Lab 9: Noise reduction using spatial-domain Techniques

Goal
The goal of this lab is to learn how to perform noise reduction using spatial domain techniques.

Objectives
- Learn how to implement the arithmetic mean filter, as well as some of its variations, such
  as the contra-harmonic mean, the harmonic mean, and the geometric mean filters.
- Learn how to perform order statistic filtering, including median, min, max, midpoint, and
  alpha-trimmed mean filters.

Tools/Software Requirement
- Anaconda,jupyter notebook, google colab
- Python 3.5

Tasks and description:-

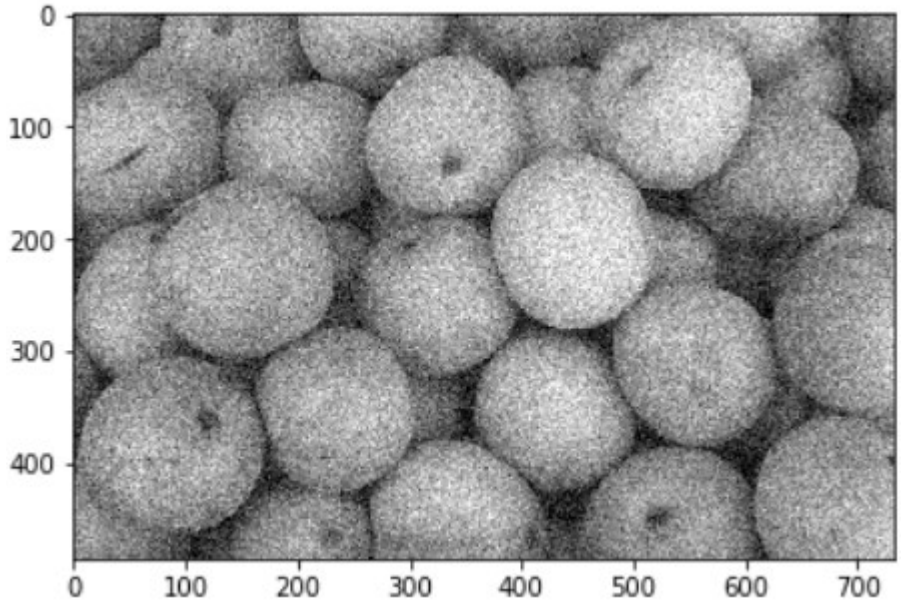
Task #1:-
Corrupt the input images with different types of noise models such as:-
1. 'Gaussian'
2. 'poisson'
3. 'salt & pepper'
4. 'speckle'
5. Salt only noise
6. Pepper only noise

Note:-
You may use the **skimage.util.random_noise** function from scikit image library.

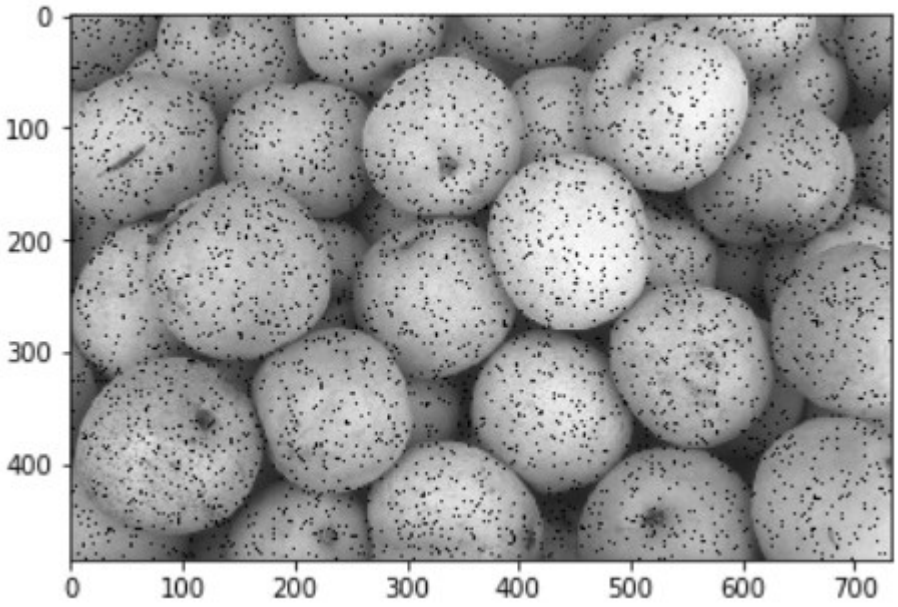
Example:-
```
guassiannoise=skimage.util.random_noise(img,"gaussian")
peppernoise=skimage.util.random_noise(img,"pepper")
```

