# INFORME TERCER PROYECTO

ESTUDIANTES:

Ricardo Murillo Jiménez
2018173697

Joel Vega
2018163840

28 de mayo de 2018

**Curso:**
**Taller de Programación**

**INSTITUTO TECNOLOGICO DE COSTA RICA**

2018

# Índice



# Índice de figuras

# 1. Introducción

Debido a la necesidad de encontrar una forma para definir los colores mediante diferentes componentes de los mismos se han creado los modelos de color. En este informe se tratara de explicar la modificación de imagenes a traves del cambio de modelo de color y sus componentes, esto debido a que la computadora solo puede cambiar los valores en el modelo de color denominado HSV por sus siglas en ingles(hue: tinte, Saturation: saturacion, value: valor) pero no los puede cambiar en el formato original de las imagenes RGB denominado tambien asi por sus iniciales (red: rojo, green: verde, blue: azul), por lo tanto se utilizaran distintos algoritmos y formulas matematica para cambiar los valores de los distntos componentes.

Basicamente lo que se debe de hacer para lograr manejar los distintos modelos de color es convertir la imagen a una matriz la cual es capaz de ser modificada por la computadora y asi pasar la imagen de formato RGB a HSV por lo cual ya le podriamos cambiar los valores de la matriz,claro para lograr cambiar hay que recurrir a diferentes lgoritmos y formulas matematicas que logren hacer los cambios pertinentes, ya sea modificar el brillo, cambiar la saturacion, el tinte o aplicar el desenfoque gaussiano, claramente para poder ver la imagen que se logro obtener despues de aplicarle los cambios respectivos se debe de volver a aplicar un algoritmo que nos devuelva la matriz a RGB.

Para lograrel efecto denominado desenfoque Gaussiano se debe de crear una matriz llamada: matriz de convolucion. esta matriz es muy usada en modificacion de imagenes ya que junto con otra matriz basada en el Kernel se pueden lograr diversos efectos. Basicamente La matriz va de pixel en pixel por toda la imagen cambiando los valores por lo que se quiere.

# 2. desarrollo

## 2.1. conceptos

Modelo de color: " Un modelo de color establece un conjunto de colores primarios a partir de los que, mediante mezclas, se pueden obtener otros colores hasta cubrir todo el espectro visible, además del propio blanco, negro y grises, y aún más".Herrera, a. (2015). Modelos de color (RGB, CMYK, HSV/HSL). Recuperado de https://ahenav.com/2014/04/09/modelos-de-color

Modelo de color RGB: Este modelo se basas en tres colores, los cuales son: rojo, verde y azul. este modelo se usa la hora de proyectar haces de luz en cualquier tipo de monitor o pantalla. Cada color se representa mediante 1 LED o diodo emisor y si todos los LED's estan apagados se obtiene el color negro, si todos tienen la misma intensidad al maximo obtenemos el blanco y mediante

los diferentes niveles de intensidad de los LED's se obtienen los colores.

Modelo de color HSV: Este se basa en los componentes, los cuales son: matiz(hue), saturacion(saturation) y el valor(value), este modelo. Definido en el en el 78 por Alvy Ray Smith. Este modelo varia las propiedades del color con lo que crea nuevos valores.

Matiz: Es el angulo que representa el matiz en un a repredentacion circular de los colores. tambien es la forma mas basica de representar un color: rojo, verde, etc.

Saturación: Es la intensidad de determinado matiz, así que cualquier color con mucha saturacion se va a ver de un color muy brillante y uno con poca saturacion se va a ver casi que en una escala de grises. Tambien está determinada por las diferentes lonitudes de onda en el espectro.

Valor: Es la amplitud de la onda de luz que define el color, es basicamente si es color queda claro o oscuro y tecnicamente solo existen dos valores, el negro y el blanco y los grises son las diferentes tonalidades de estos, los cuales se usan para aclarar o escurecer otros colores.

Matriz de convolución: La matriz de la convolucion es un tratamiento que se le aplica a ua imagen para aplicarle diferentes filtros mediante los diferentes valores de la matriz de convolucion ya sea desde desenfocar la imagen hasta lograr disminuir el ruido de dicha imagen.

