Computer Vision

Task (1)

Name	Section	Bench Number
Amr Mohamed	2	7
Adham Mohamed	1	9
Belal Mohamed	1	23
Mahmoud Emad	2	28
Ezzat Hegazy	1	56
Mina Azer	1	58

Submitted To:

Prof. Dr. Ahmed Badawi

TA. Peter Emad

Systems and Biomedical Engineering Dept.

Cairo University

Computer Vision

Adding noise to images: in image processing, adding noise to an image is a common technique used to simulate real-world conditions and test the power of image processing algorithms. Here are some ways to add noise to an image:

1. **The uniform noise**: The algorithm to generate uniform noise can be summarized as follows:

   a. Define the minimum value (a) and maximum value (b) of the uniform distribution.

   b. Generate a random number from a uniform distribution between a and b for each pixel in the image.

   c. Add the generated noise to the original image by adding the random number to the pixel value.

   d. Repeat steps 2-3 for all pixels in the image.

   e. Note that the above algorithm assumes that the image is a 2D matrix of pixel values. The algorithm can be easily adapted for color images by applying the noise to each color channel separately.

Computer Vision

2-**Gaussian noisy filter**: The algorithm to generate Gaussian noise can be summarized as follows:

f. Define the mean (mu) and standard deviation (sigma) of the Gaussian distribution. The mean represents the center of the distribution and the standard deviation represents the spread of the distribution.

g. Generate a random number from a uniform distribution between 0 and 1. This number will be used to generate the Gaussian noise.

h. Use the inverse transform method to transform the random number from the uniform distribution to the Gaussian distribution

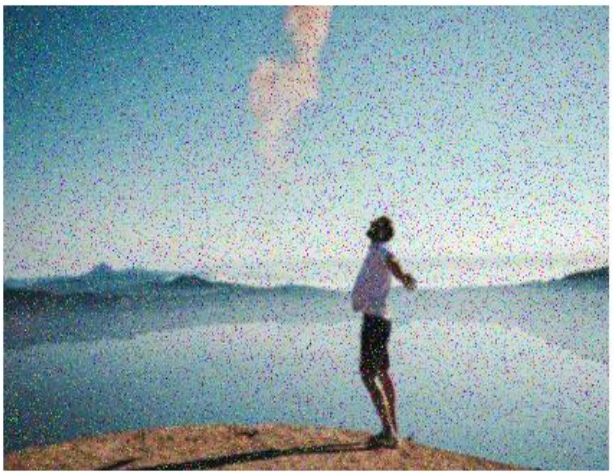
i. Use the generated value of x as the Gaussian noise to add to the original signal or image.

j. Repeat steps b-e to generate more samples of Gaussian noise.

3-**The Salt and Pepper noise:** can be summarized as follows:

a. Define the probability (p) of adding salt and the probability (q) of adding pepper. Salt is defined as a maximum pixel value (white) and pepper is defined as a minimum pixel value (black).
b. Generate a random number from a uniform distribution between 0 and 1 for each pixel in the image.
c. If the generated number is less than or equal to p, set the pixel value to the maximum pixel value (salt).
d. If the generated number is greater than 1 - q, set the pixel value to the minimum pixel value (pepper).
e. Repeat steps b-d for all pixels in the image.

Note that the above algorithm adapted for color images by applying the noise to each color channel separately.



4- **The Average filter**: can be summarized as follows: -

a. Define the size of the filter kernel, which is usually an odd number (e.g., 3x3, 5x5, 7x7).

b. Traverse the image pixel by pixel and for each pixel:

I. Define a sub-image (i.e., a kernel) centered on the pixel.
II. For each color channel (e.g., red, green, blue), compute the average of the pixel values in the kernel.
III. Replace the pixel value for each color channel with the computed average.

c. Repeat step b for all pixels in the image.

Note: The size of the filter kernel determines the degree of smoothing applied to the image. A larger kernel will result in more smoothing, but can also result in loss of fine details. The filter can be applied multiple times to achieve stronger smoothing.

