In [1]:

```
import numpy as np
```

In [2]:

```
import cv2
```

In [3]:

```
import glob
```

In [4]:

```
import os
```

In [5]:

```
import itertools
```

In [6]:

```
from sklearn.cluster import KMeans
```

In [7]:

```
import pickle
```

In [8]:

```
import os
import sys
import numpy as np
import cv2
import itertools
import argparse
from sklearn.cluster import KMeans
from sklearn.neighbors import BallTree
import pylab as pl
from scipy.cluster.vq import vq, kmeans, whiten

def get_sift(img_path):
    img = cv2.imread(img_path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    sift = cv2.xfeatures2d.SIFT_create()
    kp, des = sift.detectAndCompute(img, None)
    return kp, des

def use_cluster(descriptors, k):
    whitened = whiten(descriptors)
    codebook, variance = kmeans(descriptors, k, 1)
    print('kmeans finished and variance:{}'.format(variance))
    return codebook

def get_images_path(pics):
```

```
def get_images_path(pics):
    images = []
    for name in pics:
        images.append(os.path.join(train_path,name))
    return images

def get_all_sift_des(images):
    des_list = []
    for i in range(len(images)):
        if i%600 == 0:
            print('SIFTed:{}'.format(i/600))
        kp, des = get_sift(images[i])
        if des is not None:
            des_list.append(des)
    des_list = list(itertools.chain.from_iterable(des_list))
    des_list = np.array(des_list)
    return des_list

def get_VLAD_descriptors(SIFTdes, images):
    descriptors = []
    img_path = []
    for i in range(len(images)):
        if (i+1)%600 == 0:
            print('VLAD:{}'.format((i+1)/600))
        kp, des = get_sift(images[i])
        if des is not None:
            #des = whiten(des)
            nearest = vq(des, SIFTdes)
            centers = SIFTdes
            k = SIFTdes.shape[0]

            m, d = des.shape
            vlad = np.zeros((k,d))

            for j in range(k):
                if np.sum(nearest[0] == j) > 0:
                    vlad[j] = np.sum(des[nearest[0] == j, :] - centers[j], axis=0
            vlad = vlad.flatten()
            vlad = np.sign(vlad) * np.sqrt(np.abs(vlad))
            vlad = vlad/np.sqrt(np.dot(vlad, vlad))

            descriptors.append(vlad)
            img_path.append(images[i])
    return descriptors, img_path

def balltree(vlad, leaf=40):
    tree = BallTree(vlad, leaf_size = leaf)
    return tree

def query(img, top, des_list, tree):
    img = [img]

    vlad, img_path = get_VLAD_descriptors(des_list, img)

    dist, idx = tree.query(vlad, top)

    return dist, idx
```

In [9]:

```
train_path = './digitImage/'
query_img = './digitImage/1024.jpg'
pics = os.listdir(train_path)
```

In [10]:

```
with open('./pickle_data/des_list.pickle', 'rb') as f:
    des_list = pickle.load(f)
with open('./pickle_data/SIFTdes.pickle', 'rb') as f:
    SIFTdes = pickle.load(f)
```

In [11]:

```
all_img = get_images_path(pics)
```

In [12]:

```
vlad, each_img = get_VLAD_descriptors(SIFTdes, all_img)
```

```
VLAD:1.0
VLAD:2.0
VLAD:3.0
VLAD:4.0
VLAD:5.0
VLAD:6.0
VLAD:7.0
VLAD:8.0
VLAD:9.0
VLAD:10.0
VLAD:11.0
VLAD:12.0
VLAD:13.0
VLAD:14.0
VLAD:15.0
VLAD:16.0
VLAD:17.0
VLAD:18.0
VLAD:19.0
VLAD:20.0
```

In [13]:

```
tree = balltree(vlad, 40)
```

In [20]:

```
query_img = './digitImage/1042.jpg'
```

In [21]:

```
dist, idx = query(query_img, 5, SIFTdes, tree)
```

In [22]:

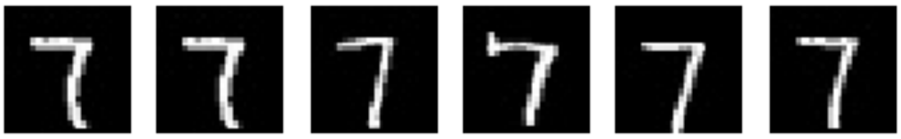
```
idx
```

Out[22]:

```
array([[28314, 32289,  9463, 34694,  6282]])
```

In [24]:

```
img = cv2.imread(query_img)
pl.figure()
pl.gray()
pl.subplot(1,6,1)
pl.imshow(img)
pl.axis('off')
for i in range(idx.shape[1]):
    retrival = cv2.imread(each_img[idx[0,i]])
    pl.gray()
    pl.subplot(1,6,i+2)
    pl.imshow(retrival)
    pl.axis('off')
pl.show()
```



In [ ]:

```

```