Desenfoque Gaussiano: basicamente usando una matriz de convolución o Kernel se recorre toda la matriz pixel por pixel hasta cambiar los valores de la matriz de la imagen original hasta lograr un efecto de borrosidad. Lo que hace la matriz es ir promediando los valores circundantes a cada pixel y junto con la desviación estandar se logra que quede mas o menos borrosa

## 2.2. Algoritmos RGB 2 HSV

```
def convertir_rgb_hsv(nuevaImagen):
    height = len(nuevaImagen)
    width = len(nuevaImagen[0])
    bands = len(nuevaImagen[0][0])
    imagenhsv = np.zeros([height,width,bands])
    for i in range(0,height):
        for j in range(0,width):
            rp = nuevaImagen[i][j][2]/255.0
            gp = nuevaImagen[i][j][1]/255.0
            bp = nuevaImagen[i][j][0]/255.0
            cmax = max(rp,gp,bp)
            cmin = min(rp,gp,bp)
            df = cmax-cmin
            if cmax == cmin:
                H = 0
            elif cmax == rp:
                H = ((60*(gp-bp)/df)+360)%360
            elif cmax == gp:
                H = (((60)*((bp-rp)/df))+120)%360
            elif cmax == bp:
                H = (((60)*(((rp-gp)/df)))+240)%360
            if cmax == 0:
                S = 0
            else:

                S = df/cmax
            V = cmax
            imagenhsv[i][j][0]=H
            imagenhsv[i][j][1]=S
            imagenhsv[i][j][2]=V

    return imagenhsv
```

Figura 1: CAMBIO DE RGB A HSV

Para poder cambiar el modelo de color que tiene la imagen lo primero que se hace es convertir la imagen original a la matriz. ya cuando tenemos la matriz se definen las variables que vamos a utilizar designandoles valores especificos de la matriz y creamos una nueva matriz que va a ser la matriz pero en el formato de HSV. Seguidamente se recorre la matriz con el uso de condicionales y se le aplica a cada pixel de la imagen la formula designada para lograr el modelo que se requiere.

$$R' = R/255$$

$$G' = G/255$$

$$B' = B/255$$

$$Cmax = \max(R', G', B')$$

$$Cmin = \min(R', G', B')$$

$$\Delta = Cmax - Cmin$$

Hue calculation:

$$H = \begin{cases} 0^\circ & \Delta = 0 \\ 60^\circ \times \left(\frac{G'-B'}{\Delta} mod 6\right) & , C_{max} = R' \\ 60^\circ \times \left(\frac{B'-R'}{\Delta} + 2\right) & , C_{max} = G' \\ 60^\circ \times \left(\frac{R'-G'}{\Delta} + 4\right) & , C_{max} = B' \end{cases}$$

Saturation calculation:

$$S = \begin{cases} 0 & , C_{max} = 0 \\ \frac{\Delta}{C_{max}} & , C_{max} \neq 0 \end{cases}$$

Figura 2: formulas para RGB2HSV

En la figura anterior podemos ver las formulas matematicas que se le alican a los diferentes valores de la matriz, se puede observar que depende del valor inicial la formula varia.

## 2.3. Algoritmos HSV 2 RGB

```
def convertir_hsv_rgb(imagenhsv):
    height = len(imagenhsv)
    width = len(imagenhsv[0])
    bands = len(imagenhsv[0][0])
    imagenrgb =
np.zeros([height,width,bands],dtype=np.uint8
)
    for i in range(0,height):
        for j in range(0,width):
            H = imagenhsv[i][j][0]
            S = imagenhsv[i][j][1]
            V = imagenhsv[i][j][2]
            C = V * S
            X = C * (1-abs((H/60%2)-1))
            m = V - C
            if (0)<=H<(60):
                rp = C
                gp = X
                bp = 0
            elif (60)<=H<(120):
                rp = X
                gp = C
                bp = 0
            elif (120)<=H<(180):
                rp = 0
                gp = C
                bp = X
    return imagenrgb
```

```
elif (180)<=H<(240):
    rp = 0
    gp = X
    bp = C
elif (240)<=H<(300):
    rp = X
    gp = 0
    bp = C
elif(300)<=H<(360):
    rp = C
    gp = 0
    bp = X

r=(rp+m)*255
g=(gp+m)*255
b=(bp+m)*255
if r>255:
    r = 255
if g>255:
    g = 255
if b>255:
    b = 255

imagenrgb[i][j][2]=r
imagenrgb[i][j][1]=g
imagenrgb[i][j][0]=b
```

Figura 3: CAMBIO DE HSV A RGB

Para poder volver a tener la imagen en el modelo RGB hay que agarrar la matriz que se creo en la funcion anterior de RGB2HSV pero ya con todos los cambios aplicados. Despues esos nuevos valores se vuelven a cambiar a componentes de modelo RGB pero no van a ser los mismos que la original debido a que ya pasaron por todo un proceso. despues de que ya se tiene una nueva matriz con los valores de HSV se le aplican distintas formulas matematicas a cada pixel para que este igual que la imagen origina pero ya con el filtro.

