Contents

[Contents 1](#_Toc1327745299)

[scipy.ndimage 1](#_Toc1682846521)

[scikit-imagehttp://scikit-image.org/docs/stable/ 3](#_Toc45275314)

[Opencv 7](#_Toc1116109040)

[Linux 安装 7](#_Toc33182983)

[Ubuntu 7](#_Toc953012748)

[Via source code 7](#_Toc1460142365)

[《学习OpenCV》 8](#_Toc220782233)

[simpleITK 9](#_Toc231595231)

[VTK 9](#_Toc1550059516)

[Python+VTK 9](#_Toc1674722145)

[VTK source code dissect 10](#_Toc98589705)

Dicom

PyDicom工具包

# scipy.ndimage

**from scipy.misc** import imread, imsave, imresize, imfilter, imrotate, imshow, bytescale, central\_diff\_weights, comb, derivative, factorial, factorial2, fromimage, info, logsumexp, pade, toimage, source, who

import matplotlib.pyplot as plt

# Read an JPEG image into a numpy array

img = imread('assets/cat.jpg')

# 自带图像

from scipy.misc import lena, ascent, face

# Resize the image to be 300 by 300 pixels.

img\_tinted = imresize(img, (300, 300))

# Write the tinted image back to disk

imsave('assets/cat\_tinted.jpg', img\_tinted)

# Show the original image

plt.subplot(1, 2, 1)

plt.imshow(img)

**from scipy.io** import loadmat/savemat/whosmat, readsav, mminfo/mmread/mmwrite, FortranFile, netcdf\_file/netcdf\_variable, hb\_read/hb\_write, wavefile/read/write, arff/loadarff/MetaData

data =scipy.io.loadmat(‘test.mat’) load matlab data

**scipy.ndimage**

建立在numpy基础上

**2D可以交给opencv, muti-dimensional交给ndimage操作**

Filters

output = ndimage.convolve(input, weights) input可以3D

correlate, gaussian\_filter, gaussian\_gradient\_magnitude, gaussian\_laplace, generic\_filter, generic\_gradient\_magnitude, generic\_laplace, laplace, prewitt, sobel

# 邻域内最大值，size定义邻域shape, footprint可以正定义邻域，类似mask

output = ndimage.maximum\_filter(input, size=None, footprint=None)

minimum\_filter, median\_filter, uniform\_filter, percentile\_filter, rank\_filter

Fourier filters

fourier\_ellipsoid, fourier\_gaussian, fourier\_shift, fourier\_uniform

Interpolation

基于各种坐标变换进行插值

affine\_transform, geometric\_transform, map\_coordinates, rotate, shift, spline\_filter, zoom

Measurements

基于label（也就是基于各区域）计算统计量

center\_of\_mass, extrema, find\_objects, histogram, label, maximum/minimum, maximum\_position/minimum\_position, sum/mean/median/standard\_deviation/variance, watershed\_ift

Morphology

generate\_binary\_structure/iterate\_structure, binary\_closing/binary\_dilation/binary\_erosion/binary\_opening, binary\_fill\_holes, binary\_hit\_or\_miss, black\_tophat/white\_tophat, binary\_propagation,

distance\_transform\_bf/distance\_transform\_cdt/ distance\_transform\_edt, grey\_closing/grey\_dilation/grey\_erosion/grey\_opening, morphological\_gradient/morphological\_laplace

# scikit-imagehttp://scikit-image.org/docs/stable/

scikit-image.org/docs/dev/

The scikit-image SciKit (toolkit for SciPy) **extends scipy.ndimage** to provide a versatile set of image processing routines

# $sudo apt-get install python-skimage

$sudo pip install scikit-image

example:

<http://scikit-image.org/docs/stable/auto_examples/index.html>

user guide:

<http://scikit-image.org/docs/stable/user_guide.html>

维数次序

Matplotlib: standard Cartesian coordinates, where x is the horizontal coordinate, y the vertical, and the origin is on the bottom left.