You can incorporate all the above mentioned 6 types of noise models by playing around with the second argument.
For further details you may consult the following link:-
https://scikit-image.org/docs/stable/api/skimage.util.html#random-noise

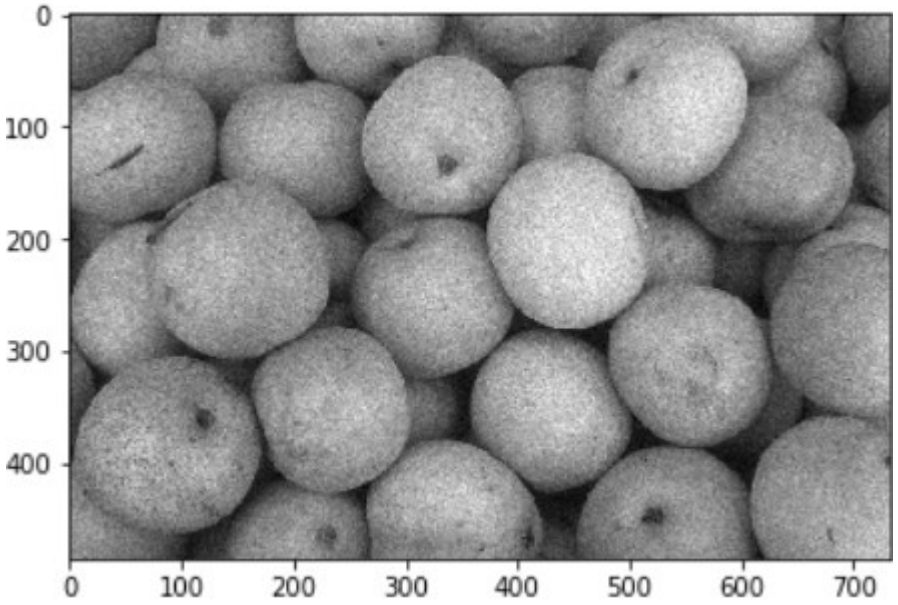
Demonstration:-
Your output for task 1 should look something like this :-
- **Image Corrupted with Gaussian Noise:**



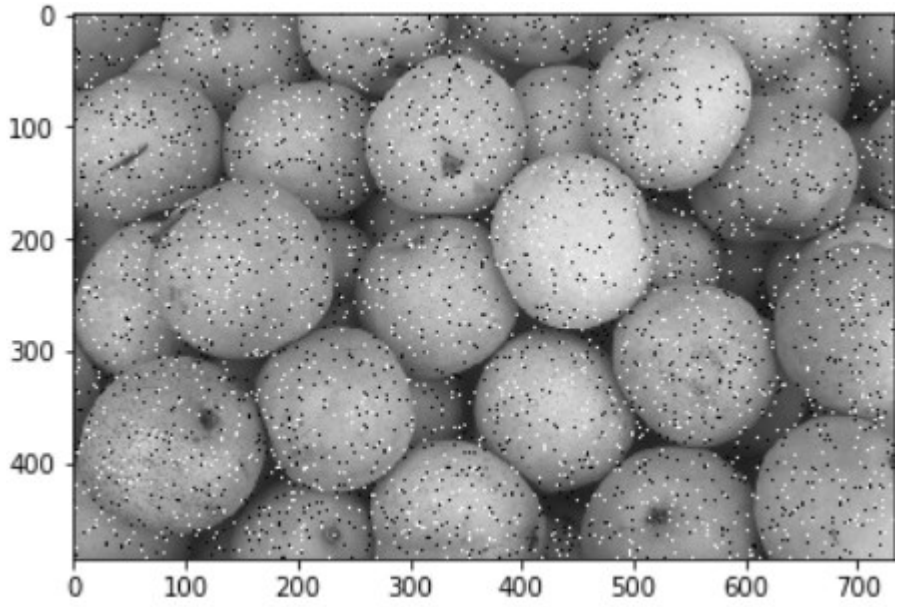
- **Image Corrupted with Pepper Noise:**



