Dharampeth Polytechnic

Nagpur

Department of Computer Technology

Micro-Project Report On

Image Processing using OpenCV (Python) and Numpy

In Subject

Programming with Python

[P.W.P. - 22616]

Under the Guidance of

Prof. Sonal S. Mohurle

Submitted by

Yash Dattatraya Desai

[Enrollment # 1711880033]

Dharampeth Education Society's

DHARAMPETH POLYTECHNIC

CERTIFICATE

This is to certify that **Yash Dattatraya Desai [1711880033]** student of **Sixth Semester** of **Department of Computer Technology [CM-6I], Dharampeth Polytechnic, Nagpur** has completed his project titled : "**Image Processing using OpenCV (Python) and Numpy**" for the subject **"Programming with Python" [P.W.P. - 22616]** during the academic year **2019-20.**

Signature of Faculty
[Prof. Sonal S. Mohurle]

Signature of Head of Department
[Prof. Meeta B. Fadnavis]

ABSTRACT

Image processing is the processing of an image through an algorithm, changing the values in the image data structure. OpenCV (with Numpy) allows a much wider range of algorithms to be applied to the input data.

OpenCV supports a wide variety of programming languages like C++, Python, Java etc and is available on different platforms including Windows, Linux, OS X, Android, iOS etc. Also, interfaces based on CUDA and OpenCL are also under active development for high-speed GPU operations. OpenCV-Python is the Python API of OpenCV. It combines the best qualities of OpenCV C++ API and Python language.

As Python is Open-Source, so does the Open-CV and Numpy the making cost of this project is zero. Thanks to all Open-Source Technologies creating another technology on top of the open-source environment has a big impact in current days.

As I am developer, who developed this program , I may offer this program as an Open-Source project. Maning any one can see my source code. As well as I can make it MIT, CC, Apache Open-Source Licensed so that any one can jump in and try to modify the code.

Best part of this industry is most of the technology is open source, you just need an spark in you to make something. There is no Capital investment, Just requires skill, time, and some other factors.

INTRODUCTION

Image = "2-Dimensional Array", we can consider image as an array having the 2-Dimesions within each element of the array the basic image data.

For example an **RGBA image which stands for (Red – Green – Blue - Alpha) may have 4 values in each element of array. This values can be intensiy of each pixel total combined can form a pixel.**

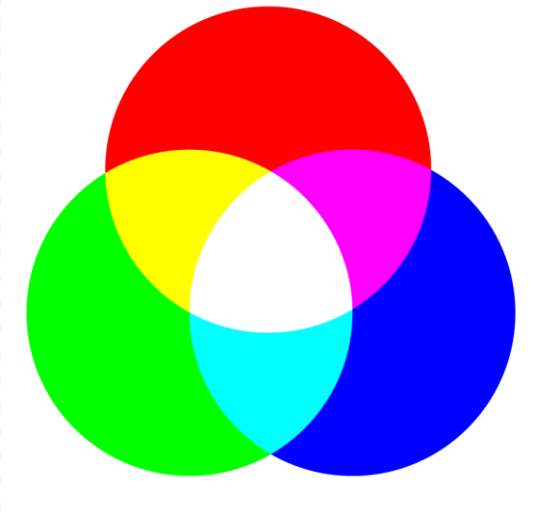
An RGBA color space looks like this represented in this images.

Each Pixel in the image is made of :

RED
GREEN
BLUE
ALPHA

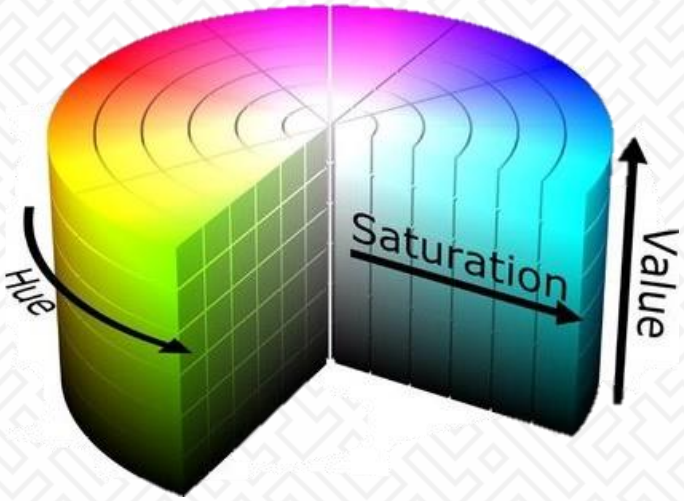
Sample Length:	8								8								8								8							
Channel Membership:	**Red**								**Green**								**Blue**								**Alpha**							
Bit Number:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

There are other Image Spaces / Models. But I Used only RGB, RGBA, HSV and LUV only, as they are popular and widwly used in industry as well as can be implemented with less space.



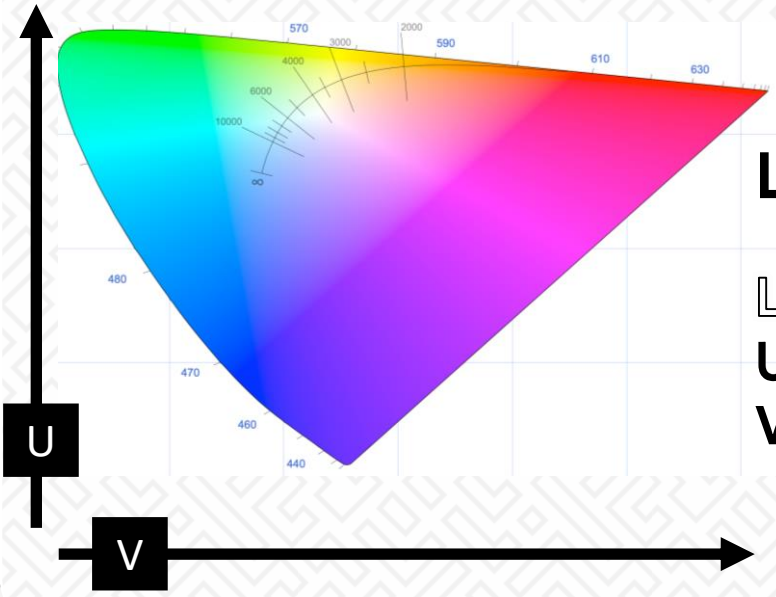
RGB Color Space

RED
GREEN
BLUE



HSV Color Space

HUE
SATURATION
VALUE



LUV Color Space

LUMINANCE
U Chroma
V Chroma

FILE STRUCTURE

- **Image_effects**
  - edge_detect.py
  - hsv_color_space.py
  - rgb_color_space.py
  - luv_color_space.py
- greeting_msg.py
- img_import_n_export.py
- main.py
- prerequisites_checker.py

