EBook Gratis

APRENDIZAJE computer-vision

Free unaffiliated eBook created from
Stack Overflow contributors.

#computer-vision

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: computer-vision

It is an unofficial and free computer-vision ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official computer-vision.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con la visión por computadora

Observaciones

El procesamiento digital de imágenes y la visión por computadora es un campo interesante, ya que se encuentra en una ubicación hermosa entre Matemáticas y Ciencias de la computación. Por lo tanto, es muy útil comprender los fundamentos y aplicarlos usando la programación para entender el tema.

Las imágenes digitales son discretización de señales bidimensionales o tridimensionales. En otras palabras, las imágenes digitales son conjuntos muestreados de píxeles o vóxeles de dominios continuos.

$$f : R^2 \supset \Omega \rightarrow R$$

Donde f es imagen digital sobre Ω: dominio de imagen rectangular

En aras de la simplicidad, solo analizaremos imágenes digitales bidimensionales, como las de nuestros avatares StackOverflow.

Acerca de los píxeles: una nota rápida sobre los valores de los píxeles antes de comenzar a analizar los tipos de imágenes. Como regla general, los píxeles comienzan con el valor 0 que denota que no hay luz (negro), llega a 1, intensidad máxima (por ejemplo, blanco) y se representan en números enteros.

Imágenes binarias: imágenes en blanco y negro solamente. Cada píxel es 0 o 1, cada píxel se puede representar con un bit. No se conocen muy popularmente, ya que se usan generalmente en aplicaciones científicas o para otras operaciones de procesamiento de imágenes como máscara, por ejemplo.

Un ejemplo de imagen binaria. (Al advertir que los valores de píxeles de imagen de este archivo

no son necesariamente binarios, esto es para demostración, también es Lena, la estrella del mundo de Procesamiento de imágenes)

Imágenes en escala de grises: gracias a los filtros en línea, todos aún conocen muy bien estas imágenes. Estas imágenes generalmente son de un byte por píxel, con 0 en negro y 255 en blanco, todo entre un tono diferente y gris, dado que los humanos solo pueden distinguir 40 tonos de gris, este rango es suficiente para muchas aplicaciones (tenga en cuenta que los valores de píxeles aquí se asignan de 0 a 1 a los valores de byte 0 - 255)

Imágenes en color: Finalmente, el tipo de imagen digital más común, imágenes en color. Tenemos que mencionar el concepto de canales aquí. Las imágenes digitales, también tienen canales, en realidad, las imágenes binarias y en escala de grises descritas anteriormente también tienen canales. La descripción más común sería el modelo RGB (Rojo-Verde-Azul), en tal caso, la imagen tiene 3 canales (No la confunda con las dimensiones, estas son imágenes en 2D) para describir el enrojecimiento, el azul y el verdor de la imagen. En este caso, cada píxel es un triplete, un valor entre 0 - 255 (Sin rojo a la mayoría de los rojos), 0 - 255 (Sin verde a la mayoría de los verdes), 0 - 255 (Sin azul a la mayoría de los azules). Para este modelo, el píxel {0,0,0} es negro, {255,255,255} es blanco, {255,0,0} es rojo, {255, 255, 0} es amarillo. Sin embargo, el color es un tema muy amplio y puede consultar las referencias para obtener más información.

Imágenes hiperespectrales:

Después de que hablamos de los canales, hablar de imágenes hiperespectrales es más fácil. Estas imágenes pueden tener cientos de canales y se usan comúnmente en microscopía, imágenes satelitales, etc.

Lecturas

1. Muestreo de señales: https://en.wikipedia.org/wiki/Sampling_(signal_processing)
2. Biblia del Procesamiento de Imágenes Digitales: RC Gonzalez, RE Woods: Procesamiento de Imágenes Digitales. Tercera edición, Pearson Prentice Hall, Upper Saddle River, 2008.
3. Revisión de la Visión por Computadora (Hasta el Aprendizaje Profundo): R. Szeliski: Visión por Computadora: Algoritmos y Aplicaciones. Springer, Nueva York, 2010.
4. Para comprender las imágenes binarias, en escala de grises y en color: https://en.wikipedia.org/wiki/Grayscale

Examples

Instalación o configuración

Instrucciones detalladas sobre cómo configurar o instalar la visión artificial.

Para esta y las siguientes series de visión artificial, usaré Python 2 como lenguaje de programación. Python es una opción común para la comunidad científica, es gratuita, tiene muchas bibliotecas gratuitas y de código abierto, y si eres nuevo en la programación, es una de las más sencillas de aprender y comenzar a programar.

Ahora configura, si usas Linux, probablemente ya tengas Python, simplemente abre el terminal y escribe "python" para verificar si todo está funcionando. Otros, pueden consultar este enlace y descargar python 2.7.