When $0 \le H < 360$, $0 \le S \le 1$ and $0 \le V \le 1$:

$$C = V \times S$$

$$X = C \times (1 - |(H / 60^\circ) \bmod 2 - 1|)$$

$$m = V - C$$

$$(R', G', B') = \begin{cases} (C, X, 0) & , 0^\circ \le H < 60^\circ \\ (X, C, 0) & , 60^\circ \le H < 120^\circ \\ (0, C, X) & , 120^\circ \le H < 180^\circ \\ (0, X, C) & , 180^\circ \le H < 240^\circ \\ (X, 0, C) & , 240^\circ \le H < 300^\circ \\ (C, 0, X) & , 300^\circ \le H < 360^\circ \end{cases}$$

$$(R,G,B) = ((R'+m)\times 255, (G'+m)\times 255, (B'+m)\times 255)$$

Figura 4: formulas para HSV2RGB

En la imagen anterior podemos observar las diferentes formulas que hay que aplicarle a los pixeles de la imagen o a las diferentes posiciones de la matriz para poder llevarla de nuevo a RGB y se puede observar como puede cambiar la formula dependiendo de los valores de la matriz.

## 2.4. Algoritmos Matriz Convolución

```
def matriz_convolucion(ker,desv):
    n=(ker*2)+1
    matriz_con=np.zeros((n,n))
    c = n//2este al
    for i in range (n):
        for k in range(n):
            x=abs(i-c)
            y=abs(k-c)
            matriz_con[i][k]=funcion_desenfoque(x,y,desv)
            lista = [matriz_con,n]
    return lista
```

Figura 5: Algoritmo matriz de convolucion

Este algoritmo se utiliza a la hora de aplicar el desenfoque gaussiano y basicamente lo que hace es definir la cantidad de borrosidad que va a tener la imagen, por ejemplo si entra una desviacion muy alta la imagen va a quedar muy borrosa ya que basicamente lo que se hace es "mover"los pixeles x cantidad de espacios

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Figura 6: Formula matriz de convolucion

Esta es la formula usada para lograr obtener la matriz de convolución la cual se usa para lograr el efecto de desenfoque Gaussiano.

## 2.5. bibliotecas externas

CV2: fue originalmente desarrollada por Intel, es multiplataforma. Debido a que esta biblioteca es libre, que existe en distintas plataformas y que posee muchas funcionalidades se ha usado en muchas aplicaciones. Pretende proporcionar un entorno muy facil de usar y muy eficiente.

numPy: Una extension de python con matematicas de alto nivel y muy bueo para manejar marices

math:

## 2.6. Documentacion de funciones

Nombre: Hue

```
def hue(nuevaImagen,angulo):
    x = convertir_rgb_hsv(nuevaImagen)
    height = len(nuevaImagen)
    width = len(nuevaImagen[0])
    bands = len(nuevaImagen[0][0])
    for i in range(0,height):
        for j in range(0,width):
            x[i][j][0]+=angulo
            if x[i][j][0] > 360:
                x[i][j][0] -= 360
    return convertir_hsv_rgb(x)
```

Figura 7: Funcion para el cambio de matiz

Parametros: nuevaImagen y angulo
NuevaImagen: es la copia de la imagen que se va a modificar, variable:local
Angulo: es el numero de grados que se quiere cambiar el matiz de la imagen, variable:local
Retorno: retorna una copia de la imagen original pero en la cual el matiz se vio modificado
Resumen: Se llama a la funcion que convierte la imagen a el formato HVS y va cambiando los datos basandose en el segundo parametro de entrada(angulo) y a lo ultimo retorna una imagen de la original pero ya con los cambios realizados

