Digital Image Processing (EEL715) Assignment 5 Report

Aditya Kumar Jain 2012EE10433	Santosh Nain 2012EE10476	Ayush Jain 2012EE10439

***Abstract*—To implement an image retrieval system for extracting similar images from a dataset (INRIA Holidays Dataset) using the individual or combination of features such as BRIEF, KAZE, SIFT.**

Keywords—Bag of Visual Words, Distance based Hasing (DBH), Indexing schemes, K means clustering

I. Introduction

The idea is to implement an image retrieval system technique to extract similar images from the datasets of images of a given image. We do not want to traverse whole dataset of images again and again for finding similar images so we store these images in terms of Bags of Visual words in a file, so whenever we have to find similar copies of an image, we first calculate its bag of Visual word and compare it with all bags stored in the file we have created earlier. Those images for which matching percentage is above a certain value are perceived as similar images. This method is known as Distance based Hashing.

II. Functions Used

A. Index.py

This function is used to store Bags of Visual Words of dataset of images into an index.csv file.

PYTHON Code:

```
# USE : python index.py --data_path
dataset # import the necessary packages
from descriptor import Descriptor import
argparse import glob import cv2

# construct the argument parser and parse the arguments
parser = argparse.ArgumentParser()
parser.add_argument("-d", "--data_path", required = True,
help = "Path to the directory that contains the images to be
indexed")
arguments = vars(parser.parse_args())
desc = Descriptor((8, 12, 3))
index_file = open("index.csv", "w")  #
opening the index.csv file for writing

# use glob to grab the image paths and loop over them for
imagePath in glob.glob(arguments["data_path"]
+
"/*.png"):
ID = imagePath[imagePath.rfind("/") + 1:]
# Name of the image image =
cv2.imread(imagePath) features =
desc.describe(image)    # getting the bag
of words for the image words = [str(f) for f in
features]

# Combining the features to form a bag of words
index_file.write("%s,%s\n" % (ID, ",".join(words))) # Writing
in the csv file (comma required because comma separated values
(csv) files) index_file.close()
```

B. Descriptor.py

This function is used to find Bags of Visual Words for an input image.

PYTHON Code:

```
import numpy as np import
cv2 class Descriptor: def
__init__(self, bins):

    self.bins = bins
def describe(self, image):
 image=cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
# Converting the RGB image to HSV colour space
 features = []
 # Array of features for the given image
 (height, width) = image.shape[:2]
# Dimensions of the image
 (centreX, centreY) = (int(width * 0.5),
int(height * 0.5))
 # Centre of the image segments = [(0, centreX,
0, centreY), (centreX, width, 0, centreY),
(centreX, width, centreY, height),(0, centreX,
centreY, height)]
 # Divide the image into 4 segments
(quadrants)

 # construct an elliptical mask of 75% dimensions
representing the central segment of the image

(axesX, axesY) = (int(width * 0.75) / 2,
int(height * 0.75) / 2) elliptical_mask =
np.zeros(image.shape[:2], dtype = "uint8")


# Mask of dimensions of image

cv2.ellipse(elliptical_mask,(centreX, centreY),
(axesX, axesY), 0, 0, 360, 255, -1)
# Making ellipse on the mask

# Making 5 segments by subtracting overlapping
areas for (startX, endX, startY, endY) in
segments:

FullMask = np.zeros(image.shape[:2], dtype =
"uint8")

cv2.rectangle(FullMask,(startX, startY), (endX,
endY), 255, -1)

Subtracted_Mask=cv2.subtract(FullMask,
elliptical_mask)

# Removing part common with ellipse hist =
self.histogram(image, Subtracted_Mask) #
Obtaining the histogram for this segment
features.extend(hist) hist =
self.histogram(image, elliptical_mask) #
Obtaining histogram for central elliptical
region features.extend(hist) return features

# Returning the bag of words for this image

def histogram(self, image, mask):

hist = cv2.calcHist([image], [0, 1, 2], mask,
self.bins, [0, 180, 0, 256, 0, 256])

# Obtaining the HSV histogram

   hist = cv2.normalize(hist,hist).flatten()
# Normalizing the histogram return hist
```

C. Retrieve.py

This function is used to output 10 images which are most similar to the input test image in terms of similarity between their Bags of Visual Words.

