result, the higher the error it got. For the output, it surprisingly got a lower result which is good. The PSNR achieved a high result and the SSIM, achieved 45%.

[ ]:
```
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('BM3D_filter.ipynb')
```

```
--2022-06-07 14:38:29--  https://raw.githubusercontent.com/brpy/colab-
pdf/master/colab_pdf.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)…
185.199.108.133, 185.199.109.133, 185.199.110.133, …
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.108.133|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 1864 (1.8K) [text/plain]
Saving to: ‘colab_pdf.py’

colab_pdf.py        100%[===================>]   1.82K  --.-KB/s    in 0s

2022-06-07 14:38:29 (31.5 MB/s) - ‘colab_pdf.py’ saved [1864/1864]


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.
```

```
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Extracting templates from packages: 100%
```