多媒體技術與應用
Spring 2021

Instructor：Yen-Lin Chen(陳彥霖), Ph.D.
Professor
Dept. Computer Science and Information Engineering
National Taipei University of Technology

Lecture 6

機器學習概論與SIFT特徵描述器

什麼是機器學習

- 傳統機器學習(特徵分類器)之優點與限制
  - 優點：對於少量的訓練集而言，比深度學習更容易訓練出好的模型，參數量少且易於分析，因此易於調整參數與線上更新訓練模型。
  - 限制：偵測效能非常依賴事先設計特徵之好壞，選擇最佳特徵並對資料進行充分降維，才可獲得有效之分類器。

影像特徵檢測

- 在一張影像中，常見的特徵可能為平坦表面(flat)(如圖中A、B)、角(corner) (如圖中C、D) 、邊緣(edge) (如圖中E、F)等

影像特徵提取

- 影像特徵可以包括顏色特徵、紋理特徵、形狀特徵以及局部特徵點等。由於一幅影像中，對電腦來說只是許多的像素點所構成，所以我們必須根據這些數據提取出影像中的關鍵資訊，例如梯度變化或輪廓等資訊，接著可將這些資訊作為機器學習分類器(如SVM)對影像進行分類的依據。

Harris Corner Detection

Harris Corner Detection-簡介

- 先前投影片中，我們看到角是影像中各個方向上強度變化很大的區域。Chris Harris和Mike Stephens在1988年的論文《A Combined Corner and Edge Detector》中嘗試找到這些影像中的角(corner)，所以現在將其稱為Harris Corner Detection。他們把這個簡單的想法變成了數學形式。將sliding window向各個方向移動(u, v)然後計算所有差異的總和。表示如下：

$$E(u,v) = \sum_{x,y} \underbrace{w(x,y)}_{\text{window function}} \underbrace{[I(x+u,y+v)}_{\text{shifted intensity}} - \underbrace{I(x,y)}_{\text{intensity}}]^2$$

- 其中$w(x,y)$表示sliding window，$I(x,y)$表示像素點灰度值強度(範圍0~255)，

Harris Corner Detection-簡介

- 角點偵測中要使$E(u,v)$的值最大。這表示我們必須最大化第二項值v。利用泰勒展開式應用於上述方程式，然後再通過幾個步驟數學換算，我們最終可推得：

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

where $$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

- 在這裡，I_x和I_y分別代表影像在x和y方向上的導數(可使用cv2.Sobel推導而得)

Harris Corner Detection-簡介

- 接下來進入主要步驟，在此建立一個公式來計算分數值，此分數將決定一個區域中是否有包含一個角點(corner)。

$$R = det(M) - k(trace(M))^2$$

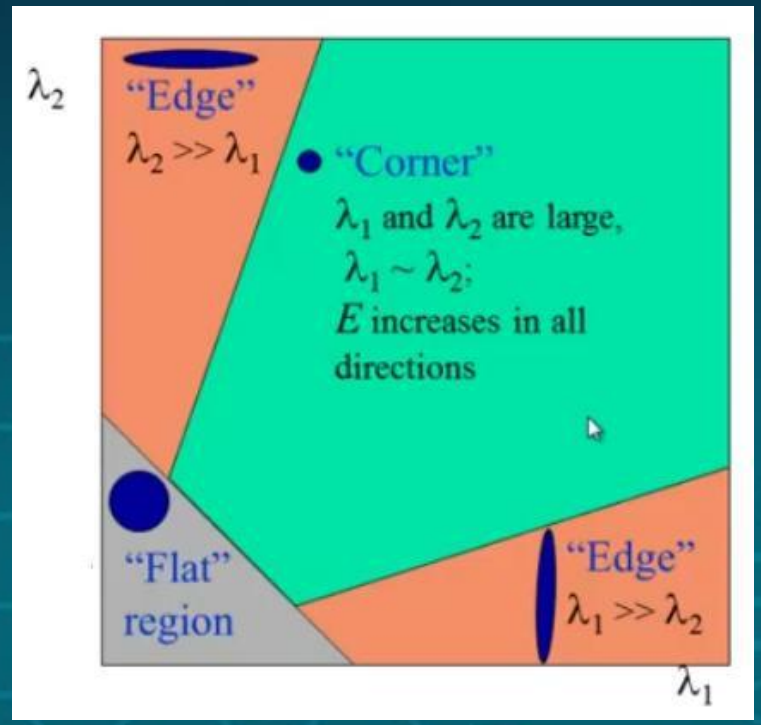
where

- $det(M) = \lambda_1 \lambda_2$
- $trace(M) = \lambda_1 + \lambda_2$
- λ_1 and λ_2 are the eigen values of M

- λ_1、λ_2是矩陣M的特徵值(eigen value)，λ_1、λ_2大小決定了區域中是平坦區域(flat)、角點(corner)或者是邊緣(edge)。

Harris Corner Detection-簡介

- 當λ_1、λ_2 都很小時， |R|也很小時，區域為平坦區域(flat)。
- 當$\lambda_1 >> \lambda_2$ 或$\lambda_1 << \lambda_2$時，R<0，區域為邊緣(edge)。
- 當λ_1與λ_2都很大，且 $\lambda_1 \sim \lambda_2$，R也很大，區域為角點(corner)。



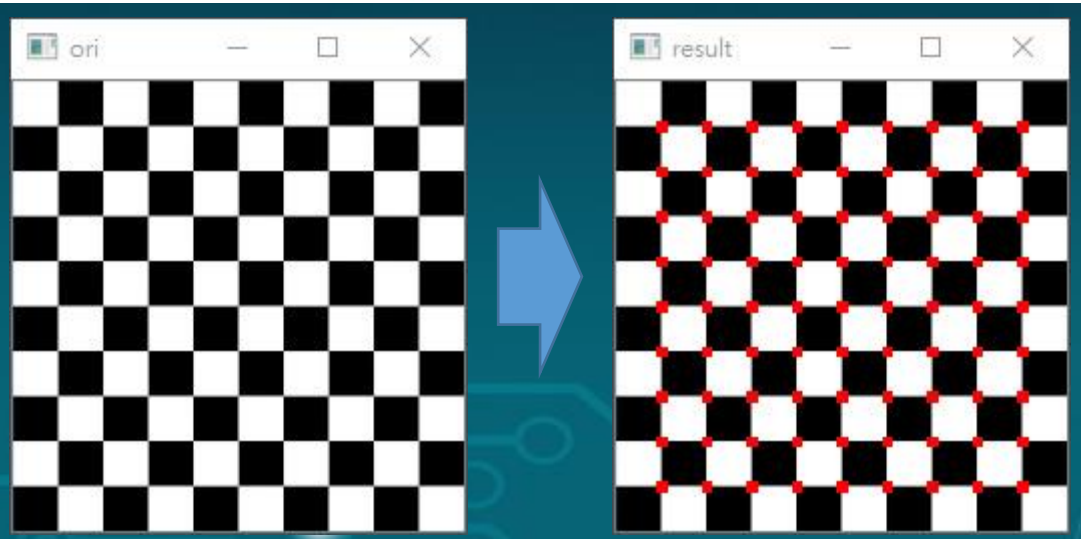
Harris Corner Detection-OpenCV

- cv2.cornerHarris(src=gray, blockSize, ksize, k, dst=None, borderType=None)
  - src:指定輸入影像(須為單通道8-bit或numpy.float型態的影像)
  - blockSize : 角點檢測中要考慮的區域大小。也就是計算矩陣時的sliding window大小
  - ksize : Sobel求梯度中使用的sliding window大小
  - k:Harris 角點檢測方程中的自由參數，取值參數為 [0.04 , 0.06]
  - dst:輸出影像
  - boarderType:邊界的類型
  - 例：dst = cv2.cornerHarris(gray, 2, 3, 0.04)

Harris Corner Detection-OpenCV

- 範例程式：
  - [4]讀取影像
  - [6]影像轉為灰階
  - [7]影像轉為float32格式
  - [8]設定2x2的sliding window大小、3x3的Sobel sliding window大小及Harris參數0.04來取出灰階影像中的角點
  - [10]對角點進行膨脹
  - [12]畫出角點
  - [13]秀圖