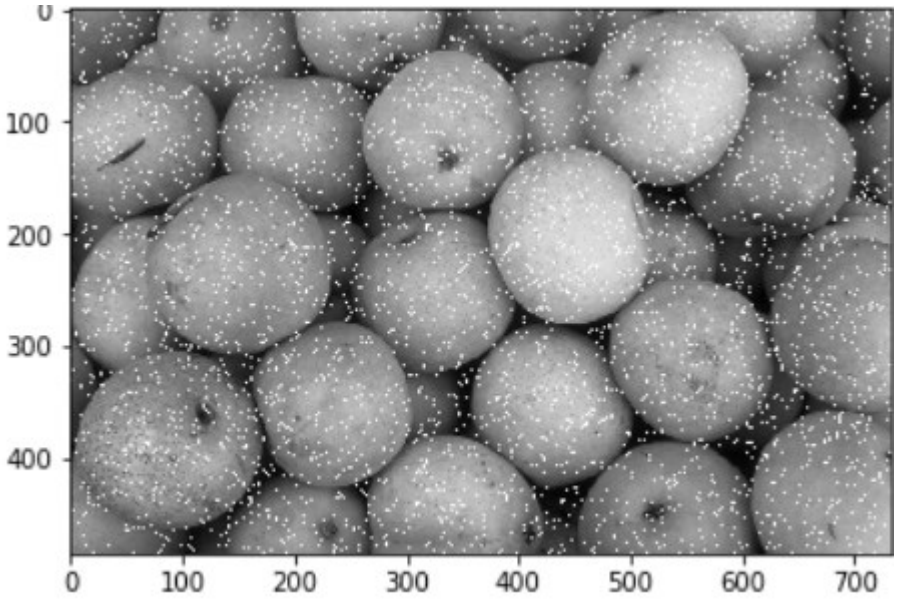
- **Image Corrupted with Poisson Noise:**



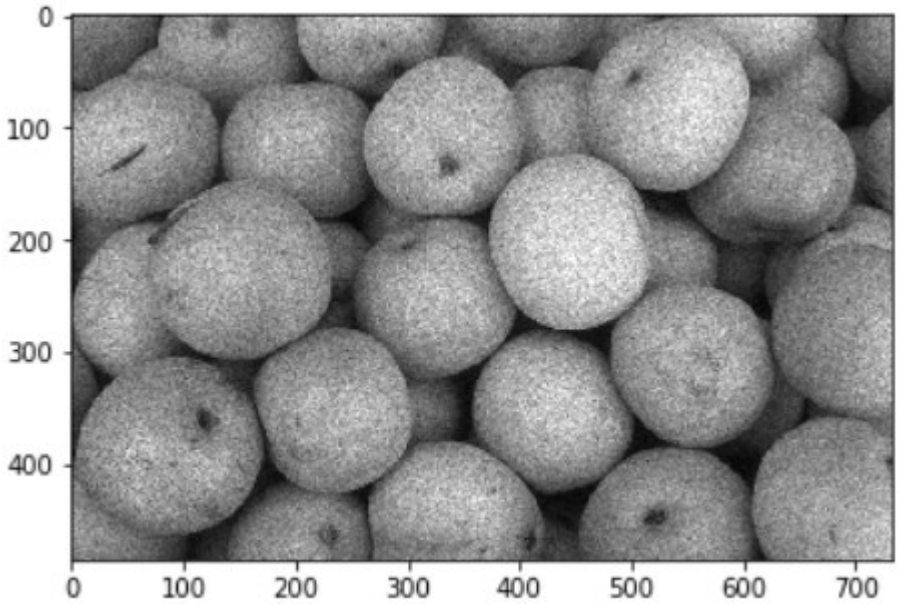
- **Image Corrupted with Salt & Pepper Noise:**



