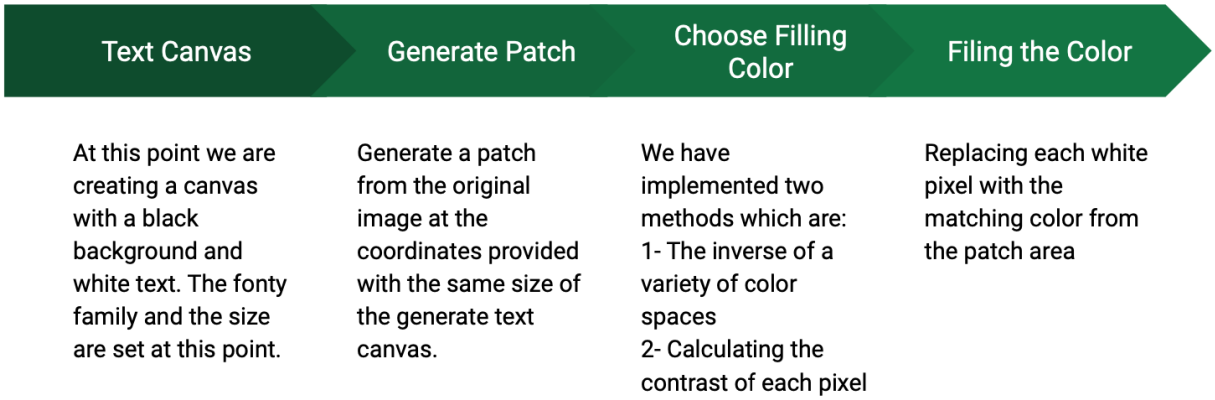
Compare image conversion techniques

In our project we were tasked with providing a method to produce readable text in a video. We propose two methods, HSV inversion and Contrast transformation. These two methods are utilized to transform the entire image, where the text will then be masked by the transformed image to produce a text that is readable on top of the original image.



For HSV inversion we transform the images current color space of red green blue, RGB, into a hue saturation value, HSV, color space. Then we flip the values of the HSV space for the entire image using the bitwise not cv2 function. The method is pretty simple and fairly effective in generating readable text on top of the original image.

$$image_{mask} = \sim image_{hsv}$$

For the contrast transformation we implement a WCAG method usually used to test the contrast level of text in a browser to produce a contrasting color for each pixel of the original. In our implementation we tranform each color value to a linear color space, where our expected input is a color value from sRGB. We expect that each color value in sRGB is in the range of 0 to 1. This method of transformation is known as the reverse transformation, referred to as $rt(c)$. Below is our function for converting a sRGB color value to Linear.

$$rt(c) = \begin{cases} 0.0002 & \text{if c} < 0.0002 \\ \frac{c+0.055}{1.055}^{2.4} & \text{if c} > 0.04045 \\ \frac{c}{12.92} & \text{else} \end{cases}$$

$$\{c_{rl}, c_{gl}, c_{bl}\} = \{rt(c_r), rt(c_g), rt(c_b)\}$$

This converts the color into linear space. We refer to our color channels as c_r for red, c_g, for green, and r_b for blue. Once the color is in linear space we can calculate its contrast depending on its relative luminance, and determine the relative luminance it needs to be at to have a contrast difference of $contrast_{diff}$. $contrast_{diff}$ is provided as an input for our program, the `-cl` parameter. The minumum $contrast_{diff}$ is 4.5 as it's the least accessible color for large texts, 7.5 for small texts. This will allow the text to be readable to the human eye, even if the text color and background color is the same color palette.

$$contrast_{current} = \frac{1.05}{(0.2126 * c_{rl}) + (0.7152 * c_{gl}) + (0.0722 * c_{bl})}$$

$$contrast_{desired} = \begin{cases} contrast_{current} + contrast_{diff} & \text{if } contrast_{current} - contrast_{diff} < 0 \\ 21.0 & \text{if } contrast_{current} + contrast_{diff} \geq 21 \\ contrast_{current} - contrast_{diff} & \text{else} \end{cases}$$

$$rl_{desired} = \frac{1.05}{contrast_{desired}} - 0.05$$

From our desired relative luminance we generate luminance constants for the red green and blue channels in linear space, and the minimum and maximum constants of the color for each channel. Cr, Cg, Cb are the channel constants. $Cr_{min}, Cg_{min}, Cb_{min}$ are the minimum constants for each channel in linear space. $Cr_{max}, Cg_{max}, Cb_{max}$ are the maximum constants for each channel in linear space.

$$C = \frac{rl_{desired}}{(0.2126 * c_{rl}) + (0.7152 * c_{gl}) + (0.0722 * c_{bl})}$$

$$Cr = C, Cg = C, Cb = C$$

$$Cr_{max} = \frac{0.00000390625}{c_{rl}}, Cg_{max} = \frac{0.00000390625}{c_{gl}}, Cb_{max} = \frac{0.00000390625}{c_{bl}}$$

$$Cr_{max} = \frac{1}{c_{rl}}, Cg_{max} = \frac{1}{c_{gl}}, Cb_{max} = \frac{1}{c_{bl}}$$

From the constants we can then set the new linear color channels with the constants, but only if the color value and the resulting constants aren't met with 6 different scenarios. The scenarios are represented below as boolean values.

