

Visión Artificial



Título: Practica 5

Nombre: Mario Gerardo Casas Miramontes

Registro: 22310165

Objetivo

Objetivo: Utilizar las funciones de umbrales para la recuperación de información.

Threshold1 binary, b_inv, Trunc, To Zero, Tz_inv, Mean, Gaus, Otsu.

Funcionamiento

El funcionamiento de este programa es el uso de umbrales para la recuperación de información, dentro de este programa vemos los siguientes métodos para la extracción de información: Threshold1 binary, b_inv, Trunc, To Zero, Tz_inv, Mean, Gaus, Otsu.

Siendo métodos con los cuales podemos extraer la información de la imagen mediante la modificación de parámetros de la imagen según sean los valores de los colores que tengan los píxeles, teniendo los anteriores métodos dichos

Código Fuente

#Mario Gerardo Casas Miramontes 22310165

```
import numpy
```

```
import matplotlib
```

```
import cv2
```

```
img = cv2.imread('bookpage.jpg') #Obtenemos la imagen
```

```
retval, threshold = cv2.threshold(img, 12, 255, cv2.THRESH_BINARY)
```

```
#Ponemos nuestra Imagen en escala de grises
```

```
grayscaled = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
#En esta parte se realiza el analisis de la imagen binaria, se puede decir que es un filtro en el que dependiendo de los valores del pixel sube a un valor o permanece
```

```
retval2, threshold2 = cv2.threshold(grayscaled, 12, 255, cv2.THRESH_BINARY)
```

```
#Aqui se usa la umbrlizacion por Gauss en donde se dividen en regiones en donde se realizaran las operaciones mediante Gauss
```

```
#El parametro de division es el 115 que es para generar division y sacar un promedio el cual se le resta la unidad y se consigue un valor
```

```
gaus = cv2.adaptiveThreshold(grayscaled, 255, cv2. ADAPTIVE_THRESH_GAUSSIAN_C,  
cv2.THRESH_BINARY, 115, 1)
```

```
#Este metodo se encarga de separar en dos picos el histograma.
```

```
retval3, otsu = cv2.threshold(grayscaled, 125, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)
```

```
retval4, threshold3 = cv2.threshold(img, 12, 255, cv2.THRESH_BINARY_INV)#Basicamente hace lo inverso a lo binario en donde los valores menores a 12 se vuelven blancos
```

```
retval5, threshold4 = cv2.threshold(img, 12, 255, cv2.THRESH_TRUNC)#Como dice su nombre, trunca los valores mayores a 12 a 12
```

```
retval6, threshold5 = cv2.threshold(img, 12, 255, cv2.THRESH_TOZERO)#El TO ZERO hace que los valores hace que los valores menores a 12 se vuelven 0
```

```
retval7, threshold6 = cv2.threshold(img, 12, 255, cv2.THRESH_TOZERO_INV)#Es lo inverso a lo anterior, los valores menores a 12 son iguales a 12 y los mayores se vuelven 0
```

```
thresh_mean = cv2.adaptiveThreshold(grayscaled, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 11, 2)
```

```
#Ponemos las imagenes en ventanas para ver el resultado
```

```
cv2.imshow('original',img)
```

```
cv2.imshow('threshold',threshold)
```

```
cv2.imshow('threshold2',threshold2)
```

```
cv2.imshow('Binary INV',threshold3)
```

```
cv2.imshow('Trunc',threshold4)
```

```
cv2.imshow('Tozero',threshold5)
```

```
cv2.imshow('Tozero INV',threshold6)
```

```
cv2.imshow('Mean',thresh_mean)
```

```
cv2.imshow('gaus',gaus)
```

```
cv2.imshow('otsu',otsu)
```

```
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```

Resultado