SOURCE CODE

main.py

```
import greetings_msg as msg

msg.intro()

import prerequisites_checker
import img_import_n_export as __img__
import image_effects.edge_detect as edge_detect
import image_effects.hsv_color_space as effects_hsv
import image_effects.luv_color_space as effects_luv
import image_effects.rgb_color_space as effects_rgb
raw_img_data = __img__.import_image()
I = raw_img_data

operations_tuple = (
    "Yash Desai - Empty Space",
    "Convert Image Data from OpenCV's BRG to RGB",
    "Convert Image Data from OpenCV's BRG to RGBA",
    "Convert Image to Grayscale Image",
    "Extract Red Channel from Image RGB values",
    "Extract Green Channel from Image RGB values",
    "Extract Blue Channel from Image RGB values",
    "Set custom Image Threshold",
    "Set custom Image Individual RGB Threshold",
    "[HSV] Convert Image Data from OpenCV's BRG to HSV",
    "[HSV] Extract Hue Channel from Image HSV values",
    "[HSV] Extract Saturation Channel from Image HSV values",
    "[HSV] Extract Value Channel from Image HSV values",
    "[LUV] Convert Image Data from OpenCV's BRG to LUV",
    "[LUV] Extract Luminance Component from Image LUV values“,
    "[LUV] Extract U Chroma Color Component from Image LUV values",
    "[LUV] Extract V Chroma Color Component from Image LUV values",
    "[SOBEL E.D. ALGO] Based on Horizontal Axis",
    "[SOBEL E.D. ALGO] Based on Vetical Axis",
    "[SOBEL E.D. ALGO] Based on Bitwised OR",
    "[SOBEL E.D. ALGO] Based on Bitwised AND",
    "[SOBEL E.D. ALGO] Based on Bitwised XOR",
    "[SOBEL E.D. ALGO] Based on Bitwised NOT",
    "[Laplacian E.D. ALGO]",
    "[Canny E.D. ALGO] Based on Custom Thresholds",
```

```
    "Change image file Format",
    "Get Image Height and Width"
)

count = 0
print("\n\n[ ? ]  Enter the operation # you would like to perform on th
e image : \n")
for operation_name in operations_tuple:
    if(count == 0) :
        count+=1
        continue
    print(f"  +---  {count} -> {operation_name}")
    count+=1

SELECT = int(input("\nEnter the Operation # (1-25) : "))

if SELECT < 1 or SELECT > 25 :
    print("[ X ]  Invalid Input")
    msg.urgent_exit()
else :
    if SELECT == 1:
        I = __img__.preview_image(operations_tuple[SELECT],effects_rgb.
bgr_2_rgb(raw_img_data))
    elif SELECT == 2:
        I = __img__.preview_image(operations_tuple[SELECT],effects_rgb.
bgr_2_rgba(raw_img_data))
    elif SELECT == 3:
        I = __img__.preview_image(operations_tuple[SELECT],effects_rgb.
grayscale_image(raw_img_data))
    elif SELECT == 4:
        I = __img__.preview_image(operations_tuple[SELECT],effects_rgb.
red_channel_only(raw_img_data))
    elif SELECT == 5:
        I = __img__.preview_image(operations_tuple[SELECT],effects_rgb.
green_channel_only(raw_img_data))
    elif SELECT == 6:
        I = __img__.preview_image(operations_tuple[SELECT],effects_rgb.
blue_channel_only(raw_img_data))
    elif SELECT == 7:
        threshold = int(input("\n[ ? ]  Threshold Value : "))
        I = __img__.preview_image(operations_tuple[SELECT],effects_rgb.
custom_rgb_threshold(raw_img_data,threshold))
    elif SELECT == 8:
        r_threshold = int(input("\n[ ? ]  Red Threshold Value : "))
        g_threshold = int(input("[ ? ]  Green Threshold Value : "))
        b_threshold = int(input("[ ? ]  Blue Threshold Value : "))
```

```
        I = __img__.preview_image(operations_tuple[SELECT],effects_rgb.
custom_rgb_gain_or_loss(raw_img_data,r_threshold,
                        g_threshold,b_threshold))
    elif SELECT == 9:
        I = __img__.preview_image(operations_tuple[SELECT],effects_hsv.
bgr_2_hsv(raw_img_data))
    elif SELECT == 10:
        I = __img__.preview_image(operations_tuple[SELECT],effects_hsv.
hue_channel_only(raw_img_data))
    elif SELECT == 11:
        I = __img__.preview_image(operations_tuple[SELECT],effects_hsv.
saturation_channel_only(raw_img_data))
    elif SELECT == 12:
        I = __img__.preview_image(operations_tuple[SELECT],effects_hsv.
value_channel_only(raw_img_data))
    elif SELECT == 13:
        I = __img__.preview_image(operations_tuple[SELECT],effects_luv.
bgr_2_luv(raw_img_data))
    elif SELECT == 14:
        I = __img__.preview_image(operations_tuple[SELECT],effects_luv.
luminance_component_only(raw_img_data))
    elif SELECT == 15:
        I = __img__.preview_image(operations_tuple[SELECT],effects_luv.
u_chroma_component_only(raw_img_data))
    elif SELECT == 16:
        I = __img__.preview_image(operations_tuple[SELECT],effects_luv.
v_chroma_component_only(raw_img_data))
    elif SELECT == 17:
        I = __img__.preview_image(operations_tuple[SELECT],edge_detect.
sobel_horizontal_edge_detect_algo(raw_img_data))
    elif SELECT == 18:
        I = __img__.preview_image(operations_tuple[SELECT],edge_detect.
sobel_vertical_edge_detect_algo(raw_img_data))
    elif SELECT == 19:
        I = __img__.preview_image(operations_tuple[SELECT],edge_detect.
sobel_bothaxis_bitws_or_edge_detect_algo(raw_img_data))
    elif SELECT == 20:
        I = __img__.preview_image(operations_tuple[SELECT],edge_detect.
sobel_bothaxis_bitws_and_edge_detect_algo(raw_img_data))
    elif SELECT == 21:
        I = __img__.preview_image(operations_tuple[SELECT],edge_detect.
sobel_bothaxis_bitws_xor_edge_detect_algo(raw_img_data))
    elif SELECT == 22:
        I = __img__.preview_image(operations_tuple[SELECT],edge_detect.
sobel_bothaxis_bitws_not_edge_detect_algo(raw_img_data))
    elif SELECT == 23:
```

