Activity Based Image Retrieval

Progress Report

In fulfilment of the requirements for the

NU 302 R&D Project

At NIIT University

Submitted by: -

- **Prakriti Pritmani**
- **Shubham Sethi**
- **Nizampatnam Mounish**
- **Shashwat Shah**
- **Preetham Venkata Paritala**

Area

NIIT University

Neemrana

Rajasthan

R&D Project

Student Names:

S. No.	Enrollment No.	Student Name
1	U101116FCS090	Prakriti Pritmani
2	U101116FCS124	Shubham Sethi
3	U101116FCS185	Nizampatnam Mounish
4	U101116FCS112	Shashwat Shah
5	U101116FCS093	Preetham Venkata Paritala

Name of the Project:

Activity Based Image Retrieval

Name of the Mentor:

Dr. Prashant Srivastava

Summary of the Project:

Our Project aims to retrieve similar images of the action performed in the query image. It incorporates a technique with the highest accuracy in the non-neural network techniques list. We have tried and tested up to 3 various methods and the best so far has been HOG (Histogram of Oriented Gradients). We have theorized that HOG along with Adaptive Thresholding and Canny Edge detection will be the most accurate method. And, for similarity measurement Chi squared distance outstands the rest. This report will discuss the various aspects and innovations proposed. The field of our topic is niche, so we wish to contribute in any way possible.

Table of Contents

CERTIFICATE

This is to certify that the present research work entitled "Activity Based Image Retrieval" being submitted to NIIT University, Neemrana, Rajasthan, in the fulfillment of the requirements for the course at NIIT University, Neemrana, embodies authentic and faithful record of original research carried out by Prakriti Pritmani, Shubham Sethi, Nizampatnam Mounish, Shashwat Shah, Preetham Venkata Paritala of B Tech (CSE) at NIIT University, Neemrana. She /He has worked under our supervision and that the matter embodied in this project work has not been submitted, in part or full, as a project report for any course of NIIT University, Neemrana or any other university.

Mentored by: - Dr. Prashant Srivastava

List of Figures:

Fig 1:

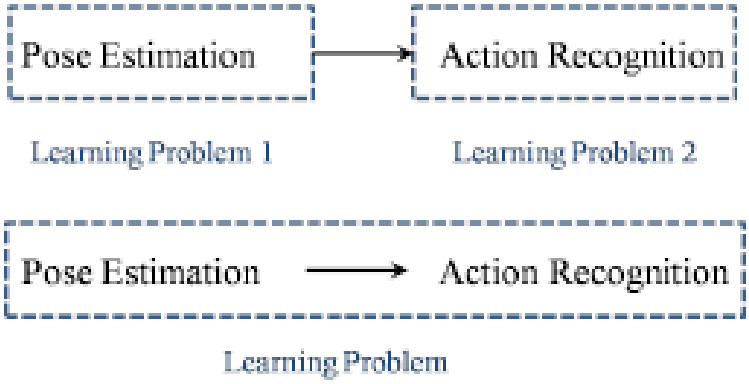


Fig 2:

Fig 3:

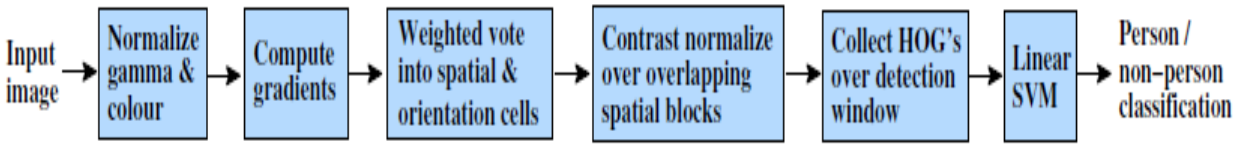


Fig 4:

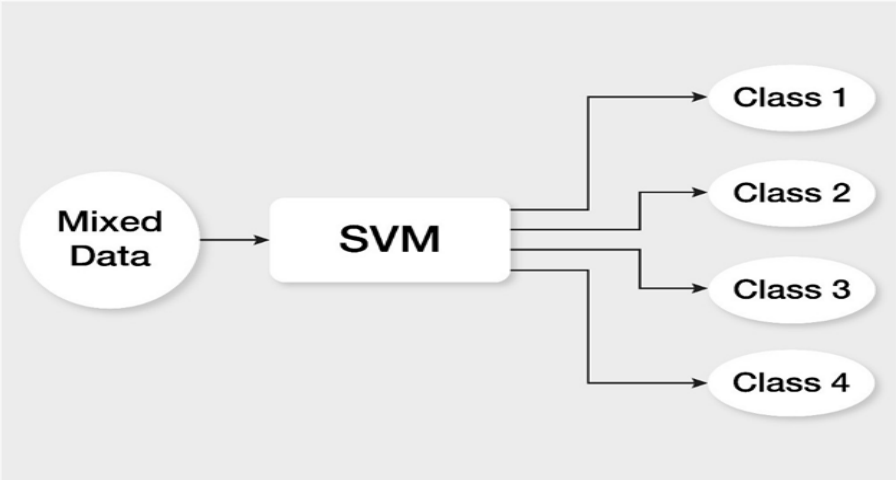


Fig 5:

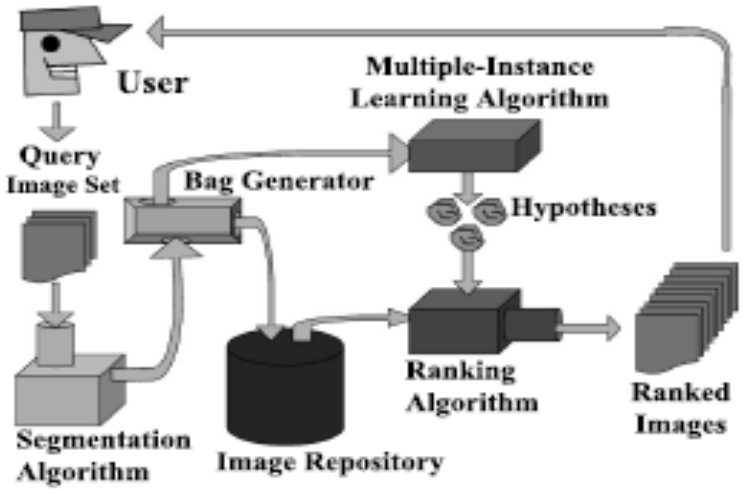


Fig 6:

Fig 7:

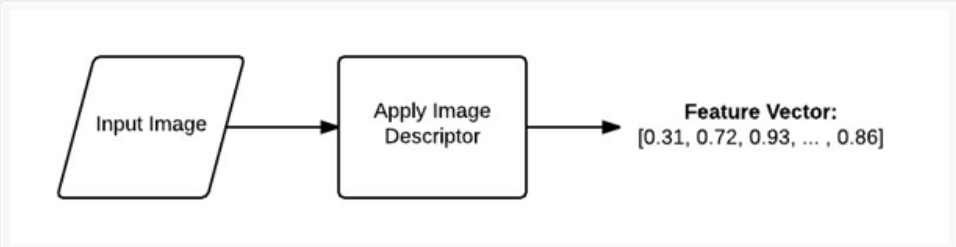


Fig 8:

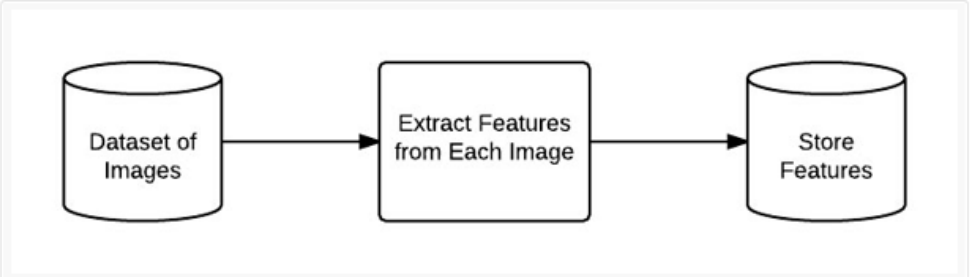


Fig 9:

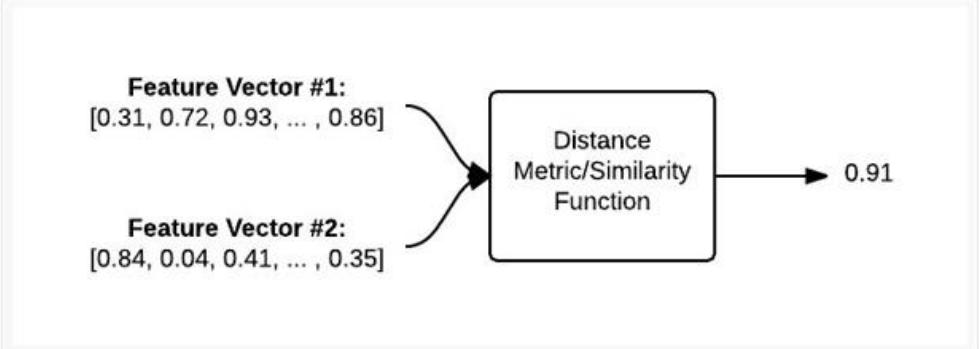
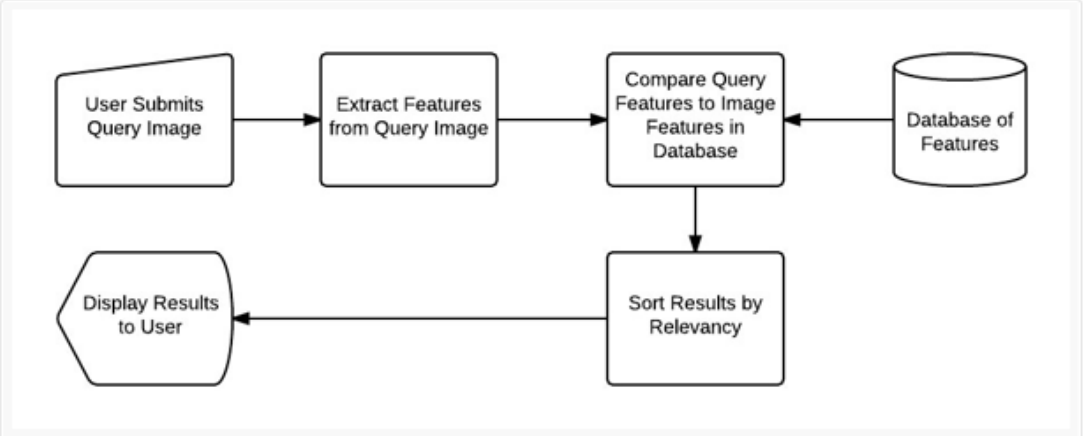


Fig 10:



Rationale of work

The Rationale of work of Activity-Based Image Retrieval project is to retrieve similar images which are related to the user-given an image of an action being performed by the human-object in the image. Activity-Based Image Retrieval is a niche version of Content-Based Image Retrieval a.k.a *CBIR*. CBIR has a vast field of methods or techniques which are used to retrieve the images from the dataset. In here Activity-based image retrieval is a small subset of the CBIR.

CBIR has been researched and developed upon since the early ’90s. The main idea of this work in this topic is to retrieve images based upon the annotate query or an image query. There were many types of research has been on this topic in the early ’90s but as we are in the 20th century there has been very less work has been done on the CBIR and comparatively less on Activity-Based image retrieval. As we work on this project, we will be adding more information and techniques for future work and self-reference. As we have seen that there is a very low contribution from researchers on this topic, this work by our will be a great contribution to this field.

Although early systems existed at the beginning of the ’80s and ’90s, the majority would recall systems such as IBM’s Query By Image Content as the start of content-based image retrieval. The commercial QBIC system is the most well-known system in an early time. Another system for images and video retrieval is Virage that has well known commercial customers such as CNN.

This work will be helpful in Data Collection and Web searching. When we have a huge amount of unorganized data but we need data which is related to one particular type, this work will add be of help to the other researching people who are working on this area. In Web searching also big company like GOOGLE SEARCH ENGINE and YAHOO SEARCH uses the same principle which is used in activity-based image retrieval.

