

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
Phòng Đào tạo Sau đại học & Khoa học công nghệ

ĐỒ ÁN MÔN HỌC
NHẬN DẠNG THỊ GIÁC VÀ ỨNG DỤNG

GVHD:

Lê Đình Duy
Nguyễn Tấn Trần Minh Khang

Học viên:

Lê Hoàng Dũng - Mã HV: CH1501022

TP. Hồ Chí Minh – tháng 12/2017

MỤC LỤC

MÔ TẢ ĐỒ ÁN	1
PHẦN 1: HỆ THỐNG TRUY VẤN HÌNH ẢNH	2
I. Tổ chức thư mục.....	2
II. Phương pháp xây dựng hệ thống.....	5
1. Rút trích đặc trưng (file index.py).....	5
2. Kỹ thuật gom cụm (Clustering) (file index.py).....	12
3. Tính khoảng cách TF-IDF (file index.py)	15
4. Truy vấn ảnh (file search.py).....	15
5. Xây dựng server (file server.py).....	18
6. Xây dựng website (NodeJS)	18
7. Demo chương trình.....	19
PHẦN 2: VẤN ĐỀ MỞ RỘNG	20
I. Global feature	20
II. Local feature	21
III. Kết hợp local feature và global feature trong bài toán nhận dạng	22
1. Stacking	22
2. Classification Hierarchy	23
IV. Kết luận	25
V. Tài liệu tham khảo	25
TÀI LIỆU THAM KHẢO	26
MỘT SỐ SOURCE CODE THAM KHẢO	26

MÔ TẢ ĐỒ ÁN

Xây dựng hệ thống truy vấn hình ảnh (Image Retrieval System)

- Github:

<https://github.com/lehoangdung0612/VRA.LeHoangDung.CH1501022/tree/master/VRA-Final>

- Youtube: <https://youtu.be/vkHdFcv3QfA>

Mục tiêu, nội dung đồ án

Tìm hiểu và áp dụng các phương pháp dùng BoW để xây dựng một hệ thống truy vấn hình ảnh. Hệ thống cung cấp một giao diện người dùng website. Tích hợp web service cung cấp các API để phục vụ cho việc truy vấn

Hệ thống sử dụng bộ dữ liệu train chuẩn là Corel-1K (tham khảo <http://wang.ist.psu.edu/docs/related/>). Vì lí do hạn chế phần cứng nên sẽ giới hạn 1000 bức ảnh train.

Ngoài chức năng retrieval bằng ảnh, hệ thống còn có tính năng cho phép người dùng lựa chọn vùng truy vấn (cropped), hoặc chọn loại đặc trưng sẽ dùng.

Các phương pháp được sử dụng: rút trích các đặc trưng như SIFT, SURF, ROOTSIFT, phương pháp gom cụm Kmeans và tính độ tương đồng bằng TF-IDF

Ví dụ, tham khảo matlab: <https://www.mathworks.com/help/vision/ug/image-retrieval-with-bag-of-visual-words.html>.

PHẦN 1: HỆ THỐNG TRUY VẤN HÌNH ẢNH

I. Tổ chức thư mục

Tập tin đồ án gồm các thư mục sau:

DAO HOC > NHAN DANG THI GIAC VA UNG DUNG > VRA.LeHoangDung.CH1501022 > VRA-Final >				
Name	^	Date modified	Type	Size
Bao cao		12/25/2017 10:27 ...	File folder	
Source		12/25/2017 11:03 ...	File folder	

- *Bao cao*: File báo cáo đồ án, phương pháp, kỹ thuật xây dựng hệ thống
- *Sources*: Chứa source code gồm
 - *web-client*: xây dựng web bằng ngôn ngữ NodeJS
 - *python-backend*: xây dựng một server REST API truy vấn ảnh bằng ngôn ngữ Python
 - *queries*: thư mục chứa các ảnh test
 - *README.md* : Thông tin cài đặt và chạy thử hệ thống

Hướng dẫn cài đặt hệ thống

Môi trường cài đặt:

- Web:
 - NodeJS version 8.9.1
 - Cách build hệ thống:
 - `cd web-client`
- Cài đặt packages
- `npm install`
- Khởi tạo server (localhost:3000)
- `npm run watch`

- Server

- Python: version 2.7.14
- Cách build hệ thống:
 - `cd python-backend`

Cài đặt packages

- `python -m pip install -r requirements.txt`

Khởi tạo data huấn luyện


















- `python index.py`

Chạy server (localhost:5000)

- `python server.py`

- Thư mục dataset:

- Copy ảnh bỏ vào thư mục: `python-backend/dataset/corel/`

HOC > NHAN DANG THI GIAC VA UNG DUNG > VRA.LeHoangDung.CH1501022 > VRA-Final > Source > python-backend > dataset > corel					
Name	Date	Type	Size	Tags	
 0.jpg	12/29/2017 3:19 PM	JPG File	35 KB		
 1.jpg	12/29/2017 3:19 PM	JPG File	24 KB		
 2.jpg	12/29/2017 3:19 PM	JPG File	42 KB		
 3.jpg	12/29/2017 3:19 PM	JPG File	28 KB		
 4.jpg	12/29/2017 3:19 PM	JPG File	42 KB		
 5.jpg	12/29/2017 3:19 PM	JPG File	46 KB		
 6.jpg	12/29/2017 3:19 PM	JPG File	40 KB		
 7.jpg	12/29/2017 3:19 PM	JPG File	29 KB		
 8.jpg	12/29/2017 3:19 PM	JPG File	43 KB		
 9.jpg	12/29/2017 3:19 PM	JPG File	25 KB		
 10.jpg	12/29/2017 3:19 PM	JPG File	32 KB		
 11.jpg	12/29/2017 3:19 PM	JPG File	41 KB		
 12.jpg	12/29/2017 3:19 PM	JPG File	34 KB		
 13.jpg	12/29/2017 3:19 PM	JPG File	45 KB		
 14.jpg	12/29/2017 3:19 PM	JPG File	39 KB		
 15.jpg	12/29/2017 3:19 PM	JPG File	34 KB		
 16.jpg	12/29/2017 3:19 PM	JPG File	30 KB		