prerequisites_checker.py

```
import greetings_msg as msg

print("[ \u25B2 ]  Checking Prerequisites......\n")



print("[ ! ]  Trying to import \"Numpy\".....")
try:
    import numpy as npy
except ImportError as NUMPY_IMPORT_EXCEPTION:
    print("[ \u2717 ]  Numpy couldn't be imported.....")
    print(f"\n[ \u2717 ]  Error Message : {NUMPY_IMPORT_EXCEPTION}")
    msg.urgent_exit()

else:
    print(f"[ \u2713 ]  Numpy [ ver. {npy.version.full_version} ] ---
 Imported Successfully.....")



print("\n[ ! ]  Trying to import \"Open-CV\".....")
try:
    import cv2 as open_cv
except ImportError as OPENCV_IMPORT_EXCEPTION:
    print("[ \u2717 ]  Open-CV couldn't be imported.....")
    print(f"\n[ \u2717 ]  Error Message : {OPENCV_IMPORT_EXCEPTION}")
    msg.urgent_exit()
else:
    print(f"[ \u2713 ]  Open-
CV [ ver. {open_cv.version.opencv_version} ] ---
 Imported Successfully.....")



print("\n\n[ \u2605 ]  All prerequisites checks are completed......")
```

greetings_msg.py

```
def intro():
    print("\n\n\n")
    print("[ * ]  Welcome to Yash Desai's PWP Micro-Project....")
    print("[ * ]  Entitled : Image Processing using OpenCV (Python) and
 Numpy\n")

def urgent_exit():
    print("\n\n[ ! ] Terminating the program execution.....")
    quit()

def greet_exit():
    print("\n\n[ ! ] Thanks for using the program.....")
    urgent_exit()

def preview_image_exit(Window_Name) :
    print(f"[ ! ]  Press any key to close preview window ({Window_Name}
)...")

def processing_img():
    print("\n\n[ ! ]  Processing Image Data. Please Wait.....")
```

image_effects \ rgb_color_space.py

```
import numpy as npy
import cv2 as open_cv
import greetings_msg as msg
import img_import_n_export as __img__

def grayscale_image(Image_Raw_Data):
    return open_cv.cvtColor(Image_Raw_Data, open_cv.COLOR_BGR2GRAY)

def bgr_2_rgb(Image_Raw_Data):
    return open_cv.cvtColor(Image_Raw_Data, open_cv.COLOR_BGR2RGB)

def bgr_2_rgba(Image_Raw_Data):
    return open_cv.cvtColor(Image_Raw_Data, open_cv.COLOR_BGR2RGBA)

def blue_channel_only(Image_Raw_Data):
    msg.processing_img()
    for y in range(0, __img__.image_height(Image_Raw_Data)):
        for x in range(0, __img__.image_width(Image_Raw_Data)):
            Image_Raw_Data[y,x,1] = 0
            Image_Raw_Data[y,x,2] = 0
    return Image_Raw_Data

def green_channel_only(Image_Raw_Data):
    msg.processing_img()
    for y in range(0, __img__.image_height(Image_Raw_Data)):
        for x in range(0, __img__.image_width(Image_Raw_Data)):
            Image_Raw_Data[y,x,0] = 0
            Image_Raw_Data[y,x,2] = 0
    return Image_Raw_Data

def red_channel_only(Image_Raw_Data):
    msg.processing_img()
    for y in range(0, __img__.image_height(Image_Raw_Data)):
        for x in range(0, __img__.image_width(Image_Raw_Data)):
            Image_Raw_Data[y,x,0] = 0
            Image_Raw_Data[y,x,1] = 0
    return Image_Raw_Data

def custom_rgb_threshold(Image_Raw_Data,threshold_factor=0):
    msg.processing_img()
    for y in range(0, __img__.image_height(Image_Raw_Data)):
        for x in range(0, __img__.image_width(Image_Raw_Data)):
            if (Image_Raw_Data[y,x] >= threshold_factor).any() :
                Image_Raw_Data[y,x] = 255
```

```
            else:
                Image_Raw_Data[y,x] = 0
    return Image_Raw_Data

def custom_rgb_gain_or_loss(Image_Raw_Data,R_factor=0,G_factor=0,B_fact
or=0):
    msg.processing_img()
    for y in range(0, __img__.image_height(Image_Raw_Data)):
        for x in range(0, __img__.image_width(Image_Raw_Data)):
            if Image_Raw_Data[y,x,0] >= B_factor :
                Image_Raw_Data[y,x,0] = 255
            else:
                Image_Raw_Data[y,x,0] = 0

            if Image_Raw_Data[y,x,1] >= G_factor :
                Image_Raw_Data[y,x,1] = 255
            else:
                Image_Raw_Data[y,x,1] = 0

            if Image_Raw_Data[y,x,2] >= R_factor :
                Image_Raw_Data[y,x,2] = 255
            else:
                Image_Raw_Data[y,x,2] = 0
    return Image_Raw_Data
```