Any search engine gives feedback of information based on the user query, similarly we retrieve feedback of images which are related to the user given query image. Therefore, our work will work as a contribution and help other researching people who are working towards this area. This work can be helpful for further research in Robotics area. Our Main objective is to give our side of contribution to get better image retrieval techniques which can give better results of feedback with a lower time of execution and which should be user-friendly so that we shouldn’t be spending more amount of time and more space to run them.

Review of literature

Recognizing Human Actions from Still Images with Latent Poses
By Weilong Yang, Yang Wang, and Greg Mori

The main objective of this research paper is to find out activity in a still image. They used the pose estimation technique as a piece of latent information to recognize activity as a novel method. Figure 1 discusses an overview. These are the 3 stages of this method to estimate pose from a given image, label an action for certain poses and combining both to get the result. In order to detect the presence of each poselet, they train a classifier for each poselet, use the standard linear SVM and the histograms of Oriented Gradients feature proposed by Dalal and Triggs and annotate them with any labels (running, dancing etc.).

Dataset they tested consists of still images from five action categories: running, walking, playing golf, sitting, and dancing (Ikizler-Cinbis) with 2458 images in it. They trained 90 poses to recognize the activity. As seen in Fig 2.

these are some of the poselets of their algorithm, so this leads to getting a better pose estimation algorithm and we need to annotate a number of poselets to get desired output.

Histograms of Oriented Gradients for Human Detection
By Navneet Dalal and Bill Triggs

In this research paper they have mentioned that human detection using HOG Histograms of Oriented Gradient is more accurate than adopting linear SVM for human detection in still image.
They have performed various performance parameters concluding that fine-scale gradients, fine orientation binning, relatively coarse spatial binning, and high-quality local contrast normalization in overlapping descriptor blocks are all important for good results. They have used more challenging dataset containing over 1800 annotated human images with a large range of pose variations and backgrounds. Fig 3 talks about this technique.
An overview of feature extraction and object detection chain. The detector window is tiled with a grid of overlapping blocks in which Histogram of Oriented

Gradient feature vectors are extracted. The combined vectors are fed to a linear SVM for object/non-object classification.

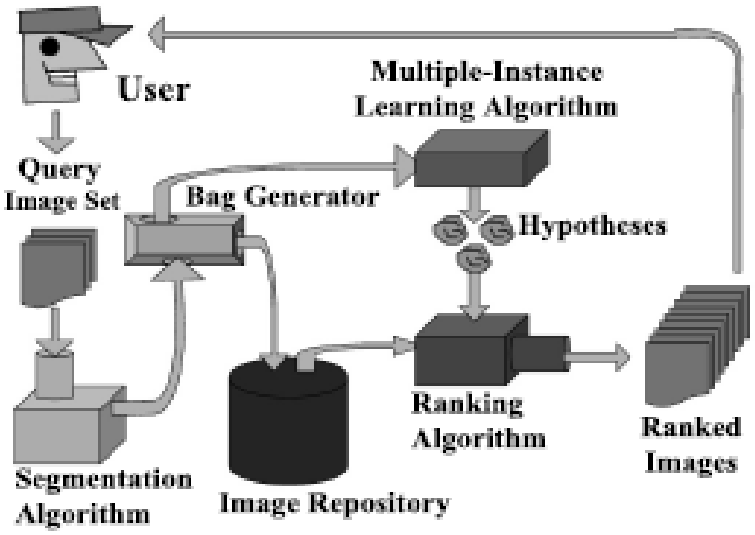
The HOG-based detectors greatly outperform the wavelet, PCA-SIFT and Shape Context ones, giving near-perfect separation on the MIT test set.
In their standard detector, each HOG cell appears four times with different normalizations and including this 'redundant' information improves performance from 84% to 89% at 10−4 FPPW. They have shown that using locally normalized histogram of gradient orientations features similar to SIFT descriptors in a dense overlapping grid gives very good results for person detection.

Localized Content Based Image Retrieval
By Rouhollah Rahmani, Sally A. Goldman, Hui Zhang, John Krettek, and Jason E. Fritts

In this paper it tells how Localized Content Based Image Retrieval works.
Classic Content-Based Image Retrieval (CBIR), when using a query-by-example interface, takes a single query image provided by the user, and retrieves similar images from an image repository. The query image set can either be directly provided by a user or can be obtained using relevance feedback by adding labeled feedback images to the original query image (labeled as positive).

In this they used Accio's ranking algorithm is general enough for the situation in which the desired images are defined by many different objects.

Below diagram show the flow of the IHS algorithm, uses a bottom-up approach that can make use of any desired similarity measure between two segments.



In this algorithm on an image of a bird flying past a mountain which has a road leading to it. The segment for the mountain selects the sky as its northern, western, and eastern neighbor. It ignores the clouds, and the bird. The trees are selected as the southern neighbor, and the smaller road is ignored. The segment for the bird, selects the mountain and the sky as its neighbors. So, a mountain is defined as being below the sky and above the trees. And a bird is defined as being an object surrounded by the sky or by a mountain.

Multiple-instance learning (MIL) is a supervised learning. The learner receives a set of labeled bags, each containing many instances, instead of receiving a set of instances which are individually labeled. In this simple a bag is labeled negative if all the instances in it are negative. On the other hand, if there is at least one instance in it which is positive, a bag is labeled positive. From a collection of labeled bags, the learner tries to (i) identify label with individual instances correctly or (ii) learn how to label bags.

A much more convenient and simpler example: Imagine several people, and each of them has a key chain that contains few keys. Some of these people will be able to enter a certain room, and some won't be. The task is to predict whether a certain key can get you into that room. To find the solution we need to find the exact key that is common for all the "positive" keychains. If we can correctly identify this key, we can also correctly classify an entire keychain - positive if it contains the required key, or negative if it doesn't. It is used in ranking of images in database.
The final stage of Accio! is to take the set of hypotheses returned by Ensemble-EMDD and combine them to rank the images in the image repository and return similar images.

Objectives

Our main objective is to recognize the activity being performed in the image and retrieve an accurate set of images of that activity from our dataset.