PYTHON Code:

```
# assumes index.csv is in the current directory
# USE: python retrieve.py --image
queries/103100.png --data_path dataset

from descriptor import Descriptor from

searcher import Searcher import

argparse import cv2

# construct the argument parser and parse the
arguments

parser = argparse.ArgumentParser()

parser.add_argument("-q", "--image", required =

True,help = "Path to the test image")

parser.add_argument("-r","--data_path",

required = True, help = "Path to the dataset")

arguments = vars(parser.parse_args())

test=cv2.imread(arguments["image"]) #reading

the test image searcher = Searcher()

#initialising the searcher object desc =

Descriptor((8, 12, 3))

 #initialising the descriptor words =

   desc.describe(test) #obtaining

the features (bag of visual words) of the test
image

results = searcher.search(words)
 #results obtained by searching in the dataset #

display the test cv2.imshow("Test image", test)

pp = 1 # loop over the results for (score, ID)

in results:

result = cv2.imread(arguments["data_path"] +
"/" + ID)

cv2.imshow("Result " + `pp`, result)

if pp == 5: cv2.waitKey(0)

pp += 1
```

D. Searcher.py

This function is used to compare similarity between two images by comparing their Bags of Visual Words.

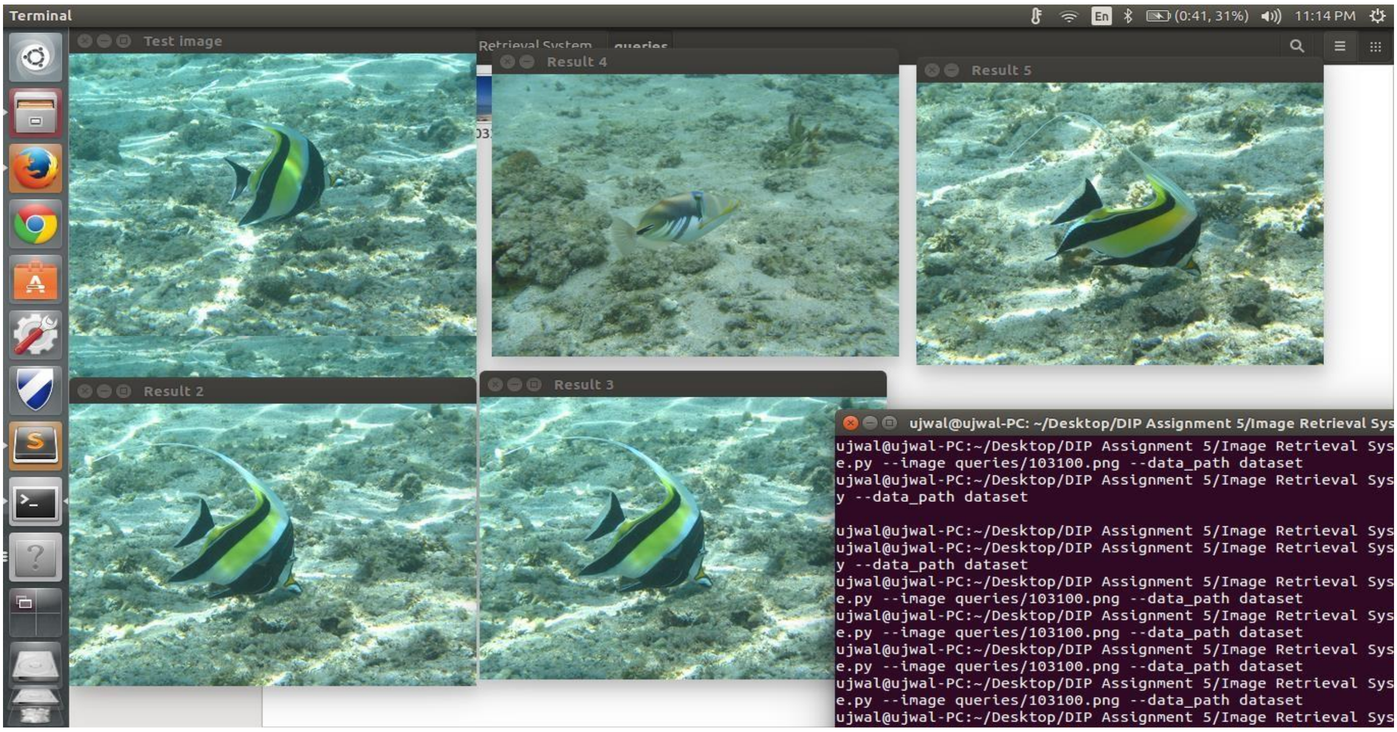
PYTHON Code:

```
 # assumes index.csv is in the same directory

import numpy as np import csv     class

Searcher:

 def search(self, queryWords, limit = 10):     results =

 {}

 #We will store the names of the matching images in this
 array

 # open the index file for reading with open("index.csv") as
index_file:     reader = csv.reader(index_file)   # initialize
the CSV reader   for row in reader:     features     =
[float(x) for x in row[1:]]

 # 0th element of every row is the name of the image, so we

start from the 1st element  distance     =

self.chi2_distance(features, queryWords)
```

```
# Computing the chi-squared distance between the Bag of
words in test image and current dataset image

results[row[0]] = distance

# row[0] contains the name of the image, d contains the
measure of similarity (distance)        index_file.close()
results = sorted([(v, k) for (k, v) in results.items()])
# sorting results on the basis of smallest distance between
images (best matches)        return results[:limit]

# returning only 10 best matches
            def chi2_distance(self, histA, histB, eps = 1e-10):
                                  # compute the chi-squared distance

    chi = 0.5 * np.sum([((a - b) ** 2) / (a + b + eps)
 # we calculate the cumulative distance between the  histograms
as the sum of individual point distances  for (a, b) in zip(histA,
histB)])

    # zip combines all the pairs of elements of the histograms
into one array

return chi
```