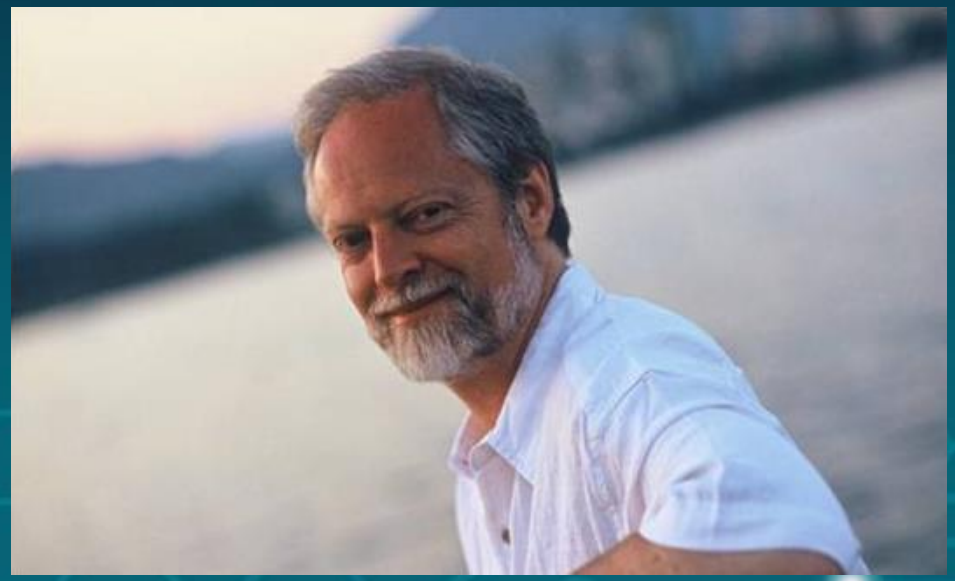
```python
import numpy as np
import cv2 as cv

img = cv.imread("chessboard.png")
cv.imshow('ori',img)
gray = cv.cvtColor(img,cv.COLOR_BGR2GRAY)
gray = np.float32(gray)
dst = cv.cornerHarris(gray,2,3,0.04)
#result is dilated for marking the corners, not important
dst = cv.dilate(dst,None)
# Threshold for an optimal value, it may vary depending on the image.
img[dst>0.01*dst.max()]=[0,0,255]
cv.imshow('result',img)
if cv.waitKey(0) & 0xff == 27:
    cv.destroyAllWindows()
```



SIFT Feature

SIFT(Scale Invariant Feature Transform)概述

- 尺度不變特徵轉換(Scale Invariant Feature Transform, SIFT)，是一種機器視覺的演算法用來偵測與描述影像中的局部性特徵。SIFT特徵**對旋轉、尺度縮放、亮度變化等保持不變性**，是一種非常穩定的局部特徵。此演算法由 David Lowe 在1999年所發表，2004年完善總結。



大不列顛哥倫比亞大學的David G.Lowe教授

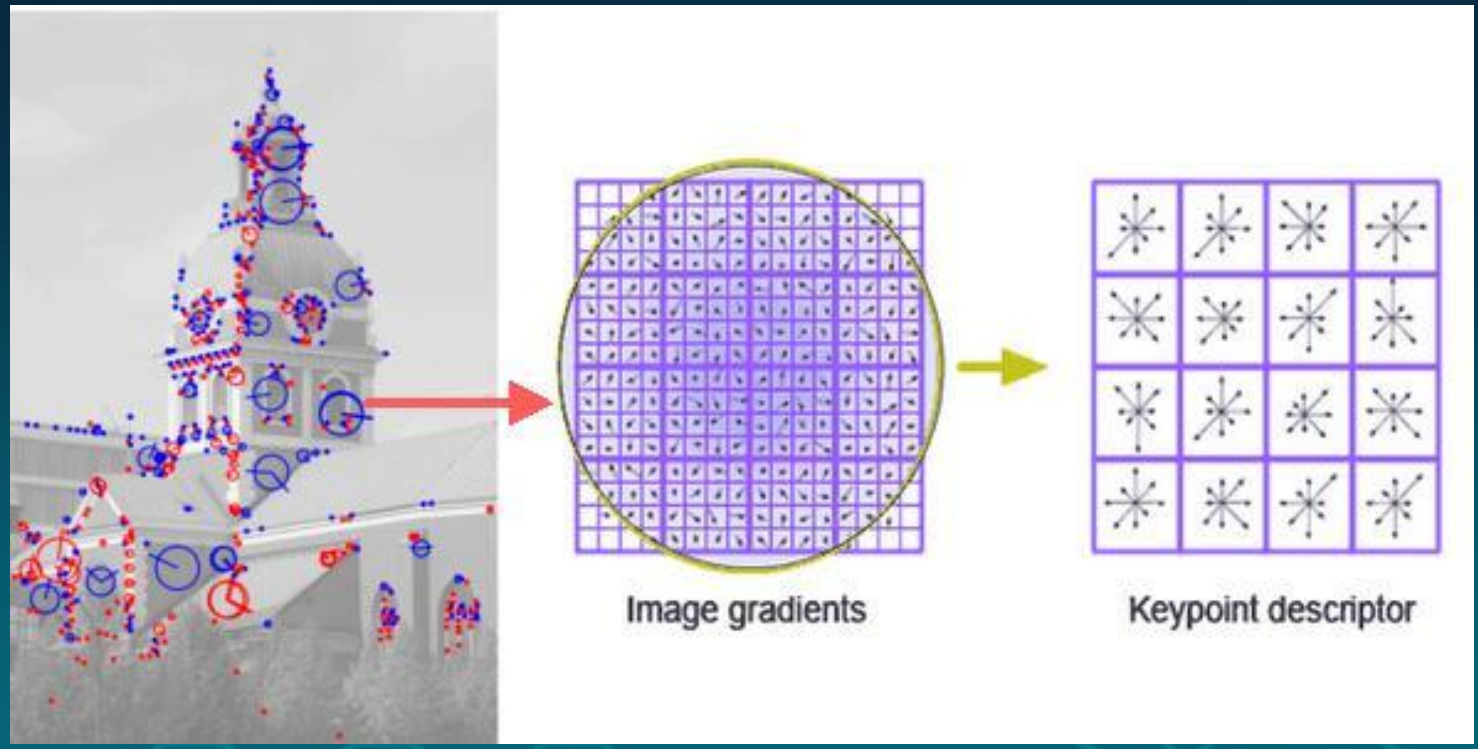
SIFT-特點

1. 影像的局部特徵，對旋轉、尺度縮放、亮度變化保持不變，對視角變化、仿射變換、噪聲也保持一定程度的穩定性
2. 獨特性好，訊息量豐富，適合於大量特徵進行快速、準確的匹配。
3. 即使是很少幾個物體也可以產生大量的SIFT特徵
4. 經最佳化的SIFT匹配演算法甚至可以達到即時性
5. 可以很方便的與其他的特徵向量進行聯合。

