Contents

[Opencv 1](#_Toc479162330)

[simpleITK 3](#_Toc479162331)

[VTK 3](#_Toc479162332)

[Python+VTK 3](#_Toc479162333)

[VTK source code dissect 3](#_Toc479162334)

# Opencv

Linux 安装

<http://www.pyimagesearch.com/2016/10/24/ubuntu-16-04-how-to-install-opencv/>

Step #1: Install OpenCV dependencies on Ubuntu 16.04

refresh and upgrade and pre-installed packages/libraries

$ sudo apt-get update

$ sudo apt-get upgrade

developer tools

$ sudo apt-get install build-essential cmake pkg-config

图像导入依赖库

$ sudo apt-get install libjpeg8-dev libtiff5-dev libjasper-dev libpng12-dev

视频导入依赖库

$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev

GUI依赖库

$ sudo apt-get install libxvidcore-dev libx264-dev libgtk-3-dev

数值计算依赖库

$ sudo apt-get install libatlas-base-dev gfortran

Python development headers and libraries

$ sudo apt-get install python2.7-dev python3.5-dev

Step #2: Download the OpenCV source

$ cd ~

$ wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.1.0.zip

$ unzip opencv.zip

$ wget -O opencv\_contrib.zip https://github.com/Itseez/opencv\_contrib/archive/3.1.0.zip

$ unzip opencv\_contrib.zip

Step #3: Setup your Python environment — Python 2.7 or Python 3

Step #4: Configuring and compiling OpenCV on Ubuntu 16.04

$ cd ~/opencv-3.1.0/

$ mkdir build

$ cd build

$ cmake -D CMAKE\_BUILD\_TYPE=RELEASE \

-D CMAKE\_INSTALL\_PREFIX=/usr/local \

-D INSTALL\_PYTHON\_EXAMPLES=ON \

-D INSTALL\_C\_EXAMPLES=OFF \

-D OPENCV\_EXTRA\_MODULES\_PATH=~/opencv\_contrib-3.1.0/modules \

-D PYTHON\_EXECUTABLE=~/.virtualenvs/cv/bin/python \

-D BUILD\_EXAMPLES=ON ..

If download ippicv\_linux\_20151201.tgz failure

Please download and *cp ippicv\_linux\_20151201.tgz ~/opencv-3.1.0/3rdparty/ippicv/downloads/linux-808b791a6eac9ed78d32a7666804320e/ippicv\_linux\_20151201.tgz*

$ make -j8

若出现编译出法成功，可能是cuda8与opencv3.1中的算法Graphcut冲突，解决方法：

<https://devtalk.nvidia.com/default/topic/986950/opencv-installation-problem-nppigraphcutinitalloc-not-declared/>

将文件opencv-3.1.0/modules/cudalegacy/src/graphcuts.cpp

#if !defined (HAVE\_CUDA) || defined (CUDA\_DISABLER) || (CUDART\_VERSION >= 8000)

$ sudo make install

$ sudo ldconfig

>>> import cv2

>>> cv2.\_\_version\_\_

特征检测

角点

某些属性上强度最大或者最小的孤立点、线段的终点，或者是曲线上局部曲率最大的点

角点是二维图像亮度变化剧烈的点或图像边缘曲线上曲率极大值的点，这些点在保留图像图形重要特征的同时,可以有效地减少信息的数据量,使其信息的含量很高,有效地提高了计算的速度,有利于图像的可靠匹配,使得实时处理成为可能

关于角点的具体描述可以有几种：

一阶导数(即灰度的梯度)的局部最大所对应的像素点；

两条及两条以上边缘的交点；

图像中梯度值和梯度方向的变化速率都很高的点；

角点处的一阶导数最大，二阶导数为零，指示物体边缘变化不连续的方向。

# simpleITK

<https://itk.org/Wiki/SimpleITK/GettingStarted>

$ pip install --trusted-host itk.org -f https://itk.org/SimpleITKDoxygen/html/PyDownloadPage.html SimpleITK

Example: <https://github.com/SimpleITK/SimpleITK/tree/master/Examples/Python>

# VTK

## Python+VTK

最简单方式：

<http://stackoverflow.com/questions/15383666/installing-vtk-for-python>

$sudo apt-get install python-vtk

Example: <http://www.vtk.org/Wiki/VTK/Examples/Python>

官方：

<http://www.vtk.org/download/>

download vtkpython-7.1.0-Linux-64bit.tar.gz

## VTK source code dissect

Observer/command 模式

Command 模式

对象（vtkRender）发送一个命令,然后执行观察者相应的动作

Void StartEvent() -> ExecuteStart()

Void EndEvent() -> ExecuteEnd()

若有很多命令，同时有相应的动作，如何统一一个接口？

Void InvokeEvent(Event) -> Execute()

命令映射，从而使每个具体命令有唯一标识符（命令的名字作为主键）,用带命令标识符参数的函数统一接口InvokeEvent(Event),对于被调用的对象依次发起的每个命令(如StartEvent, EndEvent, ProcessEvent)，在客户端注册的观察者集合中查找匹配的命令，并执行客户端的动作

动作通过接口继承统一接口Execute().

Observer模式

客户端注册命令和相应的动作（事件和相应的回调函数作为一个观察者），从而可以在对象运行时，了解对象的状态。（多个观察者接注册顺序添加到对象的变量列表中如vtkRender的vtkObservers，从而有机会被通知到干活）

应用程序运行时，会依次发起客户端注册的事件，从而触发客户端的回调函数.(一对多)

（另一个例子是界面语言切换，发起事件，从而每个注册的界面切换语言）



VTK 使用心得

VTK source directory Structure

InfoVis

-----classes for information visualization

Views

-----classes for viewing data including filters, visualization, interaction and selection

Common

-----core classes

Filtering

-----classes related to data processing in the visualization pipeline

GenericFiltering

-----an adaptor framework to interface VTK to external simulation packages

GeoVis

-----view, sources and other objects useful in terrain visualization

Graphics

-----filters that process 3D data

GUISupport

-----classes for using VTK with the MFC and Qt user interface packages

Hybrid

-----complex classes that depend on classes in multiple other directories

Imaging

-----image processing filters

IO

-----classes for reading and writing data

Parallel

-----classes used to render

Utilities

-----supporting software like expat, png, jpeg, tiff and zlib

VolumeRendering

-----classes used for volume rendering

Widgets

-----3D widget classes

Wrapping

-----support for Tcl, Python, and Java wrapping.

Examples

-----examples, grouped by topic

CMake

-----configuration files for cross-platform building

Application software: ParaView

Except creating VTK application using the Tcl ( in this case, pre-compiled binaries may be available for the windows platform), you will have to compile and link the source code to produce libraries and executables.

VTK = visualization pipeline + rendering engine

Visualization pipeline is used to acquire or create data, process that data, and either write the results to a file or pass the results to the rendering engine for display

Rendering engine is responsible for creating a visual representation of the data

Actors: serves to group rendering attributes such as surface properties(e.g., ambient, diffuse, and specular color), representation(e.g., surface or wireframe), texture maps, and a geometric definition(a mapper)

Mappers: geometric definition using analytic primitives such as points, lines, polygons and triangle strips, the mapper terminates the visualization pipeline and serves as the bridge between the visualization subsystem and the graphics subsystem

Coordinate systems:

Display: x-y pixel values in the rendering window, the original is the lower-left corner

View: x-y-z(-1,1) values in camera coordinates(z is depth)

Operation performed on image data in VTK: image processing + geometry extraction + direct rendering