Một số configuration trong hệ thống

- Web:
 - o File “config.json”:
 - **pyAPI.host**: link host server Python, giá trị default là `http://localhost:5000/`
 - **pyAPI.search**: link API đến server Python để truy vấn ảnh, giá trị default là `http://localhost:5000/search`
 - **pyAPI.upload**: link API đến server Python để upload ảnh, giá trị default là `http://localhost:5000/file-upload`
 - **externalUpload**: cho phép upload ảnh lên server Python, giá trị default là `true`
 - **searchByAjax**: cho phép gọi ajax thay vì submit ảnh lên server Python, giá trị default là `false`
- Server
 - o File “config.py”:
 - **Settings.SEARCH_RESULT**: số lượng kết quả trả về
 - **Settings.MAX_FILES**: số lượng file lớn nhất dùng để huấn luyện
 - **Settings.ROOT_DATASET_FOLDER**: thư mục lưu trữ ảnh huấn luyện, chứa các thư mục con là loại ảnh huấn luyện
 - **Settings.TRAIN_DATASET**: tên thư mục loại ảnh được huấn luyện
 - **Settings.FEATURE**: loại đặc trưng dùng để rút trích
 - **Settings.FEATURE_FILE**: tên file lưu trữ features
 - **KmeansTYPE**: phương pháp kmean sẽ dùng (type=1, 2, 3)
 - **Kmeans.NUM_WORDS**: số lượng K clusters
 - **Kmeans.ITER**: số lần lặp Kmeans
 - **Features**: các loại đặc trưng sẽ dùng
 - **Resources**: chứa các cài đặt liên quan đến thư mục ảnh, phục vụ cho việc truy vấn đường dẫn file ảnh bằng các URL

II. Phương pháp xây dựng hệ thống

1. Rút trích đặc trưng (file index.py)

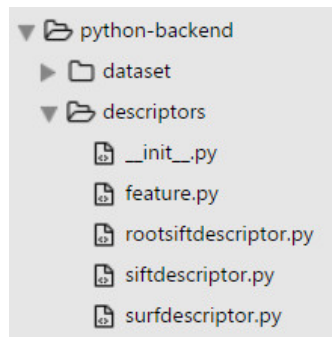
Hệ thống sử dụng ngôn ngữ Python để cài đặt hệ thống truy vấn ảnh. áp dụng một số thư viện của Python để hỗ trợ cho việc xử lý ảnh:

- **Numpy**: chứa các hàm xử lý mảng, ma trận
- **Scipy**: chứa các hàm xử lý ảnh, xử lý mảng, gom cụm
- **OpenCV**: chứa các hàm xử lý ảnh, đọc ảnh, rút trích đặc trưng, hiển thị ảnh,..
- **Pillow**: chứa các hàm để encode, lấy color,...
- **Sklearn**: xử lý ảnh, gom cụm (clustering), đọc và lưu file

Quá trình huấn luyện dữ liệu train:

Nạp ảnh → Đọc và rút trích descriptors → clustering → tính TF-IDF → lưu kết quả tập tin

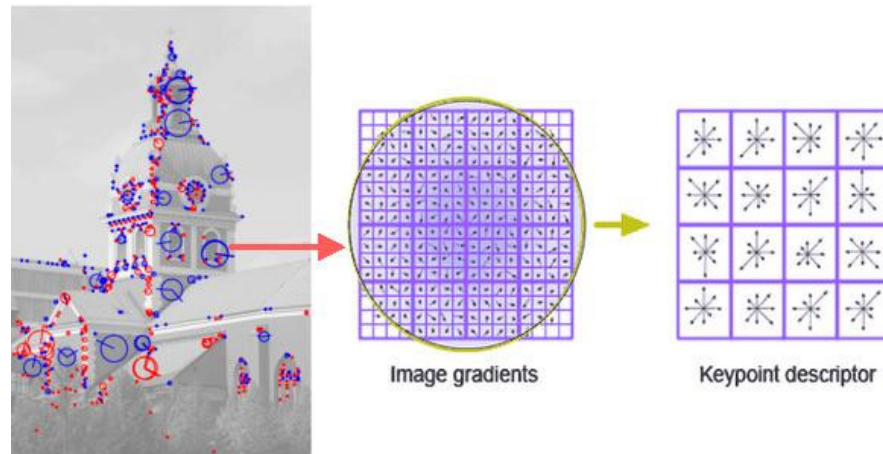
Áp dụng một số kĩ thuật rút trích đặc trưng như SIFT, SURF, ROOTSIFT. Các đặc trưng này được lựa chọn có sẵn trong thư viện của OpenCV và một số khác tham khảo từ trên internet. Các phương pháp được cài đặt trong thư mục *python-backend/descriptors*



Đặc trưng SIFT:

- Từ ảnh tìm ra các điểm ảnh đặc biệt, gọi là feature point hay keypoint. Đầu vào và đầu ra của phép biến đổi SIFT: ảnh -> SIFT -> các keypoint. Keypoint phụ thuộc rất ít vào cường độ sáng, nhiễu, che khuất (một phần ảnh bị che), góc xoay (ảnh bị xoay trong mặt phẳng 2D), thay đổi của tư thế (pose thay đổi trong không gian 3D)

- Để có thể phân biệt keypoint này với keypoint khác cần tìm ra tham số gì đó, gọi là descriptor. 2 keypoint khác nhau thì phải descriptor khác nhau. Thường thì descriptor là chuỗi số gồm 128 số (vector 128 chiều).
- Sau khi áp dụng biến đổi SIFT, ứng với mỗi keypoint, thu được (1) tọa độ keypoint (2) scale và orientation của keypoint (3) descriptor. Các mũi tên trong hình dưới vẽ nhờ vào scale và orientation



```
# import the necessary packages
import numpy as np
import cv2

class SIFT:
    def __init__(self):
        self.detector = cv2.xfeatures2d.SIFT_create()
        return;

    def describe(self, image):
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        (kps, descs) = self.detector.detectAndCompute(image, None)

        if len(kps) == 0:
            return ([], None)

        return (kps, descs)
```

Đặc trưng SURF

- SURF cũng gồm các bước như ở SIFT:
 - o Scale-space extrema detection.
 - o Keypoint localization.
 - o Orientation assignment.
 - o Keypoint descriptor.