- **Image Corrupted with Salt Noise:**



- **Image Corrupted with Speckle Noise:**



Task #2:-
Apply different kinds of noise removal filters as given below;

1. Arithmetic mean
2. Geometric mean
3. Harmonic mean
4. Contra harmonic mean, The contra harmonic mean filter is used for filtering an image with either salt or pepper noise (but not both).
5. Max filters
6. Min filters

7. Median filters

Note:-
1):-in order to find out how a filter is affecting a particular type of noise you can save the resultant image using cv2.imwrite because just plotting would might not give you a precise intuition.
2):- You can use **filters** class from **scikit image** library. It contains very handy and easy to use builtin functions for filters.
You can import it as follows:-

- import skimage
- from skimage import filters

As it is a well renowned library for image processing so it has vast variety of functions. But for today's lab as we are using filtering so **scipy.ndimage** package is useful for you. You may visit the following link to get an insight about most of the filters for task2.
https://docs.scipy.org/doc/scipy-0.14.0/reference/ndimage.html

1):- Arithmetic mean

$$\hat{f}(x,y) = \frac{1}{mn} \sum_{(r,c) \in S_{xy}} g(r,c)$$

Suppose you have a 3*3 kernel as follows:-

```
kernel = np.array([ 1, 1, 1,    1, 1 , 1,    1, 1, 1]).reshape(3, 3)/9
```

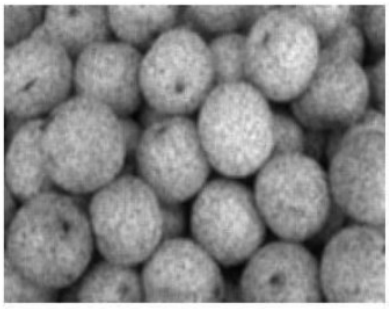
You can apply it using cv2.filter2D function as used in previous labs.

Output:-

order 9x9

Impact on Gaussian Noise

order 9x9



Impact on Pepper Noise

order 7x7

Impact on Poisson Noise

order 9x9

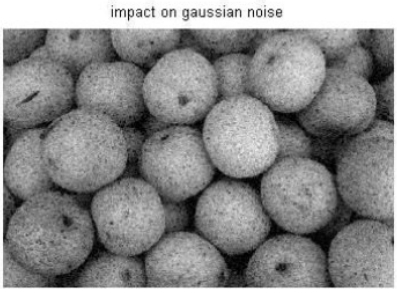
Impact on Salt Noise

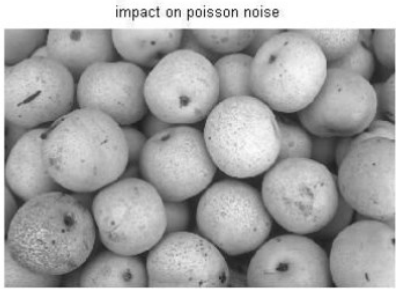
2):- Min filter

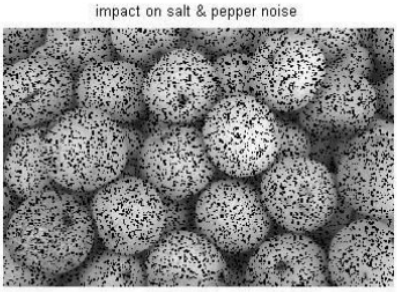
$$\hat{f}(x,y) = \min_{(r,c) \in S_{xy}} \{g(r,c)\}$$

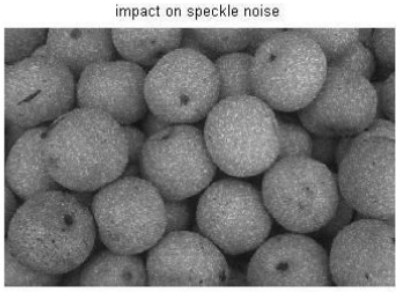
You may use the minimum_filter function from scikit image.
https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.ndimage.filters.minimum_filter.html