Computer Vision

5- **The Gaussian filter:** The algorithm for a Gaussian filter on a color image can be summarized as follows:

a. Define the size of the filter kernel, which is 3x3 size.

b. Define the standard deviation of the Gaussian distribution.

c. Compute the values of the Gaussian kernel, which is a 3D cube of weights centered on the origin of the kernel. The values of the kernel can be computed using the formula:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)},$$

d. Traverse the image pixel by pixel and for each pixel:

I. Define a sub-image (i.e., a kernel) centered on the pixel.
II. For each color channel (e.g., red, green, blue), multiply each pixel value in the kernel by the corresponding value in the Gaussian kernel for that color channel.
III. Compute the sum of the resulting values in the kernel for each color channel.
IV. Normalize the sum by dividing by the sum of the values in the Gaussian kernel for each color channel.
V. Replace the pixel value for each color channel with the normalized sum.

e. Repeat step 4 for all pixels in the image.

Note that the above algorithm assumes that the image is a 3D matrix of pixel values with dimensions (height, width, channels). The size of the filter kernel and the standard deviation of the Gaussian distribution determine the degree of smoothing applied to the image.

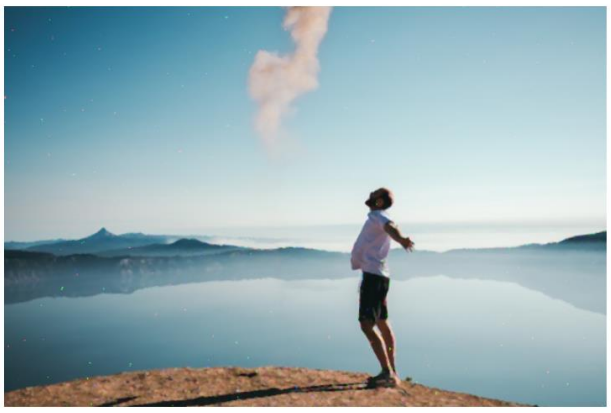
6-**The Median filter**: The algorithm for a median filter on a color image can be summarized as follows:

a. Define the size of the filter kernel with a size 3x3.

b. Traverse the image pixel by pixel and for each pixel:

   I. Define a sub-image (i.e., a kernel) centered on the pixel.
   II. For each color channel (e.g., red, green, blue), extract the pixel values in the kernel and sort them in ascending order.
   III. Compute the median value of the sorted pixel values for each color channel.
   IV. Replace the pixel value for each color channel with the median value.

c. Repeat step 2 for all pixels in the image.



Note the size of the filter kernel determines the size of the sub-image (i.e., kernel) that is used to compute the median value. A larger kernel will result in more smoothing, but can also result in loss of fine details.

To optimize the algorithm, it is recommended to use an efficient sorting algorithm (e.g., quicksort) to sort the pixel values in each color channel. In addition, it may be useful to use a sliding window approach instead of creating a new sub-image for each pixel, which can reduce the memory requirements of the algorithm.

7- **The Sobel filter algorithm**: is an algorithm for edge detection using a defined kernel with summation of 0

a. Define the two 3x3 convolution kernels:

Gx = [-1, 0, 1, -2, 0, 2, -1, 0, 1]

Gy = [-1, -2, -1, 0, 0, 0, 1, 2, 1]

where Gx and Gy are the kernels for the horizontal and vertical gradients, respectively.

b. Traverse the image pixel by pixel and for each pixel:

I. Define a sub-image (i.e., a kernel) centered on the pixel.
II. Compute the convolution of the kernel with the image using the Gx and Gy kernels.
III. Compute the magnitude of the gradient at the pixel using the formula**: mag = sqrt (Gx^2 + Gy^2)**
   i. If desired, compute the orientation of the gradient at the pixel using the formula: **angle = atan2(Gy, Gx)**

c. Replace the pixel value with the magnitude of the gradient.

d. Repeat step 2 for all pixels in the image.

Note that the above algorithm assumes that the image is a 2D matrix of pixel values with dimensions (height, width). The output of the Sobel filter is a 2D matrix of gradient magnitudes.