- Making better use of the limited resources available to us such as computing power and researched work and find an efficient **non-neural technique.**

- Develop a model for Activity-Based Image Retrieval that is accurate and reliable. Eliminate irrelevant results that are obtained from annotation-based systems used by most search engines.

- We will first take a single non-annotated query image from user and perform activity recognition followed by image retrieval from a large dataset with suitable algorithm.

- Our final objective is to get the best possible output with high accuracy using minimum lines of code. Finally, contribute to the field of Activity-Based Image Retrieval.

Methodology

Our main objective is to recognize the activity being performed in the image and retrieve an accurate set of images of that activity from our dataset.

The user is required to input a query image. The image is converted to grayscale and its edges are detected using Canny Edge Detection.

Canny Edge Detection consists of the following five steps:

1. **Noise Reduction:**

   Results of edge detection are very sensitive to noise present in the image because the calculation involves derivatives. To remove noise from the image, we use Gaussian Blur so as to smoothen it.

2. **Gradient Calculation:**

   In this step, we calculate gradient of the image using edge detection operators to detect the edge intensity and direction.

3. **Non-Maximum Suppression:**

   This is carried out to thin out edges of the image since the final image should have thin edges.

4. **Double Threshold:**

   Double thresholding helps us to divide the pixels into three categories, that is, strong, weak and non-relevant.

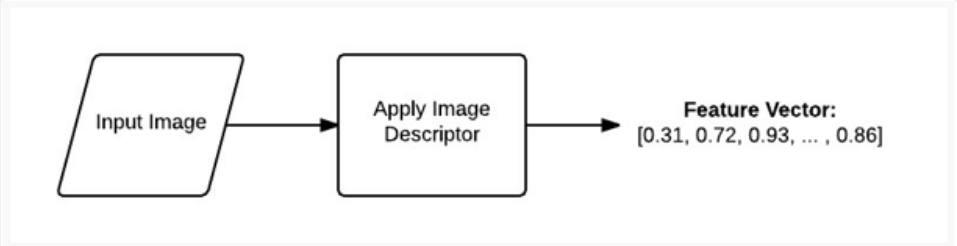
5. **Edge Tracking by Hysteresis:**

   This step aims at converting weak pixels to strong pixels only if the current pixel being processed has at least at least one strong pixel around it.

For any Activity Based Image Retrieval System, the following four steps have to be followed:

1) Decide the Image Descriptor:

A feature descriptor or an image descriptor is an algorithm which accepts an image as an input and gives its feature vectors as the output. These feature vectors are later used for comparison with the feature vectors of the images in our dataset to determine the similarity between the images.



Now we need to determine which aspect of an image we need to be described: shape of the object in the image, colour of the image or the texture.

To detect the activity being performed by the human being in the query image, we needed to determine the shape of the object (in our case a human being).

We studied and applied the following techniques to achieve the above-mentioned objective:

(a) HuMoments:

A moment in an image is a weighted average of the pixel intensities in an image. HuMoments are basically a set of seven numbers which are calculated using central moments. They are used to describe the shape as well as the structure of the object. We find the HuMoments function in OpenCV.

(b) Zernike Moments:

Zernike moments are also used for description of the shape of an object but there is no redundancy of data because they are orthogonal to each other. They can be found in the mahotas package.

We found Zernike moments better than HuMoments because it not only removes the redundancy but at the same time, it had a better accuracy than HuMoments. In our case, HuMoments had only 1 out of 10 images correctly retrieved whereas in case of Zernike moments, it was 3 out of 10 images, which is a 20% increase in the accuracy.

(c) Histogram of Oriented Gradients (HOG):

Histogram of Oriented Gradients (HOG) uses distribution of oriented gradients as features.

i. <u>Without Canny Edge Detection:</u>

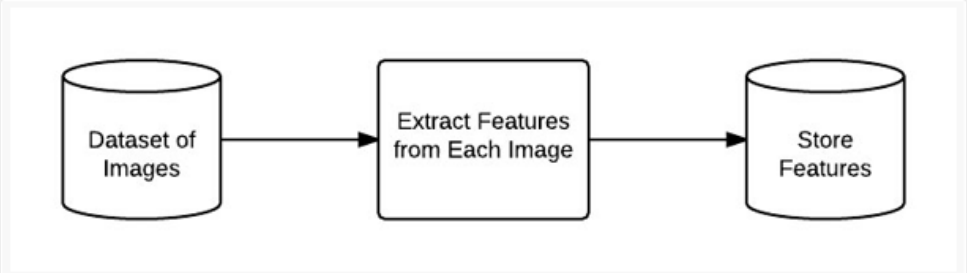
Applying HOG directly on the grayscale images, we retrieved 6 images correctly out of the 10 output images.

ii. <u>With Canny Edge Detection:</u>

When we used Canny Edge Detection followed by HOG, we were able to retrieve 8 out of 10 images correctly.

As it is clear from the above-mentioned facts, applying HOG on the images with Canny Edge Detection gave us the best possible results by far. Hence, HOG after Canny Edge Detection was our preferred technique.

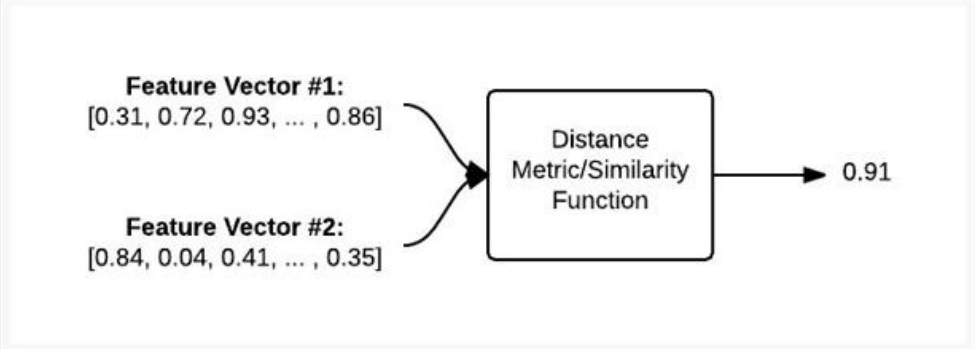
2) Index the Dataset:



After deciding our descriptor, this descriptor had to be applied to each image of our dataset. Then we had to perform feature extraction and store these features in a file so that they could be compared. We stored the extracted features in a CSV (Comma Separated Values) file named index.csv.