SIFT-特徵檢測的步驟

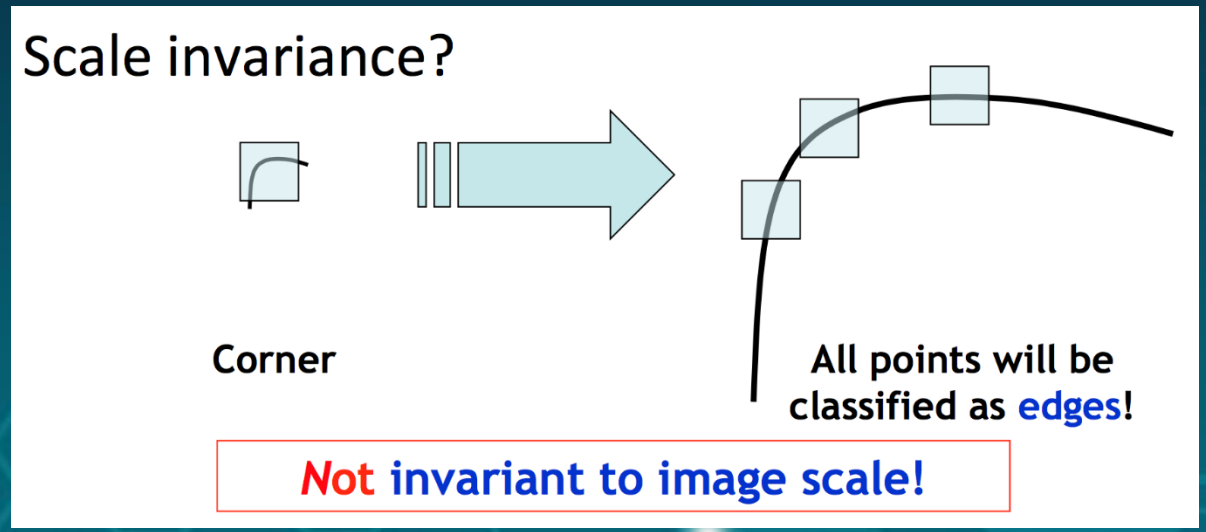
1. 尺度空間的極值檢測搜索所有尺度空間上的影像，通過高斯微分函數來偵測潛在的對尺度和選擇不變的興趣點。
2. 刪除不穩定的極值點。主要刪除兩類：低對比度的極值點以及不穩定的邊緣回應點。
3. 特徵方向賦值基於影像局部的梯度方向，分配給每個特徵點位置一個或多個方向，後續的所有操作都是對於特徵點的方向、尺度和位置進行變換，從而提供這些特徵的不變性。
4. 特徵點描述在每個特徵點周圍的鄰域內，在選定的尺度上測量影像的局部梯度，這些梯度被變換成一種descriptor，這種表示descriptor比較大的局部形狀的變形和光照變換。

SIFT-特徵檢測呈現的效果



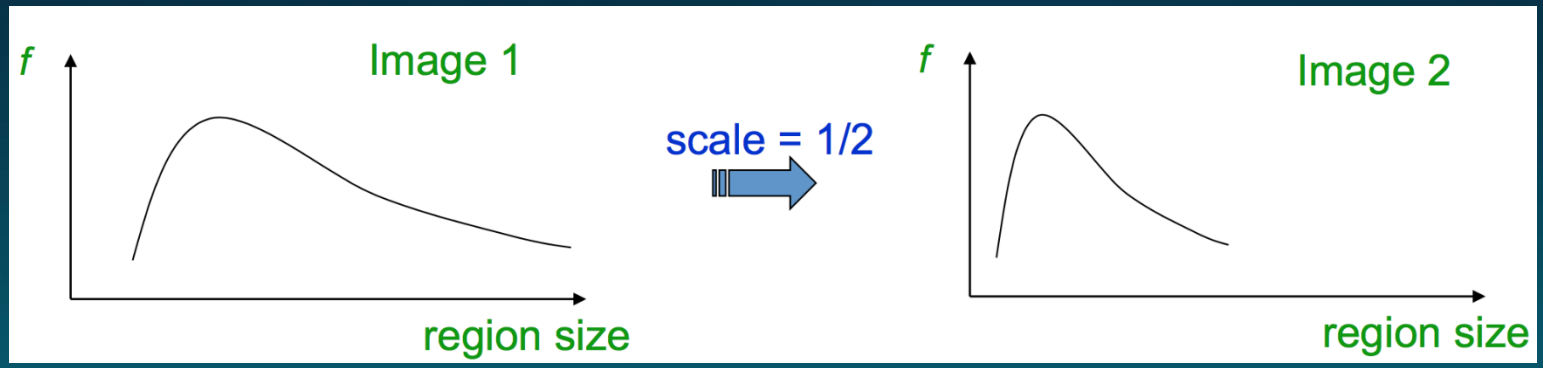
SIFT-簡介

- Harris角點方法計算簡便，並具有平移不變性和旋轉不變性。特徵f對某種變換T具有不變性，是指在經過變換後原特徵保持不變，也就是$f(I) = f(T(I))$。但是Harris角點不具有尺度變換不變性，如下圖所示。當影像被放大後，原圖的角點被判定為了邊緣點。



SIFT-簡介

- 而我們想要得到一種對尺度變換保持不變性的特徵計算方法。例如影像的像素平均亮度，如下圖所示。region size縮小為原始的一半後，亮度長條圖的形狀不變，即平均亮度不變。



- 而Lowe想到了使用局部極值來作為特徵(feature)來保證對scale變換的不變性。在演算法的具體實現中，他使用DoG來獲得局部極值。

SIFT-尺度空間極值的檢測方法

- 前人研究已經指出，在一系列合理假設下，Gaussian Kernel是唯一滿足尺度不變性的。所謂的尺度空間，就是指原始影像I(x,y)和可變尺度的Gaussian Kernel G(x,y,σ)的卷積結果。如下式所示：

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

- 其中， $G(x, y, \sigma) = \frac{1}{2\pi\sigma^2}\exp(-(x^2 + y^2)/2\sigma^2)$ 不同的σ代表不同的尺度。

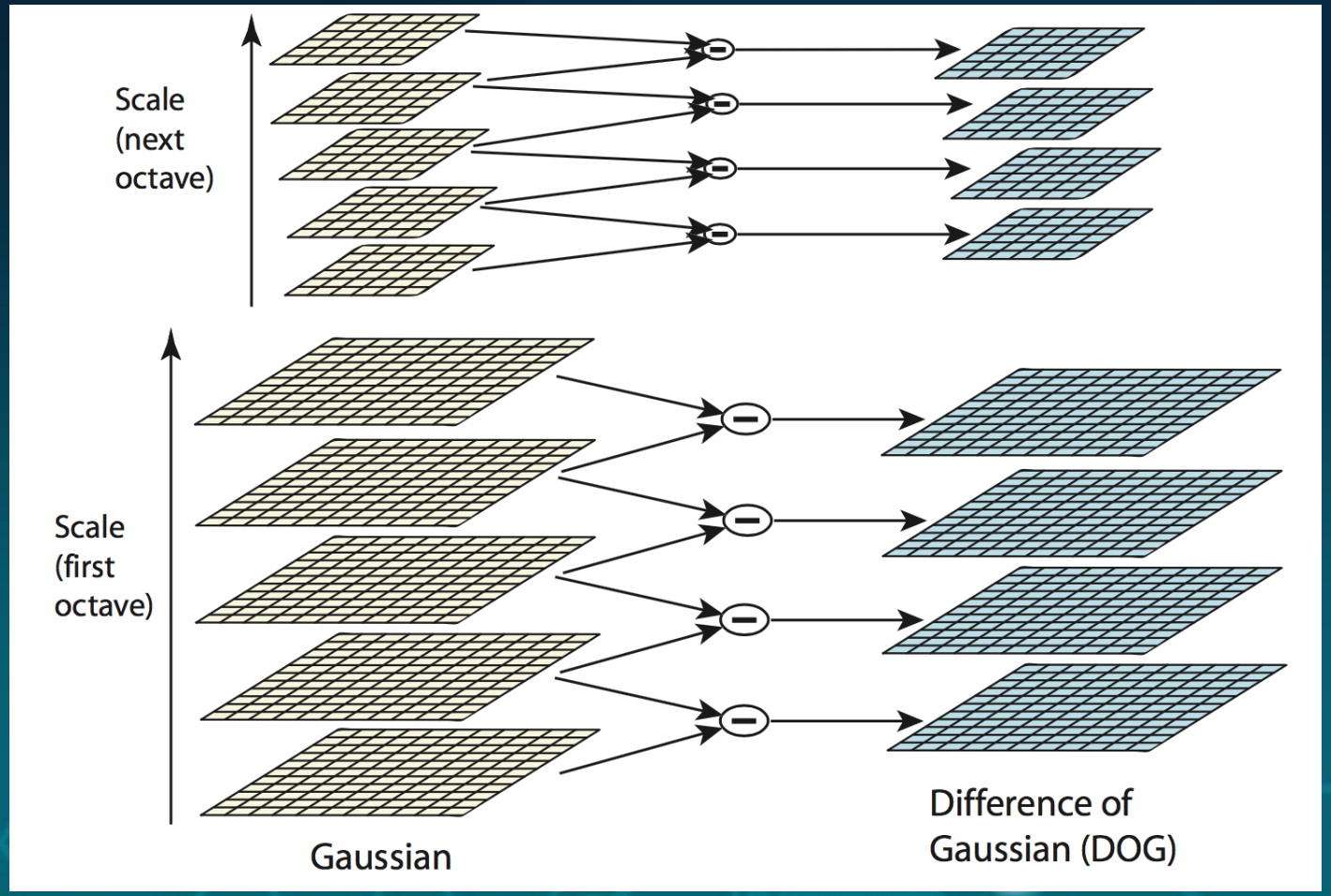
- DoG(difference of Gaussian)函式定義為不同尺度的Gaussian Kernel與影像卷積結果之差，即：