Segundo, necesitaremos instalar las bibliotecas que vamos a utilizar en el código fuente. Ahora, una nota aquí, esta configuración está diseñada para este ejemplo, en etapas posteriores, agregaremos diferentes bibliotecas y, por supuesto, diferentes aplicaciones de visión artificial pueden requerir bibliotecas específicas, como OpenCV. Solo necesitaremos una biblioteca para instalar en nuestro sistema para ejecutar el código. Cuando uso python, generalmente instalo dependencias usando 'pip'. Es una herramienta simple para instalar los módulos de python, también puede verificarlo a través de este enlace

Ahora, estamos listos para instalar la biblioteca que necesitamos, PyPNG. Si está utilizando pip todo lo que necesita hacer es

    pip instalar PyPNG

en la terminal si está usando Linux / Mac, en la línea de comandos si está usando Windows.

Además, para estos ejercicios, necesita obtener imágenes que se pueden encontrar en el enlace de github junto con el código fuente y las libretas ipython.

https://github.com/Skorkmaz88/compvis101

Ahora, deberíamos estar bien para ir a hacer ejercicio.

Ejemplos

Esta es una serie de ejercicios Python muy simples para el procesamiento de imágenes y la visión por computadora, diseñados para introducir estos temas con pocas prácticas. Lo siento de antemano por cualquier error de novato, todavía está en desarrollo. En esta serie, limitaré las imágenes digitales que podemos usar a los archivos PNG por simplicidad, que también trataré sobre el tema de la compresión de imágenes.

Clone el repositorio si aún no lo ha hecho, o simplemente puede descargarlo aquí a través de Github:

```
git clone https://github.com/Skorkmaz88/compvis101
```

Hay dos archivos que puedes usar, uno de ellos es tutorial0.py y otro es readingImages.ipynb, el segundo es un cuaderno de ipython. Si prefieres usarlo, puedes hacerlo. Pero dos archivos están haciendo lo mismo.

El código tiene explicaciones en los comentarios, estoy

```
 # libs
import png

# We create a greyscale image as described in our text.
# To do that simply, we create a 2D array in python.
# x and y, x being horizontal and y being vertical directions.

x  = []
y = []
# Play around with these pixels values to get different grayscale images, they shoud be
# in range of 0 - 255.
white = 255
gray = 128
black = 0
width  = 100
height = 300

# Add 100 x 100 rectangle as just white(255) valued pixels
for i in range(0, 100):
    for j in range(0,100):
        y.append(white); # Pixel (i,j) is being set to a value, rest is coding trick to nest
two lists
    x.append(y)
    y = []

# Add 100 x 100 rectangle as just mid-gray(128) valued pixels
for i in range(0, 100):
    for j in range(0,100):
        y.append(gray);
    x.append(y)
    y = []
```

```
# Add 100 x 100 rectangle as just black(0) valued pixels
for i in range(0, 100):
    for j in range(0,100):
        y.append(black);
    x.append(y)
    y = []

# output image file
f = open('out.png', 'wb')
w = png.Writer(width, height , greyscale=True, bitdepth=8)
w.write(f, x)
f.close()
# If everything went well, you should have 3 vertically aligned rectangles white, gray and
black
# Check your working folder

# PART 2
# Read a grayscale image and convert it to binary

# This time we will binarize a grayscale image, to do that we will read pixels and according
to threshold we set
# we will decide if that pixel should be white or black

# This file is originally 8 bit png image, can be found in github repository, you should use
only this type of
# images if you want to change the image.
f = open('./img/lenaG.png', 'r')

r=png.Reader(file=f)
# You will the details about the image, for now pay attention to size and bitdepth only.
img = r.read()

width = img[0]
height = img[1]
# Threshold value for binarizing images,
threshold = 128
print "Input image size is: "+ str(width)+ " pixels as  width, " + str(height) + " pixels as
height"

f_out = open('lenaBinary.png', 'wb')
w = png.Writer(width, height , greyscale=True, bitdepth=1)

pixels = img[2]

x = []
y = []

# Let's traverse the Lena image
for row in pixels:
    for pixel in row:
        p_value =  pixel
        # Now here we binarize image in pixel level
        if p_value > threshold:
            p_value = 1
        else:
            p_value = 0

        y.append(p_value);
    x.append(y)
    y = []
```

```
w.write(f_out, x)
f_out.close()
```

Si todo funcionó bien, felicidades! Creó una imagen desde cero y realizó la transformación del primer píxel en una imagen existente. Consulta tu carpeta de trabajo para ver las nuevas imágenes.

Lea Empezando con la visión por computadora en línea: https://riptutorial.com/es/computer-vision/topic/5710/empezando-con-la-vision-por-computadora

Creditos

S. No	Capítulos	Contributors
1	Empezando con la visión por computadora	Community, Semih Korkmaz