- Nhưng, ở từng bước SURF sẽ có những sự cải thiện để cải thiện tốc độ xử lý mà vẫn đảm bảo độ chính xác trong việc detection.
- Ở SIFT, việc tìm Scale-space dựa trên việc tính gần đúng LoG (Laplace of Gaussian) dùng DoG (Difference of Gaussian), trong khi đó SURF sử dụng Box Filter, tốc độ xử lý sẽ được cải thiện đáng kể với việc dùng ảnh tích phân (integral image)
- Ở bước Orientation Assignment, SURF sử dụng wavelet response theo 2 chiều dọc và ngang, sau đó tính hướng chính bằng cách tính tổng các response đó, có một điều đáng chú ý là wavelet response cũng dễ dàng tính được với ảnh tích phân (integral image)

```
# import the necessary packages
import numpy as np
import cv2

class SURF:
    def __init__(self):
        self.detector = cv2.xfeatures2d.SURF_create()
        return;

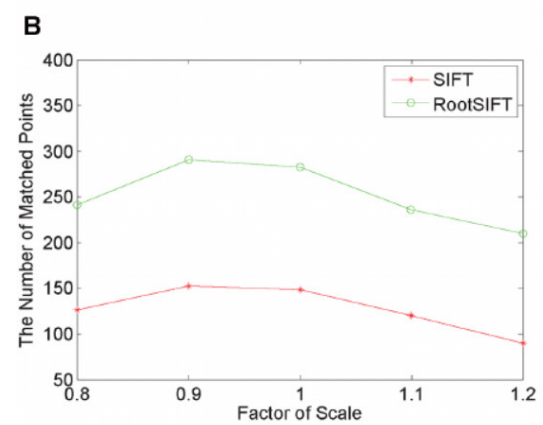
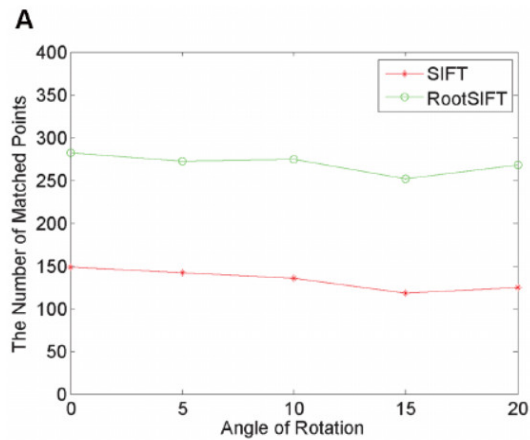
    def describe(self, image):
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        (kps, descs) = self.detector.detectAndCompute(image, None)

        if len(kps) == 0:
            return ([], None)

        return (kps, descs)
```

Đặc trưng ROOTSIFT

- ROOTSIFT là một cải tiến khác của SIFT, vẫn sử dụng khoảng cách Euclidean giống như SIFT. Tuy nhiên khoảng cách Euclidean trong ROOTSIFT của các vector đặc trưng được ánh xạ tới mô hình nhân Hellinger (Hellinger kernel).
- Vector đặc trưng của RootSIFT được tính toán bởi nhân Hellinger, và sự kết hợp giữa hai điểm đặc trưng sau đó được đánh giá bởi tỷ số giữa khoảng cách điểm gần nhất và gần nhất thứ hai.
- Hiệu suất của việc nhận dạng được cải thiện bằng cách thay thế khoảng cách Euclidean bằng nhân Hellinger, đã được chứng minh qua thực nghiệm



```
# import the necessary packages
import numpy as np
import cv2

class RootSIFT:
    def __init__(self):
        # initialize the SIFT feature extractor
        self.detector = cv2.xfeatures2d.SIFT_create()

    def describe(self, image, eps=1e-7):
        # compute SIFT descriptors
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        (kps, descs) = self.detector.detectAndCompute(image, None)

        # if there are no keypoints or descriptors, return an empty tuple
        if len(kps) == 0:
            return ([], None)

        # apply the Hellinger kernel by first L1-normalizing, taking the
        # square-root
        descs /= (descs.sum(axis=1, keepdims=True) + eps)
        descs = np.sqrt(descs)

        return (kps, descs)
```

Xây dựng bộ feature trong hệ thống:

```
# import the necessary packages
from surfdescriptor import SURF
from siftdescriptor import SIFT
from rootsiftdescriptor import RootSIFT

class Descriptor:
    def __init__(self, config):
        descriptors = {};
        descriptors[config.Features["SURF"]] = SURF()
        descriptors[config.Features["SIFT"]] = SIFT()
        descriptors[config.Features["ROOTSIFT"]] = RootSIFT()
        self.descriptors = descriptors;

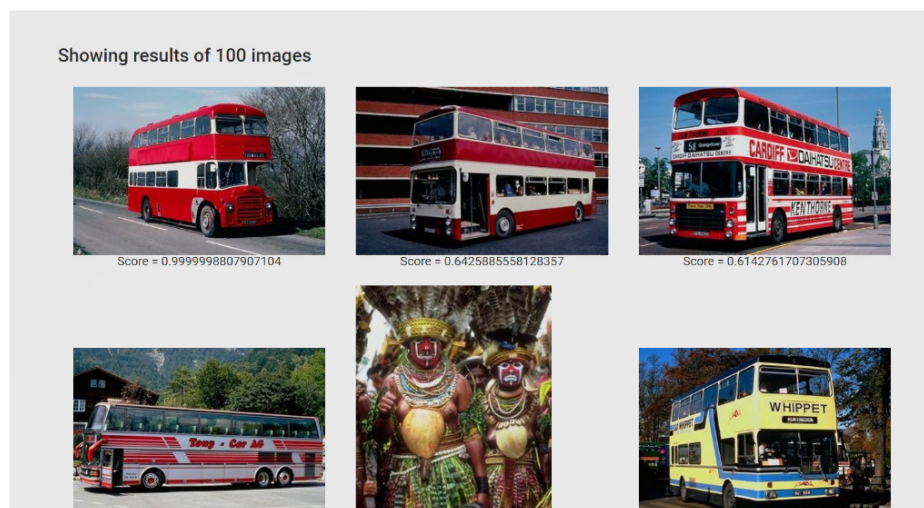
    def extractFeatures(self, feature, image):
        (kps, descs) = self.descriptors[feature].describe(image)
        return (kps, descs)
```

Thực hiện train 1000 ảnh với lần lượt các đặc trưng, nhận thấy ROOTSIFT và SURF có độ chính xác cao hơn so với SIFT

Kết quả: ROOTSIFT \geq SURF > SIFT



Query với SIFT và KMeans type =1





Query với đặc trưng SURF

Showing results of 100 images



Score = 0.999998807907104



Score = 0.6999175548553467



Score = 0.69199538230896



Score = 0.6681864857673645



Score = 0.6613956093788147



Score = 0.6598005294799805



Score = 0.6574621200561523



Score = 0.6456998586654663



Score = 0.6362499594688416



Score = 0.6362499594688416



Score = 0.6362499594688416



Score = 0.6362499594688416



Query với đặc trưng ROOTSIFT

Showing results of 100 images



Để thực hiện huấn luyện ảnh, chỉnh các cài đặt trong file config.py:

- **Settings.MAX_FILES:** giới hạn kích thước ảnh được train
- **Settings.TRAIN_DATASET:** thư mục con của thư mục dataset chứa ảnh train
- **Settings.FEATURE:** loại đặc trưng để train