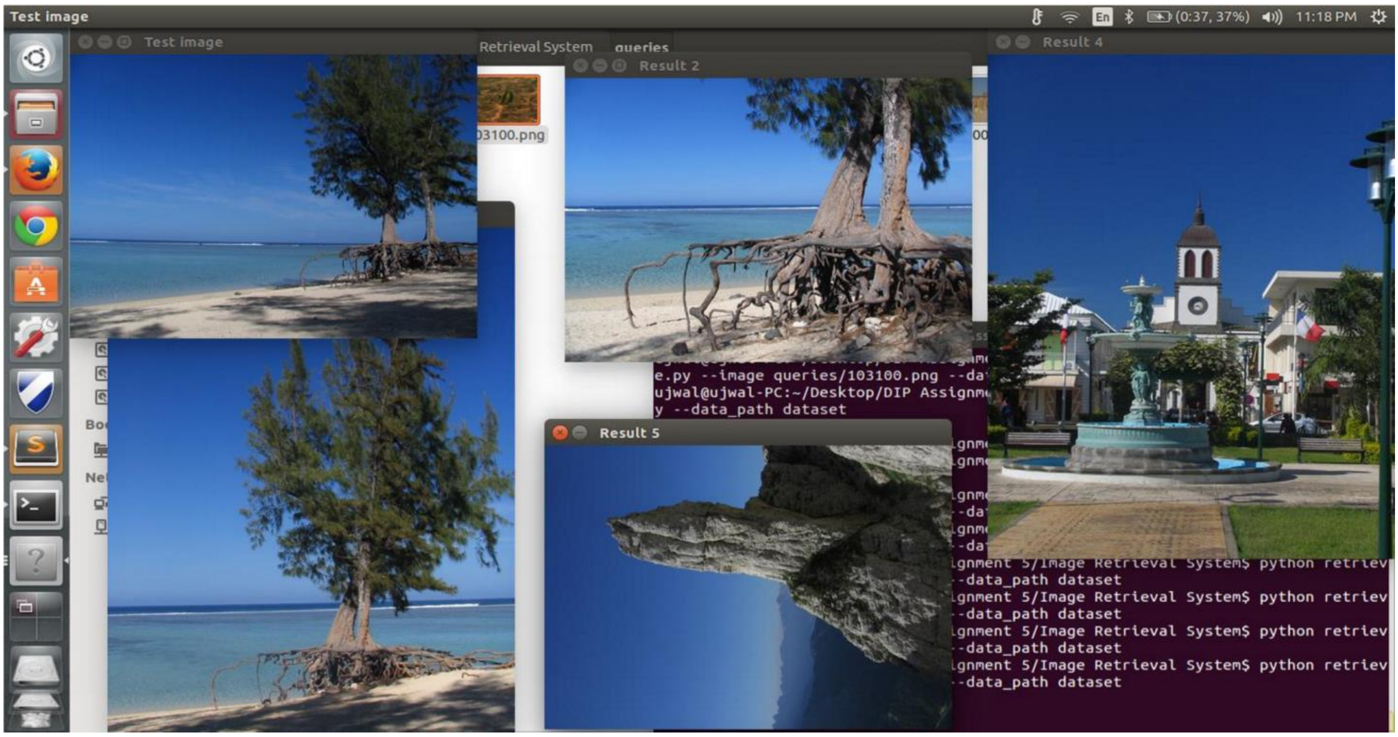
III. Observations

We have run our code for following seven images. These images along with their results found in INRIA holiday dataset are given below:



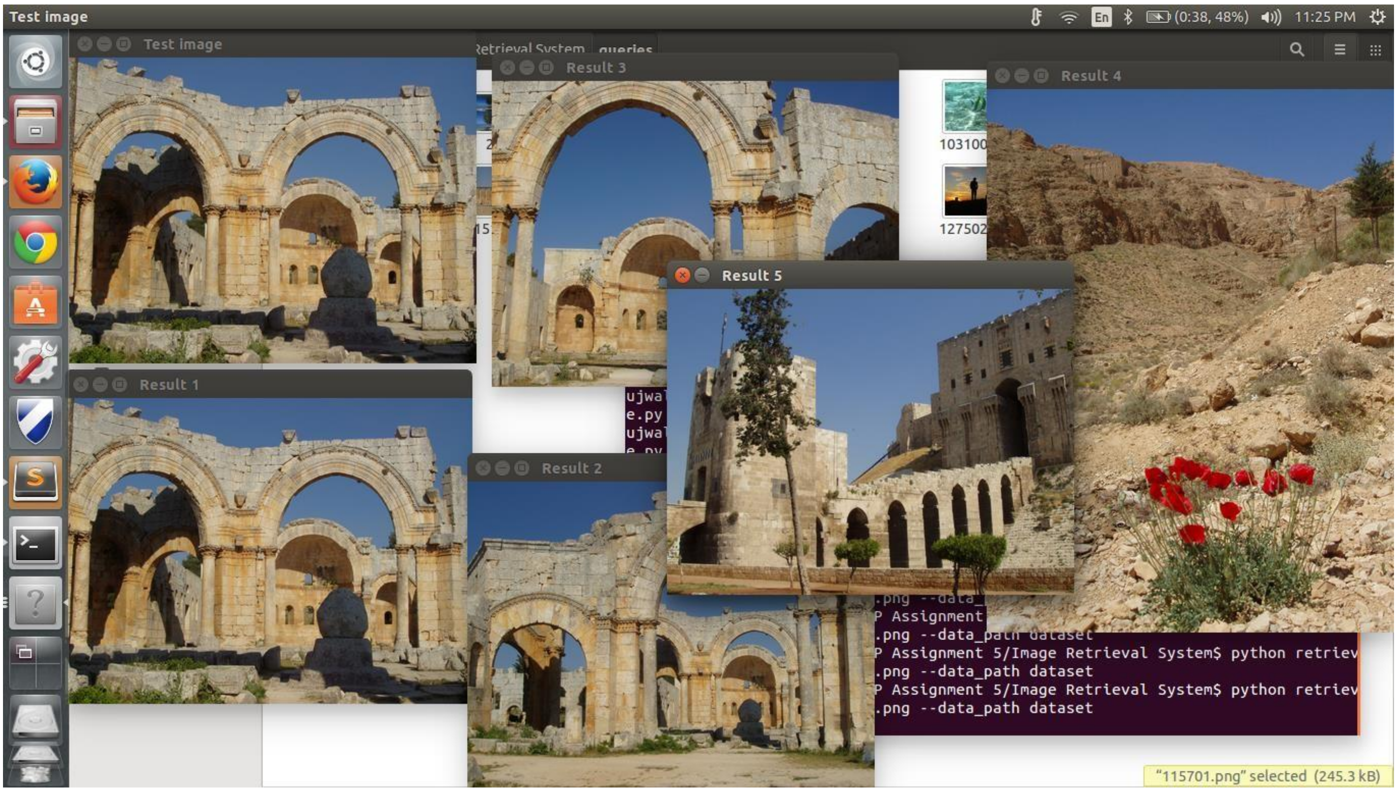
```
ujwal@ujwal-PC:~/Desktop/DIP Assignment 5/Image Retrieval Sys
e.py --image queries/103100.png --data_path dataset
ujwal@ujwal-PC:~/Desktop/DIP Assignment 5/Image Retrieval Sys
y --data_path dataset

ujwal@ujwal-PC:~/Desktop/DIP Assignment 5/Image Retrieval Sys
ujwal@ujwal-PC:~/Desktop/DIP Assignment 5/Image Retrieval Sys
y --data_path dataset
ujwal@ujwal-PC:~/Desktop/DIP Assignment 5/Image Retrieval Sys
e.py --image queries/103100.png --data_path dataset
ujwal@ujwal-PC:~/Desktop/DIP Assignment 5/Image Retrieval Sys
e.py --image queries/103100.png --data_path dataset
ujwal@ujwal-PC:~/Desktop/DIP Assignment 5/Image Retrieval Sys
e.py --image queries/103100.png --data_path dataset
ujwal@ujwal-PC:~/Desktop/DIP Assignment 5/Image Retrieval Sys
e.py --image queries/103100.png --data_path dataset
ujwal@ujwal-PC:~/Desktop/DIP Assignment 5/Image Retrieval Sys
```

