Visión por Computador
Prácticas: P0

N.Pérez de la Blanca

DECSAI

Guía de instalación del Software

- Instalar Anaconda-Python y opencv-python
- Tensorflow y Keras no son necesarios por ahora, pero lo serán más adelante.
- Abrir o acceder a la cuenta de Google
- Acceder al recurso Colab de Google y crear un directorio en Drive:
  - https://colab.research.google.com/notebooks/intro.ipynb#recent=true

Colab

- Conectarse para tener una primera impresión de las características de colab https://colab.research.google.com/notebooks/basic_features_overview.ipynb
- Para importar ficheros de DRIVE a Colab usar:
  - **from google.colab import drive**
  - **drive.mount('/content/gdrive')**
  - **!ls '/content/gdrive/My Drive'** (verificar que funciona)

OpenCV

- Espacio de nombres: cv2
- Estructura de datos: matrices de n-canales (compatibles numpy)
- Comandos python: objetos
- Matrices e imágenes con cosas distintas
- Imágenes OpenCV: matrices peculiares
  - 1 o 3 bandas
  - valores uchar en [0,255] o float en [0,1]
  - Almacena color como BGR (en python RGB)
- https://docs.opencv.org/master/

Tipos de dato: OpenCV|Numpy

- CV_8U - 8-bit unsigned integers (0..255) | np.uint8
- CV_8S - 8-bit signed integers (-128..127) | np.int8
- CV_16U - 16-bit unsigned integers (0..65535) | np.uint16
- CV_16S - 16-bit signed integers (-32768..32767) | np.int16
- CV_32S - 32-bit signed integers (-2147483648..2147483647) | np.int32
- CV_32F - 32-bit floating-point numbers (-FLT_MAX..FLT_MAX, INF, NAN) | np.float32
- CV_64F - 64-bit floating-point numbers (-DBL_MAX..DBL_MAX, INF, NAN) | np.float64

Tipos de imágenes y su almacenamiento

- Tipos de imágenes:
  - B&W : solo dos valores (np.uint8)
  - Grises: 255 valores (np.uint8)
  - RGB: 255x255x255 valores (np.uint8,np.int32)
  - Indexadas: N valores de una tabla de valores.
- Almacenamiento:
  - 1 bit, 1 byte, 3 bytes por píxel
- Cualquier matriz de 1 o 3 canales puede convertirse en imagen y ser visualizada.

¿Cómo leer y visualizar imágenes?

- Im= cv2.imread(filename,flag_type)
  - Filename debe incluir el formato (.png, .jpeg, .tiff, pgm,ppm,etc)
  - Flag_type (0|1): indica tipo de salida (gray|color)(1|3 canales)

- ¿Cómo visualizar imágenes/matrices?
  - cv2.imshow(winame, matriz) muestra una **IMAGEN OpenCV!!**
  - Las matrices hay que convertirlas a imágenes OpenCV
- ¿Cómo visualizar en ejecución?
  - La ejecución hay que pararla si queremos visualizar una imagen
  - cv2.waitKey(0) permite parar hasta la presión de cualquier tecla
  - cv2.namedWindow(winname, flag) permite crear ventanas con nombre y múltiples propiedades de forma.
  - cv2.destroyWindow(winname) elimina la ventana con nombre winname
  - cv2.destroyAllWindows() elimina todas las ventanas abiertas
  - cv2.setWindowTitle(winname, title)

Visualizar Imágenes/Matrices

- Crear una función que tomando como entrada una matriz haga:
  1. Mire si el número de canales es el adecuado.
  2. Mire el tipo de dato y adapte los valores a los intervalos permitidos
  3. Crear una ventana
  4. Visualice la imagen en la ventana
  5. Destruir la ventana

- Acceso a las propiedades de una matriz OpenCV
  - cv2.Variable.depth() (tipo de dato)
  - cv2.Variable.channels() (número de canales)
- Acceso en Numpy: variable.dtype()

Juntando matrices en Numpy

- Numpy ayuda a la visualización:
  - vstack(), hstack() (solo matrices con igual forma !!)
  - Otros casos: usar copy, etc, etc….
  - https://numpy.org/devdocs/reference/routines.array-manipulation.html

Ejercicios

- 1.- Escribir una función que lea el fichero de una imagen permita mostrarla tanto en grises como en color (im=leeimagen(filename, flagColor))
  - Lectura + visualización

- 2.- Escribir una función que visualice una matriz de números reales cualquiera ya sea monobanda o tribanda (pintaI(im)). Para ello deberá de trasladar y escalar el rango de cada banda al intervalo [0,1] sin pérdida de información.
  - Lectura + verificación de tipo + adaptación de tipo+ visuaización

- 3.- Escribir una función que visualice varias imágenes a la vez (fusionando las imágenes en una última imagen final): pintaMI(vim). (vim será una secuencia de imágenes) ¿Qué pasa si las imágenes no son todas del mismo tipo (nivel de gris, color, blanco-negro)?
  - Lectura + adaptación a profundidad común + concatenación + visulización

- 4.- Escribir una función que modifique el color en la imagen de cada uno de los elementos de una lista de coordenadas de píxeles. (Recordar que (fila, columna) es lo contrario a (x,y). Es decir fila=y, columna=x)
  - Acceso y modificación de los valores de la imagen (1D o 3D)

- 5.- Una función que sea capaz de representar varias imágenes con sus títulos en una misma ventana.
  - Igual que 3 pero con títulos en las ventanas