Guía de Instalación del Software de Prácticas

Visión por Computador

1. Instalación

Las implementaciones se realizarán usando la librería de algoritmos OpenCV a través de dos posibles lenguajes de interface Python o C++. Debido al excesivo consumo de tiempo en la carga de la imagen, no se dispondrá de una instalación del software en las aulas de prácticas. Para poder usar los ordenadores de las aulas se deberá hacer una instalación de OpenCV+Python en un pen-drive/disco removible que se usará con dichos ordenadores (>4 Gbytes).

Se recomienda que siempre que sea posible el software de prácticas se instale en el ordenador personal de cada alumno.

Requerimientos: 4 GB de memoria de disco mínimo.

Sistemas Operativos compatibles: W10, Linux, MacOS.

Lenguajes de uso: C++ y Python 3.6.

S.O. recomendado: Windows 10.

1.1. Instalación en Windows 10

Descargar la versión 3.4.3 de OpenCV.
Instalación de OpenCV para Python en Windows 10:

1. Descargar e instalar la distribución Anaconda de Python 3.6 para Windows, siguiedo las instrucciones del paquete.
2. Verificar que todo fue bien ejecutando en el prompt de Anaconda:

   `python -V` (da mensajes con información).

   `pip -V` (da mensajes con información).
3. Ejecutar `conda upgrade numpy` (decir [y] a las preguntas).
4. Ejecutar `conda upgrade spyder` (decir [y] a las preguntas).
5. Descargar los binarios de opencv de la página `https://www.lfd.uci.edu/~gohlke/pythonlibs/#opencv`

   `opencv_python-3.4.3-cp36-cp36m-win_amd64.whl` (o win32 en su caso).

6. Instalar los binarios ejecutando: `pip install <nombre del fichero descargado>`.
7. Si todo ha ido bien ejecutar spyder, se levantará un IDE de desarrollo en python.
8. Ir a `https://docs.opencv.org/3.4.3/d6/d00/tutorial_py_root.html`
   a) Copiar código python de prueba en la ventana de desarrollo y ejecutarlo, o
   b) Ejecutarlo paso a paso en la terminal.
9. Ver la documentación sobre la estructura y funcionalidad de la librería en `https://docs.opencv.org/3.4.3/`

1.2. Instalación en Ubuntu 18.04

A1. Instalar la distribución Anaconda de python 3.6.

A2. Seguir las instrucciones de `https://www.pyimagesearch.com/2018/05/28/ubuntu-18-04-how-to-install-opencv/`

Nota. Si Spyder no se inicia después de seguir las instrucciones, ejecuta `conda install pyopengl`.

O alternativamente

B1. Seguir las instrucciones de `https://www.pyimagesearch.com/2018/05/28/ubuntu-18-04-how-to-install-opencv/`.

B2. Instalar spyder: `https://pythonhosted.org/spyder/installation.html`.

En caso de que utilices un virtual environment para instalar OpenCV, con cualquiera de las dos variantes, es necesario que después configures Spyder para que use ese environment. En Herramientas > Preferencias > Intérprete de Python pinchar en Usar el siguiente intérprete y en `.virtualenvs` buscar python3.6. Cerrar y volver a abrir Spyder e instalar con pip los paquetes que nos diga en la consola de Ipython en el entorno virtual. Volver a cerrar y abrir Spyder.

1.3. Instalación en MacOS X

A1. Seguir las instrucciones de `https://www.learnopencv.com/install-opencv3-on-macos/`.

A2. Instalar Spyder con `pip install spyder`.

O alternativamente

B1. Instalar la distribución Anaconda.

B2. Instalar OpenCV usando: `conda install -channel https://conda.anaconda.org/menpo opencv3` (ver https://solarianprogrammer.com/2016/11/29/install-opencv-3-with-python-3-on-macos/).

O también

C1. Seguir las instrucciones de `https://www.pyimagesearch.com/2016/12/05/macos-install-opencv-3-and-python-3-5/`.