Output:-









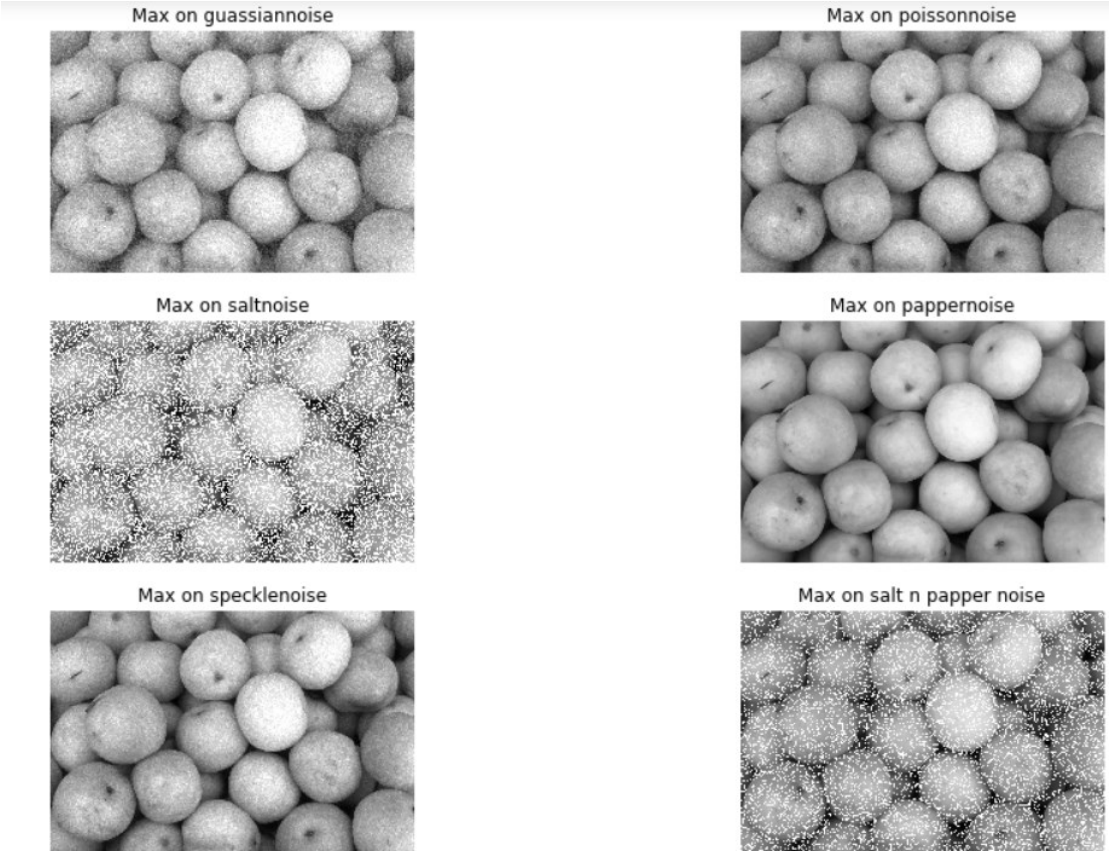
3):- Max filter

$$\hat{f}(x,y) = \max_{(r,c) \in S_{xy}} \{g(r,c)\}$$

For max filter of scikit image visit the following link.

https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.ndimage.filters.maximum_filter.html#scipy.ndimage.filters.maximum_filter

Output:-



4):- Median filter

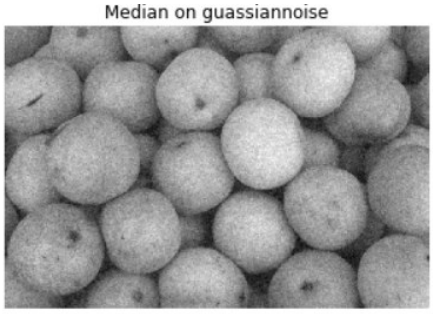
$$\hat{f}(x,y) = \underset{(r,c) \in S_{xy}}{\text{median}} \{g(r,c)\}$$

For median filter visit the following link.