3) Decide the Similarity Metric:

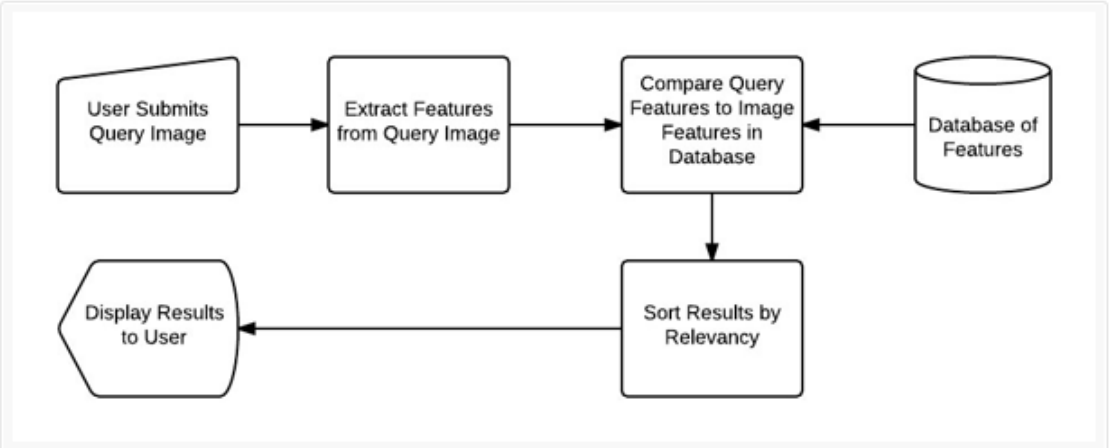
We have feature vectors stored with us now. Now we have to compare the feature vectors to find out the similarity between them.



Chi-squared distance was our preferred metric in this case. We used a function to calculate the chi-squared distance. It accepts the two histograms we wish to compare for their similarity as its two arguments.

4) Searching:

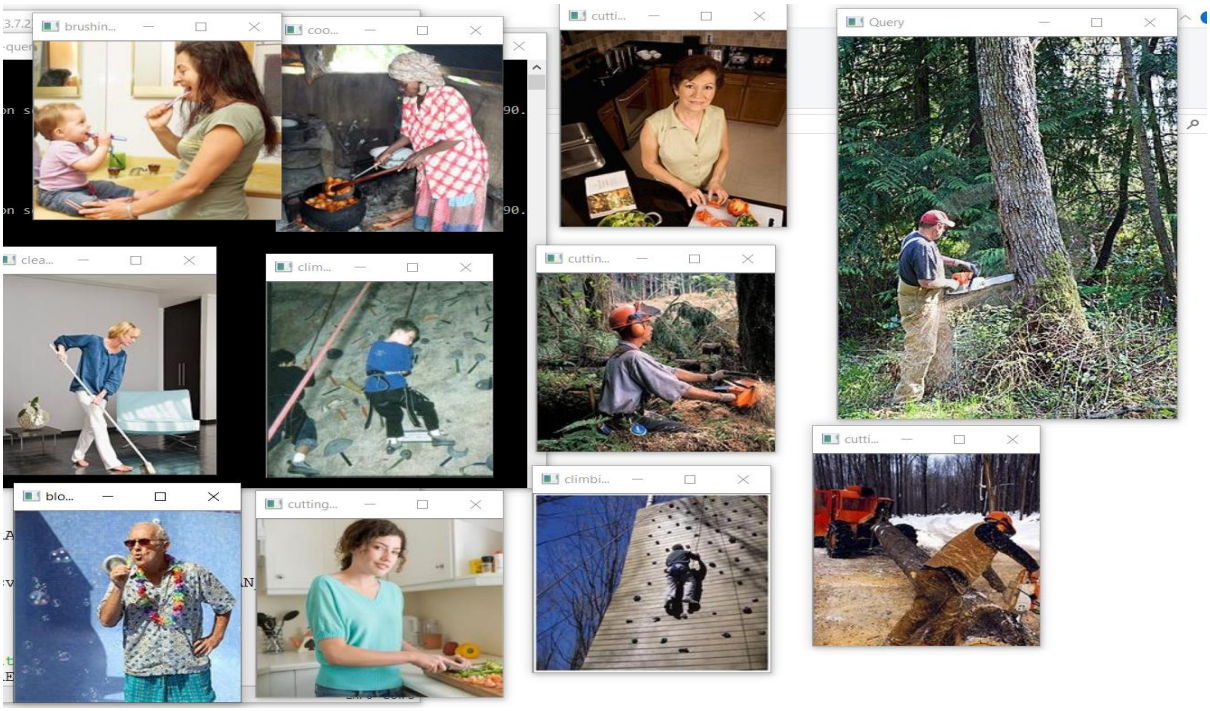
The last but the most important step is actually performing the search. The user is required to enter or give a query image, from which the feature vectors will be extracted. We then apply the chi-squared distance function, which is our preferred similarity metric to compare the feature vectors of the query image to the feature vectors of the images present in our indexed and labelled dataset and retrieve the most relevant images in terms of the similarity of the activity being performed by the human being in them.



Results of work completed

Zernike Moments

With Hu' moments (seven) the accuracy of the retrieval was far too low. After switching to Zernike moments (150) the accuracy of the retrieval increased significantly.



150 Zernike moments increased the accuracy from 1 to 3 accurate images out of 10. Cutting trees was the query image (top right).