2. Guía Rápida de Arranque con Python + Spyder

El lenguaje python es un lenguaje interpretado que comparte mucha de la filosofía del lenguaje Matlab (estándar actual para el prototipado y desarrollo de aplicaciones de cálculo). Por ello es un lenguaje que permite un rápido aprendizaje y un rápido desarrollo de programas con el objetivo de experimentar técnicas de cálculo. Una de sus pincipales ventajas es ser de dominio libre y estar disponible para todas las plataformas (Windows, Linux, MacOS).

El desarrollo de aplicaciones python puede realizarse desde un editor de texto ascii y guardando el fichero con extensión .py (ejemplo.py). Para ejecutar la aplicacion sólo habrá que llamar al ejcutable python dándole como argumento el fichero: `$python ejemplo.py` o `c:>python.exe ejemplo.py` . Aunque esta forma de trabajo puede ser útil para usuarios avanzados, para usuarios que comienzan es mejor usar una plataforma de desarrollo que permita desarrollar al mismo tiempo que integrar depuradores para la correción de errores. Spyder es una de las plataformas actuales más avanzadas además de ser de dominio libre.

Para que el lenguaje python sea útil en el desarrollo de aplicaciones es neceasrio añadirle toda una serie de librerías (paquetes o módulos en terminología python) que permitan realizarlas (p.e. cálculo numérico, dibujo de gráficas, lectura de ficheros, algoritmos de aprendizaje, etc). La distribución Anaconda integra todas las librerias básicas necesarias menos las de algoritmos de aprendizaje. Por tanto además de instalar Anaconda instalaremos el módulo `scikit-learn`.

2.1. Comencemos

Una vez tenga instalado todo el software de acuerdo a la guía de instalación, ejecute Spyder, se le abrirá un entorno de ventanas con tres ventanas principales (editor, [object inspector+variables+files] y consola).

En consola se ha levantado una terminal de IPython donde se pueden ejecutar de forma interactiva cualquier comado o llamada python. Con el comando print podrá mostrar valores de variable (`print(a)`, mostrará el valor de `a`).

En editor se muestra un editor de texto con información sobre el sistema de codificación que se usa y el nombre del autor del código. (# es el símbolo para la línea de comentario). En este fichero podemos comenzar a escribir nuestro código y podemos ir verificando su corrección ejecutándolo dándole al triángulo verde en la barra de herramientas de Spyder o en Ejecutar en la pestaña Ejecutar.

El código propio se puede estructurar con distintos niveles de complejidad: a) código plano sin llamadas a funciones propias; b) código con llamadas a funciones propias previamente definidas, c) código con clases propias. Se adjuntan ejemplos de cada caso en la comprobación de la instalación.

2.2. Depuración

Leamos el fichero de prueba `scikit0.py` desde la opción File+Open file. Podemos verificar que es un código correcto ejecutándolo con Run y viendo sus resultados. Además, podemos verificar que en la consola no nos aparece ningún mensaje de error. Si queremos ejecutar paso a paso pincharemos en la opcion Depurar del menú Depurar y veremos que en la terminal aparece el símbolo (ipdb) que implica que el depurador está funcionando. Además, en el editor la primera linea ejecutable se ha enfatizado en color, indicando en qué línea estamos. Para avanzar debemos ir ejecutando en la consola ipdb>n [Return]. Veremos que la línea enfatizada del editor se desplaza hacia abajo y que en el la ventana "variables" deben ir apareciendo las variables que se van instanciando. Obviamente esto es sólo si estamos intereados en hacerlo paso a paso. Si queremos ir hasta una instrucción de forma directa, entonces debemos hacer lo siguiente: a) pinchar en la instrucción a la que queremos ir; b) El el menu Run poner un breakpoint (aparecerá un círculo de color rojo); c) en la ventana del depurador ejecutar: (ipdb) c [Return]. Se ejecuta de forma continua hasta que encuentre un punto de interrupción, acabe el código o encuentre un error. (de momento no tratar de entenderlo).