$$S_r = Cr_{min} > CrorCr_{max} < Cr$$
$$S_g = Cg_{min} > CgorCg_{max} < Cg$$
$$S_b = Cb_{min} > CborCb_{max} < Cb$$
$$S_{rg} = S_r and S_g$$
$$S_{rb} = S_r and S_b$$
$$S_{gb} = S_g and S_b$$

To resolve the scenarios we set the constant values unaffected by the scenarios relative to the current color value and the channels constant. $set(c_i, C_i)$ represents the change below.

$$set(c_i, C_i) = \begin{cases} \frac{1}{c_i} & \text{if } \frac{1}{c_i} < C_i \\ \frac{0.00000390625}{c_i} & \text{if } \frac{0.00000390625}{c_i} > C_i \\ C_i & \text{else} \end{cases}$$

When we come across a scenario we alter the constants of the color based. The alterations are reflected below.

$$\{C_r, C_g, C_b\} = \begin{cases} \{\frac{rl_{desired} - (0.7152 * c_{gl} * C_g) - (0.0722 * c_{bl} * C_b)}{0.2126 * c_{rl}}, set(c_{gl}, C_g), set(c_{bl}, C_b)\} & \text{if } S_{gb} \\ \{set(c_{rl}, C_r), \frac{rl_{desired} - (0.2126 * c_{rl} * C_r) - (0.0722 * c_{bl} * C_b)}{0.7152 * c_{gl}}, set(c_{bl}, C_b)\} & \text{if } S_{rb} \\ \{set(c_{rl}, C_r), set(c_{gl}, C_g), \frac{rl_{desired} - (0.2126 * c_{rl} * C_r) - (0.7152 * c_{gl} * C_g)}{0.0722 * c_{bl}}\} & \text{if } S_{rg} \\ \{\frac{rl_{desired} - (0.0722 * c_{bl} * C_b)}{(0.2126 * c_{rl}) + (0.7152 * c_{gl})}, C_r, set(c_{bl}, C_b)\} & \text{if } S_b \\ \{\frac{rl_{desired} - (0.7152 * c_{gl} * C_g)}{(0.2126 * c_{rl}) + (0.0722 * c_{bl})}, set(c_{gl}, C_g), C_r\} & \text{if } S_g \\ \{set(c_{rl}, C_r), \frac{rl_{desired} - (0.2126 * c_{rl} * C_r)}{(0.7152 * c_{gl}) + (0.0722 * c_{bl})}, C_g\} & \text{if } S_r \\ \{C_r, C_g, C_b\} & \text{else} \end{cases}$$

From the resulting constants of $\{C_r, C_g, C_b\}$ we can then convert them back into sRGB space using the $t(c)$ function shown below.

$$t(c) = \begin{cases} 0.0002 & \text{if c} < 0.0002 \\ (c^{\frac{1}{2.4}} * 1.055) - 0.055 & \text{if c} > 0.003131594552688991 \\ c * 12.92 & \text{else} \end{cases}$$

$$c_r = t(C_r), c_g = t(C_g), c_b = t(C_b)$$

This method is applied to each unique color value to get the contrast transformation of the entire image.