Below, "moments" contain values of Zernike moment. For Hu' moments, *moments=cv2.moments(outline)* must be written instead of *desc.describe(outline).*

```
#image = cv2.copyMakeBorder(image, 15, 15, 15, 15,
 #       cv2.BORDER_CONSTANT, value = 255)
#thresh = cv2.bitwise_not(image)
#thresh[thresh > 0] = 255
#outline = np.zeros(image.shape, dtype = "uint8")
#cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
#        cv2.CHAIN_APPROX_SIMPLE)
#cnts = imutils.grab_contours(cnts)
#cnts = sorted(cnts, key = cv2.contourArea, reverse = True)[0]
#cv2.drawContours(outline, [cnts], -1, 255, -1)
#moments = desc.describe(outline)
#huMoments=cv2.HuMoments(moments)
#huMoments=[str(f)[1:-1] for f in huMoments]
```

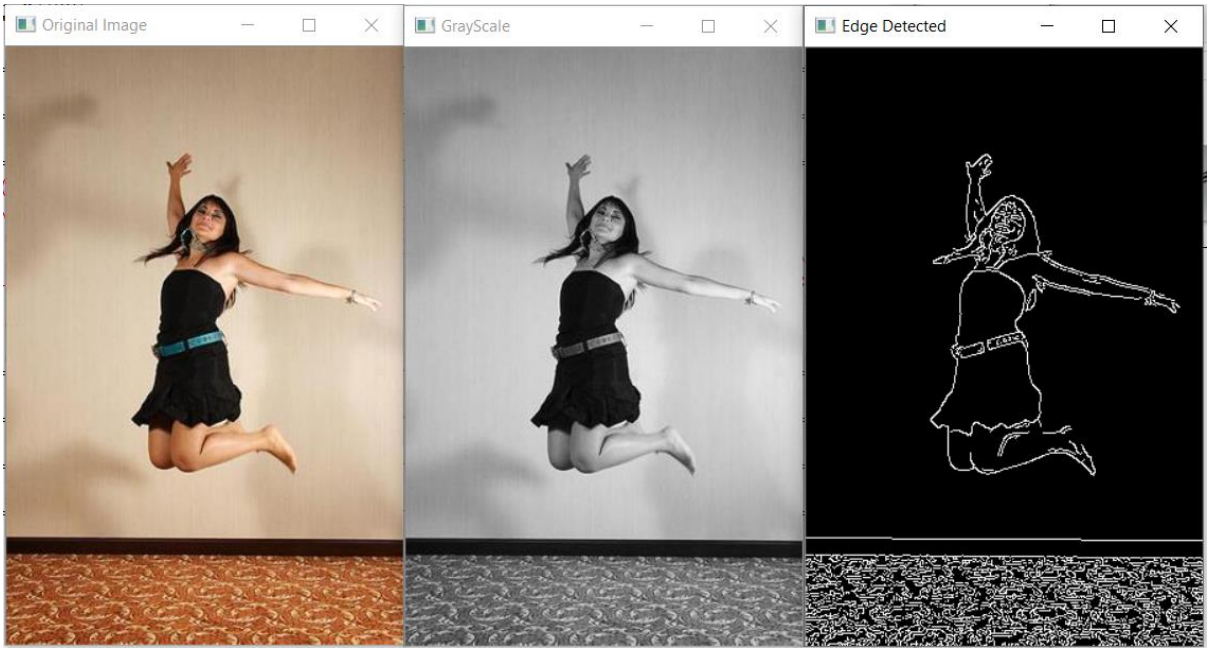
Histogram of Oriented Gradients

i) Canny Edge detection

The following code is the above technique plus adaptive thresholding:

```
    image = cv2.imread(imagePath)
    activity = imagePath[imagePath.rfind("/") + 1:]
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    image = cv2.Canny(image,100,200)
    image = cv2.adaptiveThreshold(image,255,cv2.ADAPTIVE_THRESH_MEAN
cv2.THRESH_BINARY,11,2)
```

According to the proposed methodology explained above, conversion to grayscale and detection of edges using Canny Edge Detection technique is shown below:



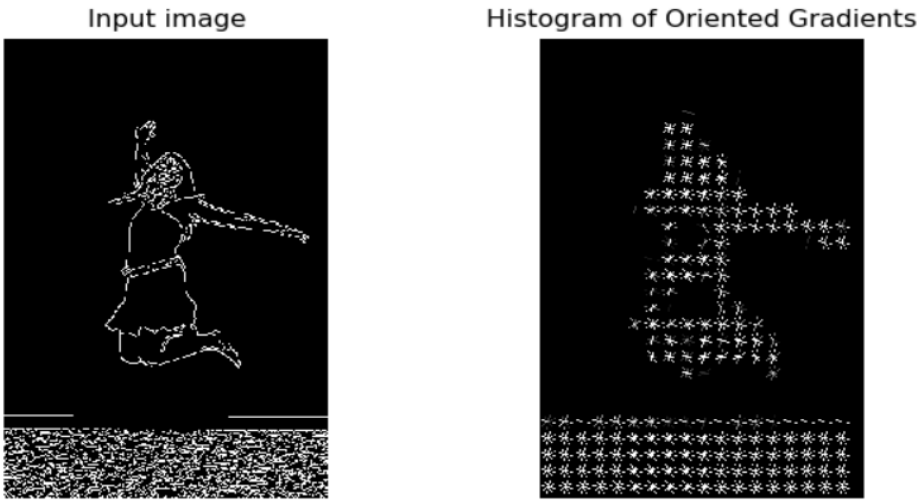
(a) Original image (b) Grayscale version (c) Edges detected

(ii) Applying HOG

The edge detected image is converted to HOG which is stored as a feature vector as follows:

```
fd, hog_image = feature.hog(image, orientations=8,
    pixels_per_cell=(16, 16),
    cells_per_block=(1, 1), visualize=True,
    multichannel=False)
```

The result of the above code as an array:



We do this for every picture in our dataset which is almost 9532 images and store the feature vector values in CSV (comma separated values) file.