Chạy lệnh:

python index.py

2. Kỹ thuật gom cụm (Clustering) (file index.py)

Hệ thống sử dụng thư viện **sklearn** và **scipy** để phục vụ cho việc clustering. Bởi vì các thư viện này cung cấp phương pháp KMeans khác nhau. Thực hiện lần lượt các phương pháp và nhận thấy KMeans của thư viện **sklearn** cho thời gian khá lâu so với **scipy**, nhưng bù lại có độ chính xác cao hơn



Query với SIFT và KMeans type =1

Showing results of 100 images





SIFT + KMeans 2 (scipy.cluster.vq.kmeans2)

Showing results of 100 images



Score = 1



Score = 0.6381916403770447



Score = 0.627290666103363



Score = 0.6107041835784912



Score = 0.6088675260543823



Score = 0.6074529886245728



Score = 0.5989221334457397



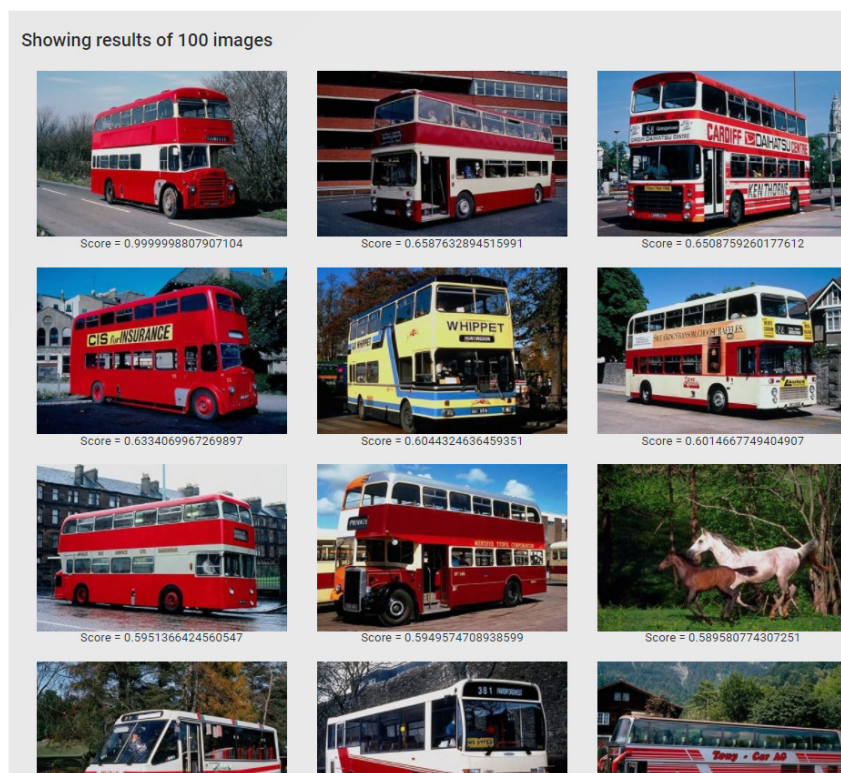
Score = 0.5963426232337952



Score = 0.5954108238220215



SIFT + KMeans 3 (sklearn.cluster.KMeans)



Kết quả gom cụm ta sẽ thu được các center-clusters hay còn gọi là vocabularies

```
def kMeansClustering(kmeansType, descriptors, numWords, iterations):
    print "Clustering k-means: %d words, %d descriptors" %(numWords, descriptors.shape[0])

    if kmeansType == 3:
        kmeans3 = sklearn.cluster.KMeans(n_clusters=numWords, max_iter=iterations)
        kmeans3.fit(descriptors)
        (voc, variance) = (kmeans3.cluster_centers_, kmeans3.labels_)
    elif kmeansType == 2:
        (voc, variance) = scipy.cluster.vq.kmeans2(descriptors, numWords, iterations)
    else:
        (voc, variance) = scipy.cluster.vq.kmeans(descriptors, numWords, iterations)
    return (voc, variance)
```


Một số cài đặt cho Kmeans

- **KMeans.TYPE**: phương pháp Kmeans được chọn(1, 2, 3)
- **KMeans.NUM_WORDS**: số cụm (số K)
- **KMeans.ITER**: số lần lặp của thuật toán kmeans

3. Tính khoảng cách TF-IDF (file index.py)

Dựa vào kết quả bộ từ điển từ việc gom cụm, và danh sách các descriptors đã rút trích, tính toán khoảng cách vector dựa vào công thức TF-IDF và áp dụng phương pháp chuẩn hóa L2

```
def calculateTFIDF(des_list, voc, variance, numWords, numImages):
    print "Calculating Tf-Idf vectorization of features"
    im_features = np.zeros((numImages, numWords), "float32")
    for i in xrange(numImages):
        # In kmeans, labels = scipy.cluster.vq.vq(ar, voc)
        words, _ = scipy.cluster.vq.vq(des_list[i], voc)

        for w in words:
            im_features[i][w] += 1

    # Perform Tf-Idf vectorization
    nbr_occurences = np.sum( (im_features > 0) * 1, axis = 0)
    idf = np.array(np.log((1.0*numImages+1) / (1.0*nbr_occurences + 1)), 'float32')

    # Perform L2 normalization
    im_features = im_features*idf
    im_features = preprocessing.normalize(im_features, norm='l2')
    return (im_features, idf)
```

Kết quả được lưu trong tập tin:

{tên thư mục train}_{loại đặc trưng}_{loại kmean}_features.bin

4. Truy vấn ảnh (file search.py)

Khi người dùng nhập ảnh query, ảnh sẽ được đọc và xử lý thông qua thư viện OpenCV.