image_effects \ hsv_color_space.py

```
import numpy as npy
import cv2 as open_cv
import greetings_msg as msg
import img_import_n_export as __img__

def bgr_2_hsv(Image_Raw_Data):
    return open_cv.cvtColor(Image_Raw_Data, open_cv.COLOR_BGR2HSV)

def hue_channel_only(Image_Raw_Data):
    msg.processing_img()
    Image_Raw_Data = bgr_2_hsv(Image_Raw_Data)
    for y in range(0, __img__.image_height(Image_Raw_Data)):
        for x in range(0, __img__.image_width(Image_Raw_Data)):
            Image_Raw_Data[y,x,1] = 0
            Image_Raw_Data[y,x,2] = 0
    return Image_Raw_Data

def saturation_channel_only(Image_Raw_Data):
    msg.processing_img()
```

```
    Image_Raw_Data = bgr_2_hsv(Image_Raw_Data)
    for y in range(0, __img__.image_height(Image_Raw_Data)):
        for x in range(0, __img__.image_width(Image_Raw_Data)):
            Image_Raw_Data[y,x,0] = 0
            Image_Raw_Data[y,x,2] = 0
    return Image_Raw_Data

def value_channel_only(Image_Raw_Data):
    msg.processing_img()
    Image_Raw_Data = bgr_2_hsv(Image_Raw_Data)
    for y in range(0, __img__.image_height(Image_Raw_Data)):
        for x in range(0, __img__.image_width(Image_Raw_Data)):
            Image_Raw_Data[y,x,0] = 0
            Image_Raw_Data[y,x,1] = 0
    return Image_Raw_Data
```

image_effects \ luv_color_space.py

```
import numpy as npy
import cv2 as open_cv
import greetings_msg as msg
import img_import_n_export as __img__

def bgr_2_luv(Image_Raw_Data):
    return open_cv.cvtColor(Image_Raw_Data, open_cv.COLOR_BGR2LUV)

def luminance_component_only(Image_Raw_Data):
    msg.processing_img()
    Image_Raw_Data = bgr_2_luv(Image_Raw_Data)
    for y in range(0, __img__.image_height(Image_Raw_Data)):
        for x in range(0, __img__.image_width(Image_Raw_Data)):
            Image_Raw_Data[y,x,1] = 0
            Image_Raw_Data[y,x,2] = 0
    return Image_Raw_Data

def u_chroma_component_only(Image_Raw_Data):
    msg.processing_img()
    Image_Raw_Data = bgr_2_luv(Image_Raw_Data)
    for y in range(0, __img__.image_height(Image_Raw_Data)):
        for x in range(0, __img__.image_width(Image_Raw_Data)):
            Image_Raw_Data[y,x,0] = 0
            Image_Raw_Data[y,x,2] = 0
    return Image_Raw_Data
```

```
def v_chroma_component_only(Image_Raw_Data):
    msg.processing_img()
    Image_Raw_Data = bgr_2_luv(Image_Raw_Data)
    for y in range(0, __img__.image_height(Image_Raw_Data)):
        for x in range(0, __img__.image_width(Image_Raw_Data)):
            Image_Raw_Data[y,x,0] = 0
            Image_Raw_Data[y,x,1] = 0
    return Image_Raw_Data
```

image_effects \ edge_detect.py

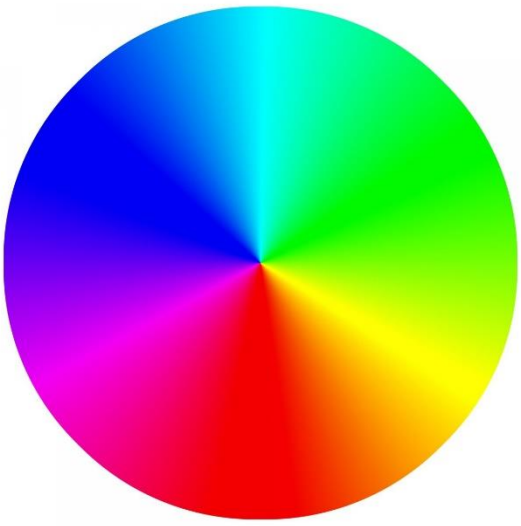
```
import numpy as npy
import cv2 as open_cv
import greetings_msg as msg
import img_import_n_export as __img__

def sobel_horizontal_edge_detect_algo(Image_Raw_Data):
    return open_cv.Sobel(Image_Raw_Data,open_cv.CV_64F,0,1)

def sobel_vertical_edge_detect_algo(Image_Raw_Data):
    return open_cv.Sobel(Image_Raw_Data,open_cv.CV_64F,1,0)

def sobel_bothaxis_bitws_or_edge_detect_algo(Image_Raw_Data):
    return open_cv.bitwise_or(sobel_horizontal_edge_detect_algo(Image_R
aw_Data),
        sobel_vertical_edge_detect_algo(Image_Raw_Data))

def sobel_bothaxis_bitws_and_edge_detect_algo(Image_Raw_Data):
    return open_cv.bitwise_and(sobel_horizontal_edge_detect_algo(Image_
Raw_Data),
        sobel_vertical_edge_detect_algo(Image_Raw_Data))

def sobel_bothaxis_bitws_xor_edge_detect_algo(Image_Raw_Data):
    return open_cv.bitwise_xor(sobel_horizontal_edge_detect_algo(Image_
Raw_Data),
        sobel_vertical_edge_detect_algo(Image_Raw_Data))

def sobel_bothaxis_bitws_not_edge_detect_algo(Image_Raw_Data):
    return open_cv.bitwise_not(sobel_horizontal_edge_detect_algo(Image_
Raw_Data),
        sobel_vertical_edge_detect_algo(Image_Raw_Data))

def laplacian_edge_detect_algo(Image_Raw_Data):
    return open_cv.Laplacian(Image_Raw_Data,open_cv.CV_64F)

def canny_edge_detect_algo(Image_Raw_Data,Threshold1,Threshold2):
    return open_cv.Canny(Image_Raw_Data,Threshold1,Threshold2)
```