Fig 1. Image at top left – test image, rest images are retrieved from dataset



```
e.py --image queries/103100.png --da
ujwal@ujwal-PC:~/Desktop/DIP Assignme
y --data_path dataset
ignment 5/Image Retrieval System$ python retriev
--data_path dataset
ignment 5/Image Retrieval System$ python retriev
--data_path dataset
ignment 5/Image Retrieval System$ python retriev
--data_path dataset
ignment 5/Image Retrieval System$ python retriev
--data_path dataset
```

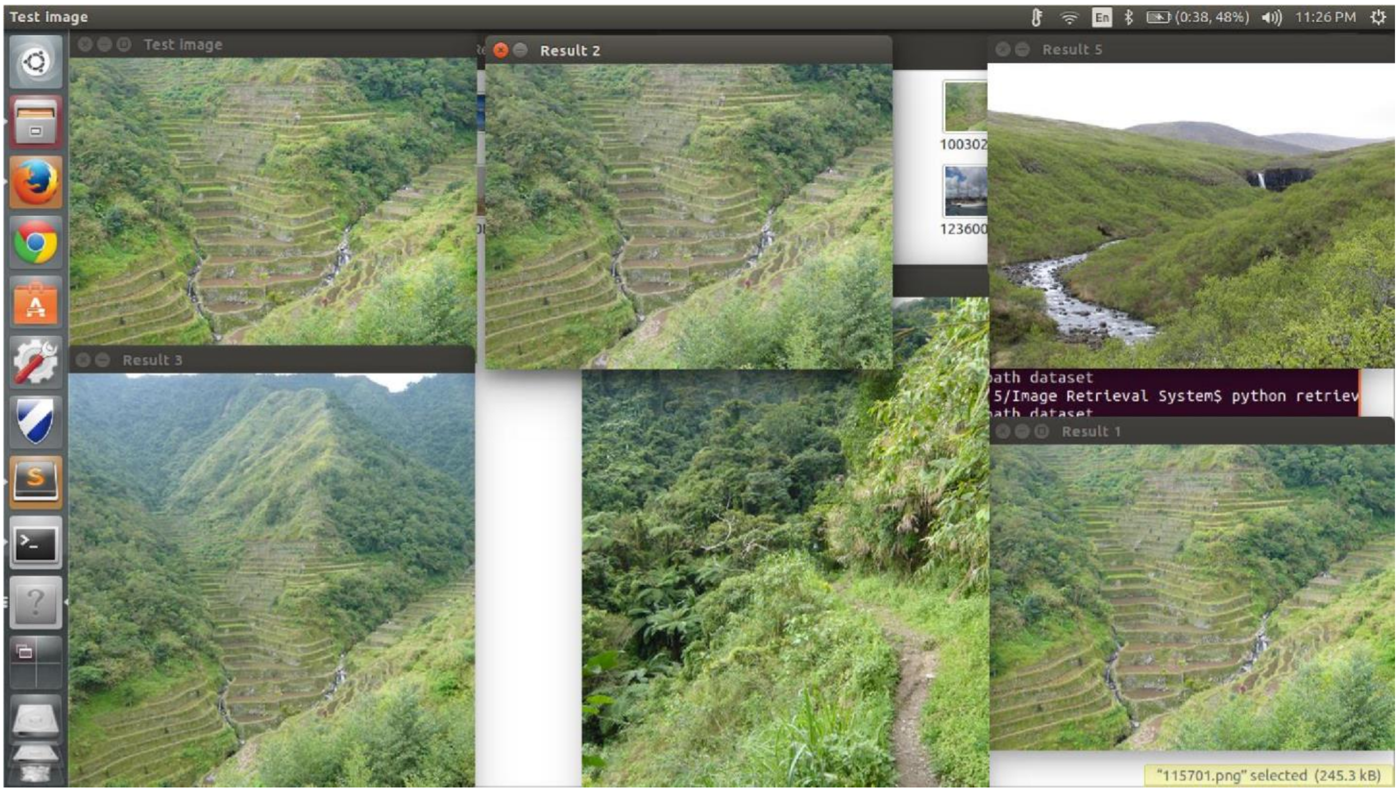
Fig 2. Image at top left – test image, rest images are retrieved from dataset



```
ujwa
e.py
ujwal
.png --data_
P Assignment
.png --data_path dataset
P Assignment 5/Image Retrieval System$ python retriev
.png --data_path dataset
P Assignment 5/Image Retrieval System$ python retriev
.png --data_path dataset
```