En python al igual que en C++ toda estructura que no sea una variable simple está definida por una clase que tiene métodos asociados (matrices, vectores, listas, etc). En la ventana variables, podemos ver los valores de todas las variables que han sido instanciadas hasta el momento e incluso editar sus valores, pinchando en celda de valores y editándola. Tambien nos muestra el tipo de dato que aloja. Esto es muy útil ya que en python como en C/C++ los resultados de las operaciones númericas pueden depender del tipo de dato (p.e. división).

2.3. Ayuda on-line

Programación: Python se estructura en tres niveles básicos, funciones, módulos y paquetes. La función es el elemento básico del ejecución y es el concepto equivalente a la función en C++. Sin embargo en python los argumentos sólo se pasan por valor. Un módulo es un conjunto de funciones y variables que están dentro de un mismo fichero (equivalente a una clase en C++). Un paquete es una agregación de módulos. Cuando estamos programando el entorno Spyder nos proporciona informacion de los objetos que estamos manejando en la ventana "Ayuda"(Ctrl+I), sólo tenemos que escribir el nombre del objeto. Ademas en el editor podemos acceder a todos los métodos y variables de un objeto poniendo por ejemplo `nombre_objeto`. Esto debe visualizar todos los métodos y variables de `nombre_objeto`. Para conocer los parámetros de una función sólo hay escribir el primer paréntesis y aparecerá la cabecera [ej. `nombre_metodo(`]. Se pueden hacer llamadas al sistema operativo a través de os.system() para ejecutar comandos que nos interesen. Cuando esto ocurra en alguna práctica y con objeto de aislar la llamda del S.O. que se esté usando deberemos definir un fichero script ejecutable con la llamada y la llamada desde desde python será el fichero.

Python y librerías: En la ayuda de Spyder hay un excelente compedio de documentos de ayuda tanto del lenguaje python como del resto de paquetes y módulos (numpy, scipy, etc). Se recomienda leer el Tutorial que hay en Python Documentation dentro de la pestaña Ayuda del menu principal de Spyder.

Manejo de Spyder: Usar la documentación disponible dentro de la pestaña Ayuda del menú principal de Spyder.

Scikit_Learn: La mejor ayuda al uso de esta librería se encuentra en su propia página web. Numerosos ejemplos en python muestran no sólo el uso de los algoritmos, sino que además dan muchas pistas sobre cómo es posible desarrollar resultados gráficos de interés a partir de los resultados de los experimentos.

2.4. Comprobando la instalación y aprendiendo Python

1. Abrir el fichero `P0.zip` en un directorio pruebas.
2. Situarse en ese directorio con el entorno Spyder.
3. Leer y ejecutar el fichero `python_basics.py`
4. Generará una salida por la consola phyton. Este fichero contiene comentarios de interés sobre el lenguaje python. Leer el fichero para irse familiarizando con la sintaxis básica de python. Ejecutar el fichero paso a paso e ir viendo la salida y el código que la produce.
5. Leer y ejecutar python `exercises_test.py`. ¿Qué ves?
6. El fichero `exercises_test.py` realiza un conjunto de test para verificar la funcionalidad de las funciones y clases definidas en el fichero `ejercicios.py`. Las funciones están intencionadamente vacías. Modificar este fichero rellenando el contenido de las funciones para que `exercises_test.py` se ejecute correctamente. (Atención: en la lectura del fichero recordad que python sólo lee cadenas y por tanto hay que convertirlas a valores numéricos con int() o float(). Además, si en el fichero hay caracteres como "\n" u otros, habrá que eliminarlos antes de extraer el número, format()).
7. Leer y ejecutar el fichero `scikit0.py` de nuevo. Si leemos el código ahora aprenderemos algo sobre el uso de la funciones de aprendizaje y los gráficos.

Cualquier error que se pueda producir en la instalación se intentará subsanar en clase de prácticas.