Hàm **getQueryImage** có chức năng đọc ảnh và crop, nếu có đủ các tham số (x,y,w,h).

```
def getQueryImage(image_path, cx=0, cy=0, cw=0, ch=0):
    if image_path.find("http") != -1:
        imgdata = urllib.urlopen(image_path)
        arr = np.asarray(bytearray(imgdata.read()), dtype = np.uint8)
        query = cv2.imdecode(arr, -1)
    else:
        if (image_path.find("/") == -1) and (image_path.find("\\") == -1):
            query = cv2.imread("{} / {}".format(Config.Resources["UPLOAD_FOLDER"], image_path))
        elif (image_path.find(settings["ROOT_DATASET_FOLDER"]) == -1):
            query = cv2.imread("{} / {}".format(settings["ROOT_DATASET_FOLDER"], image_path))
        else:
            query = cv2.imread(image_path)

    if (cx and cy and cw and ch):
        query = query[cy:cy + ch, cx:cx + cw]

    return query
```

Tiếp đến ảnh query (hoặc crop) được rút trích đặc trưng và tính toán khoảng cách dựa vào giá trị TF-IDF và bộ từ điển lấy ra từ tập tin `*features.bin`.

```
def calculateTFIDF(descriptors, idf, voc, numWords):
    test_features = np.zeros((1, numWords), "float32")
    (words, distance) = vq(descriptors, voc)
    for w in words:
        test_features[0][w] += 1

    # Perform Tf-Idf vectorization and L2 normalization
    test_features = test_features * idf
    test_features = preprocessing.normalize(test_features, norm='l2')
    return test_features
```

Sau đó ảnh kết quả sẽ được trả về với số lượng được cho trước (`SEARCH_RESULT`) và sắp xếp (hàm **ranking**) theo thứ hạng từ cao đến thấp, sao cho ảnh gần giống query nhất sẽ xuất hiện ở vị trí đầu tiên

Để tính score, ta dựa vào biến `im_features` chứa các đặc trưng rút ra từ tập train. Biến này có kích thước là (nRow, nCol) với nRow là số lượng ảnh train, nCol là số features rút trích. Nghĩa là mỗi dòng trong `im_features` tương ứng với features của 1 ảnh. Tương tự `test_features` là feature của ảnh query (chỉ có 1 dòng). Score được tính nhờ vào phép nhân ma trận `test_features x invert(im_features)`. Kết quả trả về là một danh sách có n ảnh result (n là tổng số ảnh train) mà mỗi phần tử có giá trị là độ tương đồng với ảnh query. Dựa vào danh sách này để trả về danh sách ảnh kết quả.

```
def ranking(test_features, im_features, searchResult):
    score = np.dot(test_features, im_features.T)
    rank_ID = np.argsort(-score[0])
    # sorted decending score
    decScore = sorted(score[0], reverse=True)
    # reduce ranking
    rank_ID = rank_ID[:searchResult]
    return (rank_ID, decScore)
```

Hàm **searchImage** gồm toàn bộ quy trình để tìm kiếm ảnh phù hợp câu query, tham số là query, loại feature và các thông số crop ảnh (cx, cy, cw, ch)

```
def searchImage(q, feature, searchResult, cx=0, cy=0, cw=0, ch=0):
    try:
        if (searchResult is None):
            searchResult = settings["SEARCH_RESULT"]
        if (feature is None):
            feature = settings["FEATURE"]

        # load data features
        (image_paths, im_features, idf, numWords, voc) = Datasets[feature]

        # load query image and crop
        query = getQueryImage(q, cx, cy, cw, ch)

        # create descriptors vertically in a numpy array
        descriptors = createDescriptors(descriptorHandler, feature, query)

        # Calculate the histogram of features
        test_features = calculateTFIDF(descriptors, idf, voc, numWords)

        # get distances and ranking result list
        (rank_ID, decScore) = ranking(test_features, im_features, searchResult)

        result = []
        for i, ID in enumerate(rank_ID):
            result += [{
                "image": image_paths[ID],
                "score": float(min(decScore[i], 1))
            }]

        return result

    except Exception as e:
        tb = traceback.format_exc()
        print 'Error found: %s' % (tb)
        return None
```

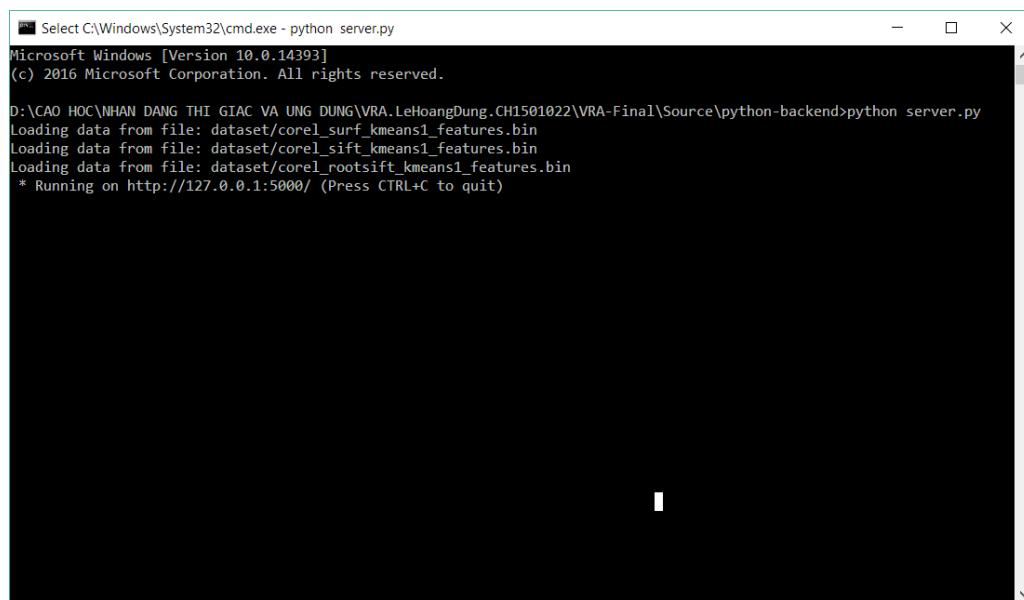
5. Xây dựng server (file server.py)

Sử dụng thư viện Flask của Python và các thư viện con (Flask-Cors, Flask-Jsonpify, Flask-RESTful, Flask-Uploads) của nó để xây dựng một web service cung cấp API để truy vấn ảnh

Việc cài đặt server được thực hiện trong tập tin server.py. Server được chạy trên localhost:5000