Nombre: sat

```
def sat(nuevaImagen,proporcion):
    x = convertir_rgb_hsv(nuevaImagen)
    height = len(nuevaImagen)
    width = len(nuevaImagen[0])
    bands = len(nuevaImagen[0][0])
    for i in range(0,height):
        for j in range(0,width):
            x[i][j][1]*=proporcion
    return convertir_hsv_rgb(x)
```

Figura 8: Funcion para el cambio de la saturacion

Parametros: nuevaImagen y proprocion
NuevaImagen: es la copia de la imagen que se va a modificar, variable:local

Proporcion: El nivel de saturacion que se quiere obtener en la imagen, variable: local
Retorno: retorna una copia de la imagen original pero en la cual la saturacion es mayor o menor
Resumen: Se llama a la funcion que convierte la imagen a el formato HVS y mientras recorre la imagen va cambiando la saturacion segun el valor que se le dio al programa y retorna una copia de la original pero con la saturación modificada

Nombre: val

```
def val(nuevaImagen,proporcion):
    x = convertir_rgb_hsv(nuevaImagen)
    height = len(nuevaImagen)
    width = len(nuevaImagen[0])
    bands = len(nuevaImagen[0][0])
    for i in range(0,height):
        for j in range(0,width):
            x[i][j][2]*=proporcion
    return convertir_hsv_rgb(x)
```

Figura 9: Funcion para el cambio del valoe

Parametros: nuevaImagen y proporcion
NuevaImagen: es la copia de la imagen que se va a modificar, variable:local
Proporcion: El nivel de valor que se quiere obtener en la imagen, se usa para controlar el brillo en la imagen variable, ya puede ser una imagen muy negra o muy brillante: local
Retorno: retorna una copia de la imagen original pero ya con los cambios aplicados, o sea va a salir una imagen muy oscura y sin brillo o brillosa
Resumen: Se llama a la funcion que convierte la imagen a el formato HVS y mientras recorre la imagen va cambiandoel valor segun el valor que se le dio al programa y retorna una copia de la original pero con mas o menos brillo modificada

Nombre: value

```
def funcion_desenfoque(x,y,des):
    funcion =(1/(2*math.pi*des**2))*math.e**(-((x**2+y**2)/(2*des**2)))
    return funcion
```

Figura 10: Funcion desenfoque

Parametros: x, y y desviacion estandar
x e y: posición en la matriz de convolucion desviación estandar: Es el grado de dispersión. En este caso es el rango de alcanze de la matriz de convolución
Retorno: retorna la funcion para calcular la matriz de convolución para ser llamada despúes con diferentes valores de x e y.
Resumen: Recibe la posicion de la matriz y el tamaño que se quiere de desviación.

Nombre: blur

```
def blur (Imagen_desenfoque,ker,desv):
    lista = matriz_convolucion(ker,desv)
    matriz_con = lista[0]
    n = lista[1]
    height = len(Imagen_desenfoque)
    width = len(Imagen_desenfoque[0])
    bands = len(Imagen_desenfoque[0][0])
    imagenblur = np.zeros([height,width,bands],dtype=np.uint8)
    suma=0
    for i in range (n):
        for j in range (n):
            suma +=matriz_con[i][j]
    for i in range (n):
        for j in range (n):
            matriz_con[i][j]=matriz_con[i][j]/suma

    for i in range (height):
        for j in range (width):
            acum=0
            for k in range (n):
                for l in range (n):
                    Ipri=i-ker+k
                    Jpri=j-ker+l
                    if Ipri>=0 and Ipri<height and Jpri>=0 and Jpri<width:
                        acum += (Imagen_desenfoque[Ipri][Jpri]*matriz_con[k][l]
            imagenblur[i][j] = acum

    return imagenblur
```

Figura 11: Desenfoque

Parametros: imagen desenfoque, kernel y desviacion
imagen desenfoque: la imagen a la cua, se le va a aplicar el desenfoque
Kernel: La matriz con la cual se va a modificar la copa de la imagen original
desviación estandar: Es el grado de dispersión. En este caso es el rango de alcanze de la matriz de convolución
Retorno: Retorna a copia de la imagen orginal pero ya con el desenfoque Gaussiano aplicado
Resumen: Se llama a la funcion que nos da la matriz de convolución despues va apromediando todos los valores que entren en el rango de la matriz de convolución y hace lo mismo por toda a matriz

