Image retrieval by example query image

Experiment with one query image

In [1]:
```
%reset
```
```
Once deleted, variables cannot be recovered. Proceed (y/[n])? y
```

In [2]:
```
from PIL import Image
import torch
import matplotlib.pyplot as plt
import numpy as np
import os
import sys
from time import time
import pickle

experimentPath = r'/Users/jeremywan/Desktop/MMTech/lab1_cbir_student'
os.chdir(experimentPath)
# lab related module
from ai_pytorch_module import *
from cbir_module import *  # LabelDic defined here
import cbir_module
from cbir_module import *
import importlib
importlib.reload(cbir_module)
```
Out[2]:
```
<module 'cbir_module' from '/Users/jeremywan/Desktop/MMTech/lab1_cbir_student/cbir_module.py'>
```

In [3]:
```
#%% Set Path

imgpath = r'./images'
sys.path.append(os.getcwd())
sys.path.append(imgpath)
```

In [4]:
```
#%% load database
# Need this since pickle store a list of Database objects
# Pickle need to refer to this class
class Database :
    def __init__(self) :
        self.imageName =  None
        self.featCNN = None

with open("CBIR_database.pickle","rb") as f:
    dataDict = pickle.load(f)
database = dataDict['database']
```

In [5]:
```
print(database[1].featColorHist.shape)
print(database[1].featCNN.shape)
```
```
(768,)
(1, 4096)
```

Question 6

Implement the following functions. Test your function with 1 image from each label category

1. retrievedID = doRetrieval(featQuery , k, database, imgpath, showImage=True)
2. Precision_K = getPrecisionRank_K(k, queryLabel, retrievedID, database)

In [6]:
```
# student code for function definition
# def showImageInfoFromDB(id, imgpath, database):
# def doRetrieval(featQuery , k, database, imgpath, showImage=True):
#  hint use np.argsort()
def showImageInfoFromDB(id, imgpath, database):
    # your code

    label = database[id].classLabel
    feat1 = database[id].featCNN
    feat2 = database[id].featColorHist

    print("Image name = " , database[id].imageName)
    print("Label ID = " , label)
    print("Label Name = " , LabelDic[label])
    print("Feature dimension CNN = " , feat1.shape)
    print("Feature dimension Colour Histogram = " , feat2.shape)

    imFile = database[id].imageName
    imFile = os.path.join(imgpath, imFile)
    im = Image.open(imFile)
    plt.figure(figsize=(8,6))

    plt.imshow(im) , plt.axis('off')
    titleStr = " Image {}.jpg label = {} Label name = {}".format(str(id), label, LabelDic[label])
    plt.title(titleStr)
```

In [7]:
```
# hint use argsort()
def doRetrieval(featQuery , k, database, imgpath, showImage=True):

    numImages = len(database)
    dist_cnn = []
    idx_k = []

    for f in range (0,numImages) :
        dist = np.linalg.norm(featQuery - database[f].featCNN)
        dist_cnn.append(dist)

    idx_k = np.argsort(dist_cnn)

    return idx_k[1:k+1]
```

In [8]:
```
#%% Test your code with the script in this cell for CNN feature

# Do retrieval by nearest neighbour search
# Use query by example

k=10 # select the top K image to be retrieved
queryID=101  # Select query image ID
featQuery = database[queryID].featCNN
print("Display Query Image id = ", queryID)
showImageInfoFromDB(queryID, imgpath, database)

featQueryCNN = database[queryID].featCNN
retrievedID = doRetrieval(featQueryCNN , k, database, imgpath, showImage=True)
```
```
Display Query Image id =  101
Image name =  101.jpg
Label ID =  2
Label Name =  Beach
Feature dimension CNN =  (1, 4096)
Feature dimension Colour Histogram =  (768,)
```

In [9]:
```
# student code for function definition
# def getPrecisionRank_K(k, queryLabel, retrievedID, database):
def getPrecisionRank_K(k, queryLabel, retrievedID, database):

    rel_img = 0

    for f in retrievedID:
        label = database[f].classLabel
        print(label, end=' ')
        if queryLabel == label:
                rel_img += 1

    precision_k = rel_img/k

    return precision_k
    #endfunc()
```

In [10]:
```
# Report the precision result
print("\n Experiment on CBIR with CNN feature as image feature")
print("\n Class labl of retrieve img")
queryLabel = database[queryID].classLabel
Precision_K = getPrecisionRank_K(k, queryLabel, retrievedID, database)
print(" Query image label :" , queryLabel)
print("\n Precision when retrieving {} images for query image {} = {:02.3f}".format(k, queryID, Precision_K))
```
```
 Experiment on CBIR with CNN feature as image feature

 Class labl of retrieve img
2 2 2 2 2 2 2 2 2 2  Query image label : 2

 Precision when retrieving 10 images for query image 101 = 1.000
```

In [11]:
```
#%% Repeat the experiment above for colour histogram feature

def doRetrieval2(featQuery , k, database, imgpath, showImage=True):

    numImages = len(database)
    dist_hist = []
    idx_k = []

    for f in range (0,numImages) :
        dist = np.linalg.norm(featQuery - database[f].featColorHist)
        dist_hist.append(dist)

    idx_k = np.argsort(dist_hist)

    return idx_k[1:k+1]
```

In [12]:
```
database[1].featColorHist.shape
```
Out[12]:
```
(768,)
```

In [13]:
```
print("\n Experiment on CBIR with color histogram as image feature")
print("\n Class labl of retrieve img")
k=10 # select the top K image to be retrieved
queryID=101  # Select query image ID
featQuery = database[queryID].featColorHist
retrievedID = doRetrieval2(featQuery, k, database, imgpath, showImage=True)
# Report the precision result
queryLabel = database[queryID].classLabel
Precision_K = getPrecisionRank_K(k, queryLabel, retrievedID, database)
print("Query image label :" , queryLabel)
print("\nPrecision when retrieving {} images for query image {} = {:02.3f}".format(k, queryID, Precision_K))
```
```
 Experiment on CBIR with color histogram as image feature

 Class labl of retrieve img
9 9 9 10 9 6 2 6 9 6 Query image label : 2

Precision when retrieving 10 images for query image 101 = 0.100
```