INPUT (Example)

Python 3.X.X

OpenCV

Numpy

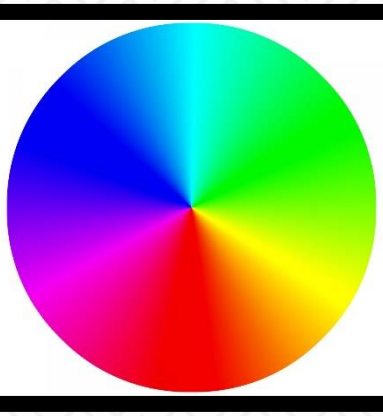


color_wheel.jpg

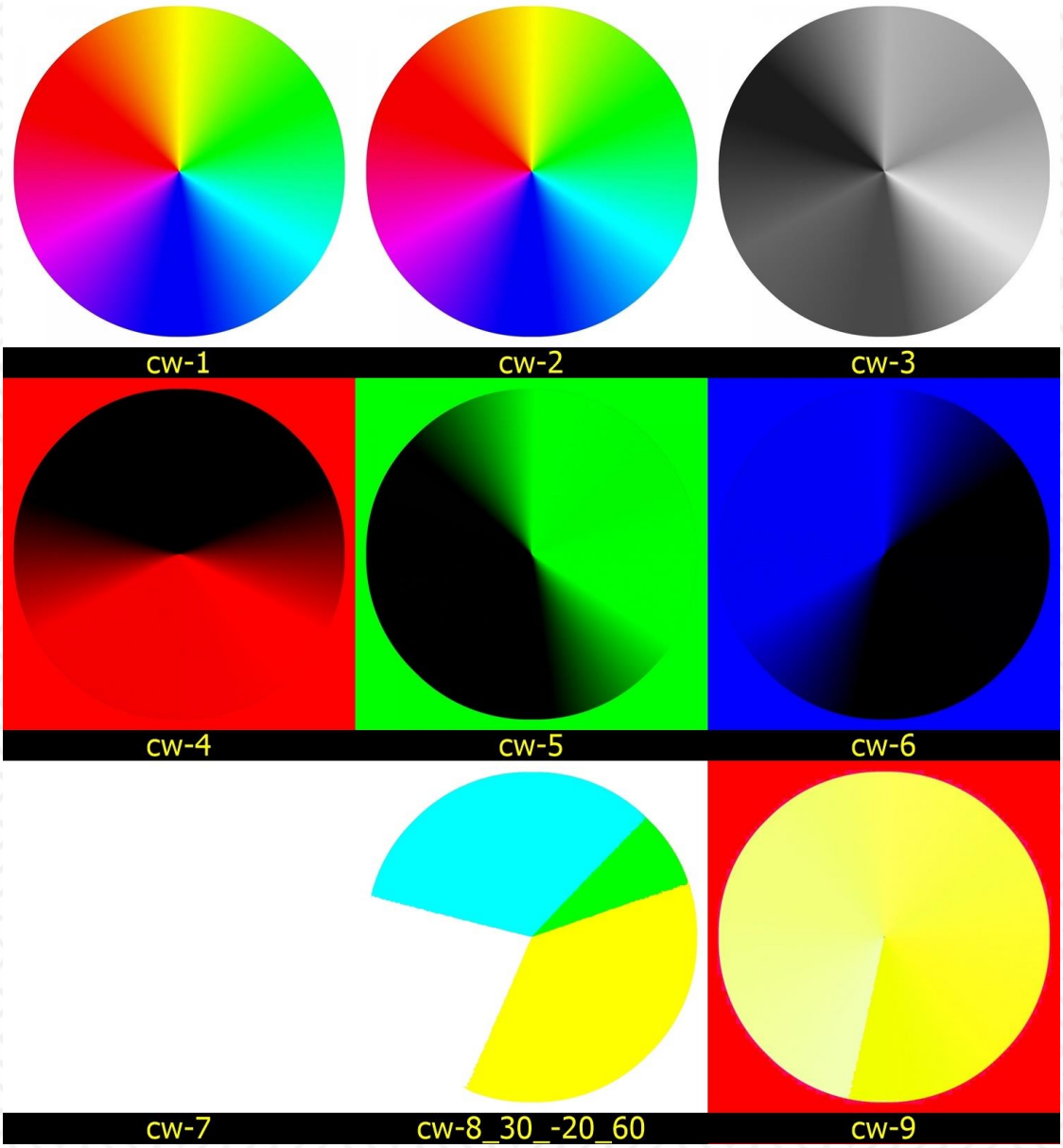
mona-lisa.jpg

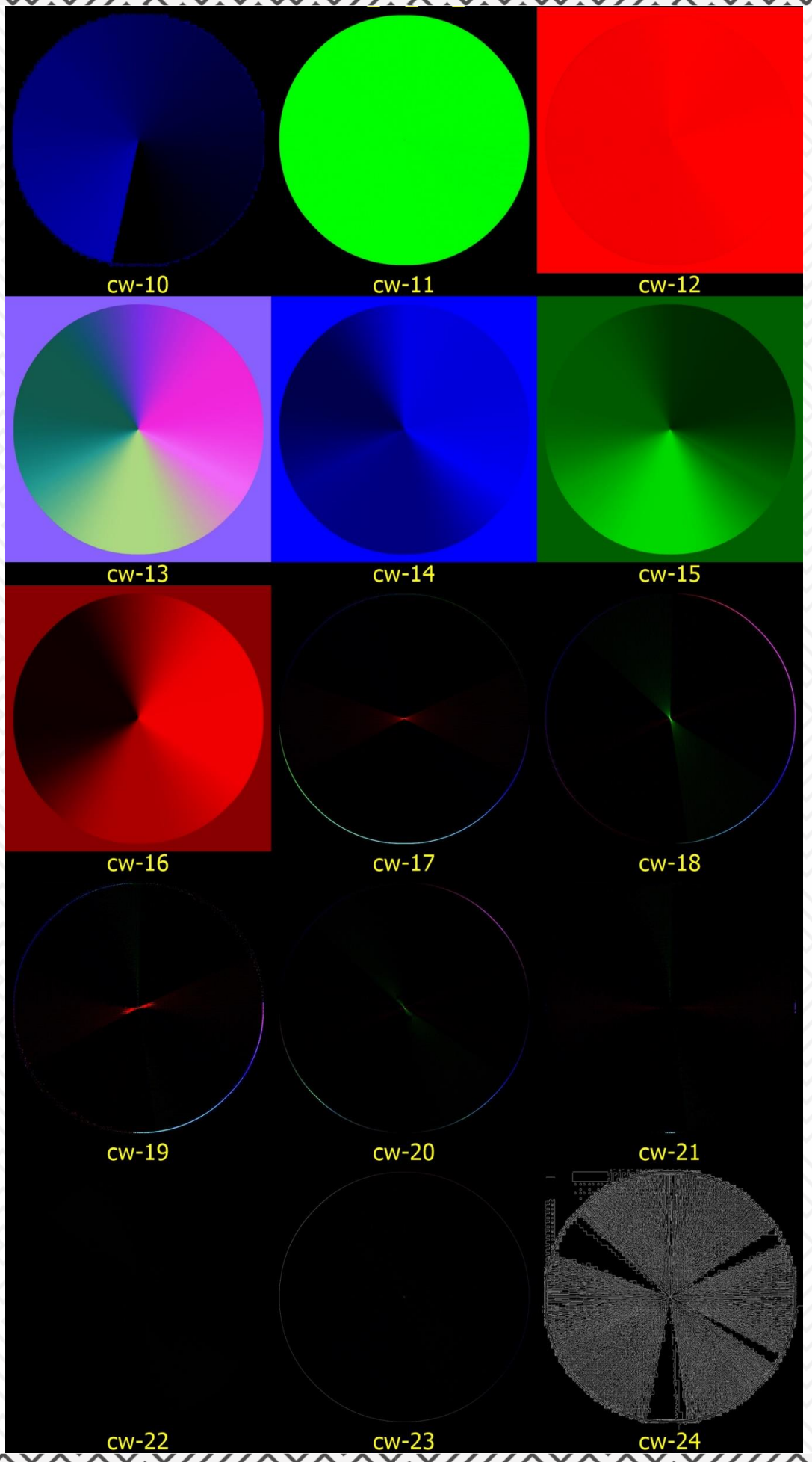
holi.jpg

OUTPUT



color_wheel.jpg




cw-10
cw-11
cw-12
cw-13
cw-14
cw-15
cw-16
cw-17
cw-18
cw-19
cw-20
cw-21
cw-22
cw-23
cw-24

mona-lisa.jpg


mona_lisa-7
mona_lisa-8
mona_lisa-9
mona_lisa-10
mona_lisa-11
mona_lisa-12
mona_lisa-13
mona_lisa-14
mona_lisa-15


mona_lisa-16
mona_lisa-17
mona_lisa-18
mona_lisa-19
mona_lisa-20
mona_lisa-21
mona_lisa-22
mona_lisa-23
mona_lisa-24