Two-dimensional (2D) grayscale images: are indexed by (row, col), with the lowest element (0, 0) at the top-left corner

Dimension name and order conventions in scikit-image

Image type coordinates

2D grayscale (row, col)

2D multichannel (eg. RGB) (row, col, ch)

3D grayscale (pln, row, col)

3D multichannel (pln, row, col, ch)

2D color video (t, row, col, ch)

3D multichannel video (t, pln, row, col, ch)

# 2D 处理方式

from skimage import filters

edges = np.zeros\_like(im3d)

for pln, image in enumerate(im3d):

# iterate over the leading dimension (planes)

edges[pln] = filters.sobel(image)

# 3D 处理方式， skimage很多图像操作支持3D处理

from scipy.ndimage import label

from skimage import morphology

seeds = label(im3d < .1)[0]

ws = morphology.watershed(im3d, seeds)

# 索引，ndarray该有的都有，因为返回的就是ndarray表示图像

camera[10, 20]

mask = camera < 87

lower\_half = row > cnt\_row

camera[np.logical\_and(lower\_half, mask)] = 255

图像数据类型及转换

**from skimage import** dtype\_limits, img\_as\_bool, img\_as\_float, img\_as\_float32, img\_as\_float64, img\_as\_int, img\_as\_ubyte, img\_as\_uint

Data type Range Function name

uint8 0 to 255 img\_as\_ubyte()

uint16 0 to 65535 img\_as\_uint()

uint32 0 to 232

float -1 to 1 or 0 to 1 img\_as\_float ()

int8 -128 to 127

int16 -32768 to 32767 img\_as\_int()

int32 -231 to 231 - 1

min, max = dtype\_limits(image)

图像输入输出

**from skimage.io** import imread/imread\_collection/imread\_collection\_wrapper, imsave, show/imshow/imshow\_collection, load\_sift/load\_surf, concatenate\_images, ImageCollection/MultiImage

moon = imread(filename) 返回ndarray

out = img\_as\_uint(sobel(image))

plt.imshow(out)

**from skimage.viewer** import CollectionViewer/ImageViewer

CollectionViewer([data.coins(), data.astronaut()]).show() 显示图片序列

# 自带图像

**from skimage.data** import astronaut, binary\_blobs, camera, checkerboard, chelsea, clock, coffee, coins, horse, hubble\_deep\_field, moon, page, rocket, stereo\_motorcycle, text

图像Augment

from skimage.util import random\_noise 添加噪声

from skimage.util import crop, pad,

画图形

**from skimage.draw** import bezier\_curve, circle/circle\_perimeter/circle\_perimeter\_aa, ellipse/ellipse\_perimeter, ellipsoid, line/line\_aa, polygon/polygon\_perimeter, rectangle,

rr, cc = bezier\_curve(1, 5, 5, -2, 8, 8, 2) (1,5)起点，(5, -2)控制点，(8, 8)终点

img = np.zeros((10, 10), dtype=np.uint8)

img[rr, cc] = 1

rr, cc = circle(4, 4, 5) (4, 4)中心点， 5半径

img[rr, cc] = 1 圆盘

rr, cc = circle\_perimeter(4, 4, 5)

img[rr, cc] = 1 圆形

rr, cc = ellipse(5, 6, 3, 5, rotation=np.deg2rad(0)) (5, 6)中心点，(3, 5)长短轴半径

ellip = ellipsoid(a, b, c) 隋球体

vol, surf = ellipsoid\_stats(a, b, c) 隋球体体积和表面积

rr, cc = line(1, 1, 8, 8)

r = np.array([x0, x1, x2, …])

c = np.array([y0, y1, y2, ...])

rr, cc = polygon(r, c) 多边形区域

图像顡色空间转换

**from skimage.color** import gray2rbg, rbg2gray, rgb2hsv, rgba2rgb, …

图像灰度变换

**from skimage.exposure** import adjust\_gamma/adjust\_log/adjust\_sigmoid, cumulative\_distribution, equalize\_adapthist/equalize\_hist, histogram, is\_low\_contrast, rescale\_intensity

Gamma Correction: 0 = I \*\* gamma

Logarithmic Correction: 0 = gain \* log(1 + I)

Sigmoid Correction: 0 = 1/(1 + exp(gain\*(cutoff-I)))

equalize\_adapthist: Contrast Limited Adaptive Histogram Equalization (局部对比度增强）

equalize\_hist: 直方图均衡化

is\_low\_contrast: 亮度范围与数据类型范围的比率低于阀值

[0, 127, 255] = rescale\_intensity(np.array([51, 102, 153]) [min, max] → [0, 255]