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

SIFT-尺度空間極值的檢測方法

- 如下頁圖所示，輸入的影像重複地與Gaussian Kernel進行卷積得到結果影像L（左側），這樣構成了一個octave。相鄰的L之間作差得到DoG影像（右側）。當一個octave處理完之後，將當前octave的最後一張高斯卷積結果影像降採樣兩倍，按照前述方法構成下一個octave。在圖中，我們將一個octave劃分為s個間隔（即s+1個影像）時，設σ最終變成了2倍（即σ加倍）。那麼，顯然有相鄰兩層之間的$k = 2^{\frac{1}{s}}$。不過為了保證首尾兩張影像也能夠計算差值，我們實際上需要再補上兩張（做一個padding），也就是說一個octave內的總影像為s+3。

SIFT-尺度空間極值的檢測方法



SIFT-尺度空間極值的檢測方法

- 為何我們要費力氣得到DoG呢？論文中作者給出的解釋是：DoG對scale-normalized後的Gaussian Laplace函數$\sigma^2\Delta G$提供了足夠的相似程度。其中前面的σ^2係數正是為了保證尺度不變性。而前人的研究則指出，$\sigma\Delta G$函數的極值，和其他一大票其他function比較時，能夠提供最穩定的feature。
- 對於Gaussian kernel，有以下性質：

$$\frac{\partial G}{\partial \sigma} = \sigma \Delta G$$

- 將左式的微分變成差分，得：

$$\sigma\Delta G \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}$$

- 也就是：

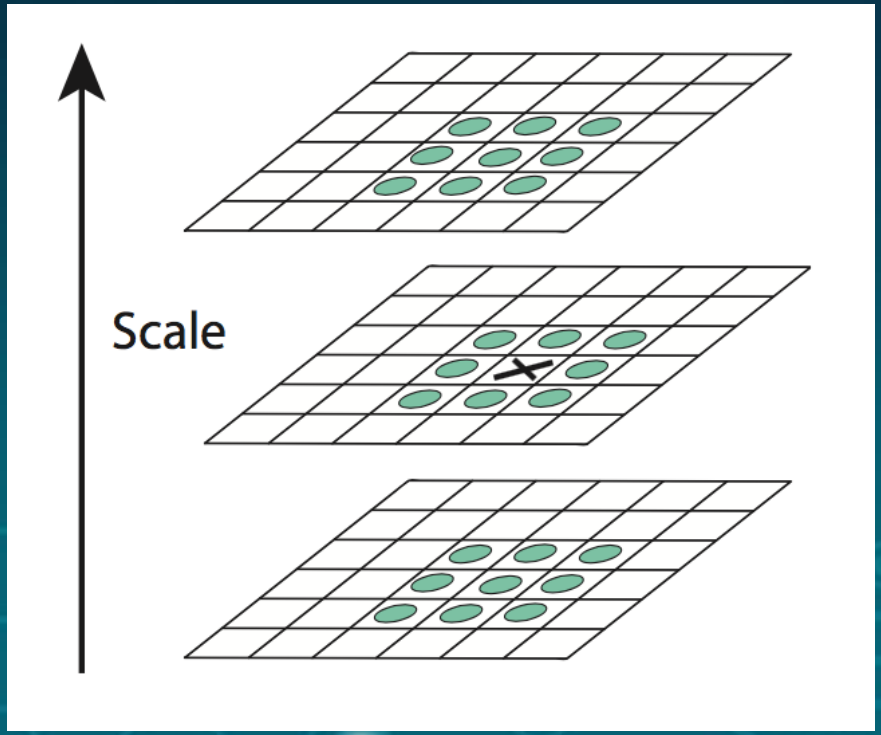
$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \Delta G$$

SIFT-尺度空間極值的檢測方法

- 當k=1時，上頁式子的近似誤差為0（即上面的s=∞，也就是說octave的劃分是連續的）。但是當k較大時，如$\sqrt{2}$（這時s=2），仍然保證了穩定性。同時，由於相鄰兩層的比例k是定值，所以k−1這個因數對於檢測極值沒有影響。我們在計算時，無需除以k−1。

SIFT-尺度空間極值的檢測方法

- 建構完了DoG影像金字塔，下面我們的任務就是檢測其中的極值。如下圖，對於每個點，我們將它與金字塔中相鄰的26個點作比較，找到局部極值。



SIFT-尺度空間極值的檢測方法

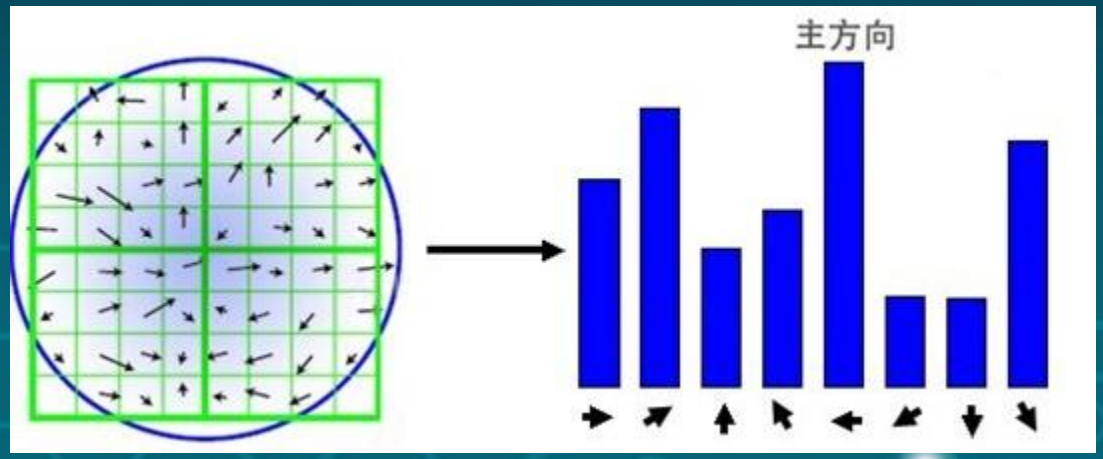
- 另外，我們將輸入影像行列預先使用線性插值的方法resize為原來的2倍，獲取更多的key point。
- 此外，在Lowe的論文中還提到了使用DoG的泰勒展開和Hessian矩陣進行key point的精細確定方法，並對key point進行過濾。

SIFT-定位特徵點

- 一旦找到潛在的關鍵點位置，就必須對其提取出更精準的結果。作者使用了泰勒展開來獲得更精確的極值位置，並且如果該極值的強度小於門檻值（論文中為0.03，在OpenCV中，此門檻值稱為ContrastThreshold），則拋棄該關鍵點。DOG它對邊緣的影響較高，因此也需要刪除邊緣。 為此作者使用2x2的Hessian矩陣（H）計算主曲率。
- 從Harris Corner Detection可以知道，對於邊緣，一個特徵值大於另一個特徵值。 因此，這裡他們使用了一個簡單的函數，如果該比率大於一個門檻值（在OpenCV中稱為edgeThreshold），則拋棄該關鍵點。經由以上方法消除了許多低對比度的關鍵點和邊緣關鍵點，剩下的就是所要的就是感興趣點了。