1105	climbing\climbing_150.jpg	0.622424	0	0.370543	0	0.622424	0	0.296435	0	0.5
1106	climbing\climbing_151.jpg	0.5	0	0.5	0	0.5	0	0.5	0	0.5
1107	climbing\climbing_152.jpg	0.636317	0	0.272443	0	0.636317	0	0.340554	0	0.539813
1108	climbing\climbing_153.jpg	0.65842	0	0.725246	0	0.193653	0	0.054773	0	0.694147
1109	climbing\climbing_154.jpg	0	0	0	0	0	0	0	0	0
1110	climbing\climbing_155.jpg	0.566958	0	0.566958	0	0.566958	0	0.188881	0	0.522057
1111	climbing\climbing_156.jpg	0.55645	0	0.55645	0	0.55645	0	0.266627	0	0.570303
1112	climbing\climbing_157.jpg	0.55234	0	0.291141	0	0.55234	0	0.55234	0	0.544284
1113	climbing\climbing_158.jpg	0.5	0	0.5	0	0.5	0	0.5	0	0.5
1114	climbing\climbing_159.jpg	0	0	0	0	0	0	0	0	0
1115	climbing\climbing_160.jpg	0.5	0	0.5	0	0.5	0	0.5	0	0.5
1116	climbing\climbing_161.jpg	0	0	0	0	0	0	0	0	0
1117	climbing\climbing_162.jpg	0.558257	0	0.558257	0	0.558257	0	0.255045	0	0.546407
1118	climbing\climbing_163.jpg	0.566176	0	0.195787	0	0.566176	0	0.566176	0	0.5
1119	climbing\climbing_164.jpg	0.5	0	0.5	0	0.5	0	0.5	0	0.5
1120	climbing\climbing_165.jpg	0.57735	0	0.57735	0	0.57735	0	0	0	0.505309

Once we input the query image, all the above steps apply to the query image and we send the processed image to a searcher function.

```
    def search(self, queryFeatures, limit = 10):
            results = {}
            with open(self.indexPath, encoding='UTF-8') as f:
                    reader = csv.reader(f)
                    for row in reader:
                            features = [float(x) for x in row[1:]]
                            d = self.chi2_distance(features, queryFeatures)
                            results[row[0]] = d
                    f.close()
            results = sorted([(v, k) for (k, v) in results.items()])
            return results[:limit]
```

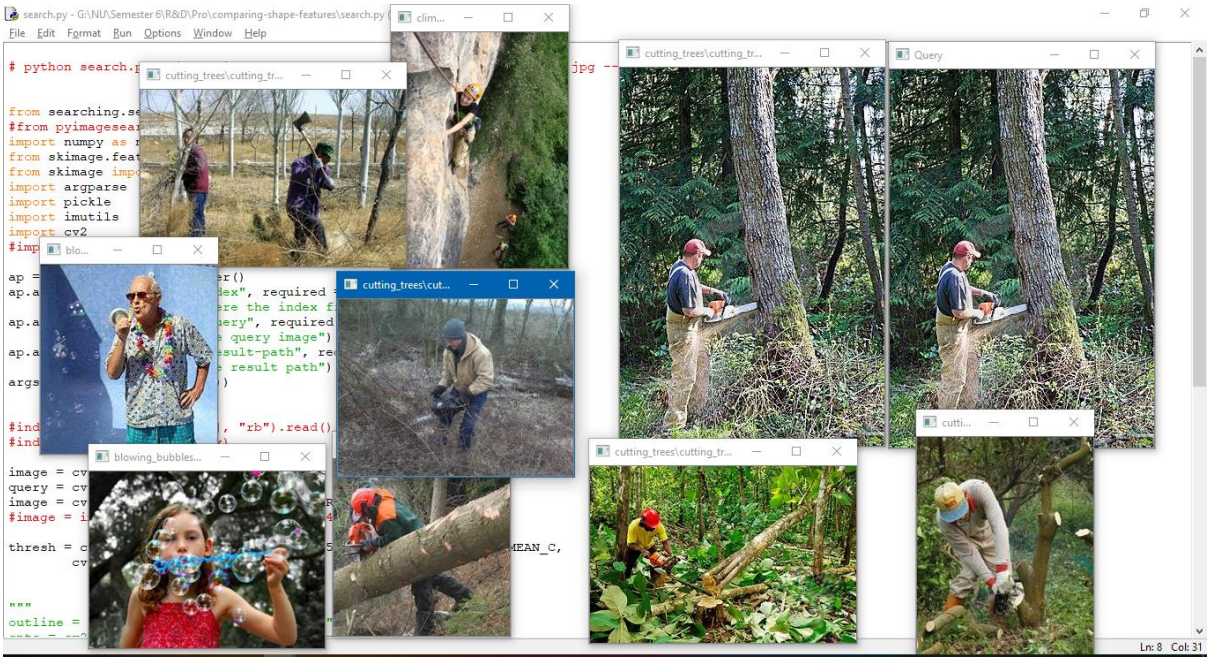
The above code snippet picks every feature vector stored and calculates similarity between the datasets and query image with a “chi squared distance” formula.

Notice the search function sorts the result into ascending order. This means that the most similar image with the least chi square distance will be the first ones in the result.