Chạy lệnh:

python server.py



```
Select C:\Windows\System32\cmd.exe - python server.py
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

D:\CAO HOC\NHAN DANG THI GIAC VA UNG DUNG\VRA.LeHoangDung.CH1501022\VRA-Final\Source\python-backend>python server.py
Loading data from file: dataset\corel_surf_kmeans1_features.bin
Loading data from file: dataset\corel_sift_kmeans1_features.bin
Loading data from file: dataset\corel_rootsift_kmeans1_features.bin
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

6. Xây dựng website (NodeJS)

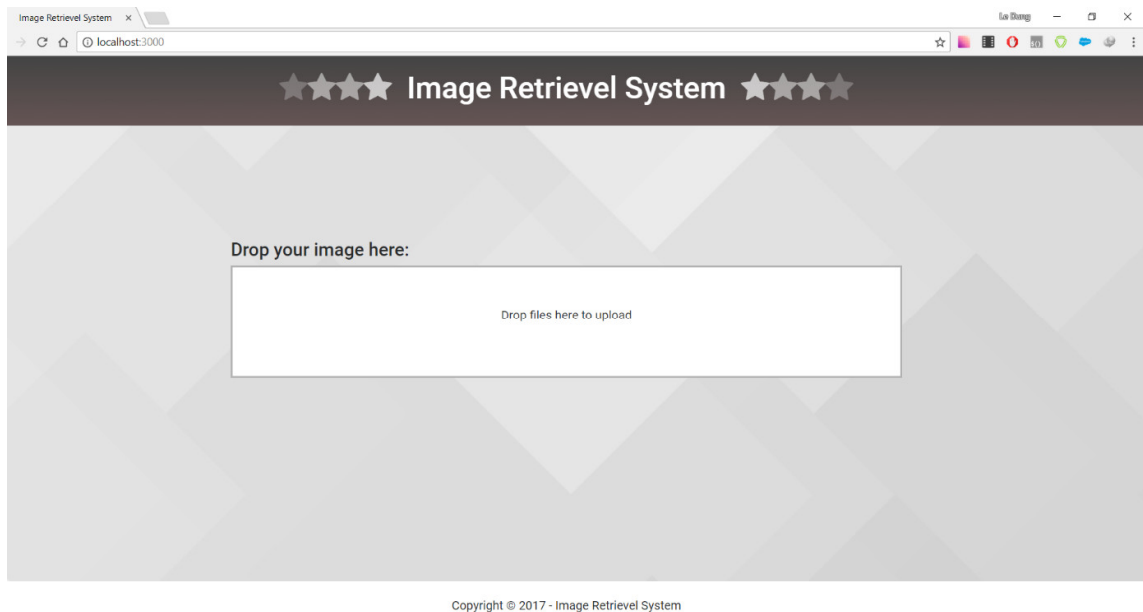
Website được cài đặt bằng NodeJS với các plugin jQuery (Jcrop) để crop ảnh.

Để kết nối đến server Python, cần cung cấp các cài đặt trong file config.json

- **pyAPI.search**: link API đến server Python để truy vấn ảnh, giá trị default là `http://localhost:5000/search`
- **pyAPI.upload**: link API đến server Python để upload ảnh, giá trị default là `http://localhost:5000/file-upload`
- **externalUpload**: cho phép upload ảnh lên server Python, giá trị default là `false`
- **searchByAjax**: cho phép gọi ajax thay vì submit ảnh lên server Python, giá trị default là `true`

Chạy lệnh (localhost: 3000):

npm run watch



7. Demo chương trình

Link: <https://youtu.be/vkHdFcv3QfA>

PHẦN 2: VẤN ĐỀ MỞ RỘNG

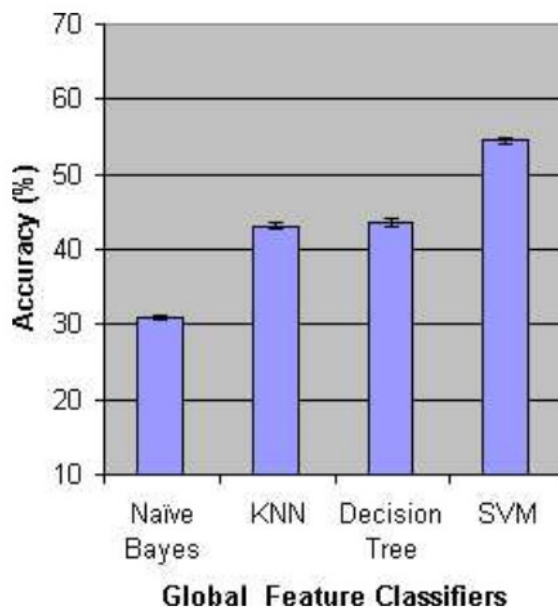
Phương pháp kết hợp local feature và global feature trong bài toán nhận dạng thị giác Object Class Recognition

I. Global feature

Global feature mô tả bức ảnh như một tổng thể để khái quát toàn bộ đối tượng mà trong đó các local features mô tả những phần của bức ảnh. Global features gồm các mô tả về đường biên (contour representations), hình dạng (shape descriptors), và đặc điểm cấu trúc (texture features). Ví dụ: Shape Matrices, Invariant Moments (Hu, Zerinke), Histogram Oriented Gradients (HOG) và Co-HOG

Có rất nhiều hệ thống nhận dạng ảnh sử dụng global feature để mô tả nội dung một bức ảnh. Bởi vì chúng tạo ra các hình ảnh rất nhỏ gọn, ở đó mỗi hình ảnh tương ứng với một điểm trong một không gian đặc trưng cao. Kết quả là, bất kỳ bộ phân loại chuẩn nào cũng có thể được sử dụng.

Tuy nhiên, phần lớn các global features thường giả định rằng mỗi hình ảnh chỉ chứa một loại đối tượng. Do đó chúng rất hạn chế trong trường hợp bức ảnh chứa nhiều loại đối tượng được sắp xếp trộn lẫn



II. Local feature

Local feature bao gồm những descriptor mô tả một phần cục bộ của bức ảnh được tính toán dựa trên nhiều interest point.

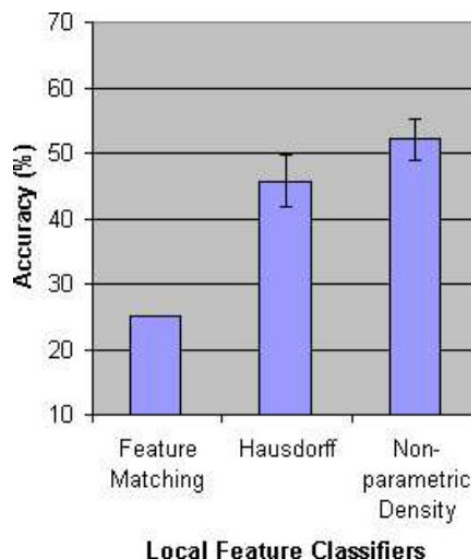
Local feature đề cập đến một mẫu hoặc cấu trúc riêng biệt được tìm thấy trong ảnh, chẳng hạn như một điểm, cạnh hoặc vùng ảnh nhỏ. Những đặc trưng đại diện không quan trọng, chỉ là nó là khác biệt với môi trường xung quanh của nó. Ví dụ: SIFT, SURF, LBP, ORB...

Local feature được sử dụng theo hai cách cơ bản:

- Để khoanh vùng các điểm neo để sử dụng trong khâu hình ảnh hoặc tái tạo 3 chiều.
- Để mô tả các nội dung hình ảnh một cách gọn nhẹ để phát hiện hoặc phân loại, mà không yêu cầu phân chia hình ảnh.

Một ưu điểm của việc sử dụng các local feature là chúng có thể được sử dụng để nhận ra đối tượng mặc dù có sự lộn xộn và tắc nghẽn đáng kể. Chúng cũng không đòi hỏi sự phân chia của đối tượng từ nền, không giống với các đặc trưng về kết cấu hoặc hình dạng

Local feature thường được dùng trong các hệ thống so khớp (matching), đòi hỏi việc so sánh các bộ mô tả đặc trưng



III. Kết hợp local feature và global feature trong bài toán nhận dạng

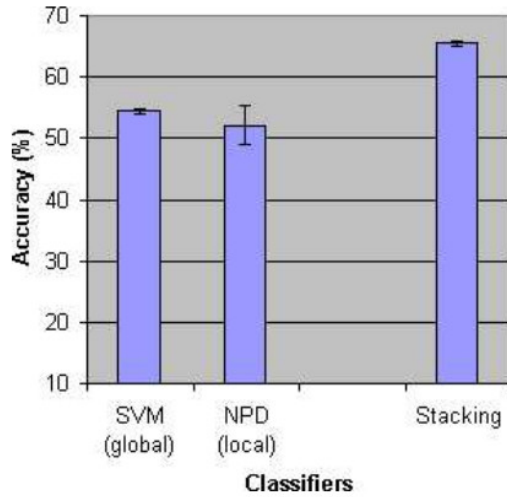
