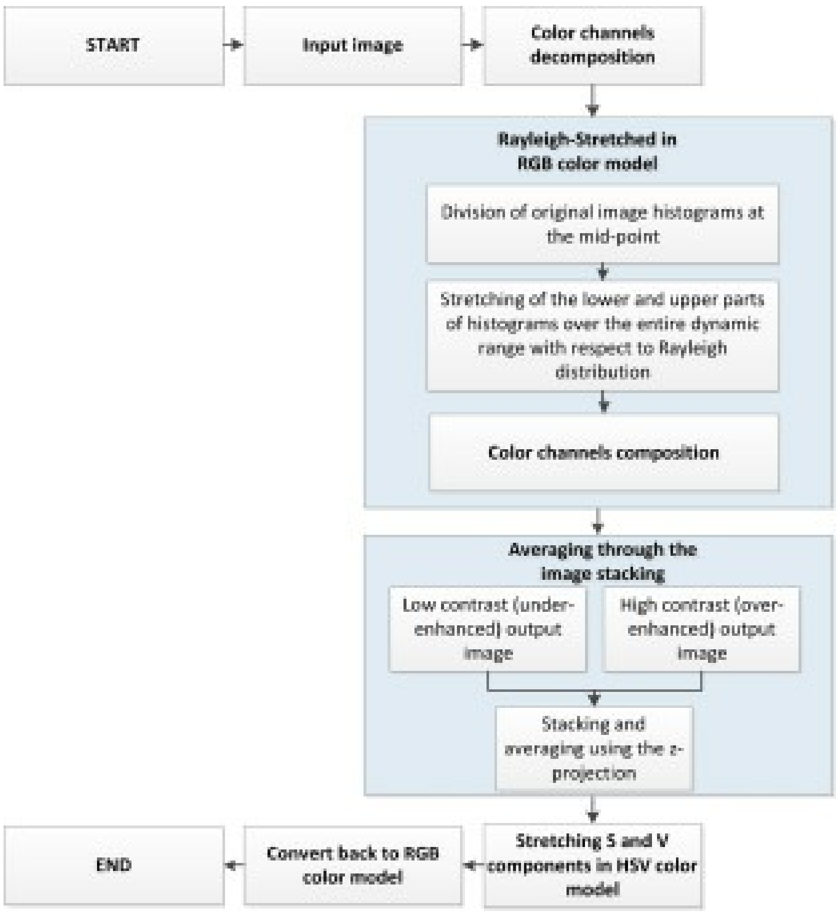
Image Enhancement

Introduction

Visibility in underwater images is usually poor because of the attenuation of light in the water that causes low contrast and color variation. In this , a new approach for underwater image quality improvement is presented,based on [this paper (https://www.sciencedirect.com/science/article/pii/S2092678216302588/pdfft?md5=0fb801ccd8c3b98979cb631428a2c3e9&pid=1-s2.0-S2092678216302588-main.pdf)](https://www.sciencedirect.com/science/article/pii/S2092678216302588/pdfft?md5=0fb801ccd8c3b98979cb631428a2c3e9&pid=1-s2.0-S2092678216302588-main.pdf). The proposed method aims to improve underwater image contrast, increase image details, and reduce noise by applying a new method of using contrast stretching to produce two different images with different contrasts.

Mehodology

Here we are dividing a image from its average pixel value to highest & lowest respectively. Calling it as high contrast & low contrast respectively.Followed by this a Rayleigh stretching of each pixel over its entire range. We get two images till now, we use their extremities to get an average image with averaging the noises.
The flow of process goes like this:


START
Input image
Color channels decomposition
Rayleigh-Stretched in RGB color model
Division of original image histograms at the mid-point
Stretching of the lower and upper parts of histograms over the entire dynamic range with respect to Rayleigh distribution
Color channels composition
Averaging through the image stacking
Low contrast (under-enhanced) output image
High contrast (over-enhanced) output image
Stacking and averaging using the z-projection
Stretching S and V components in HSV color model
Convert back to RGB color model
END

**Input Image

1] Color channel decomposition

```
   b,g,r=cv2.split(img2)
   b1,g1,r1=cv2.split(img1)
```

2] Finding average pixel
First we split down the image in its respective b,g,r channel. We need to find average pixel for each channel.

```
   imdb = ((b2.max()-b2.min())/2) + b2.min()
   imdg = ((g2.max()-g2.min())/2) + g2.min()
   imdr = ((r2.max()-r2.min())/2) + r2.min()
```

3] Rayleigh stretching
Its based on Rayleigh Distribution algorithm (https://wikimedia.org/api/rest_v1/media/math/render/svg/8a06343fb89f74d188c25aef4931739
Here using the basic principle of stretching the pixels over entire range,as mentioned in eq2 (https://www.sciencedirect.com/science/article/pii/S2092678216302588#formula2).
Integrating this equation results in dynamic stretching of pixels over the entire range as shown in eq4 (https://www.sciencedirect.com/science/article/pii/S2092678216302588#formula4).
Here Rayleigh stretching of 'b' channel for high contrast:

```
   b[b<imdb] = imdb
   for index,value in np.ndenumerate( b ):
       new_value = ((255* (value-imdb))/ ((b2.max()-b2.min())/(alpha**2)))
        b[index] = new_value
```

4] Image composition
We merges both set of channel to obtain high contrast & low contrast images:

```
    res=cv2.merge((b,g,r))
    res1=cv2.merge((b1,g1,r1))
```

5] Averaging two images, here we can use diffetent methods of averaging images, but the simplest for two images is weighted addition:

```
    res2= cv2.addWeighted(res,.5,res1,.5,0)
```

6] Final stage is gto is increasing the contrst of the image, there are no. of ways from which we can do this, like stretching s & v component image over entire range, gamma correction, histogram equalization.We need to maintain image for segmentation so we are omitting histogram stretching, instead we uses gamma correction:

```
    def adjust_gamma(image, gamma=1.0):
        invGamma = 1.0 / gamma
          table = np.array([((i / 255.0) ** invGamma) * 255
            for i in np.arange(0, 256)]).astype("uint8")
        return cv2.LUT(image, table)
```