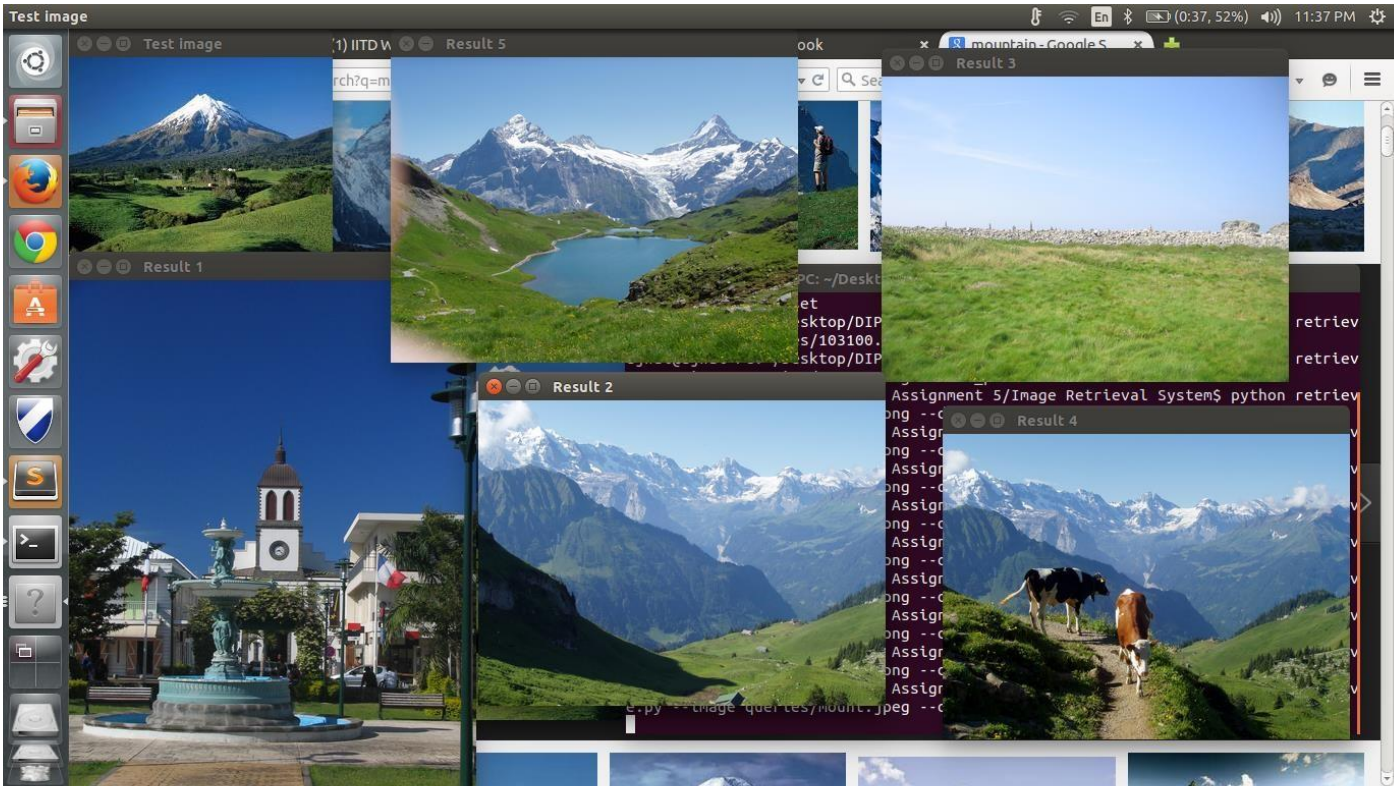
“115701.png” selected (245.3 kB)