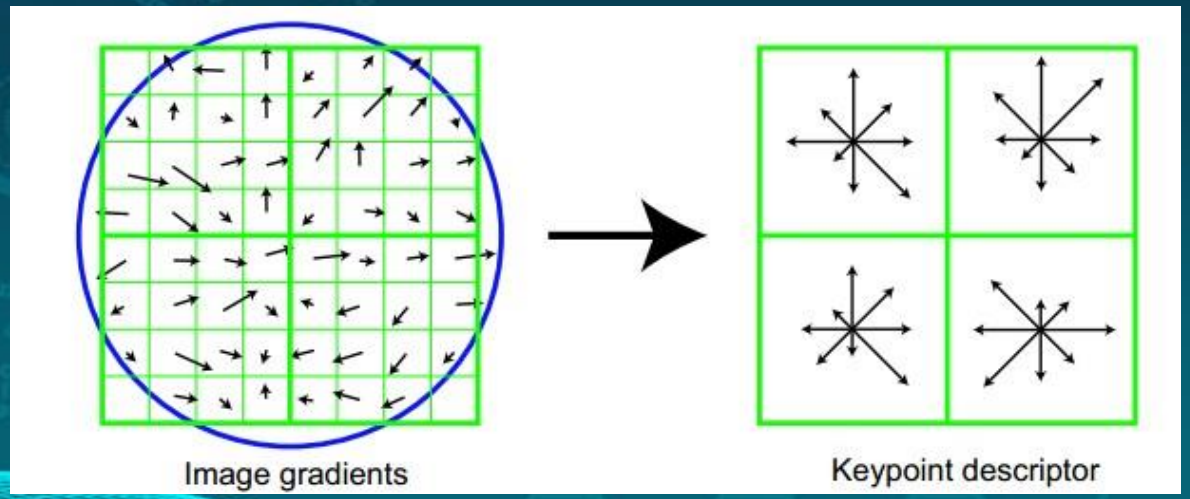
SIFT-指定特徵點方向

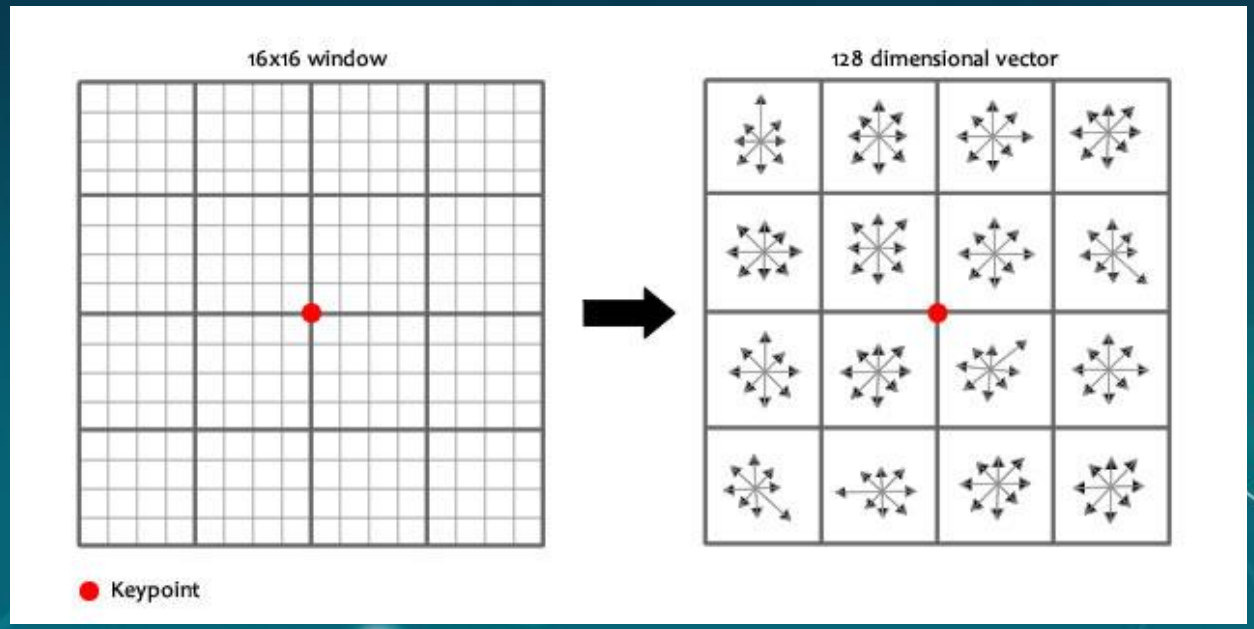
- 現在，將方向分配給每個關鍵點，以實現影像旋轉的不變性。 根據比例在關鍵點位置附近採取鄰域，並在該區域中計算梯度大小和方向。 創建了一個具有36個覆蓋360度的bin方向的直方圖（通過梯度的幅度和Gaussian-weighted circular window進行加權）。 提取直方圖中的最高峰，並且將其超過80%的任何峰也視為計算方向。所以創建了位置和比例相同但方向不同的關鍵點。如此一來有助於匹配的穩定性。



SIFT-生成特徵點描述子(Keypoint Descriptor)

- 現在創建了關鍵點描述符。 在關鍵點周圍採用了16x16的鄰域。 它分為16個4x4大小的子塊。 對於每個子塊，創建8 bin方向直方圖。 因此共有128個bin值可用。 這些值被表示為形成關鍵點描述子的vector。 除此之外，還採取了幾種措施來實現對照明變化，旋轉等的穩定性。

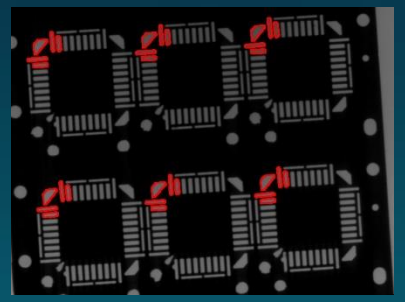




SIFT應用與SURF(加速穩健特徵)簡介

- SIFT及SURF(Speeded-Up Robust Features, 加速穩健特徵)進行局部特徵描述，用於辨識物體是否相同，在智慧製造中常常用來比對成品是否符合標準，會有一個Golden Sample(標準樣品)與成品進行比對。如下圖

