108122015 Anish Rajan ECE-A

Arithmetic Mean Filter

import numpy as np

import cv2

import os

*def* arithmetic\_mean(*img*, *kernel\_size*):

    height, width = *img*.shape[:2]

    processed\_image = np.zeros((height, width), *dtype*=np.uint8)

    buffer = *kernel\_size* // 2

    for i in range(buffer, height - buffer):

        for j in range(buffer, width - buffer):

            total = 0

            for m in range(-buffer, buffer + 1):

                for n in range(-buffer, buffer + 1):

                    total += *img*[i + m, j + n]

            processed\_image[i, j] = total // (*kernel\_size* \*\* 2)

    return processed\_image

files = os.listdir("image\_processing/Noisy\_Images")

for filename in files:

    os.mkdir("./image\_processing/processed\_images/"+filename[:-4])#image\_processing\processed\_images

for filename in files:

    img = cv2.imread("./image\_processing/Noisy\_Images/"+filename, cv2.IMREAD\_GRAYSCALE)

    processed\_image = arithmetic\_mean(img, 3)

    cv2.imwrite("./image\_processing/processed\_images/"+filename[:-4]+"/arithmetic\_mean.png", processed\_image)

Geometric Mean Filter

import numpy as np

import cv2

import os

*def* geometric\_mean(*img*, *kernel\_size*):

    height, width = *img*.shape[:2]

    processed\_image = np.zeros((height, width), *dtype*=np.uint8)

    buffer = *kernel\_size* // 2

    for i in range(buffer, height - buffer):

        for j in range(buffer, width - buffer):

            product = 1.0

            for m in range(-buffer, buffer + 1):

                for n in range(-buffer, buffer + 1):

                    product \*= *img*[i + m, j + n]

            geometric\_mean = int(product \*\* (1.0 / (*kernel\_size* \*\* 2)))

            processed\_image[i, j] = geometric\_mean

    return processed\_image

files = os.listdir("image\_processing/Noisy\_Images")

#for filename in files:

#    os.mkdir("./image\_processing/processed\_images/"+filename[:-4])#image\_processing\processed\_images

for filename in files:

    img = cv2.imread("./image\_processing/Noisy\_Images/"+filename, cv2.IMREAD\_GRAYSCALE)

    processed\_image = geometric\_mean(img, 3)

    cv2.imwrite("./image\_processing/processed\_images/"+filename[:-4]+"/geometric\_mean.png", processed\_image)

Harmonic Mean Filter

import numpy as np

import cv2

import os

*def* harmonic\_mean(*image*, *kernel\_size*):

    height, width = *image*.shape[:2]

    filtered\_image = np.zeros((height, width), *dtype*=np.uint8)

    buffer = *kernel\_size* // 2

    for i in range(buffer, height - buffer):

        for j in range(buffer, width - buffer):

            sum\_pix= 0

            for m in range(-buffer, buffer + 1):

                for n in range(-buffer, buffer + 1):

                    sum\_pix+= 1/(*image*[i + m, j + n]+1e-8)

            harmonic\_mean = int(*kernel\_size*\*\*2/sum\_pix)

            filtered\_image[i, j] =harmonic\_mean

    return filtered\_image

files = os.listdir("image\_processing/Noisy\_Images")

#for filename in files:

#    os.mkdir("./image\_processing/processed\_images/"+filename[:-4])#image\_processing\processed\_images

for filename in files:

    img = cv2.imread("./image\_processing/Noisy\_Images/"+filename, cv2.IMREAD\_GRAYSCALE)

    processed\_image = harmonic\_mean(img, 3)

    cv2.imwrite("./image\_processing/processed\_images/"+filename[:-4]+"/harmonic\_mean.png", processed\_image)

Maximum Filter

import numpy as np

import cv2

import os

*def* max\_filter(*img*, *kernel\_size*):

    height, width = *img*.shape[:2]

    processed\_image = np.zeros((height, width), *dtype*=np.uint8)

    buffer = *kernel\_size* // 2

    for i in range(buffer, height - buffer):

        for j in range(buffer, width - buffer):

    # Extract the pixel values within the kernel

            kernel\_values = []

            for m in range(-buffer, buffer + 1):

                for n in range(-buffer, buffer + 1):

                    kernel\_values.append(*img*[i + m, j + n])