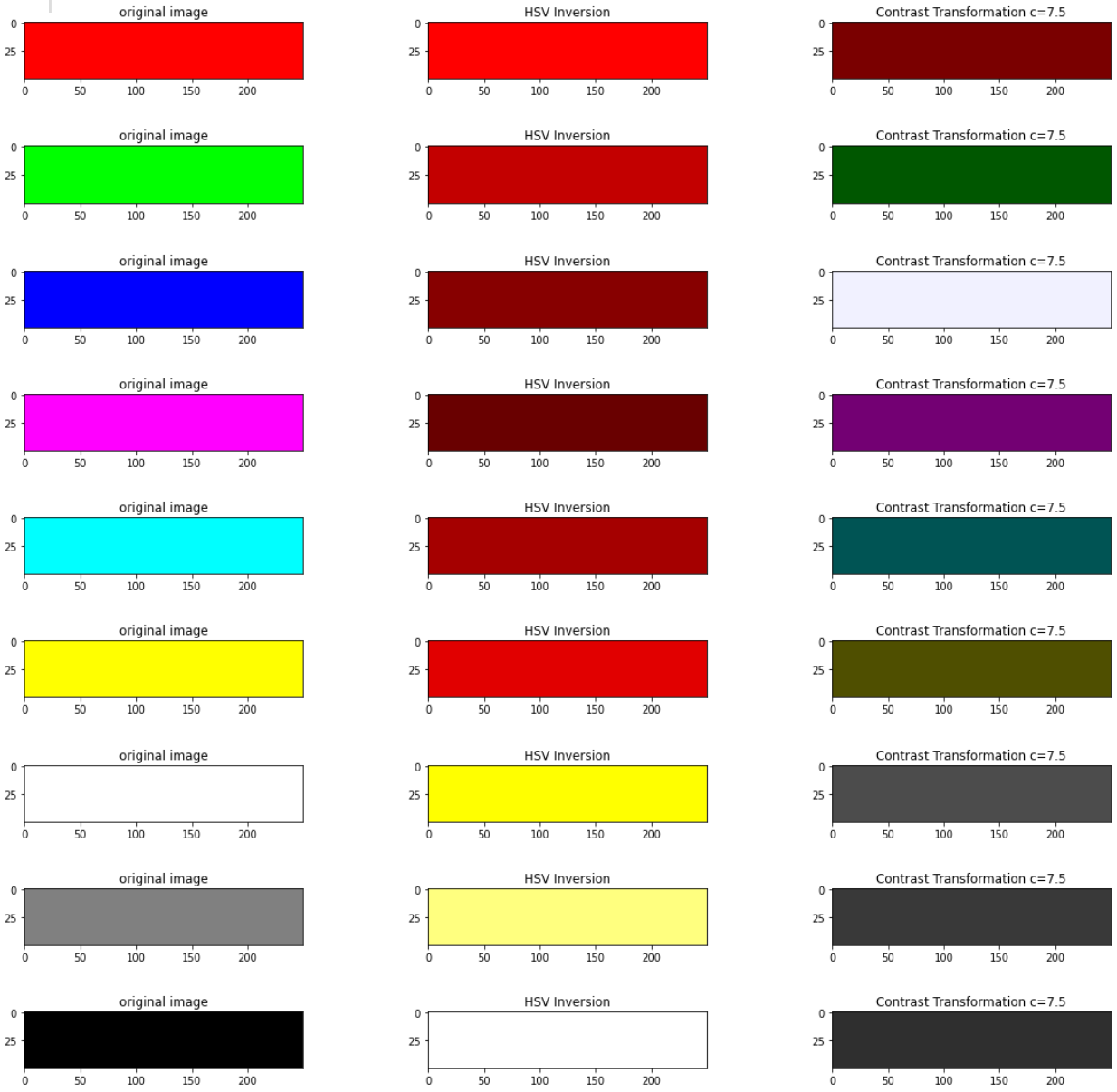
Experimental Results

Below we compare the outputs of the HSV inversion and Contrast transformation for several different images, most of strict color. We do this as a litmus test for both methods. In each plot we provide the original image on the left, the HSV inversion image in the center, and the Constrast transformation method on the right.

In [4]:

```python
from contrast.convert import Convert
import matplotlib.pyplot as plt
import numpy as np
import cv2

def generateTestImages(size = 50):
    blank_image = np.zeros((size,size*5,3)).astype('uint8')
    red = blank_image.copy();
    red[:,:,0] = 255
    green = blank_image.copy();
    green[:,:,1] = 255
    blue = blank_image.copy();
    blue[:,:,2] = 255
    magenta = blank_image.copy();
    magenta[:,:,0] = 255
    magenta[:,:,2] = 255
    cyan = blank_image.copy();
    cyan[:,:,1] = 255
    cyan[:,:,2] = 255
    yellow = blank_image.copy();
    yellow[:,:,0] = 255
    yellow[:,:,1] = 255
    white = blank_image.copy();
    white[:,:,0] = 255
    white[:,:,1] = 255
    white[:,:,2] = 255
    gray = blank_image.copy();
    gray[:,:,0] = 128
    gray[:,:,1] = 128
    gray[:,:,2] = 128
    black = blank_image.copy();
    return [red, green, blue, magenta, cyan, yellow, white, gray, black

def getHSVInversion(image):
    return cv2.bitwise_not(cv2.cvtColor(image, cv2.COLOR_RGB2HSV))

def getContrastTransformation(image, contrast = 7.5):
    return Convert().image(image, contrast).astype("uint8")

def plotResults(image, hsvInversion, contrastTransformation, contrast =
    plt.rcParams["figure.figsize"] = (20,1)
    fig, (axs1, axs2, axs3) = plt.subplots(1,3);
    axs1.set_title("original image")
    axs1.imshow(image);
    axs2.set_title("HSV Inversion")
    axs2.imshow(hsvInversion);
    axs3.set_title(f"Contrast Transformation c={contrast}")
    axs3.imshow(contrastTransformation);
    plt.show();

# images = generateTestImages();
# getHSVInversion(images[0].copy())
for image in generateTestImages():
    plotResults(
        image,
        getHSVInversion(image),
```

```
        getContrastTransformation(image.copy())
    );
```



Experiments

In [1]:

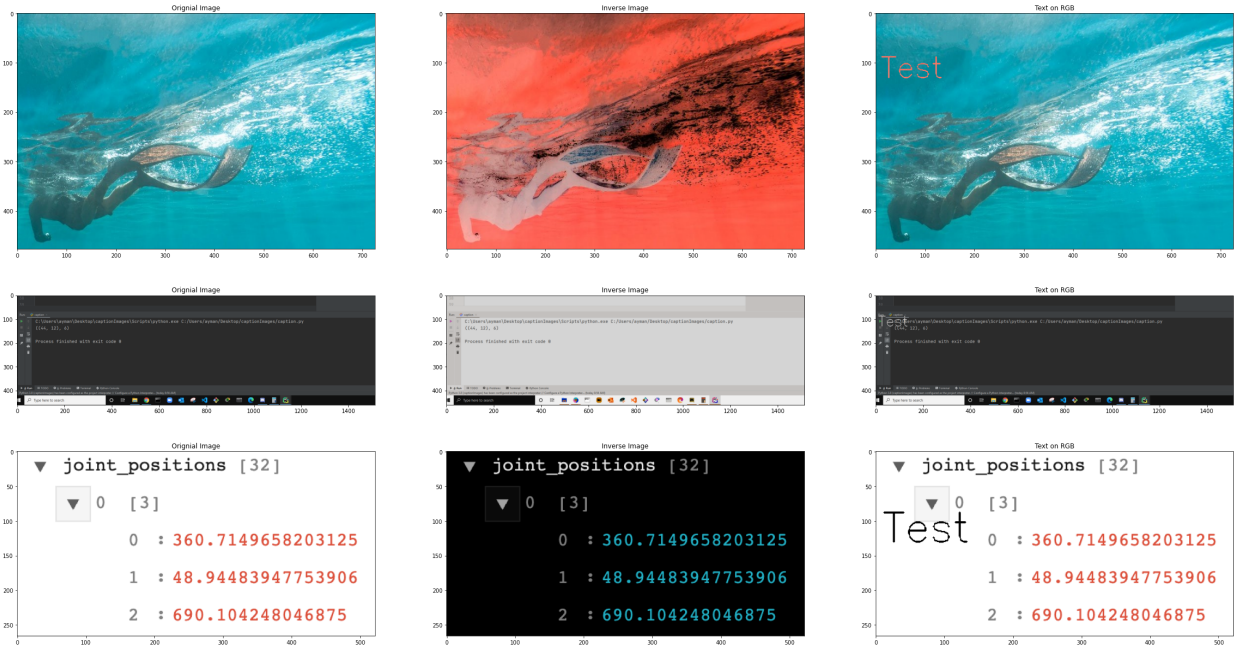
```
import numpy as np
import os
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2
import os
from contrast.convert import Convert
images = ['Capture.PNG', 'Capture2.PNG', 'image.PNG', 'image3.PNG']
```

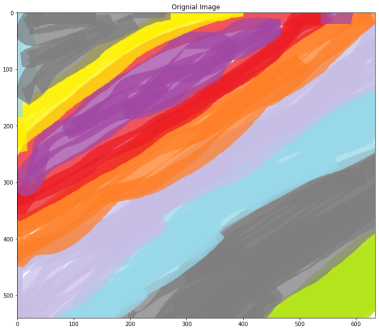
RGB Color Space

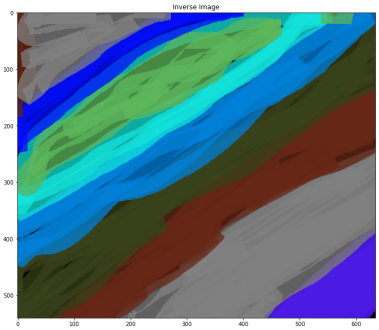
- All Three channels has the amount of light included in each component (brightness)

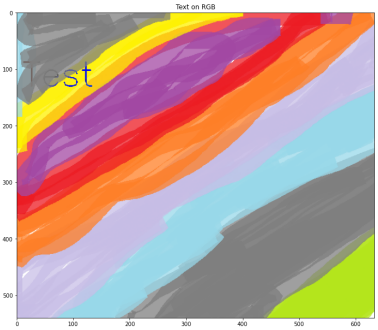
In [2]:

```
for image in images:
    imageBGR = cv2.imread(os.path.join("images", image))
    blankImage = np.zeros(shape=imageBGR.shape, dtype=np.uint8)
    cv2.putText(
            blankImage,
            "Test",
            (10, 130),
            0,
            2,
            (255, 255, 255),
            2,
            cv2.LINE_AA,
            False
        );
    imageRGB = cv2.cvtColor(imageBGR, cv2.COLOR_BGR2RGB);
    imageRGBwithText = imageRGB.copy()
    imagePatch = imageRGB[0:200, 0:200, :]
    indices = np.where(np.all(blankImage == 255, axis=-1))
    imageRGBInverse = cv2.bitwise_not(imageRGB)
    coordinates = [*zip(indices[0], indices[1])]
    for i in coordinates:
        imageRGBwithText[i[0], i[1], :] = imageRGBInverse[i[0], i[1], :
    f, (ax1, ax2, ax3) =  plt.subplots(1, 3, figsize=(40, 40))
    ax1.set_title("Orignial Image")
    ax1.imshow(imageRGB)
    ax2.set_title("Inverse Image")
    ax2.imshow(imageRGBInverse)
    ax3.set_title("Text on RGB")
    ax3.imshow(imageRGBwithText)
```