Fig 3. Image at top left – test image, rest images are retrieved from dataset



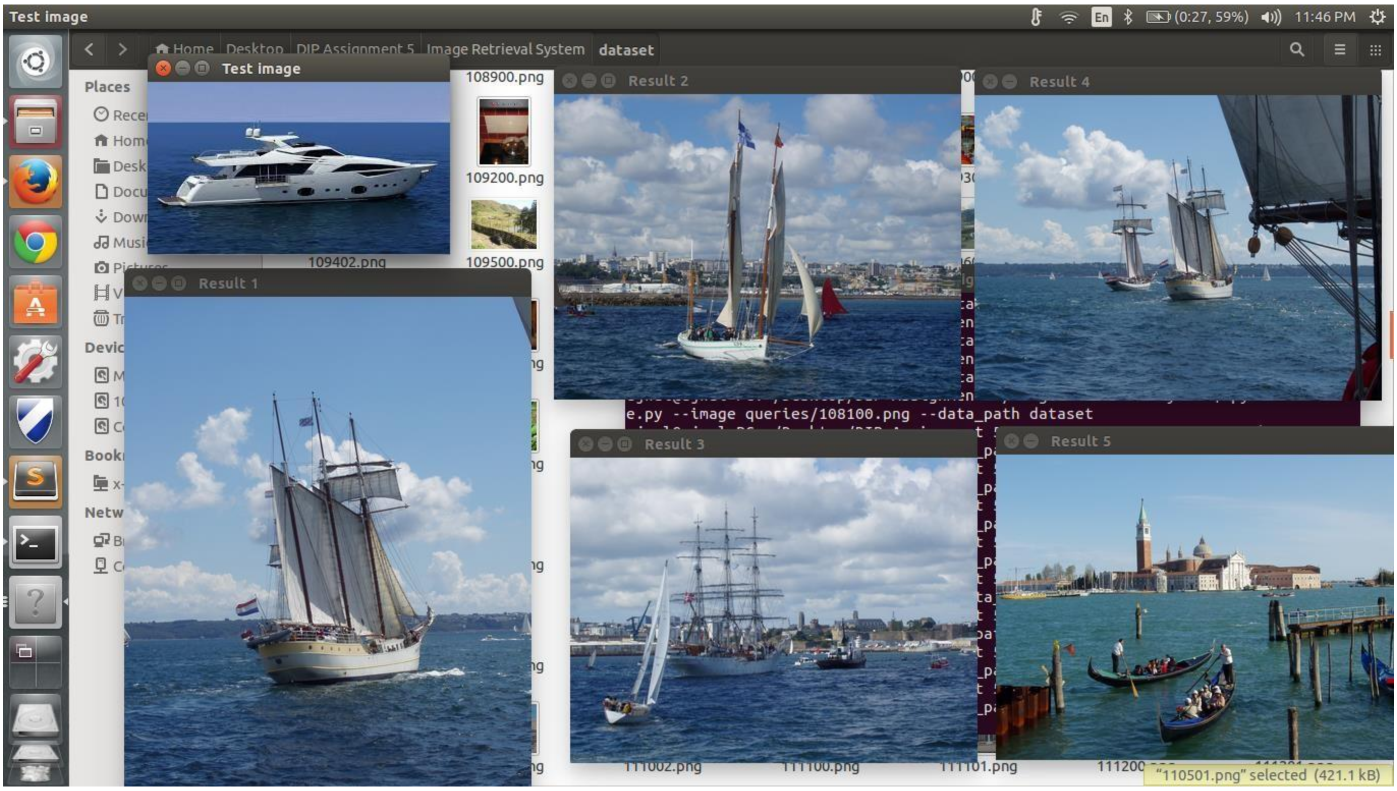
```
path dataset
5/Image Retrieval System$ python retriev
path dataset
```