holi.jpg

holi-1

holi-2

holi-3

holi-4

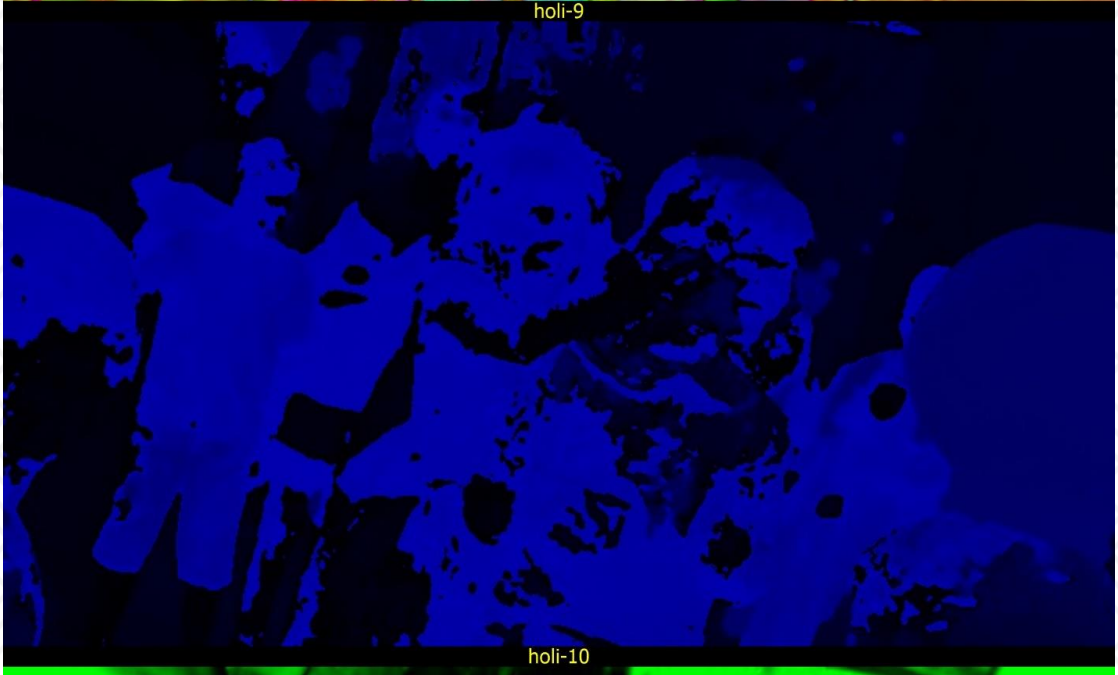
holi-5

holi-6

holi-7

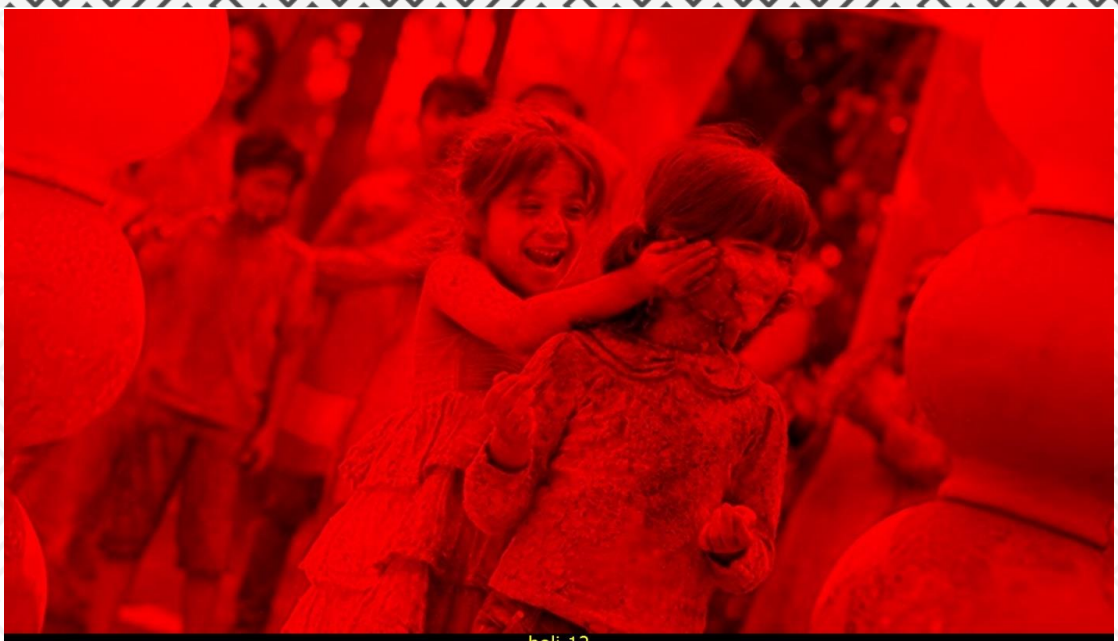
holi-8

holi-9

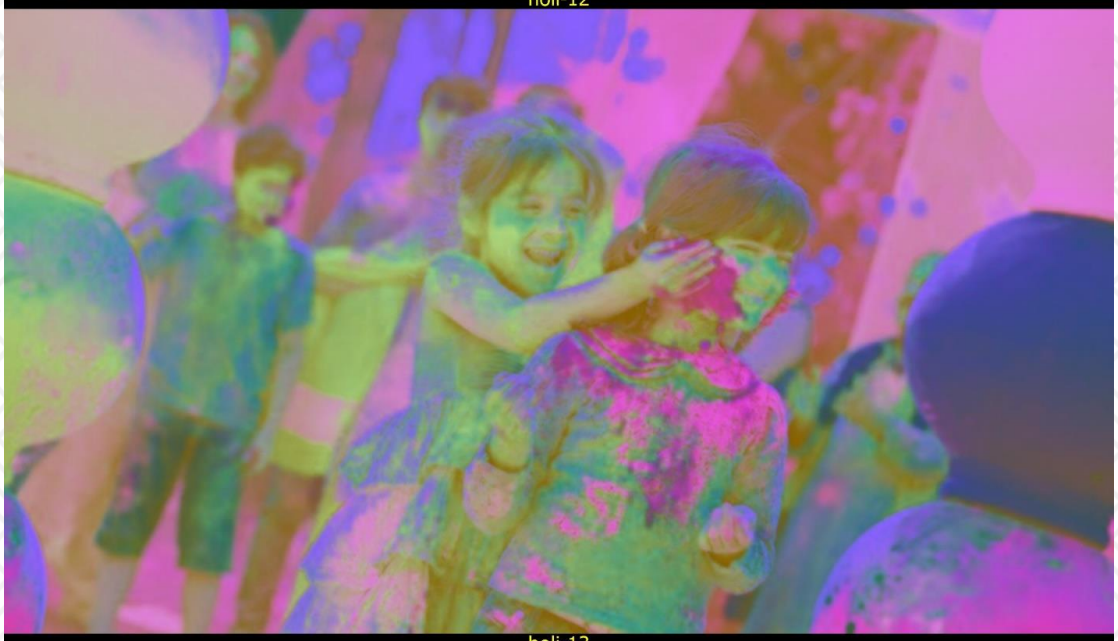


holi-10

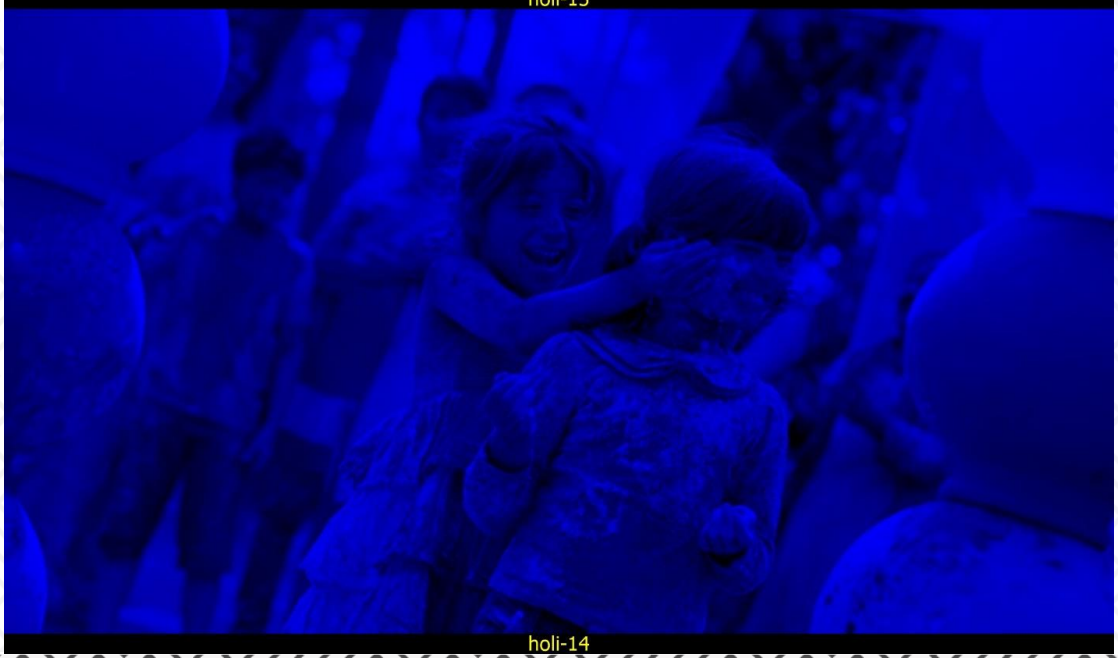