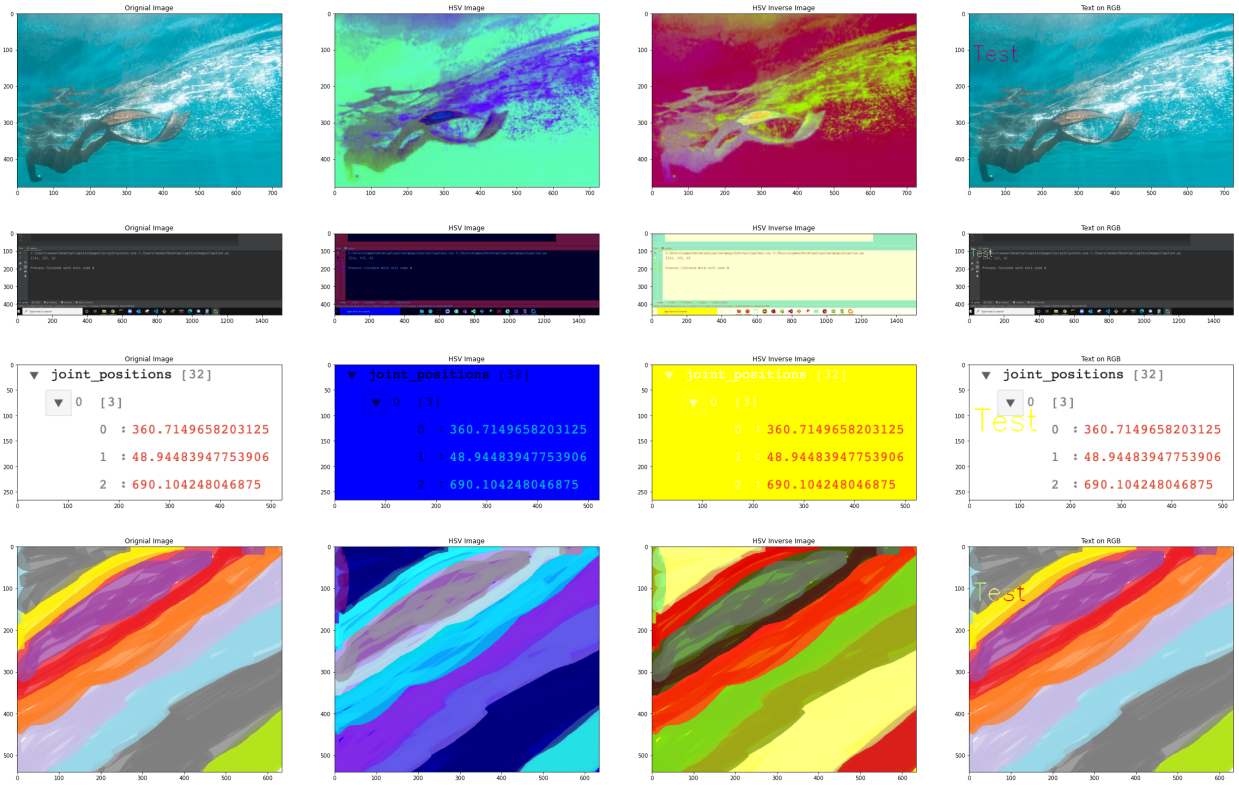


HSV Color Space

- H – Hue (Dominant Wavelength).
- S – Saturation (Purity / shades of the color).
- V – Value (Intensity)

In [3]:

```python
for image in images:
  imageBGR = cv2.imread(os.path.join("images", image))
  blankImage = np.zeros(shape=imageBGR.shape, dtype=np.uint8)
  cv2.putText(
            blankImage,
            "Test",
            (10, 130),
            0,
            2,
            (255, 255, 255),
            2,
            cv2.LINE_AA,
            False
        );
  imageRGB = cv2.cvtColor(imageBGR, cv2.COLOR_BGR2RGB);
  imageHSV = cv2.cvtColor(imageBGR, cv2.COLOR_BGR2HSV);
  imageRGBwithText = imageRGB.copy()
  imagePatch = imageRGB[0:200, 0:200, :]
  indices = np.where(np.all(blankImage == 255, axis=-1))
  imageInverse = cv2.bitwise_not(imageHSV)
  coordinates = [*zip(indices[0], indices[1])]
  for i in coordinates:
    imageRGBwithText[i[0], i[1], :] = imageInverse[i[0], i[1], :]
  f, (ax1, ax2, ax3, ax4) =  plt.subplots(1, 4, figsize=(40, 40))
  ax1.set_title("Orignial Image")
  ax1.imshow(imageRGB)
  ax2.set_title("HSV Image")
  ax2.imshow(imageHSV)
  ax3.set_title("HSV Inverse Image")
  ax3.imshow(imageInverse)
  ax4.set_title("Text on RGB")
  ax4.imshow(imageRGBwithText)
```

