Project Documentation

For

Virtual Lab for Image Processing Project (Image Arithmetic) Using JavaScript

Prepared by Shoaib Areeb

Documentation

Prepared by Shoaib Areeb

1. INTRODUCTION

The Image Arithmetic virtual lab is the JavaScript version of the experiment written originally in PHP.
It is used to demonstrate how various arithmetic operations between two images produce different results according to the operations performed.

Languages used :

HTML and CSS
JavaScript

External Libraries:

JQuery.js
Bootstrap.css

2. THEORY

In image arithmetic, we consider operations such as I(x,y) = A (x,y) o B(x,y) where o is an arithmetic operation such as addition, subtraction, multiplication or division. Here, A and B could be derived from different sources. Such operations are particularly used in modelling image acquisition as a result of a perfect image corrupted with (additive or multiplicative) noise. The noise is often introduced either by a transmission medium or the camera.
In this experiment we access the RGBA pixel values of an image which ranges from (0-255) and perform the arithmetic operation accordingly.

Arithmetic operations are done pixelwise. Let p = A(x,y) and q = B(x,y) be the pixel values to be operated on and r =I(x,y) be the result of the operation.

Addition : $I(x,y) = A(x,y) + B(x,y) \rightarrow r = p + q$

Subtraction : $I(x,y) = A(x,y) - B(x,y) \rightarrow r = p - q$

Difference : $I(x,y) = |A(x,y) - B(x,y)| \rightarrow r = |p - q|$

Multiplication : $I(x,y) = A(x,y) \times B(x,y) \rightarrow r = p \times q$

Division : $I(x,y) = A(x,y) / B(x,y) \rightarrow r = p / q$

Implementation issues : Digital images are stored as b - bit images. Hence, the range of values a pixel can take is restricted to the range [0, 1,... $(2^b -1)$]. With b= 8 this range is [0,1,...255]. The closed interval poses a problem when performing arithmetic operations in practice, as the results are not guaranteed to be within this interval. For an 8-bit image the intervals for the output pixel for each operation are:

Addition: $r \in [0, (2x255=510)]$

Subtraction: $r \in [-255, 255]$

Difference: $r \in [0, 255]$

Multiplication: $r \in [0, (255^2 = 65025)]$

Division: $r \in [0,\infty]$

Since we need r to be in [0,255], we will have an underflow or overflow. A final processing step is generally required to handle this problem.

There are two options:

Clipping- Map all negative pixel values ($r < 0$) to zero and all values above 255 to 255.

Auto scaling - This operation remaps the range of r to fit to be in [0, 255] as follows. Let r_a be the autoscaled value.

$r_a = 255 \times (r - r_{min})/(r_{max}-r_{min})$

Where, r_{max} and r_{min} are the maximum and minimum values of an arithmetic operation.

3. OVERALL DESCRIPTIONS

1. Image Grid Box

1. Image Grid
2. Run Button
3. Reset Button

2. Operations Menu

1. Select Operation
    1. Addition
    2. Subtraction
    3. Difference
    4. Multiplication
    5. Division
2. Select Fitting
    1. Clipping
    2. Auto-Scaling
3. Select Secondary Image
    1. Dull
    2. Bright
    3. Gradient

3.1.1 Image Grid : It is used to select the primary image which will be displayed in the input layer box. It consists of nine <img> tags which are carefully placed in a grid layout using css float property.

3.1.2 - Run Button : It is used to Run the experiment to observe the result in the output layer. It can only be used once all the fields have been selected.

3.1.3 - Reset Button :It is used to reset all fields of the experiment to original conditions. This is done using the function FormReset() .

3.2.1.1 - Addition : This option is used to perform Addition Operation.

3.2.1.2 - Subtraction : This option is used to perform Subtraction Operation.

3.2.1.3 - Difference : This option is used to perform Difference Operation.

3.2.1.4 - Multiplication: This option is used to perform Multiplication Operation.

3.2.1.5 - Division : This option is used to perform Division Operation.

3.2.2.1 - Clipping : This option clips the image pixels to values between 0 and 255.

3.2.2.2 - Auto-Scaling : This option autoscales the pixel values to remain between 0 and 255.

3.2.3.1 - Dull : This option chooses a dull image as a secondary image from the images folder.

3.2.3.1 - Bright : This option chooses a bright image as a secondary image from the images folder.

3.2.3.2 - Gradient : This option chooses a gradient image as a secondary image from the images folder.

4. WORKING DESCRIPTION

Here are all the functions and their working defined in the file **arith.js** :

Global Variables :

canvas : stores the canvas value of the output layer from id “output_layer”
context : stores the context of canvas
canvas2 : stores the canvas value of the secondary image id “secondary”
context2 : stores the context of canvas2
operation : stores the radio input value from the select operation menu
fitting : stores the radio input value from the select fitting menu
RunButton : Stores the value of button id “Run”

Functions :

doOperation(data1,data2) :

Function Introduction: This function performs the main objective of the experiment i.e. image arithmetic operations. It takes the pixel data of the primary and secondary images and performs the operation as per user choice.

Function Implementation: The values of operation and fitting are first calculated and two switch-case statements are used to do arithmetic operations pixel-by-pixel. Variables num1, num2 and num3 are used to store pixel values separately since the canvas data values automatically clips the pixel values to stay between 0 and 255.The num array stores these values to create pixel data of the entire processed image. These values are used to calculate max and min of a given operation (stored in variables max and min) which can be used if a user selects Auto-Scaling option.

drawImage(image) :
Function Introduction: This function is used to access the image from its associated canvas.

secimgdisplay() :
Function Introduction: This function is specifically used to draw the secondary image on an invisible canvas so that its pixel data can be accessed. It is using the DOM property of JavaScript to access the secondary image.

addEffect() :

Function Introduction: This function basically calls the doOperation Function after the user has pressed the Run button.This functions first calls the secimgdisplay() function and after that it stores the two image data into the imageData1 and imageData2 variables using the getImageData() canvas function and the sends those data to the doOperation() function. It finally displays the image into the output layer using the putImageData function.

Function Implementation: The function first calls the secimgdisplay() function and after that it stores the two image data into the imageData1 and imageData2 variables using the getImageData() canvas function and the sends those data to the doOperation() function. It finally displays the image into the output layer using the putImageData function.

init() :

Function Introduction: This function is only called after the user has selected his/her primary image. It initialises the canvas for the output and starts accessing pixel data of primary image and

thereby initialising the values of variable canvas and context. It also uses an event listener for the Run button so that when it is clicked , the funtion call the addEffect() function.

preview(img) :
Function Introduction: This function displays the primary image on the Input Layer.

Funtion Implementation: The function first sets the visibility of the input layer image to “visible”. The function is called using an onclick listener in the img tags of the primary images and using the DOM model of JavaScript it displays the respective image which is passed to the function.

previewSec(img) :
Function Introduction: This function displays the secondary image on the Secondary Image box . It is called using an onclick listener on the radio input tags of the secondary images.

checkSRC() :
Function Introduction: This funtion is used to check whether the user has selected an image or not.

Function Implementation: The function uses an event listener for the Run button to check the when the user has pressed the Run button , the src attribute of the input_layer is empty or not using the DOM property of JavaScript. If it is found empty it displays an alert box message. If it is not empty the input_layer is made visible to proceed with the experiment.

FormReset() :

Function Introduction: This functions resets all the forms of the page and sets the visibility of the input_layer and output_layer as “hidden”. It also makes the src attribute of the input_layer empty.