Việc sử dụng global feature hay local feature đều đem lại hiệu quả cao tùy thuộc vào yêu cầu bài toán. Tuy nhiên ở phần này, em sẽ trình bày một nghiên cứu kết hợp cả global và local features. Có hai phương pháp để tiếp cận. Thứ nhất là phương pháp xếp chồng xếp cổ điển (Stacking) và thứ hai sử dụng phân cấp phân loại (Classification Hierarchy). Cả hai đều cải thiện đáng kể kết quả so với các phương pháp sử dụng riêng lẻ global feature hay local feature.

1. Stacking

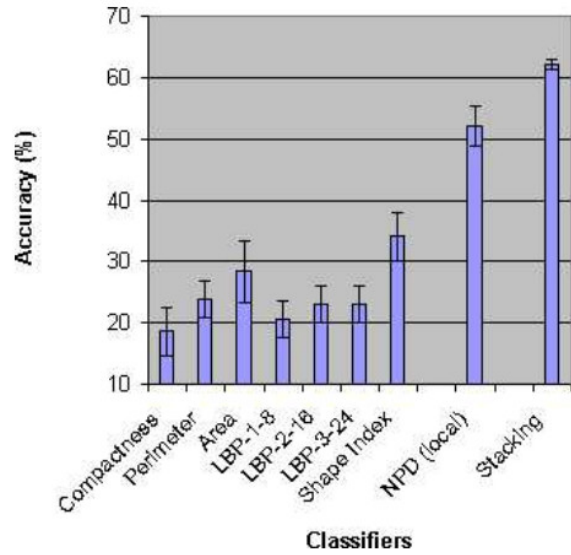
Đối lập với các kỹ thuật trong đó một chiến lược cố định được sử dụng, các kỹ thuật học siêu (meta-learning) sử dụng một bộ phân lớp meta tổng quát trong không gian của các kết quả đầu ra từ các bộ phân lớp cơ sở. Trong phương pháp Stacking các đầu ra của các phân lớp cấu thành được nối và được sử dụng như một vector dữ liệu đầu vào cho một bộ phân lớp meta.

Trong phần này thảo luận về hai biến thể chính của Stacking. Phần đầu tiên, đầu vào của bộ phân lớp meta là sự kết hợp các nhãn lớp do mỗi bộ phân lớp thành phần tạo ra. Trong biến thể thứ hai, mỗi bộ phân lớp thành phần cho ra một sự phân phối hậu nghiệm trên các nhãn lớp, chứ không phải là một nhãn duy nhất. Sự phân phối từ các bộ phân lớp thành phần được ghép nối và được sử dụng làm đầu vào cho bộ phân lớp meta. Phối hợp với các phân phối xác suất, về bản chất, ước lượng độ tin cậy phân loại từ bộ phân lớp cấp cơ sở. Bất kỳ phân lớp nào cũng có thể được sử dụng ở cấp cơ sở nếu chỉ yêu cầu một nhãn đơn lẻ, nhưng với Stacking sự phân phối xác suất sẽ giới hạn với các bộ phân lớp có phân phối trên nhãn. Sự lựa chọn của phân lớp meta không bị giới hạn dưới bất kỳ hình thức nào.

Xét một thí nghiệm ứng dụng Stacking, sử dụng SVM làm bộ phân lớp meta, thử nghiệm với hai biến thể của bộ phân lớp cơ sở cho các global features. Trong lần đầu tiên, thí nghiệm đánh giá mật độ phi tham số để xây dựng một số bộ phân lớp maximum-likelihood sử dụng các global features. Sử dụng phương pháp Stacking để kết hợp các features, kết quả đạt được độ chính xác 50,32%. Kỹ thuật thứ hai sử dụng bộ phân lớp SVM ở cấp cơ sở. Bộ phân lớp meta lấy một vector bao gồm một số phần tử trả về bộ phân lớp cấp cơ sở. Độ chính xác cho các thí nghiệm này được tóm tắt trong hình



(a) Stacking using a SVM global feature classifier, and a NPD local feature classifier.



(b) Stacking using many NPD global feature classifiers, and a NPD local feature classifier

2. Classification Hierarchy

Do các local features và global features cung cấp nhiều loại thông tin khác nhau về một hình ảnh, có thể tồn tại một cặp lớp không thể tách rời trong không gian global feature sẽ được phân biệt bằng các local feature. Trong phần này sẽ giới thiệu về một hệ thống phân lớp hai cấp (2-tier hierarchical classification system) sử dụng cả global và local features.

Ở cấp độ cao nhất, các lớp không thể tách rời bằng các global features được hợp nhất thành các lớp bậc cao (super-classes). Sau đó bộ phân lớp global (global feature classifier) được đào tạo về các lớp này. Một phân lớp local (local feature classifier) được đào tạo để phân biệt giữa các lớp gốc có trong mỗi lớp bậc cao (superclass). Khi một hình ảnh truy vấn được phân loại là thuộc về một lớp bậc cao, nó được chuyển đến bộ phân lớp local, nó sẽ xác định xem lớp nào thuộc về hình ảnh. Lý do giải thích cho điều này là ở cấp cao nhất các hình ảnh được phân loại thành các nhóm rộng hơn, dễ phân biệt hơn, và ở cấp độ dưới cùng bộ phân lớp sẽ phải làm việc với ít lớp hơn. Bộ phân loại global được sử dụng ở cấp cao nhất vì nó nhanh hơn, dẫn đến tăng tốc cho toàn bộ hệ thống.

Áp dụng phương thức kết hợp này, sử dụng bộ phân lớp có hiệu suất tốt nhất cho các global features (SVM) và local features (NPD).

Table 1: Confusion matrix for SVM with global features