8-**The Prewitt filter**: Prewitt filter algorithm to a gray-scale image:

a. Convert the input image to grayscale if it is not already in grayscale.

b. Initialize the two 3x3 kernels for detecting vertical and horizontal edges, as shown in the previous answer:

Vertical edge kernel = [-1, 0, 1, -1, 0, 1, -1, 0, 1]

Horizontal edge kernel = [-1, -1, -1, 0, 0, 0, 1, 1, 1]

c. Create an output image with the same dimensions as the input image.

d. Loop over each pixel in the input image, excluding the border pixels which cannot be convolved with the 3x3 kernels.

e. For each pixel, calculate the sum of products of the input image values and the corresponding kernel values within the 3x3 neighborhood of the current pixel. The vertical kernel is applied to the rows of the input image, while the horizontal kernel is applied to the columns.

f. Take the absolute value of the sum of products for both kernels and sum them together.

g. Set the output pixel value to this sum.

h. Repeat steps d-g for all pixels in the input image.

9-**The Robert filter:** The Robert operator is a simple and fast algorithm that uses two 2x2 kernels to detect edges in the horizontal and vertical directions. Here is the step-by-step algorithm for the Robert operator:

a. Convert the input image to grayscale if it is not already in grayscale.
b. Define the two 2x2 kernels for detecting horizontal and vertical edges:

Horizontal edge kernel = [1, 0, 0, -1]

Vertical edge kernel = [0, 1, -1, 0]

c. Create an output image with the same dimensions as the input image.
d. Loop over each pixel in the input image, excluding the border pixels which cannot be convolved with the 2x2 kernels.
e. For each pixel, calculate the gradient in the horizontal and vertical directions using the two kernels.
f. Calculate the gradient magnitude for each pixel by combining the horizontal and vertical gradients:
g. Gradient **magnitude = sqrt(horizontal gradient^2 + vertical gradient^2)**
h. Apply a threshold to the output image to highlight the edges.

9-**The Canny filter:** is a popular and effective algorithm for detecting edges in images. It involves multiple steps to extract edges from the image. Here is the step-by-step algorithm for the Canny edge detection:

a. Apply a Gaussian filter to the image to smooth out noise and reduce the number of false positives.

b. Calculate the gradient of the image using the Sobel operator to find the edges. This involves convolving the image with two kernels.

c. Compute the gradient magnitude and direction at each pixel using the results of the previous step.

d. Apply non-maximum suppression to thin the edges. This involves removing pixels that are not part of a strong edge. For each pixel, compare its gradient magnitude with the gradient magnitudes of its neighbors in the direction of the gradient. If the magnitude of the central pixel is not the maximum, set it to zero.

e. Apply hysteresis thresholding to remove weak edges. This involves defining two threshold values: a high threshold and a low threshold. Pixels with gradient magnitudes above the high threshold are considered strong edges, while pixels with magnitudes below the low threshold are considered non-edges. Pixels with magnitudes between the high and low thresholds are considered weak edges. Weak edges are only kept if they are connected to strong edges.

Note that canny filter produces high-quality edge detection results. The choice of parameters such as the filter size and the threshold values can greatly affect the quality of the output, and may require experimentation to get optimal results.

Computer Vision

10-**Histogram Equalization:** For gray scale images

a. Calculate histogram of the image (each gray level (0:255) with its corresponding intensity (number of pixels)):

   For pixel in pixels:

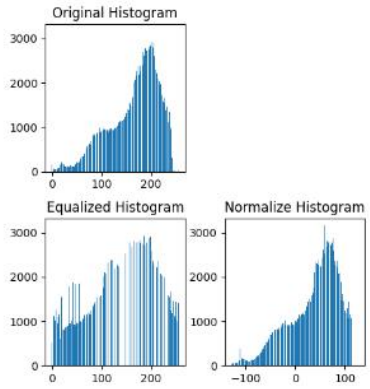
   Histogram[pixel] += 1

b. Calculate Probability Distribution Function:
   I. Get sum of intensities (pixels).
   II. Divide each intensity by the sum.
c. Calculate Cumulative Distribution Function:
   i. By summing each intensity with all previous ones.
d. Mapping so that range is from 0 to 255:
   i. Multiply each CDF by max gray level.
e. Put new pixel values in the equalized histogram:
   a. For row in range of rows:
      i. For each pixel in that row:
         1. Equalized image of that pixel = new intensity level

Histogram Normalization: For gray scale images

Get min and max value in the image.