Fig 4. Image at top left – test image, rest images are retrieved from dataset



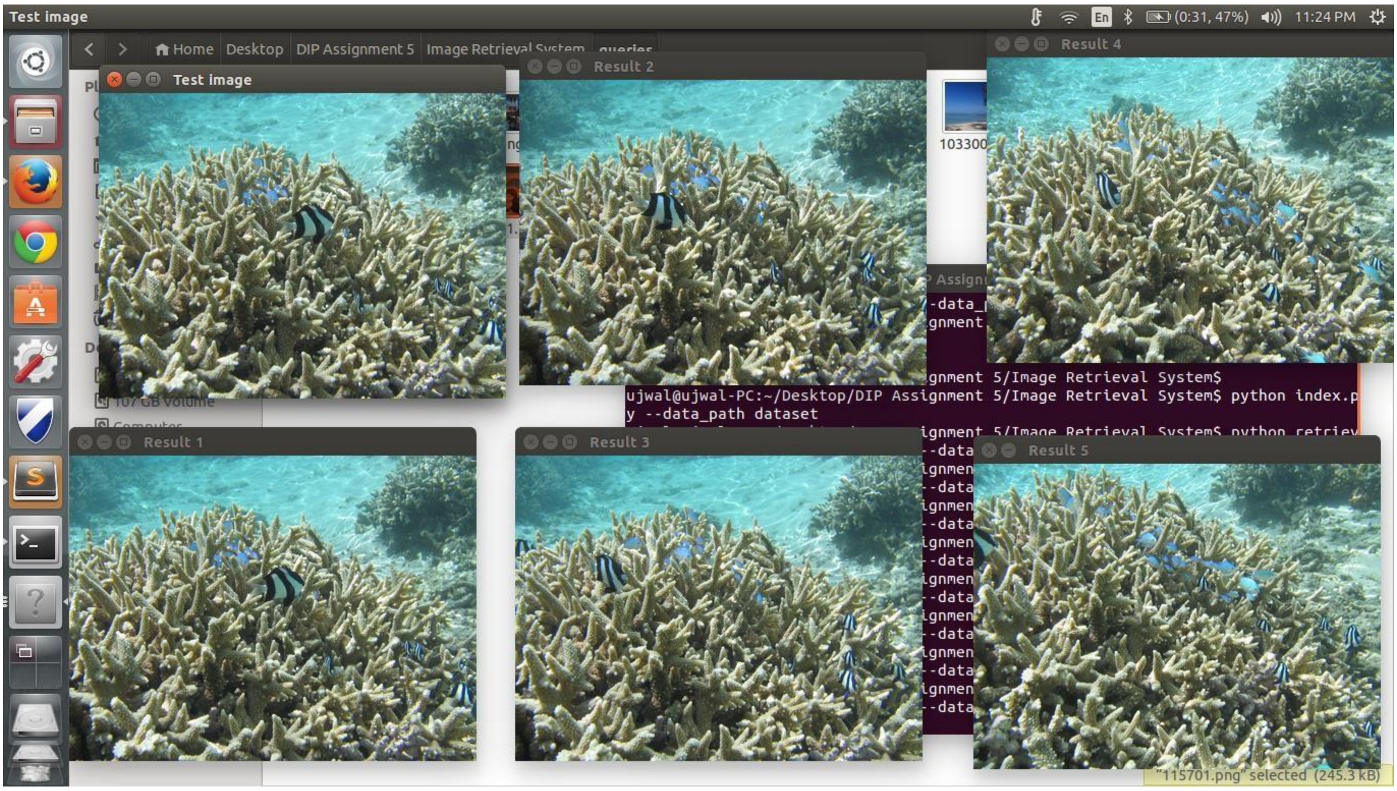
```
PC: ~/Deskt
et
sktop/DIP
s/103100.
sktop/DIP
retriev
retriev
Assignment 5/Image Retrieval System$ python retriev
png --c
Assign
png --c
Assign
png --c
Assign
png --c
Assign
png --c
Assign
png --c
Assign
png --c
Assign
png --c
Assign
e.py --image queries/mount.jpeg --c
```

Fig 5. Image at top left – test image, rest images are retrieved from dataset



```
e.py --image queries/108100.png --data_path dataset
```

Fig 6. Image at top left – test image, rest images are retrieved from dataset



```
gnment 5/Image Retrieval System$
ujwal@ujwal-PC:~/Desktop/DIP Assignment 5/Image Retrieval System$ python index.p
y --data_path dataset
ignment 5/Image Retrieval System$ python retriev
```

Fig 7. Image at top left – test image, rest images are retrieved from dataset

ACKNOWLEDGMENT

We are thankful to Prof Brejesh Lall for his guidance. We are also thankful to Ms. Prerna Mukherjee to give us ideas like Bag of Visual Worlds (BOVW), indexing schemes and distance based hashing (DBH) and K means clustering.

search-engine-python-opencv/ -guide-

REFERENCES

[1] http://www.researchgate.net/post/What_is_chi-squared_distance_I_need_help_with_the_source_codehttp://in.mathwor ks.com/discovery/object-detection.html

image-search-engine-python-opencv/

[3] http://www.pyimagesearch.com/2014/12/01/complete-guide-buildingimage-search-engine-python-opencv/

[2] http://www.pyimagesearch.com/2014/12/01/complete-guide-building-

[4] http://www.pyimagesearch.com/2014/12/01/completebuilding-image-

[5] http://www.pyimagesearch.com/2014/12/01/complete-guide-buildingimage-search-engine-python-opencv/