Nombre: Main

```
def main():
   linea = input().split(" ")
   operacion = linea[1]

   image = cv2.imread(linea[0])
   cv2.imshow("image",image)
   cv2.waitKey(0)
   cv2.destroyAllWindows()
   height = len(image)
   width = len(image[0])
   bands = len(image[0][0])
   nuevaImagen =
np.zeros([height,width,bands])
   for i in range(height):
      for j in range(width):
         for b in range(bands):
            nuevaImagen[i][j][b] =
image[i][j][b]

   Imagen_desenfoque =
np.zeros([height,width,bands],dtype=np.ui
nt8)
   for i in range(height):
      for j in range(width):
         for b in range(bands):
            Imagen_desenfoque[i][j][b] =
image[i][j][b]
   if operacion == "hue":
      n = int(linea[2])
      h = hue(nuevaImagen,n)
      cv2.imshow("hue.jpg",h)
      cv2.waitKey(0)
      cv2.destroyAllWindows()
      cv2.imwrite("Copia_hue.jpg",h)
   elif operacion == "sat":
      n = float(linea[2])
      s = sat(nuevaImagen,n)
      cv2.imshow("sat.jpg",s)
      cv2.waitKey(0)
      cv2.destroyAllWindows()
      cv2.imwrite("Copia_sat.jpg",s)
   elif operacion == "val":
      n = float(linea[2])
      v = val(nuevaImagen,n)
      cv2.imshow("val.jpg",v)
      cv2.waitKey(0)
      cv2.destroyAllWindows()
      cv2.imwrite("Copia_val.jpg",v)
   elif operacion == "blur":
      n = int(linea[2])
      n1 = int(linea[3])
      b = blur(Imagen_desenfoque,n,n1)
      cv2.imshow("blur.jpg",b)
      cv2.waitKey(0)
      cv2.destroyAllWindows()
      cv2.imwrite("Copia_blur.jpg",b)


main()
```



Figura 12: principal

Retorno: Retorna una copia de la imagen original con alguno de los filtros que se pueden aplicar
Resumen: Basicamente a puros condicionales se elige una de las salidas que se quieren, y segun lo que pida el programa, va a llamar a alguna de todas las funciones

Nombre: Convertir-rgb-hsv
Parametros: nuevaImagen= es la copia de la imagen original sobre la cual se va a trabajar
Retorno: devuelve unacopia de la imagen en formato HSV
Resumen: va recorriendo toda la matriz de la imagen cambiando los valores para poder mostarla

Nombre: Convertir-hsv-rgb
Parametros: nuevaImagen= es la copia de la imagen original sobre la cual se va a trabajar
Retorno: devuelve unacopia de la imagen en formato RGB
Resumen: va recorriendo toda la matriz de la imagen cambiando los valores para poder mostrarla

Nombre: Matriz-convolución
Parametros: Kernel y desviacion
Kernel: La matriz con la cual se va a modificar la copa de la imagen original
Desviación estandar: Es el grado de dispersión. En este caso es el rango de alcanze de la matriz de convolución
Retorno: Retorna la matriz de convolución la cual se va a poder usar en otra funcion para el desenfoque Resumen: Mediante el uso del kernel y la desviación que nos va a indicar el rango de la matriz, vamos a crear la matriz de convolución mediante el uso de una formula matematica.

# 3. Conclusiones

Durante la realizacion de este proyecto surgierón varios problemas debido a la informacion de internet. A pesar de que toda la informacion necesaria para la realizacion del proyecto se encontraba ahí, cuando se visitaban varias paginas las formulas matematicas que se ocupaban, a veces eran diferentes, otro problema que hubo que resolver era que por mas que leiamos algunos conceptos no nos quedaban claros, pero al final todo se logro solucionar.
una vez que ya se sabían cuales eran las formulas correctas para la realizacion

del proyecto, las cosas se facilitaron bastante, por que apartir de ahí solamente era aplicar las formulas mediante diversos algoritmos, a pesar de todos los problemas logramos hacer muuy bien el proyecto y a continuacion se dan algunos ejemplos para ejemplificar los resultados.

Figura 13: imagen principal de itunes

Figura 14: itunes con filtro hue

Figura 15: itunes con filtro sat

Figura 16: itunes con filtro val

Figura 17: itunes con filtro blur

# 4. Referencias

.Herrera, a. (2015). Modelos de color (RGB, CMYK, HSV/HSL). Recuperado de https://ahenav.com/2014/04/09/modelos-de-color