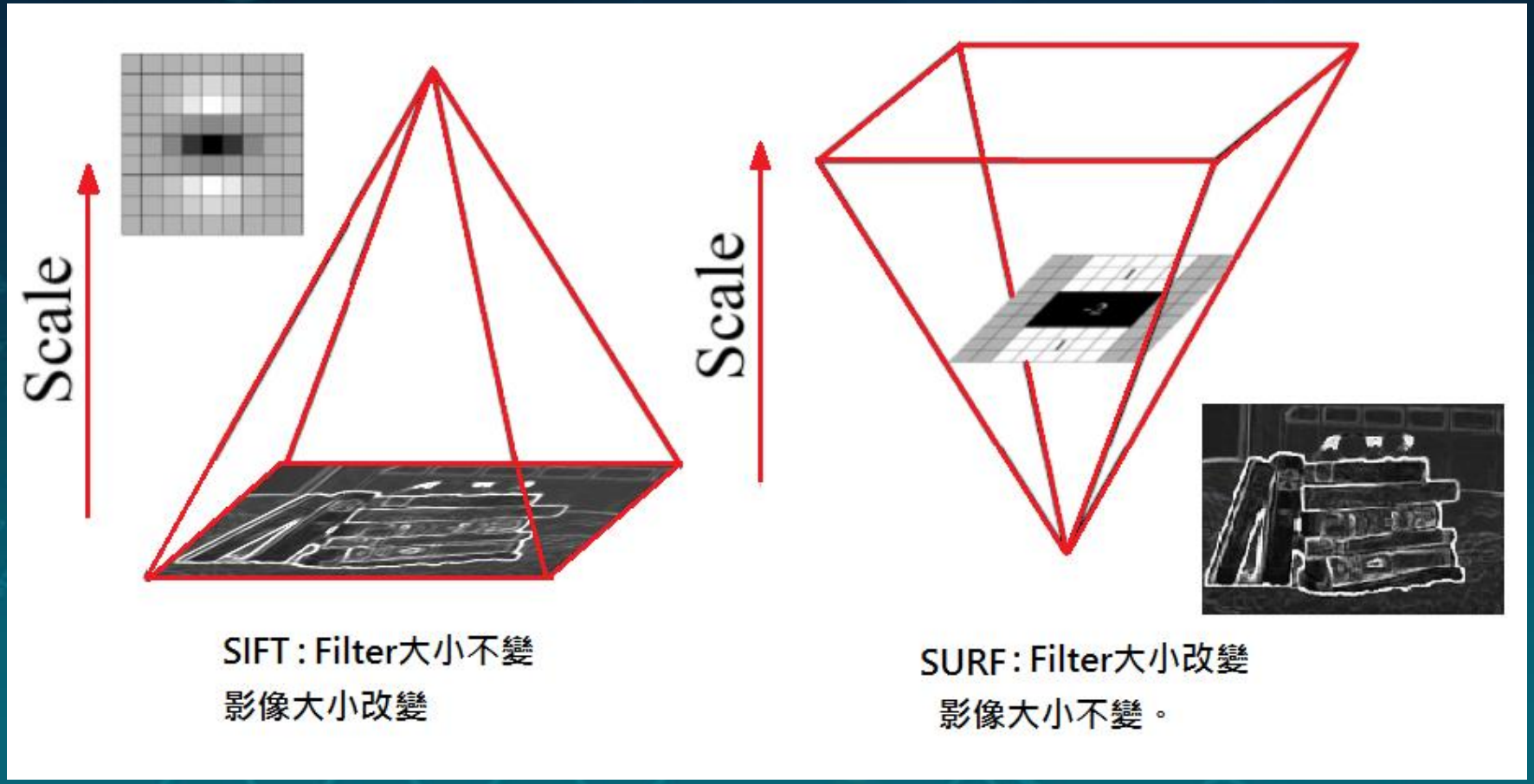
標準樣品



比對結果

- 具有旋轉，尺度，平移，視角及亮度不變性，也就是同一個物體在不同的角度，形變，位置及光影變化時，它的特徵應該是相同的。
- SURF的概念及步驟建立在SIFT上，速度比SIFT快。

SIFT與SURF 比較



SIFT與SURF 比較

	SIFT	SURF
Scale space	Filter與不同大小影像卷積	不同大小的Filter與原影像卷積
Feature Point detection	先進行非極值抑制， 再由Hessian矩陣去除邊緣點。	先由Hessian矩陣確認候選點， 然後再進行非極值抑制。
方向	在正方形區域內進行梯度直方統計，找到Max對應的方向。可以有多個方向。	在圓形區域內，計算各個扇形範圍內x、y方向的haar小波轉換，找出最大的扇形方向。
執行時間	較慢	較快
光影變化影響	較大	較小
Scale and Rotation	較好	較差

SIFT-OpenCV

- sift = cv2.SIFT_create()
  - 初始化SIFT特徵點檢測器
- kp, des = sift.detectAndCompute(<img>, None)
  - img:輸入影像
  - kp:即所有特徵點資訊，包含：
    - angle：角度，表示特徵點的方向，通過作者Lowe的論文可以知道，為了保證方向不變形，SIFT演算法通過對特徵點周圍鄰域進行梯度運算，求得該點方向。
    - class_id：當要對影像進行分類時，我們可以用class_id對每個特徵點進行區分。
    - octave：代表是從影像金字塔中的哪一層提取得到的資料。
    - pt：特徵點的坐標。
    - response：回應程度，代表該點強度大小，更確切的說，是該點角點的程度。
    - size：該點直徑的大小
  - des: 為descriptor特徵描述陣列。
  - 範例見下下頁

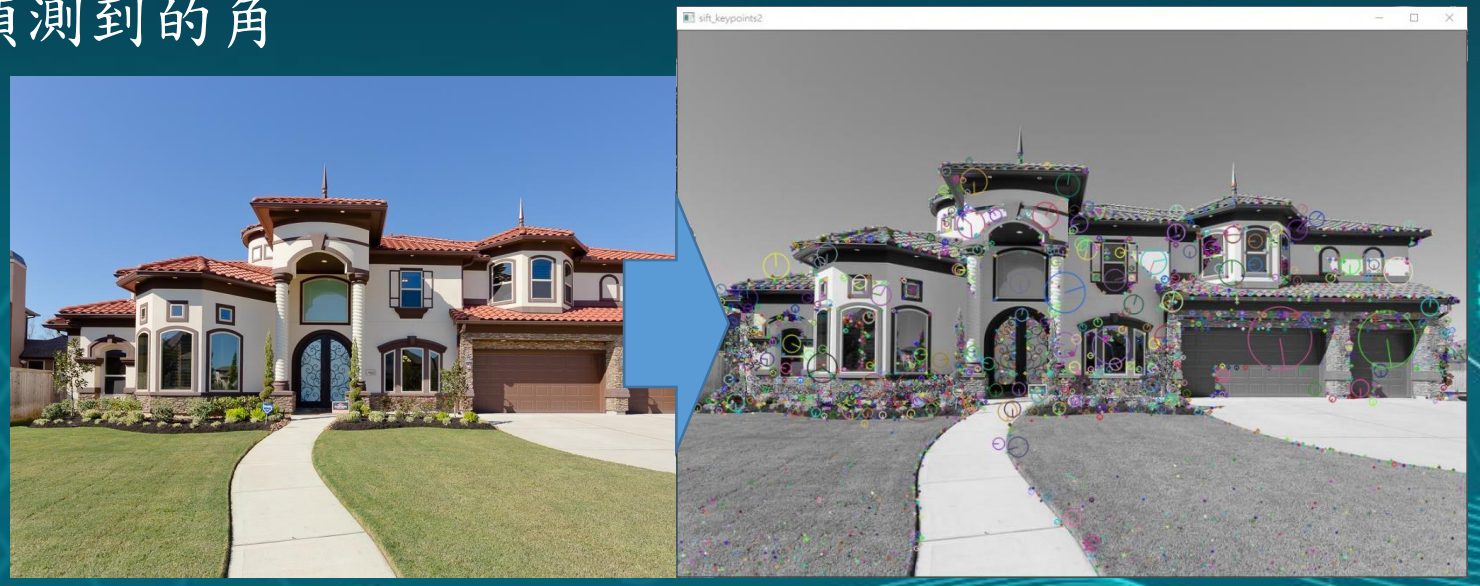
SIFT-OpenCV

- outImage = cv2.drawKeypoints(<image>, <keypoints>, <outImage>, <color>, <flags>)
  - image:輸入影像(須為灰階)
  - keypoints:特徵點向量，向量內每一個元素是一個KeyPoint物件，包含了特徵點的各種屬性資訊；
  - outImage：特徵點繪製的畫布影像，可以是原始影像
  - color：繪製的特徵點的顏色資訊，預設繪製的是隨機彩色；
  - flags：特徵點的繪製模式，其實就是設置特徵點的那些資訊需要繪製，那些不需要繪製，有以下幾種模式可選
    - DEFAULT：只繪製特徵點的座標點,顯示在影像上就是一個個小圓點，每個小圓點的圓心座標都是特徵點的座標。
    - DRAW_OVER_OUTIMG：函數不創建輸出的影像，而是直接在輸出影像變數空間繪製，要求本身輸出影像變數就是一個初始化好了的，size與type都是已經初始化好的變量。
    - NOT_DRAW_SINGLE_POINTS：單點的特徵點不被繪制。
    - DRAW_RICH_KEYPOINTS：繪製特徵點的時候繪製的是一個個帶有方向的圓，這種方法同時顯示影像的坐標、大小和方向，是最能顯示特徵的一種繪製方式。
    - 範例見下頁