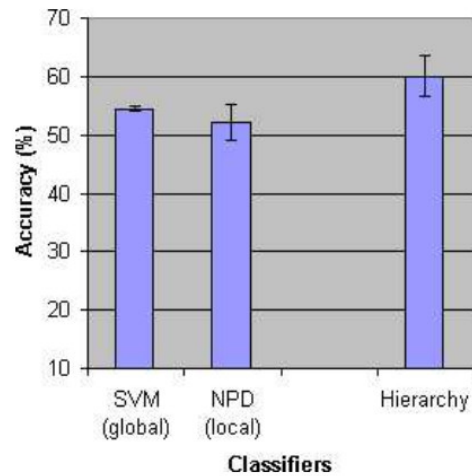
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	21	5	3	1	1	2	0	1	2	96	0	0	0	1
2	4	33	0	0	6	3	0	1	3	30	0	5	1	0
3	3	2	22	0	0	0	11	1	1	54	3	0	1	2
4	3	2	0	10	8	1	0	0	1	2	0	7	0	0
5	1	4	0	2	94	2	0	0	0	5	0	10	13	0
6	1	2	1	0	11	4	1	0	0	39	1	1	7	0
7	0	3	11	0	0	1	29	0	3	70	21	0	1	3
8	3	3	2	0	0	0	0	83	2	1	1	1	0	1
9	5	0	1	2	2	0	3	0	89	24	3	1	0	3
10	12	11	10	0	1	2	21	3	6	339	16	0	4	8
11	0	0	3	0	0	0	16	1	2	18	67	0	0	1
12	1	13	1	5	11	0	1	6	3	5	0	155	1	0
13	3	0	0	0	19	5	0	0	0	23	0	1	30	0
14	4	2	2	0	0	0	2	1	17	33	5	1	1	10

Table 2: Confusion matrix for NPD with local features

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	18	0	0	0	17	0	0	0	0	41	0	56	1	0
2	2	13	0	0	8	0	1	0	1	8	0	50	1	1
3	1	0	47	2	4	0	7	0	7	24	0	7	0	0
4	0	0	0	0	15	1	0	0	0	2	0	16	0	0
5	0	0	0	0	87	0	0	0	0	2	0	39	2	0
6	1	0	1	0	23	6	0	0	0	12	1	20	4	0
7	3	0	10	0	6	0	74	0	4	27	2	10	3	2
8	0	0	0	0	0	0	0	89	0	0	0	8	0	0
9	4	0	0	0	6	1	3	0	57	29	3	24	0	6
10	7	1	5	1	29	2	14	0	3	274	0	79	16	0
11	1	0	1	0	0	0	3	0	1	12	88	1	0	1
12	0	0	0	0	7	0	0	21	0	1	0	173	0	0
13	0	0	0	0	42	0	0	0	0	3	0	28	8	0
14	1	1	1	0	3	0	0	0	8	21	1	28	0	14

Các nhóm được xây dựng bằng cách kết hợp hai lớp A và B một cách liên tục, sao cho tỷ lệ phần trăm các trường hợp của A được phân loại B là cao nhất. Quá trình này dừng lại, khi phần trăm rơi xuống dưới ngưỡng.

Thủ tục hợp nhất trong tập dữ liệu VPR tạo ra hai lớp bậc cao, bao gồm 2 và 4 lớp. Độ chính xác tổng thể đã tăng lên 60%



(c) Hierarchical classification using SVM global feature classifier, and a NPD local feature classifier.

IV. Kết luận

Thông qua các kết quả thực nghiệm, việc sử dụng kết hợp các global features và local features phân nào đã giúp giảm tỉ lệ sai sót hơn 20% trong các bài toán nhận dạng đối tượng.

Hiện nay phương pháp này vẫn còn một số hạn chế như việc cài đặt và huấn luyện phức tạp hơn so với các phương pháp trước, giới hạn trong bài toán classification và detection cũng như chi phí thiết kế cao hơn. Tuy nhiên phương pháp này là một có thể xem là một cải tiến đáng kể trong Computer Vision hay Machine Learning nói chung.

V. Tài liệu tham khảo

- [1]. Jobin Wilson, Muhammad Arif. Scene Recognition by Combining Local and Global Image Descriptors, February 2017
- [2]. Dimitri A. Lisin, Marwan A. Mattar. Combining Local and Global Image Features for Object Class Recognition, 2005
- [3]. A. K. Seewald. Towards Understanding Stacking - Studies of a General Ensemble Learning Scheme. PhD thesis, Austrian Research Institute for Artificial Intelligence (FAI), 2003

TÀI LIỆU THAM KHẢO

- [1]. Slide bài giảng môn Tìm kiếm thông tin thị giác
- [2]. Slide bài giảng môn Nhận dạng thông tin thị giác
- [3]. TF-IDF and L2-norm. Link: <http://blog.christianperone.com/2011/10/machine-learning-text-feature-extraction-tf-idf-part-ii/>
- [4]. Implementing RootSIFT in Python and OpenCV. Link: <https://www.pyimagesearch.com/2015/04/13/implementing-rootsift-in-python-and-opencv/>
- [5]. K Means Algorithm. Link: <http://stanford.edu/~cpiech/cs221/handouts/kmeans.html>
- [6]. Jcrop jQuery (crop ảnh). Link: <https://github.com/tapmodo/Jcrop>
- [7]. B-lazy.js jQuery (tích hợp lazy load cho ảnh kết quả). Link: <http://dinbror.dk/blazy/>
- [8]. Dropzone jQuery (cung cấp giao diện upload ảnh). Link: <http://www.dropzonejs.com/>
- [9]. <http://zeus.robots.ox.ac.uk/oxfordbuildings/>
- [10]. https://en.wikipedia.org/wiki/Hellinger_distance

MỘT SỐ SOURCE CODE THAM KHẢO

- [1]. <https://www.pyimagesearch.com/2014/12/01/complete-guide-building-image-search-engine-python-opencv/>
- [2]. <https://github.com/devashishp/Content-Based-Image-Retrieval>
- [3]. <https://gist.github.com/anabranh/48c5c0124ba4e162b2e3>