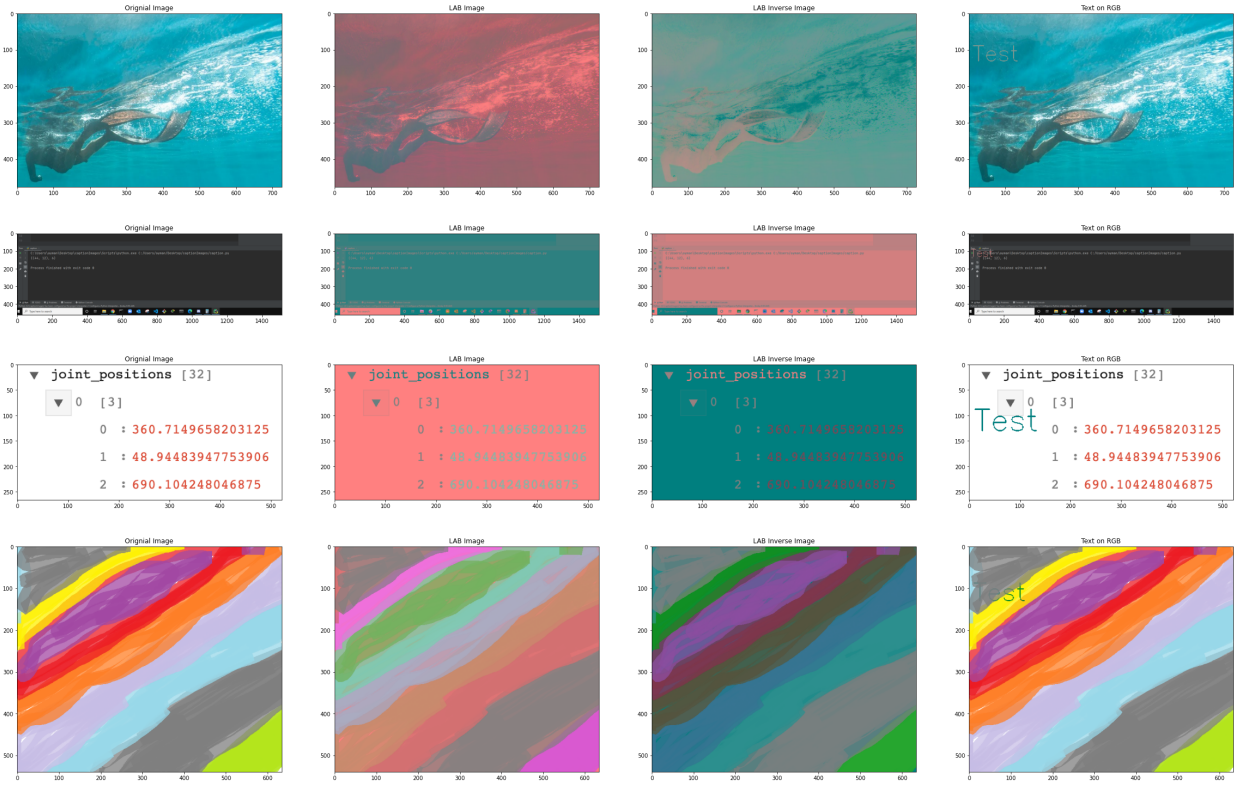


LAB Color Space

- L - Lightness
- A - Green to Magenta
- B - Blue to Yellow

In [4]:

```python
for image in images:
  imageBGR = cv2.imread(os.path.join("images", image))
  blankImage = np.zeros(shape=imageBGR.shape, dtype=np.uint8)
  cv2.putText(
            blankImage,
            "Test",
            (10, 130),
            0,
            2,
            (255, 255, 255),
            2,
            cv2.LINE_AA,
            False
        );
  imageRGB = cv2.cvtColor(imageBGR, cv2.COLOR_BGR2RGB);
  imageHSV = cv2.cvtColor(imageBGR, cv2.COLOR_BGR2LAB);
  imageRGBwithText = imageRGB.copy()
  imagePatch = imageRGB[0:200, 0:200, :]
  indices = np.where(np.all(blankImage == 255, axis=-1))
  imageInverse = cv2.bitwise_not(imageHSV)
  coordinates = [*zip(indices[0], indices[1])]
  for i in coordinates:
    imageRGBwithText[i[0], i[1], :] = imageInverse[i[0], i[1], :]
  f, (ax1, ax2, ax3, ax4) =  plt.subplots(1, 4, figsize=(40, 40))
  ax1.set_title("Orignial Image")
  ax1.imshow(imageRGB)
  ax2.set_title("LAB Image")
  ax2.imshow(imageHSV)
  ax3.set_title("LAB Inverse Image")
  ax3.imshow(imageInverse)
  ax4.set_title("Text on RGB")
  ax4.imshow(imageRGBwithText)
```