SIFT-OpenCV

- 範例程式：
  - [4]讀取影像
  - [5]轉為灰階影像
  - [6]初始化SIFT特徵點提取檢測器
  - [7]進行SIFT特徵點提取
  - [9, 10]在灰階影像上繪製偵測到的角點

```python
import numpy as np
import cv2 as cv

img = cv.imread('home3.jpg')
gray= cv.cvtColor(img,cv.COLOR_BGR2GRAY)
sift = cv.SIFT_create()
kp, des = sift.detectAndCompute(gray,None)

img=cv.drawKeypoints(gray,kp,img,
    flags=cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv.imshow('sift_keypoints2',img)

if cv.waitKey(0) & 0xff == 27:
    cv.destroyAllWindows()
```



SIFT-結合SVM分類器

- SIFT特徵結合SVM分類器目前已經被廣泛應用於影像識別中，以下會說明如何將影像中提取出的SIFT特徵送入SVM分類器中以訓練模型，並使用該模型辨識輸入影像。

SIFT-結合SVM分類器

- 基於OpenCV實現SIFT特徵提取與生成向量資料，然後使用scikit-learn的線性SVM分類器訓練模型，實作影像分類預測。其訓練過程大致分為以下四步驟：
  1. SIFT特徵提取與生成描述子
  2. 生成BoW詞袋
  3. SVM分類訓練與生成模型
  4. 使用模型進行預測

SIFT-結合SVM分類器

- 其中SIFT特徵提取演算法主要有以下步驟：
  1. 構建高斯金字塔(Gaussian Pyramid)影像尋找極值點
  2. 極值點像素級別定位
  3. 影像梯度與角度長條圖建立
  4. 特徵描述子建立K-Means分群

SIFT-結合SVM分類器

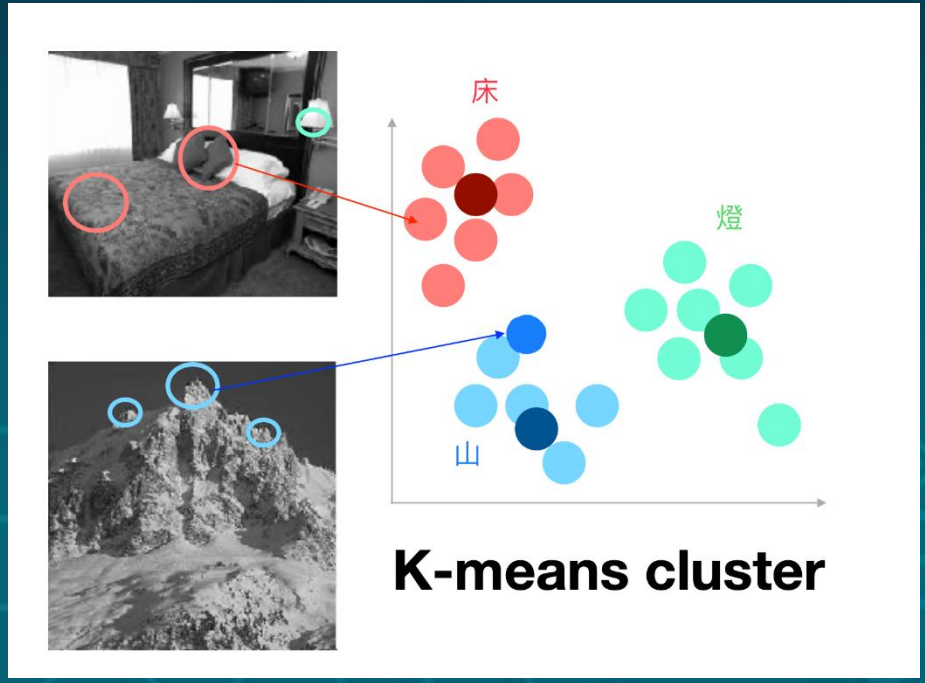
- 利用Bag of Visual Words進行影像分類：
  - Bag of Visual Words (BoVW) 模型源自於 Bag of Words (BoW, 詞袋模型)，是一種影像分類方法。
  - 在文件檢索中，BoW 模型將文件所包含的字彙 (Words) 分裝在不同的袋子中，每個文件所含的字彙種類不同則會對應到不同的袋子。舉例來說，假設有三個袋子分別包含「汽車、機車、馬路」、「醫生、感冒、生病」、「貓咪、金魚、狗狗」，那今天如果有一個文件裡面提到「汽車」，那我們就比較相信這文件屬於第一個袋子。BoW 簡單來說就是這樣做分類。

SIFT-結合SVM分類器

- 利用Bag of Visual Words進行影像分類：
  - 同理，在 BoVW 中，因為我們做的是影像分類，所以這邊的 Word 就會是影像的特徵，例如一個袋子包含一個人的「眼睛、鼻子、嘴巴」，那麼當我們在辨識一張影像的時候，看到類似的影像特徵，例如裡面包含「鼻子、嘴巴」，就可以將他歸類在是這個袋子，反之如果影像特徵是「桌腳、扶手」那麼就不會是這個袋子，以此方法我們就可以做影像分類。

SIFT-結合SVM分類器

- 利用Bag of Visual Words進行影像分類：
  - 詞袋生成最常見就是首先通過K-Means實現對描述子資料的群集分析，若分成100個群集、得到每個群集的中心資料，就生成了100 詞袋(一個K-Means群集代表一個詞袋)，根據每個描述子到這些群集中心的距離，決定了它屬於哪個群集。