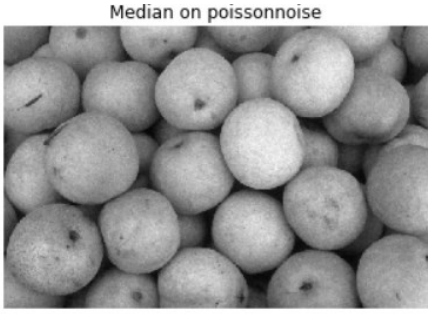
https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.ndimage.filters.median_filter.html#scipy.ndimage.filters.median_filter

Output:-

Median on guassiannoise

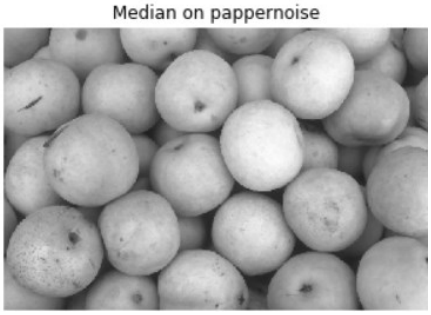


Median on poissonnoise

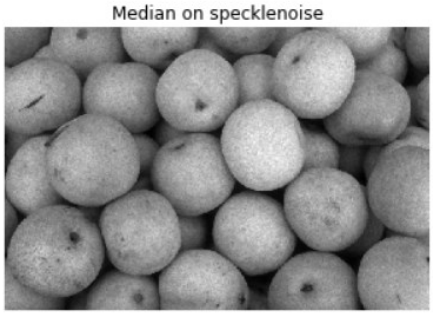


Median on saltnoise

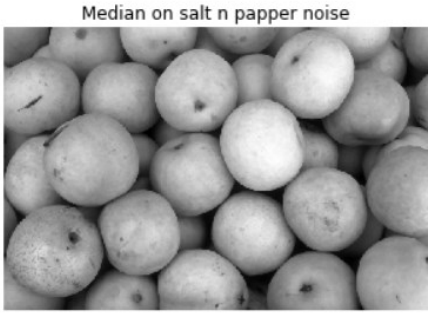
Median on pappernoise



Median on specklenoise



Median on salt n papper noise

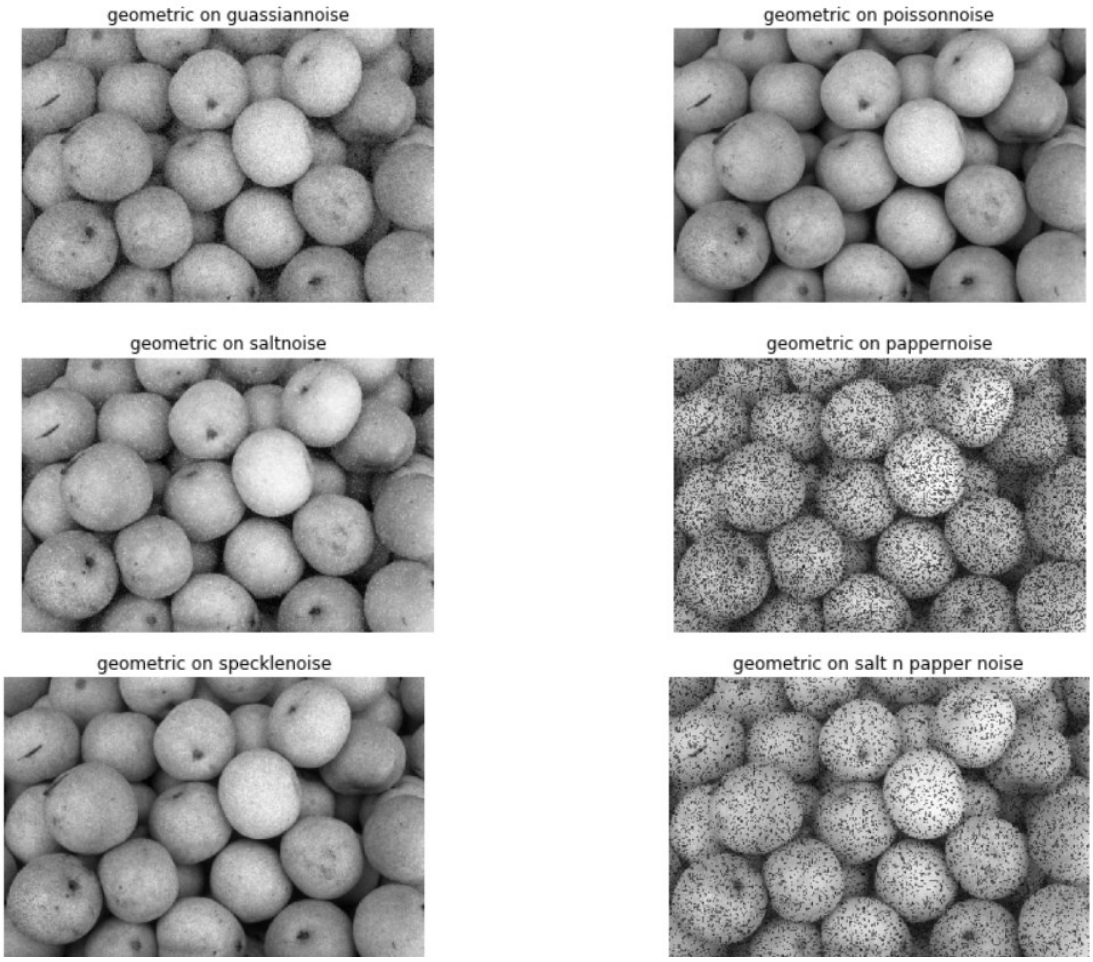


5):- Geometric mean

$$\hat{f}(x,y) = \left[\prod_{(r,c) \in S_{xy}} g(r,c) \right]^{\frac{1}{mn}}$$

Geometric mean filter can be found in rank class of skimage.filters
Visit the following link to read further.
https://scikit-image.org/docs/dev/api/skimage.filters.rank.html#skimage.filters.rank.geometric_mean