holi-11



holi-12



holi-13



holi-14

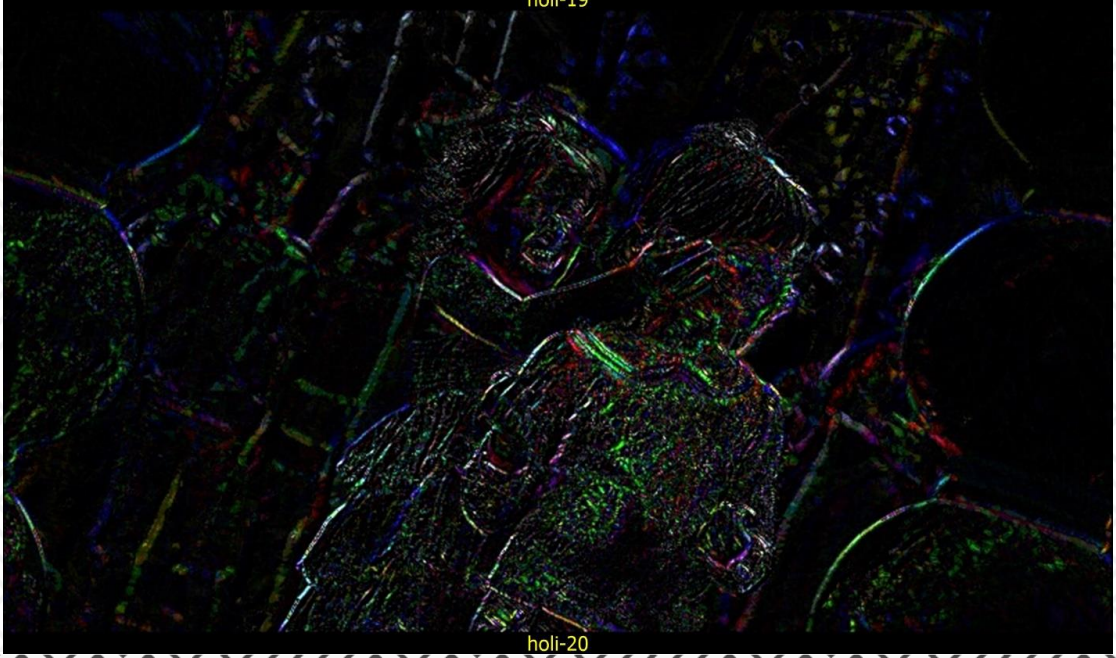
holi-15

holi-16

holi-17

holi-18

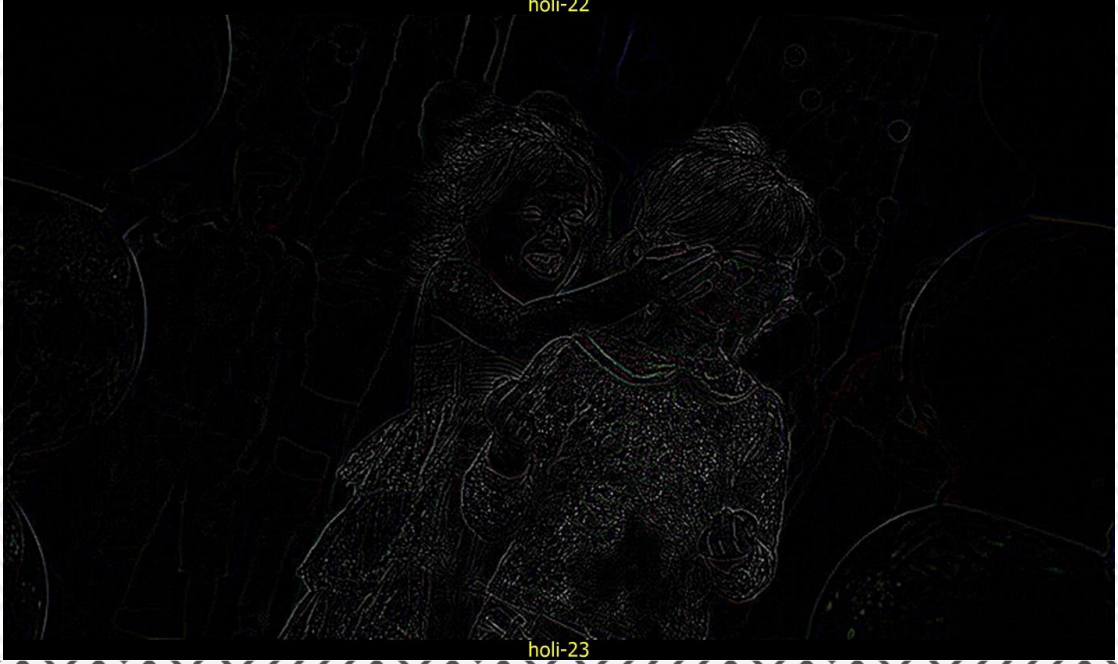
holi-19



holi-20

holi-21

holi-22



holi-23


holi-24