**Shri Ramdeobaba College of Engineering and Management, Nagpur-13.   
Department of Computer Science Engineering (AIML)  
CAP-309 - Fundamental of Digital Image & Video Processing  
Even Semester – 2023-24**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name:** | Om Ghumre | | | | |
| **Batch / Roll No.** | E3-43 | | | | |
| **Semester/Section:** | VI / E | | | | |
| **Date of Performance:** | 8 Feb 24 | | | | |
| **Date of Submission:** |  | | | | |
| **Particulars** | **Experiment Performance** | **Result & Discipline** | **Viva** | **Journal** | **Total** |
| **Max. Marks** | **03** | **03** | **03** | **01** | **10** |
| **Marks Obtained** |  |  |  |  |  |
| **Name &**  **Signature of Faculty** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | |

Practical No:- 02

**Aim:-**

Lab2- To study and perform basic arithmetic and logical operations used in image processing. Arithmetic Logical i) Addition i) AND operation ii) Subtraction ii) OR Operation iii) Multiplication iii) XOR Operation iv) Division iv) NOT Operation

**Software used:-** Pycharm/Jupyter/Colab

**Theory:-**

**1. Arithmetic Operations:**

**i) Addition:**

* Addition is a basic operation where pixel values at corresponding locations in two images are added together.
* It is often used for tasks such as image blending, contrast adjustment, and intensity transformation.

**ii) Subtraction:**

* Subtraction involves subtracting pixel values at corresponding locations in two images.
* It is used for tasks like image differencing, highlighting changes, and background removal.

**iii) Multiplication:**

* Multiplication is an operation where pixel values at corresponding locations in two images are multiplied.
* It is employed for tasks like image masking, contrast enhancement, and edge detection.

**iv) Division:**

* Division involves dividing pixel values at corresponding locations in two images.
* It can be used for tasks like normalization, where the pixel values are scaled to a certain range.

**2. Logical Operations:**

**i) AND Operation:**

* AND operation involves taking the logical AND of pixel values at corresponding locations in two images.
* It is used for tasks like image masking, where only certain regions are retained.

**ii) OR Operation:**

* OR operation involves taking the logical OR of pixel values at corresponding locations in two images.
* It is useful for tasks like combining multiple binary masks or merging segmented regions.

**iii) XOR Operation:**

* XOR (exclusive OR) operation involves taking the exclusive OR of pixel values at corresponding locations in two images.
* It is employed for tasks like image differencing, highlighting changes between two images.

**iv) NOT Operation:**

* NOT operation involves taking the logical NOT of pixel values, essentially inverting the pixel values.
* It is used for tasks like creating negative images or inverting binary masks.

**Code:-**

|  |
| --- |
| import cv2  image1 = cv2.imread("boat.png") image2 = cv2.imread("fruits.png") image2 = cv2.cvtColor(image2, cv2.COLOR\_BGR2GRAY) image1 = cv2.cvtColor(image1, cv2.COLOR\_BGR2GRAY)  added\_image = cv2.add(image1, image2) |

**Input image:-**

|  |
| --- |
|  |

**Output image:-**

|  |
| --- |
|  |

**Code:-**

|  |
| --- |
| import cv2  image1 = cv2.imread("boat.png") image2 = cv2.imread("fruits.png") image2 = cv2.cvtColor(image2, cv2.COLOR\_BGR2GRAY) image1 = cv2.cvtColor(image1, cv2.COLOR\_BGR2GRAY)  subtracted\_image = cv2.subtract(image1, image2) |

**Input image:-**

|  |
| --- |
|  |

**Output image:-**

|  |
| --- |
|  |

**Code:-**

|  |
| --- |
| import cv2  image1 = cv2.imread("boat.png") image2 = cv2.imread("fruits.png") image2 = cv2.cvtColor(image2, cv2.COLOR\_BGR2GRAY) image1 = cv2.cvtColor(image1, cv2.COLOR\_BGR2GRAY)  multiplied\_image = cv2.multiply(image1, 2) |

**Input image:-**

|  |
| --- |
|  |

**Output image:-**

|  |
| --- |
|  |

**Code:-**

|  |
| --- |
| import cv2  image1 = cv2.imread("boat.png") image2 = cv2.imread("fruits.png") image2 = cv2.cvtColor(image2, cv2.COLOR\_BGR2GRAY) image1 = cv2.cvtColor(image1, cv2.COLOR\_BGR2GRAY)  divided\_image = cv2.divide(image1,2) |

**Input image:-**

|  |
| --- |
|  |

**Output image:-**

|  |
| --- |
|  |

**Code:-**

|  |
| --- |
| import cv2  image1 = cv2.imread("boat.png") image2 = cv2.imread("fruits.png") binary1 = cv2.threshold(image1, 128, 255, cv2.THRESH\_BINARY)[1] binary2 = cv2.threshold(image2, 128, 255, cv2.THRESH\_BINARY)[1]  and\_image = cv2.bitwise\_and(binary1, binary2) |

**Input image:-**

|  |
| --- |
|  |

**Output image:-**

|  |
| --- |
|  |

**Code:-**

|  |
| --- |
| import cv2  image1 = cv2.imread("boat.png") image2 = cv2.imread("fruits.png") binary1 = cv2.threshold(image1, 128, 255, cv2.THRESH\_BINARY)[1] binary2 = cv2.threshold(image2, 128, 255, cv2.THRESH\_BINARY)[1]  or\_image = cv2.bitwise\_or(binary1, binary2) |

**Input image:-**

|  |
| --- |
|  |

**Output image:-**

|  |
| --- |
|  |

**Code:-**

|  |
| --- |
| import cv2  image1 = cv2.imread("boat.png") image2 = cv2.imread("fruits.png") binary1 = cv2.threshold(image1, 128, 255, cv2.THRESH\_BINARY)[1] binary2 = cv2.threshold(image2, 128, 255, cv2.THRESH\_BINARY)[1]  xor\_image = cv2.bitwise\_xor(binary1, binary2) |

**Input image:-**

|  |
| --- |
|  |

**Output image:-**

|  |
| --- |
|  |

**Code:-**

|  |
| --- |
| import cv2  image1 = cv2.imread("boat.png") image2 = cv2.imread("fruits.png") binary1 = cv2.threshold(image1, 128, 255, cv2.THRESH\_BINARY)[1] binary2 = cv2.threshold(image2, 128, 255, cv2.THRESH\_BINARY)[1]  not\_image = cv2.bitwise\_not(binary1) |

**Input image:-**

|  |
| --- |
|  |

**Output image:-**

|  |
| --- |
|  |

**Conclusion :** In This practical we have successfully performed arithmetic and logical operation on greyscale and binary image.