Output:-



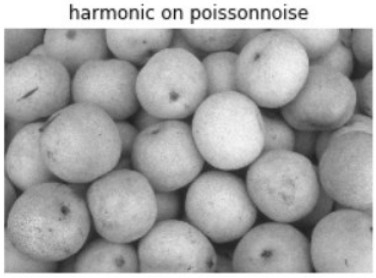
6):- Harmonic mean

$$\hat{f}(x,y) = \frac{mn}{\sum_{(r,c) \in S_{xy}} \frac{1}{g(r,c)}}$$

Output:-

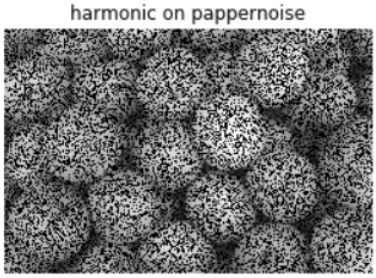
harmonic on guassiannoise

harmonic on poissonnoise

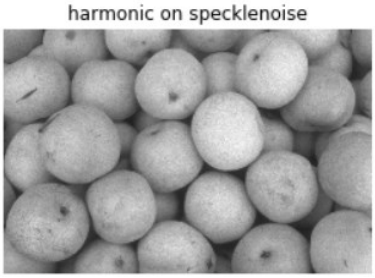


harmonic on saltnoise

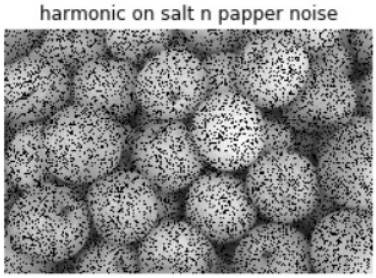
harmonic on pappernoise



harmonic on specklenoise



harmonic on salt n papper noise



7):- Contra Harmonic

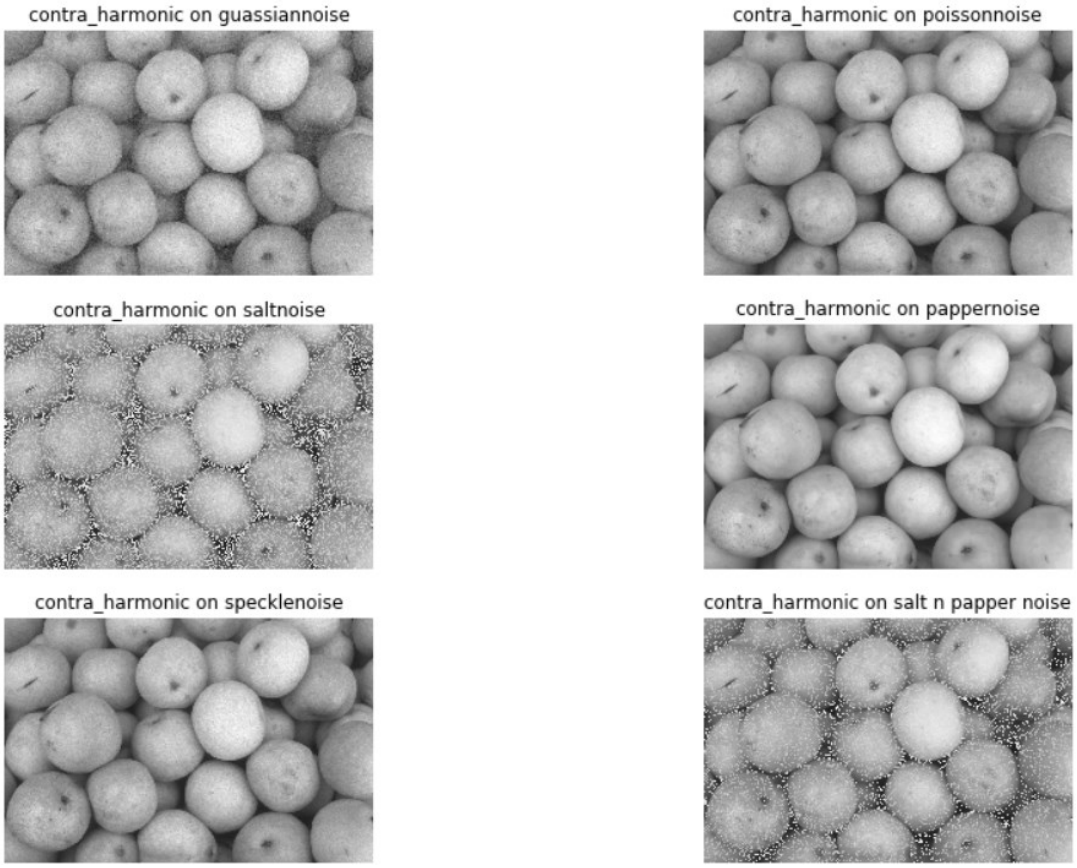
$$\hat{f}(x,y) = \frac{\sum_{(r,c) \in S_{xy}} g(r,c)^{Q+1}}{\sum_{(r,c) \in S_{xy}} g(r,c)^{Q}}$$

Output:-

For Q > 0

for positive values of Q it eliminates pepper noise.