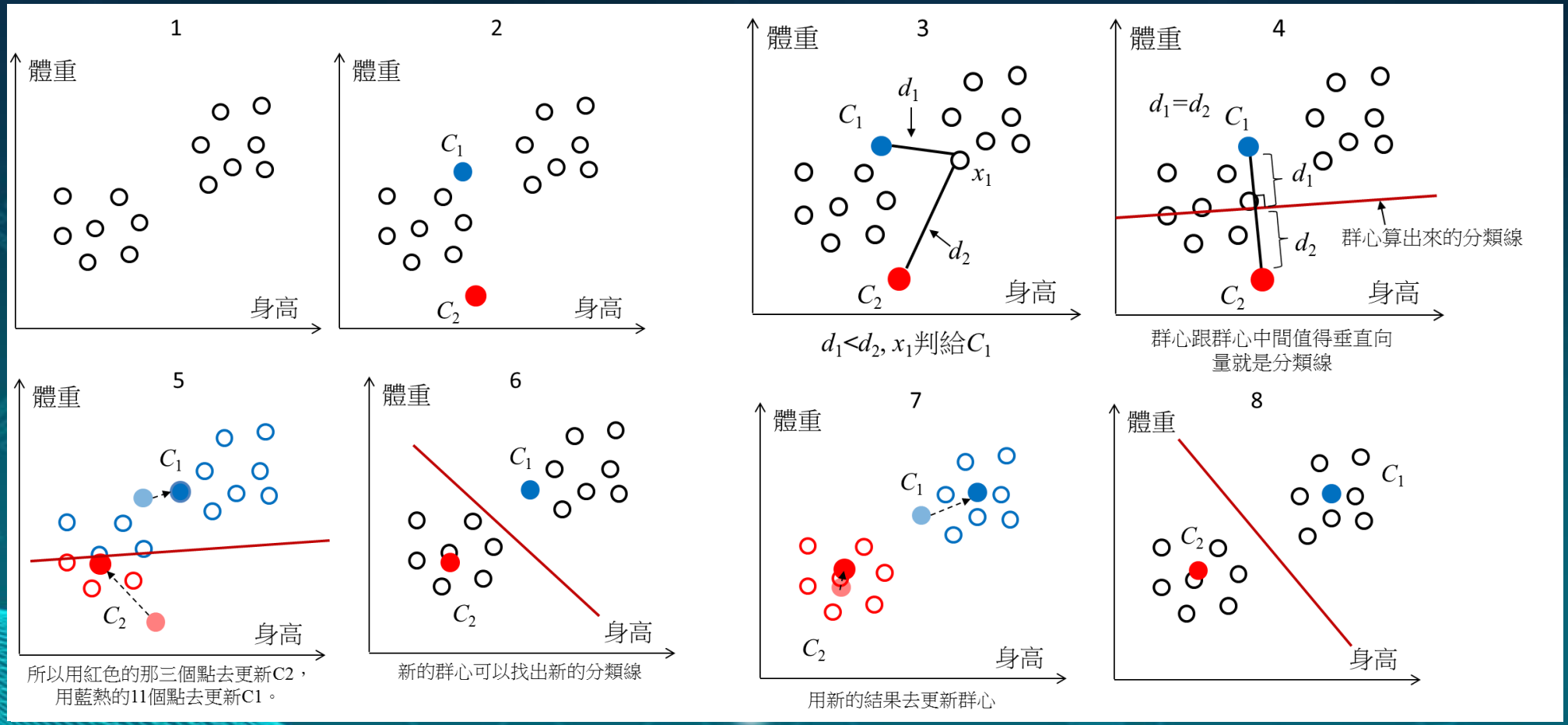


SIFT-結合SVM分類器

- K-Means分群簡介：
  - 分群(Clustering)就是對所有數據進行分組，將相似的數據歸類為一起，每一筆數據僅能屬於一個分組，每一組稱作為群集 (Cluster)
    1. 我們先決定要分k組，並隨機選k個點做群集中心。
    2. 將每一個點分類到離自己最近的群集中心(可用直線距離)。
    3. 重新計算各組的群集中心(常用平均值)。
    4. 反覆 2、3 動作，直到群集不變，群集中心不動為止。

SIFT-結合SVM分類器

- K-Means分群流程範例：



- 範例程式：
  - [1~9]引入函式庫
  - [11~13]指定dataset路徑、dataset類別
  - [17~26]將路徑和底下資料夾名稱結合、尋訪所有底下的dataset影像，並同時給予類別target
  - [31]創建SIFT特徵提取器
  - [34~42]讀入影像，統一縮放長寬至300x300，並提取出SIFT特徵。

dataset路徑結構舉例如下:

```
dataset/train/
├car/
│ ├ img0.jpg
│ ├ img1.jpg
│ └ …
└bike/
  ├ img0.jpg
  ├ img1.jpg
  └ …
```

```python
import cv2
import imutils
import numpy as np
import os   #os.path.join, os.listdir
from sklearn.svm import LinearSVC
from scipy.cluster.vq import *
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from imutils import paths
# Get the training classes names and store them in a list
train_path = "dataset/train/"
training_names = os.listdir(train_path)

# Get all the path to the images and save them in a list
# image_paths and the corresponding label in image_paths
image_paths = []
image_classes = []
class_id = 0
for training_name in training_names:
    dir = os.path.join(train_path, training_name)
    class_path = list(paths.list_images(dir))
    # class_path = imutils.imlist(dir)
    image_paths += class_path
    image_classes += [class_id] * len(class_path)
    class_id += 1
# 創建SIFT特徵提取器
sift = cv2.xfeatures2d.SIFT_create()

# 生成特徵提取描述子
des_list = []

for image_path in image_paths:
    im = cv2.imread(image_path)
    im = cv2.resize(im, (300, 300))
    kpts = sift.detect(im)
    kpts, des = sift.compute(im, kpts)
    des_list.append((image_path, des))
    print("image file path : ", image_path)
```

ref: https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.vq.kmeans.html#scipy.cluster.vq.kmeans

ref: https://cloud.tencent.com/developer/article/1165870

- 範例程式：
  - [46~47]生成特徵描述子向量
  - [51~52]將特徵描述子進行k-means分群
  - [55~59]生成特徵直方圖
  - [72~75]將特徵和其對應的target分割成8:2的比例
  - [78~79]將特徵和其對應的target送入SVM進行訓練
  - [81~87]將SVM訓練結果進行預測、並輸出以data進行predict後的結果與target進行比對計算準確率

```
45  # 描述子向量
46  descriptors = des_list[0][1]
47  for image_path, descriptor in des_list[1:]:
48      descriptors = np.vstack((descriptors, descriptor))
49
50  #  20  K-Means
51  k =  20
52  voc, variance = kmeans(descriptors, k, 1)
53
54  # 生成特徵直方圖
55  im_features = np.zeros((len(image_paths), k), "float32")
56  for i in range(len(image_paths)):
57      words, distance = vq(des_list[i][1], voc)
58      for w in words:
59          im_features[i][w] += 1

72  X = im_features
73  y = np.array(image_classes)
74  X_train, X_test, y_train, y_test = \
75      train_test_split(X, y,test_size=0.2,random_state=0)
76
77  # Train the Linear SVM
78  clf = LinearSVC()
79  clf.fit(X_train, y_train)
80
81  print("predict:")
82  print(clf.predict(X_train)) #target = y_train
83  print(clf.predict(X_test))  #target = y_test
84
85  print("Accuracy:")
86  print(clf.score(X_train,y_train))
87  print(clf.score(X_test, y_test))
68  # (將資料常態分布化，平均值會變為0, 標準差變為1，便離群值影響降低)
69  stdSlr = StandardScaler().fit(im_features)
70  im_features = stdSlr.transform(im_features)
```

Kmeans 的群集數可自行調整，一般會取辨識種類的 10 倍，亦即若有 5 種類別影像，Kmeans 會區分成 50 個群集，一個群集可以視為是一個特徵的集合。

https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.vq.kmeans.html#scipy.cluster.vq.kmeans ref: https://cloud.tencent.com/developer/article/1165870

SIFT-結合SVM分類器

- 取特徵：
  - 此處對2類別來源影像進行SIFT特徵提取，並同時給予target。
  - kp為特徵影像，des為特徵陣列。後者為我們想要的結果。
  - 對所有des進行k-means分群
  - codebook, distortion = scipy.cluster.vq.kmeans(obs, k_or_guess, iter=20)
    - codebook: 回傳一個kxN的ndarray陣列，帶有k個分群中心點的座標。
    - distortion: 回傳每個點對於各自分群中心點的平均歐基里德距離。
    - obs: 指定輸入的特徵ndarray陣列
    - k_or_guess: 指定要生成的分群中心點數量
    - iter: 指定要進行k-mean分群的次數

https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.vq.kmeans.html#scipy.cluster.vq.kmeans

ref: https://cloud.tencent.com/developer/article/1165870

SIFT-結合SVM分類器

- 訓練：
  - train_test_split(data, target, test_size=0.2, random_state=0): 將data和其對應的target分割成8:2的比例
  - clf.fit(data, target): 給予data和其對應的target送入SVM進行訓練
- 預測：
  - clf.predict(img): img預測對象可為多個影像的陣列或單一影像
  - clf.score(data, target): 輸出以data進行predict後
  - 的結果與target進行比對計算準確率

```
predict:
[0 1 1 0 0 1 0 0 0 1 1 0 1 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 1 0 0 0 0 0 1 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 0 0 1 0 1 0 0 1 1 0 0 0 0 1 0 0 1 0 1 0 1 1 0 1 0
 0 1 0 0 1 1 0 1 1 1 1 0 0 0 0 0 1 1 0 1 0 0 0 1 1 1 0 0 1 0 0 1 0 1 1 0 0
 0 0 0 0 0 1 1 0 0 0 1 0 0 1 1 0 1 1 1 1 0 0 0 0 1 0 1 0 0 0 0 1 0 1 1 1 0
 1 1 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 1 1 1 1 1 0 1 1 0 1 1 0 1 0 1 1 0
 0 0 0 1 1 0 0 1 0 1 0 1 0 0 0 1 1 0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 0 0 1 1 0
 0 1 1 0 0 1 0 1 0 0 0 0 1 1 0 0 0 0 1 1 1 0 1 1 0 0 1 1 0 1 0 0 0 1 0 1 0
 1 1 0 0 0 1 1 1 1 0 1 1 1 1 0 0 0]
[1 0 1 1 1 0 0 1 0 0 1 1 0 1 0 0 0 0 0 0 1 0 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1
 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 0 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0]
```

```
Accuracy:
1.0
0.782608695652174
```

ref: https://cloud.tencent.com/developer/article/1165870