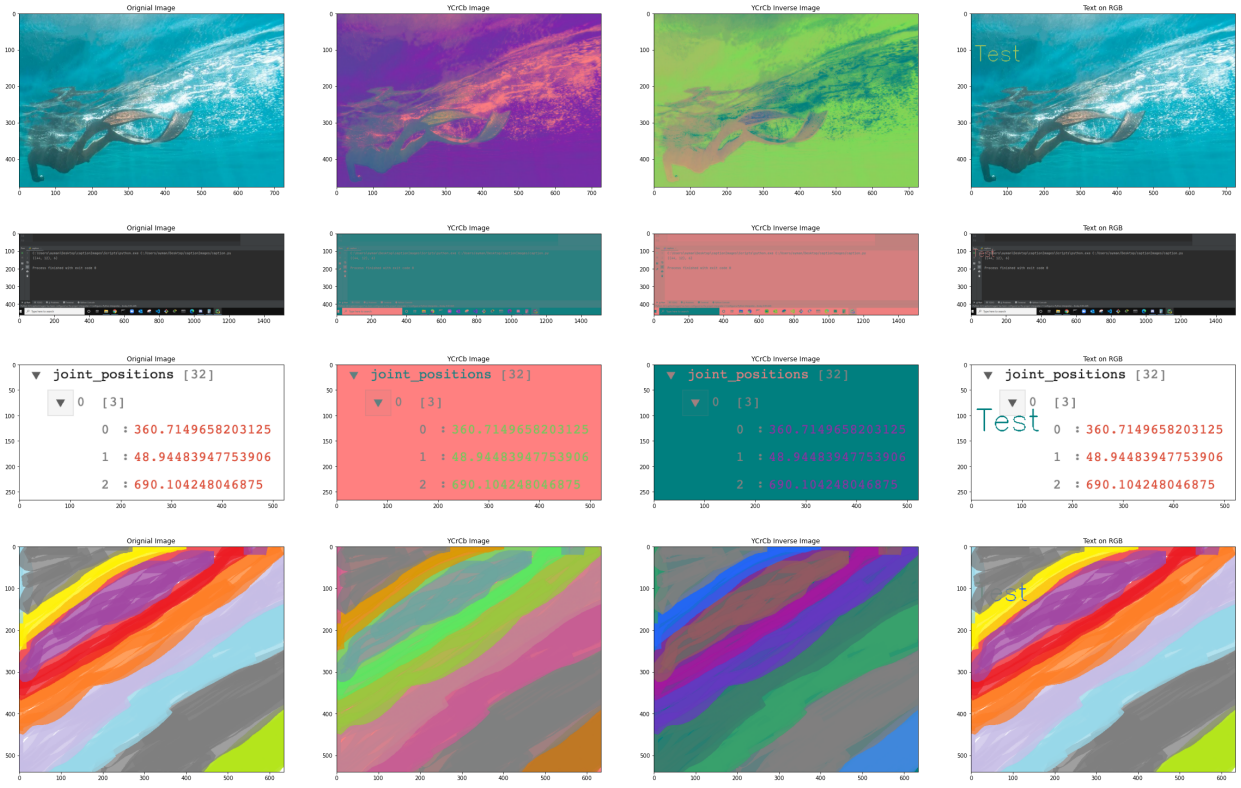


YCrCb Color Space

- Y - Luminance (Luma) which is obtained from RGB after gamma correction
- Cr - Red - Y
- Cb - Blue - Y

In [5]:

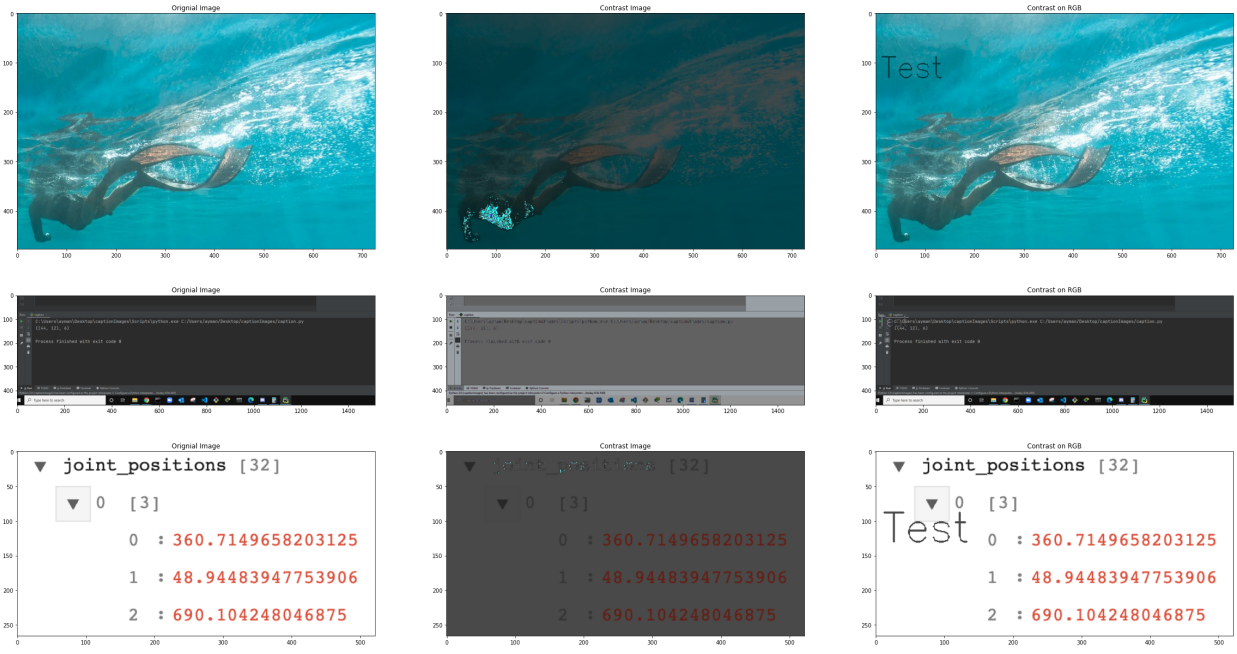
```python
for image in images:
  imageBGR = cv2.imread(os.path.join("images", image))
  blankImage = np.zeros(shape=imageBGR.shape, dtype=np.uint8)
  cv2.putText(
            blankImage,
            "Test",
            (10, 130),
            0,
            2,
            (255, 255, 255),
            2,
            cv2.LINE_AA,
            False
        );
  imageRGB = cv2.cvtColor(imageBGR, cv2.COLOR_BGR2RGB);
  imageHSV = cv2.cvtColor(imageBGR, cv2.COLOR_BGR2YCrCb);
  imageRGBwithText = imageRGB.copy()
  imagePatch = imageRGB[0:200, 0:200, :]
  indices = np.where(np.all(blankImage == 255, axis=-1))
  imageInverse = cv2.bitwise_not(imageHSV)
  coordinates = [*zip(indices[0], indices[1])]
  for i in coordinates:
    imageRGBwithText[i[0], i[1], :] = imageInverse[i[0], i[1], :]
  f, (ax1, ax2, ax3, ax4) =  plt.subplots(1, 4, figsize=(40, 40))
  ax1.set_title("Orignial Image")
  ax1.imshow(imageRGB)
  ax2.set_title("YCrCb Image")
  ax2.imshow(imageHSV)
  ax3.set_title("YCrCb Inverse Image")
  ax3.imshow(imageInverse)
  ax4.set_title("Text on RGB")
  ax4.imshow(imageRGBwithText)
```