Output:-



For Q < 0

for negative values of Q it eliminates salt noise.

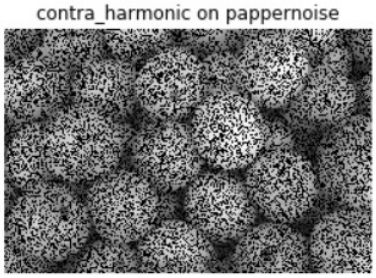
Output:-

contra_harmonic on guassiannoise

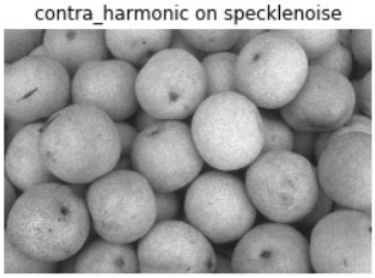
contra_harmonic on poissonnoise

contra_harmonic on saltnoise

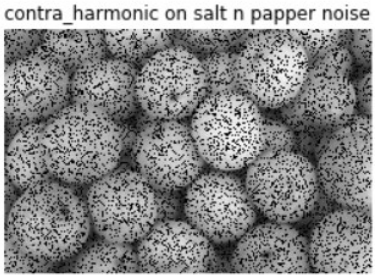
contra_harmonic on pappernoise



contra_harmonic on specklenoise



contra_harmonic on salt n papper noise



Submission Guidelines:-

Deliverables:

You have to submit a python notebook with ipynb extension containing all the codes and screen shots of output of all codes. Do not submit any zip folder containing code and screenshots separately.