[0., .5, 1.] = rescale\_intensity(np.array([51., 102., 153.]) [min, max] → [0, 1]

[.2, .4, .6] = rescale\_intensity(np.array([51., 102., 153.], in\_range=(0, 255)) in\_range → [0, 1]

[.5, 1., 1.] = rescale\_intensity(np.array([51., 102., 153.], in\_range=(0, 102)) 截断

特征提取

**from skimage.feature** import blob\_dog/blob\_doh/blob\_log, corner\_fast/corner\_foerstner/corner\_harris/corner\_kitchen\_rosenfeld/corner\_moravec/corner\_orientations/corner\_shi\_tomasi/CENSURE, corner\_peaks/peak\_local\_max/corner\_subpix, daisy, draw\_multiblock\_lbp, greycomatrix/greycoprops/, hessian\_matrix/hessian\_matrix\_det/hessian\_matrix\_eigvals, structure\_tensor/structure\_tensor\_eigvals, local\_binary\_pattern/multiblock\_lbp, match\_descriptors/match\_template/plot\_matches/register\_translation/BRIEF/ORB, shape\_index, canny, hog

滤波

**from skimage.filters import** apply\_hysteresis\_threshold/threshold\_adaptive/threshold\_isodata/threshold\_li/threshold\_local/threshold\_mean/threshold\_minimum/threshold\_niblack/threshold\_otsu/threshold\_sauvola/threshold\_triangle/threshold\_yen, frangi, gabor/gabor\_kernel/wiener/LPIFilter2D, gaussian/hessian/laplace/median/prewitt/prewitt\_h/previtt\_v/roberts/roberts\_net\_diag/roberts\_pos\_diag/scharr/scharr\_h/scharr\_v/sobel/sobel\_h/sobel\_v, inverse

形态学

**from skimage.morphology import** ball/cube/diamond/disk/octagon/octahedron/rectangle/square/star, binary\_closing/binary\_dilation/binary\_erosion/binary\_opening/binary\_closing/thin, black\_tophat/white\_tophat, dilation/erosion/opening, h\_maxima/h\_minima, label/recontruction/watershed, remove\_small\_holes/remove\_small\_objects, local\_maxima/local\_minma, medial\_axis, convex\_hull\_image/convex\_hull\_object, skeletonize/skeletonize\_3d,

图论

**from skimage.graph import** route\_through\_array, shortest\_path, MCP, MCP\_Connect, MCP\_Flexible, MCP\_Geometric

测量

**from skimage.measure import** approximate\_polygon, block\_reduce, compare\_mse, compare\_nrmse, compare\_psnr, compare\_ssim, correct\_mesh\_orientation, find\_contours, grid\_points\_in\_poly, label, marching\_cubes\_classic/marching\_cubes\_lewiner/mesh\_surface\_are, moments/moments\_central/moments\_hu/moments\_normalized, perimeter, points\_in\_poly, regionprops, shannon\_entropy, subdivide\_polygon, CircleModel/EllipseModel/LineModelND

与OpenCV区别

OpenCV image data can be accessed (without copying) in NumPy (and, thus, in scikit-image). **OpenCV uses BGR (instead of scikit-image’s RGB) for color images**, and its dtype is uint8 by default

image = image[:, :, ::-1] #Converting BGR to RGB or vice versa

from skimage import img\_as\_float

**image = img\_as\_float(any\_opencv\_image) # skiimage <- opencv image**

from skimage import img\_as\_ubyte

**cv\_image = img\_as\_ubyte(any\_skimage\_image) # opencv image <- skiimage**

Image processing pipeline

def custom\_func(image):

image = img\_as\_float(image)

from skimage import img\_as\_float

processed\_image = custom\_func(func1(func2(image)))

Image adjustment: transforming image content

# Opencv

Linux 安装

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

Ubuntu

$ sudo pip install opencv-contrib-python

Via source code

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\_\_

《学习OpenCV》

直方图

在分析图像、物体和视频信息的过程中，我们常常把眼中看到的物体用直方图表示。直方图可以用来描述物体的色彩分布、物体边缘梯度模板、以及表示目标位置的当前假设的概率分布等