    # Calculate the maximum value from the kernel values

            max\_value = max(kernel\_values)

            processed\_image[i, j] = max\_value

    return processed\_image

files = os.listdir("image\_processing/Noisy\_Images")

#for filename in files:

#    os.mkdir("./image\_processing/processed\_images/"+filename[:-4])#image\_processing\processed\_images

for filename in files:

    img = cv2.imread("./image\_processing/Noisy\_Images/"+filename, cv2.IMREAD\_GRAYSCALE)

    processed\_image = max\_filter(img, 3)

    cv2.imwrite("./image\_processing/processed\_images/"+filename[:-4]+"/max\_filter.png", processed\_image)

Minimum Filter

import numpy as np

import cv2

import os

*def* min\_filter(*img*, *kernel\_size*):

    height, width = *img*.shape[:2]

    processed\_image = np.zeros((height, width), *dtype*=np.uint8)

    buffer = *kernel\_size* // 2

    for i in range(buffer, height - buffer):

        for j in range(buffer, width - buffer):

    # Extract the pixel values within the kernel

            kernel\_values = []

            for m in range(-buffer, buffer + 1):

                for n in range(-buffer, buffer + 1):

                    kernel\_values.append(*img*[i + m, j + n])

            min\_value = min(kernel\_values)

            processed\_image[i, j] = min\_value

    return processed\_image

files = os.listdir("image\_processing/Noisy\_Images")

#for filename in files:

#    os.mkdir("./image\_processing/processed\_images/"+filename[:-4])#image\_processing\processed\_images

for filename in files:

    img = cv2.imread("./image\_processing/Noisy\_Images/"+filename, cv2.IMREAD\_GRAYSCALE)

    processed\_image = min\_filter(img, 3)

    cv2.imwrite("./image\_processing/processed\_images/"+filename[:-4]+"/minimum\_filter.png", processed\_image)

Median Filter

import numpy as np

import cv2

import os

*def* median\_filter(*img*, *kernel\_size*):

    height, width = *img*.shape[:2]

    processed\_image = np.zeros((height, width), *dtype*=np.uint8)

    buffer = *kernel\_size* // 2

    for i in range(buffer, height - buffer):

        for j in range(buffer, width - buffer):

            kernel\_values = []

            for m in range(-buffer, buffer + 1):

                for n in range(-buffer, buffer + 1):

                    kernel\_values.append(*img*[i + m, j + n])

            median\_value = np.median(kernel\_values)

            processed\_image[i, j] = median\_value

    return processed\_image

files = os.listdir("image\_processing/Noisy\_Images")

#for filename in files:

#    os.mkdir("./image\_processing/processed\_images/"+filename[:-4])#image\_processing\processed\_images

for filename in files:

    img = cv2.imread("./image\_processing/Noisy\_Images/"+filename, cv2.IMREAD\_GRAYSCALE)

    processed\_image = median\_filter(img, 3)

    cv2.imwrite("./image\_processing/processed\_images/"+filename[:-4]+"/median\_filter.png", processed\_image)

Contraharmonic Filter

import numpy as np

import cv2

import os

*def* contraharmonic\_mean(*img*, *kernel\_size*):

    height, width = *img*.shape[:2]

    processed\_image = np.zeros((height, width), *dtype*=np.uint8)

    buffer = *kernel\_size* // 2

    q=2

    for i in range(buffer, height - buffer):

        for j in range(buffer, width - buffer):

            sum\_pix= 0

            for m in range(-buffer, buffer + 1):

                for n in range(-buffer, buffer + 1):

                    sum\_pix+= *img*[i + m, j + n]

            contraharmonic\_mean=(sum\_pix\*\*(q+1))/(sum\_pix\*\*q)

            processed\_image[i, j] = contraharmonic\_mean

    return processed\_image

files = os.listdir("image\_processing/Noisy\_Images")

#for filename in files:

#    os.mkdir("./image\_processing/processed\_images/"+filename[:-4])#image\_processing\processed\_images

for filename in files:

    img = cv2.imread("./image\_processing/Noisy\_Images/"+filename, cv2.IMREAD\_GRAYSCALE)

    processed\_image = contraharmonic\_mean(img, 3)

    cv2.imwrite("./image\_processing/processed\_images/"+filename[:-4]+"/contraharmonic\_mean.png", processed\_image)