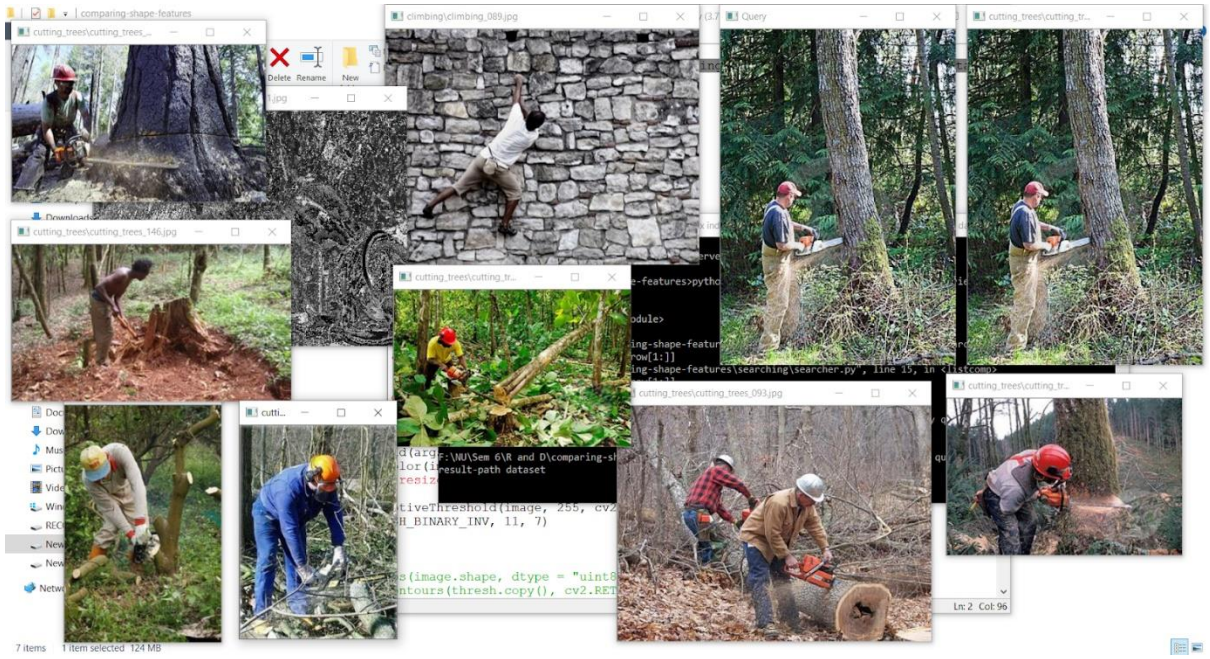
```
def chi2_distance(self, histA, histB, eps = 1e-10):
        d = 0.5 * np.sum([((a - b) ** 2) / (a + b + eps)
                for (a, b) in zip(histA, histB)])

        return d
```

Traversal of the top 10 results shall be done and displayed on the screen.

```
for (score, resultID) in results:
        result = cv2.imread("dataset\\"+resultID)
        cv2.imshow(resultID, result)
```



The above image is a result of HOG technique without Canny Edge detection.



The above image is a result of HOG technique with Canny Edge detection

Comparison of the accuracy of various techniques.	
Hu Moments	1/10 accurate images
Zernike Moments	3/10 accurate images
HOG without Canny Edge	6/10 accurate images
HOG with Canny Edge	8/10 accurate images

FUTURE OF WORK

Improve our algorithm to further increase the accuracy of the images being retrieved. With advanced computing, more accuracy can be attained. Using Histogram of oriented gradients (Hog) with canny edge our accuracy was about 80% but with more endeavor, we would come out with a better algorithm to increase accuracy rate. Since we decided to choose non-neural or machine learning techniques due to lack of skills and computation power our accuracy was a bit less. Techniques like Linear SVM would improve accuracy. Linear SVM is advanced and extreme speedy machine learning algorithm for solving multiclass classification problems from very-large data sets that implements an original proprietary version of a cutting plane algorithm for designing a linear support vector machine or (LSVM). It supports by performing image classification. For example, given an image from the user, the classification task is to classify whether the human is running or walking. We used an edge detection technique for recognizing the activity. But with the use of Linear SVM, we can improve efficiency in recognizing the activity.

Linear SVM entirely depends on by mapping large set of data to a high-dimensional feature space so that data points can be classified, even when the data are not otherwise linearly separable. A difference between the categories is found, then the data are transformed in such a way that the separator could be drawn as a hyperplane.

Another thing which we would like as a team to in future is to improve the speed of the algorithm by reducing the time it takes to traverse through the dataset for feature matching. This can be done by using more efficient computers or increasing the random-access memory of it and using AMD razon processor for the same. Size of the RAM decides how much of dataset you can hold in memory. For Deep learning applications to work, it is suggested to have a minimum of 16GB memory (Jeremy Howard Advises to get 32GB). Regarding the Clock, The higher the better. It ideally explains that the Speed — Access Time but a minimum of 2400 MHz is advised.

Currently, with our system, we are able to process 2000 images from our dataset at a speed of five minutes. Due to the lack of system requirements, our system gets overload with a lot of work which causes the crashing of the system. With more powerful and advanced system our model can work on more images from dataset.

We would like to implement it in real time for public use for data collection in Home entertainment and Advertisements. With the rise in the internet, many people believe in buying smart televisions. Millions of smart TVs in are collecting data. Then with the data analysis software with the combination of image recognition the search engine will recommend more of the stuff that the user has watched. It can be also be implemented in Web Searching and Robotics. We can train the robots with the amount of dataset we have to help them to recognize the activity being performed and can also learn the same on itself to do some basic human activity like running, walking, standing, sitting and lot more.

References:

- G. Guo and A. Lai, "*A survey on still image based human action recognition*", Pattern Recognition, vol. 47, no. 10, pp. 3343-3361, 2014.

- S. Venkatesha and M. Turk, "*Human Activity Recognition Using Local Shape Descriptors,*" 2010 20th International Conference on Pattern Recognition, Istanbul, 2010, pp. 3704-3707.

- W. Yang, Y. Wang and G. Mori, "*Recognizing human actions from still images with latent poses*," 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, 2010

- R. Rahmani, S. A. Goldman, H. Zhang, S. R. Cholleti and J. E. Fritts, "*Localized Content-Based Image Retrieval,*" in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 11, pp. 1902-1912, Nov. 2008.

- B. Singh and M. Aggarwal, "*A Review On Content Based Image Retrieval*", IJCT, vol. 17, issue no. 2, pp. 7226-7235,

- A. Khokher and R. Talwar, *Content-based Image Retrieval: Feature Extraction Techniques and Applications*, in International Conference on Recent Advances and Future Trends in Information Technology, Proceedings Published in International Journal of Computer Applications, PP 9-14.

- Navneet Dalal, Bill Triggs. Histograms of Oriented Gradients for Human Detection. International Conference on Computer Vision & Pattern Recognition (CVPR '05), Jun 2005, San Diego, United States. pp.886—893

- Veltkamp, R. C., & Tanase, M. (2002). A Survey of Content-Based Image Retrieval Systems. Content-Based Image and Video Retrieval,47-101. doi:10.1007/978-1-4615-0987-5_5

- Sigal L. (2014) Human Pose Estimation. In: Ikeuchi K. (eds) Computer Vision. Springer, Boston, MA - 05 February 2016. Doi: 978-0-387-31439-6