在每帧中，从输入的视频中检测感兴趣的色彩区域，然后计算这些感兴趣区域周围的边缘梯度方向，将得到的边缘梯度方向放到一个方向直方图相应的bin中，然后将该直方图与手势模板进行匹配，从而识别出各种手势

通过标记帧与帧之间显著的边缘和颜色的统计变化，直方图被用来检测视频中场景的变换。边缘、色彩、角等直方图构成了可以被传递给目标识别分类器的一个通用特征类型。

直方图的正确性依赖于网格大小：如果网格太宽，则直方图统计中有太多的空间平均，如果网格太窄，则因太小的平均产生尖锐和单个效果

求直方图相似度？（直方图匹配）

在对比直方图之前，都应该自行进行归一化操作。

相关，卡方，直方图相交以及Bhattacharyya距离

当光线变化能引起图像颜色值的漂移，尽管这些漂移没有改变颜色直方图的形状，但是这些漂移引起了颜色值位置的变化，从而导致前述匹配策略失效。可以用直方图的距离测量来代替直方图的匹配策略：EMD（Earth Mover’s Distance）度量的是怎样将一个直方图的形状转变为另一个直方图的形状

基于直方图的反射投影？

对于归一化直方图模型来说，结果图像可以被解释为一个概率图，它表示目标是否可能出现

cvCalcBackProject：为一个特定像素是否可能是一个特定目标类型的成员（以一个直方图作为该目标类型的模型建模） --计算一个像素是否是一个已知目标的一部分

cvCalcBackProjectPatch：考虑图像子区域以及子区域的特征（比如颜色）直方图，并且想知道子区域特征的直方图是否与模型的直方图匹配 --计算一块区域是否包含已知的目标

一量我们得到了目标图像的概率值，又应怎样利用该图像来找到目标呢？

cvMinMaxLoc()寻找最大值的位置（第一步先平滑）最有可能是目标在图像中的位置

模板匹配

通过在输入图像上滑动图像块对实际的图像块和输入图像进行匹配

方法：平方差匹配法、相关匹配法

一个好的匹配位置附近应该有许多好的匹配位置，因为模板的轻度变化不应该有不同的匹配位置。在寻找最小（对于平方差度量来说）或最大值（对于互相关或互相关系数来说）之前要对结果图像进行平滑操作

图像分割

金字塔分割：cvPyrSegmentation

分水岭算法: cvWatershed

均值漂移分割: cvPyrMeanShiftFiltering

图像修复

利用已被破坏区域的边缘的颜色和结构，繁殖和混合到损坏的图像里面。如果被破坏区域并不太大，并且在被破坏区域边缘包含足够多的纹理和颜色，那么Inpainting可以很好地恢复图像。

cvInpaint

轮廓

OpenCV允许得到的轮廓被聚合成一个轮廓树，从而把包含关系编码到树结构中

轮廓参数：

轮廓类型（检测最外的轮廓，检测所有的轮廓并保存成list, hierarchy, or tree）

轮廓如何被近似

轮廓表示有两种：

由一系列顶点的序列表示

Freeman链码表示（多边形被表示为一系列的位移，每一个位移用8个方向，Freeman链码对于识别一些形状的物体很有帮助）

轮廓的多边形逼近cvApproxPoly，长度cvContourPerimeter，面积cvContourArea，矩形框cvBoundingRect，最小外接矩形cvMinAreaRect2, 拟合椭圆cvFitEllipse2, 凸包cvConvexHull2, 凸缺陷cvConvexityDefects

求轮廓的相似度？（轮廓匹配）

比较轮廓矩 （通过对轮廓上所有点进行积分运算而得到的特征）

简单的矩依赖于所选坐标系，这意味着物体旋转后就无法正确匹配

Hu不变矩: 归一化中心距的线性组合。之所以这样做是为了能够获取代表图像某个特征的矩函数，这些矩函数对某些变化如缩放、旋转和镜像映射具有不变性

使用Hu矩进行轮廓匹配: cvMatchShapes

Freeman链码比对物体？

Freeman链码编码：是对一个多边形序列如何“移动”的描述，每个这样的移动有固定的长度和特定的方向。

链码编码直方图chain code histogram：用来统计一个轮廓的Freeman链码编码每一种走法的数字，具有良好的性质，如将物体旋转45度，那么新的直方图是老直方图的循环平移。这就提供了一个不被此类旋转影响的形状识别