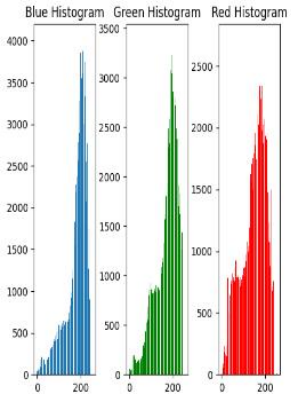
Normalize: [(pixel value - min) / (max – min)] * 255

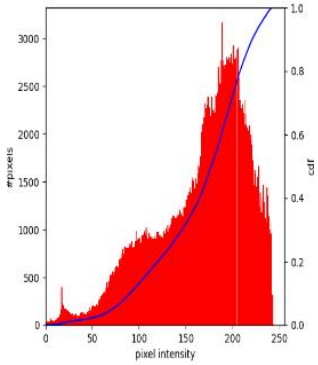


Computer Vision

-**The Equalized image**:

-**The RGB histogram:**





11-**Local Thresholding Algorithm**:

a. Choose a window size (also known as neighborhood size) around each pixel in the image.
b. Compute the mean or median intensity value of the pixels in the selected window size.
c. Assign a threshold value to each pixel in the image based on its local mean or median intensity value. For example, if the intensity value of a pixel is greater than the local mean or median intensity value, the pixel is assigned a value of 1 (foreground). Otherwise, it is assigned a value of 0 (background).

12- **Global Thresholding Algorithm**:

a. Choose an initial threshold value (typically the mean or median intensity value of the image).
b. Assign a pixel as foreground or background based on whether its intensity value is above or below the threshold value.
c. Calculate the mean intensity values of the foreground and background pixels.
d. Update the threshold value as the average of the foreground and background mean intensity values.
e. Repeat steps b-d until the threshold value converges to a stable value.

Computer Vision

13-**Frequency Domain filter**: Here are the general steps for a frequency domain filtering algorithm:

a. Load the input image.

b. Convert the image from the spatial domain to the frequency domain by applying the Fourier Transform.

c. Design a filter function to attenuate certain frequency components of the image. This filter function can be designed based on the specific requirements of the application. For example:

   I. a low-pass filter can be designed to attenuate high-frequency components of the image and retain low-frequency components:

II. while a high-pass filter can be designed to attenuate low-frequency components of the image and retain high-frequency components.

d. Multiply the Fourier Transform of the input image by the Fourier Transform of the filter function to obtain the filtered image in the frequency domain.

e. Convert the filtered image from the frequency domain back to the spatial domain by applying the Inverse Fourier Transform.

f. Normalize the filtered image by scaling its pixel values to the range of 0 to 255 or 0 to 1, depending on the requirements of the application.

g. Save the filtered image to a file or display it on the screen.

14-**Hybrid image algorithm:** is combination of two images - a low-pass filtered version of one image and a high-pass filtered version of another image. Here are the general steps for creating a hybrid image:

1. Load two input images, one that will provide the high-frequency content and one that will provide the low-frequency content.

2. Apply a low-pass filter to the first image to obtain the low-frequency content. The resulting image should retain the overall structure and shape of the original image, but with less fine detail.
3. Apply a high-pass filter to the second image to obtain the high-frequency content.
4. Combine the low-frequency content from the first image and the high-frequency content from the second image to create the hybrid image. This can be done by adding the low-pass filtered image to the high-pass filtered image.
5. Normalize the pixel values of the resulting hybrid image to the range of 0 to 255 or 0 to 1, depending on the requirements of the application.
6. Save the hybrid image to a file or display it on the screen.

Computer Vision