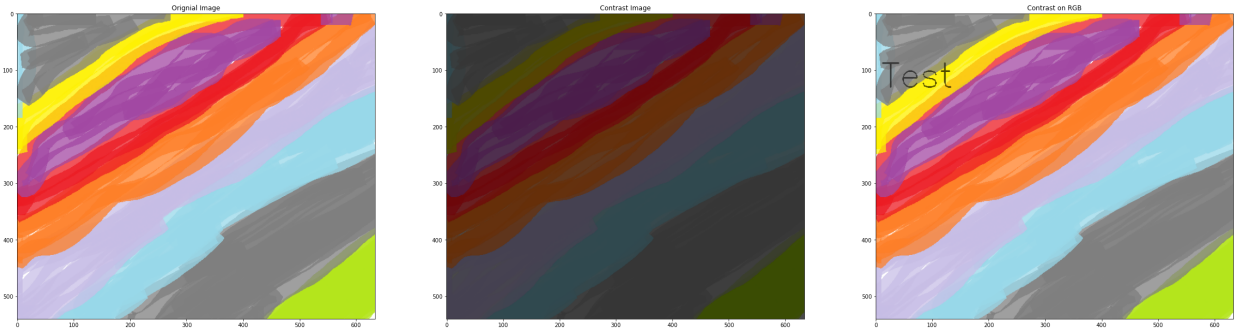


Contrast Transformation

In [9]:

```python
for image in images:
    imageBGR = cv2.imread(os.path.join("images", image))
    blankImage = np.zeros(shape=imageBGR.shape, dtype=np.uint8)
    cv2.putText(
            blankImage,
            "Test",
            (10, 130),
            0,
            2,
            (255, 255, 255),
            2,
            cv2.LINE_AA,
            False
        );
    imageRGB = cv2.cvtColor(imageBGR, cv2.COLOR_BGR2RGB);
    imageConverted = Convert().image(imageRGB.copy(), 8.0)
    imageRGBwithText = imageRGB.copy()
    imagePatch = imageRGB[0:200, 0:200, :]
    indices = np.where(np.all(blankImage == 255, axis=-1))
    coordinates = [*zip(indices[0], indices[1])]
    for i in coordinates:
        imageRGBwithText[i[0], i[1], :] = imageConverted[i[0], i[1], :]
    f, (ax1, ax2, ax4) =  plt.subplots(1, 3, figsize=(70, 40))
    ax1.set_title("Orignial Image")
    ax1.imshow(imageRGB)
    ax2.set_title("Contrast Image")
    ax2.imshow(imageConverted)
    ax4.set_title("Contrast on RGB")
    ax4.imshow(imageRGBwithText)
```





Utilizing the Project

The project was written using opencv to gather image data and place text onto the image. To get the project to run in the terminal use the following command.

```
python main.py \
    -v videos/sample.mp4 \
    -st 00:00:10 \
    -et 00:00:15 \
    -t "Ayman"  \
    -p false \
    -c false \
    -x 20 \
    -y 20 \
    -hsv true \
    -ff 3 \
    -fs 1
```

Parameter	Alternate	Description
-t	--text_caption	The text to be inserted
-hsv	--hsv_inverse	The inverse of HSV Color fast in some situation. If true, use HSV, else use Contrast transformation
-c	--contrast_method	Calculated contrast slower but more robust
-cl	--contrast_level	The level of contrast between 1 to 21
-v	--video	The input video file
-x	--x_coordinate	X Coordinate value
-y	--y_coordinate	Y coordinate value
-fs	--font_size	You may use 1 for small, 2 for medium, or 3 for large
-ff	--font_family	You may use the number from 0 to 7 for the supported fonts
-p	--pause_video	you may pass True to pause the video or False for having the text in all frames
-st	--start_time	the starting time must be in the format of hh:mm:ss
-et	--end_time	the ending time must be in the format of hh:mm:ss

AWS Implementation

Utilized `boto3` to handle requests. To make multiple requests at one time `concurrent.futures.ThreadPoolExecutor` was utilized, shown below. The `request_image` method handled uploading the object, `obj` , to the lambda and provided a response.

```
def request(self, objs):
    with ThreadPoolExecutor(max_workers=100) as executor:
        futs = []
        for obj in objs:
            futs.append(
                executor.submit(
                    self.request_image,
                    obj = obj
                )
            )

        self.results = []
        for fut in futs:
            self.results.append(fut.result())
    return self.results
```

When calling the `request` function an array of parameters needs to be passed, the image as a rgb array, the text to overlay, the x and y coordinates, the properties of the text, and the contrast to be used. If the contrast is set to `-1` then HSV will be utilized, otherwise the Contrast Transformation is used.

```
request([
    {
        image: image_rgb_arr,
        text: "text",
        x: 300,
        y: 300,
        textProperties: textProperties,
        contrast: -1 # -1 for HSV and 1-21 for Contrast Transformat
ion
    }
    ...
])
```

In [ ]:

```
1
2
```