特征检测

角点

如果墙上的所有点都是一样的或者是相似的，我们就不会有太好的运气能在随后的视频帧中跟踪到这个点了。相反，如果选择一个独一无二的点，那么再找到这点的几率就非常大

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。

Harris角点

SIFT: scale-invariant feature transform

在一点处检测主要梯度方向，根据这个方向记录局部梯度直方图结果，所以SIFT也是旋转不变的。SIFT特征在小的仿射变换中有相对不错的表现。

光流：

稠密光流dense optical flow：可以将图像中的每个像素与速度关联，或者等价地，与表示像素在连续两帧之间的位移关系。

稀疏光流sparse optical flow: 稀疏光流的计算需要在被跟踪之前指定一组点。如果这些点具有明显的特征，如角点，斑点，那么跟踪就会相对稳定和可靠

Lucas-Kanade算法： 只需要每个感兴趣点周围小窗口的局部信息。三个基本假设：亮度恒定（假设像素被逐帧跟踪时其亮度不发生变化）；时间连续或者运动是“小运动”（运动相对于帧率是缓慢的）；相邻的点保持相邻（一个场景中同一表面上邻近的点具有相似的运动

对于大多数30Hz的摄像机，大而不连贯的运动是普遍存在的，所以Lucas-Kanade光流正因为这个原因在实际中的跟踪效果交不是很好，我们需要一个大的窗口来捕获大的运动，而大窗口往往会违背运动连贯的假设！图像金字塔可以解决这个问题，即最初在较大的空间尺度上进行跟踪，再通过对图像金字塔向下直至图像像素的处理来修正初始运动速度的假定

建议的跟踪方法：在图像金字塔的最高层计算光流，用得到的运动估计结果作为下一层金字塔的起始点，重复这个过程直至到达金字塔的最底层

视频分割

从图像中分割出前景目标

“背景”在不同的应用场合下是一个很难定义的问题。例如，若正在观测一条高速公路，那么或许平均流动的车流应该被认为是背景。通常情况下，背景被认为是在任何所感兴趣的时期内，场景中保持静止或周期运行的目标。

高级的场景建模：要对前景状态和背景状态定义多重指标，以时间为基础将不变的前景模型缓慢转换为背景模块。当场景完全发生变化时我们还必须检测并建立一个新的模型。

通常，一个场景模型可能包含许多层次，从“新的前景”到旧的背景再到背景。一个新的前景目标就会放进“新前景”目标级别，标识一个真目标或一个空洞。在没有任何前景物体的地方，我们将继续更新我们的背景模型。如果一个前景物体在给定的时间内没有发生移动，就将它降级为“旧的前景”

方法1：帧差

用前一帧减去当前帧，然后 将足够大的差别标为前景（作阈值化处理），忽略像素值受到噪声和波动的影响（作形态学处理），连通域法清理离散噪声（因为噪声不会有很大的空间相关性，这些信号有大量的非常小的区域来描述）

方法 2: 计算每个像素的平均值和标准差作为它的背景模型

以上简单的方法只能用于背景场景中不包含运行的部分（比如摆动的窗帘和在风中摇曳的树）。而且这种方法还要求光线保持不变（如在室内静止的场景）。

更高级的背景建模：

很多背景场景都包含复杂的运动目标，诸如摇曳在风中的树，转动的风扇，摆的窗帘等，通常这样的场景中还包含光线的变化。比如云彩掠过，门窗中照进来不同的光线。

解决这种问题的转好方法是得到每个像素或一组像素的时间序列模型。

方法1: 均值漂移

acc(x, y) = (1-alpha)\*acc(x, y) + alpha\*image(x, y) if mask(x, y) != 0

可以作为跟踪器，给予最近值较大的权值，解释：

(2, 3, 4)，均值为3，而均值漂移值为3.25

方法2: codebook方法

Codebook不能很好处理不同模式的光（如早晨、中午和傍晚的阳光，或在室内有人打开和熄灭灯）

目标跟踪

* mean-shift: 在一组数据的密度分布中寻找局部极值的稳定的方法

步骤：

1. 选择搜索窗口
2. 计算窗口（可能带权重）的重心
3. 将窗口的中心设置在计算出的重心处
4. 重复2~3,直到窗口的位置不再变化

收敛的位置在窗口中像素分布的局部最大值（峰值）处

cvMeanShift: 需要首先选择代表物体的特征的分布（例如，颜色+纹理），然后在物体的特征分布上开始mean-shift窗口搜索，最后计算下一帧视频中所选择的特征的分布。从当前的窗口位置开始，mean-shift算法寻找特征分布的新的峰值，它们被假定设置在最初始产生颜色和纹理的物体的中心。这样mean-shift窗口就可以逐帧跟踪物体的运动

* camShift: continuously adaptive meanshift 连续自适应均值漂移

搜索窗口会自我调整尺寸，此算法可以根据物体靠近或远离摄像机时的尺寸而自动调整窗口的尺寸

* 运动模板： motion template

应用在姿态识别中，需要知道物体的轮廓或者轮廓的一部分

* Kalman滤波器

预测+校正

Kalman滤波器依赖于线性动态性和Markov独立性（也就是它假设当前状态只依赖于刚过去的状态而不是所有过去的状态）

Delaunay三角剖分和Voronoi划分

将空间点连接为三角形，使得所有三角形中最小的角最大的一个技术。也就是说Delaunay三角剖分力图避免出现瘦长三角形，同时任何三角形的外接圆都不包含任何其他顶点，这叫外接圆性质。

应用：

通过Delaunay三角剖分，我们可以直接找到一组点的外部轮廓

运用原始点的Delaunay三角测量，你可以快速搜索新点的最近邻居

Voronoi划分是Delaunay三角剖分的对偶图像，任何包含在Voronoi单元中的点都比其他Delaunay点更接近于它们自己的Delaunay点

摄像机模型与标定

摄像机标定camera calibration,来矫正（数字方式）因使用透镜而给针孔模型带来的的主要偏差

摄像机标定的过程既给出摄像机几何模型，也给出透镜的畸变模型。

投影变换：空间坐标点投影到摄像机上

“筒形”或“鱼眼”畸变：实际摄像机的透镜总是在成像仪的边缘产生显著的畸变。对某些透镜，光线在远离透镜中心的地方比靠近中心的地方更加弯曲。对常用的普通透镜来说，这种现象更加严重。鱼眼畸变在便宜的网络摄像机中非常厉害，但在高端摄像机中不明显，因为这些透镜系统做了很多消除鱼眼畸变的工作

标定方法：把摄像机对准一个有很多独立可标识点的物体。通过在不同角度观看这个物体。可以利用通过每个图像计算摄像机的相对位置和方向以及摄像机的内参数

OpenCV常用标定物是一个用不同黑白方块构成的棋盘

手持棋盘以各种方向得到的棋盘图像，以确保为完全求解这些图像在整个坐标系（相对于摄像机）的位置和摄像机内参数提供足够的信息，从而可以计算出每个视场的旋转和平移，同时也计算摄像机的内参数（对所有视场不变）

求解未知量：摄像机4个内参数和5个畸变参数（对所有视场不变），还有6个外参数含三个旋转参数和三个平移参数（依赖于每个视场）？

为了得到高质量结果，至少需要10幅7\*8或者更大棋盘的图像（而且只在移动棋盘在不同图像中足够大以从视场图像中得到更加丰富的信息）

cvCalibrateCamera2

标定摄像机通常是想做两件事：一个是矫正畸变效应，另一个是根据获得的图像重构三维场景

cvUndistort2: 基于摄像机标定时获得的内参数矩阵和畸变系数

# simpleITK

<https://itk.org/Wiki/SimpleITK/GettingStarted>$ sudo pip install 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>

source → filter → mapper → actor → render → window → interactor

Keypress j/t toggle between joystick (position sensitive) and trackball (motion sensitive)

shift + mouse pan

middle mouse zoom

Keypress e/q exit or quit the application

Keypress r reset the camera view

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