Using Haar Cascade Classifiers to build a Reverse Image Search Engine of Cartoon Characters

L Pranau Kumar, Somdutta Ukil, Ayush Roy

SCOPE, VIT University

Abstract

In this paper, we aim to build an image search engine that identifies a cartoon character and the corresponding television series from a database of cartoon characters for a given input image. For this, we propose the use of multiple Haar cascade classifier objects of different characters' features that are used in succession to recognise a character. The television is then determined by doing a simple database lookup for the detected character. Existing methods of building reverse image search engines include tagging images either automatically or manually, using machine learning algorithms to detect features in the image or doing it manually. The training of the classifier was done using verified positive samples and tested against unlabelled samples from the internet. The method we propose is faster on an average to detect the features of the character from the image and more accurate compared to previous methods of building the search engine.

Keywords- *Object recognition; Haar Classifiers; Image Search Engine*

1 Introduction

For avid watchers of animated content such as Japanese anime, it is difficult to keep track of the various characters that are introduced and to what series they belong to. We come across various pictures that feature unknown animated faces on the internet but looking at that picture increases our curiosity of knowing more about the series. This motivated us to build a system that does this automatically for us. We decided to train a classifier based on positive images of a cartoon character and make it detect that character in any image. We then extended this concept by training more classifiers for each character. Next, we built a database of these characters. Next, we took sample from the internet and passed it to the database which checked the image against the classifier and returned the character and the television series that character belonged to. What we present in this paper is but a proof of concept. We believe that the promising results will enable us to extend the database further and make further improvements to the process of searching, detecting and returning the information.

2 Literature Survey

Our application in this paper is to build a reverse search engine. We carried out a literature survey and found these existing works where a few authors have proposed image search engines and others have successfully used Haar cascade classifiers in various other applications.

"ECG Image Classification in Real time based on the Haar-like Features and Artificial Neural Networks" [1] by B. Mohammed et al.

The paper presents a ECGs classification system that uses powerful algorithms image processing and artificial intelligence. The descriptor Haar-like is based on the concept of the integral image to accelerate the calculation of Haar features and the classifier multilayer perceptron type. The training and testing of the proposed system were performed on two basic types: a learning base containing labelled data (normal ECG and ECG sick) and another base unlabelled data. The experimental results have shown that the system combines between the respective advantages of Haar-like descriptor and artificial neuron networks in terms of robustness and speed.

"Fast pedestrian detection system with a two-layer cascade of classifiers" [2] by Ying-Che et al.

This work presents a novel pedestrian detection system that uses Haar-like feature extraction and a covariance matrix descriptor to identify the distinctions of pedestrians. An approach that adopts an integral image is also applied to reduce the computational loads required in both the Haar-like feature extraction and evaluation of the covariance matrix descriptor.

"Automatic Craniofacial Anthropometry Landmarks Detection and Measurements for the Orbital Region" [3] by S. Mohammed et al.

Facial landmarks detection is undoubtedly important in many applications in computer vision for example the face detection and recognition. In craniofacial anthropometry, consistent landmarks localization as per standard definition of the craniofacial anthropometry landmarks is very important in order to get accurate craniofacial anthropometry data. In this article we demonstrated an automatic detection of craniofacial anthropometry landmarks at the orbital region.

"An algorithm for accuracy enhancement of license plate recognition" [4] by Zheng et al.

This paper presents an algorithm for extraction (detection) and recognition of license plates in traffic video datasets. For license plate detection, we introduce a method that applies both global edge features and local Haar-like features to construct a cascaded classifier consisting of 6 layers with 160 features. The characters on a license plate image are extracted by a method based on an improved blob detection algorithm for removal of unwanted areas.

"Evaluation of Haar cascade classifiers designed for face detection." [5] by R. Padilla et al.

The human face contains important features that can be used by vision-based automated systems in order to identify and recognize individuals. Face location, the primary step of the vision-based automated systems, finds the face area in the input image. An accurate location of the face is still a challenging task. This work focuses on the evaluation of face detection classifiers minding facial landmarks.

"Rapid object detection using a boosted cascade of simple features." by P. Viola and M. Jones.

This paper describes a machine learning approach for visual object detection which is capable of processing images extremely rapidly and achieving high detection rates. This work is distinguished by three key contributions. The first is the introduction of a new image representation called the Integral Image which allows the features used by our detector to be computed very quickly.

3 Methodology

This section describes the process we followed to train the cascade classifier to detect the images of our choice and the steps we took to build the database.

Pre-processing:

First, we collected 4-5 positive samples of the character we wanted to detect. Then, we converted the images to grayscale and resized them to 50x50.

Next, we collected negative image samples of background images from various datasets. We converted each image to grayscale and resized them to 100x100. This was done using the OpenCV library for Python.

Training the classifier:

Using the positive samples, we used OpenCV to create multiple variations of the positive samples by placing them in different positions and rotations and perspectives on the negative samples. We created around 2000 positives for every positive image. We then merged these images into a single image that was created by stitching together all the positive samples. This was the positive vector for the image. We also downsized all the images to 24x24.

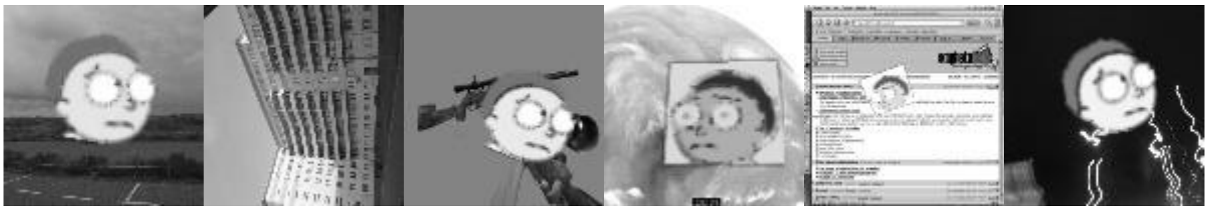


Fig. The generated positive samples.

Next, we generated positive vectors for each positive image of the character. After doing this, we merged all the vectors to get one vector for the particular character.

We then passed this vector to the classifier with the following parameters-

Number of positives – 1800

Number of negatives – 900

Number of stages – 15 (on average)

Dimensions of the training images – 24x24

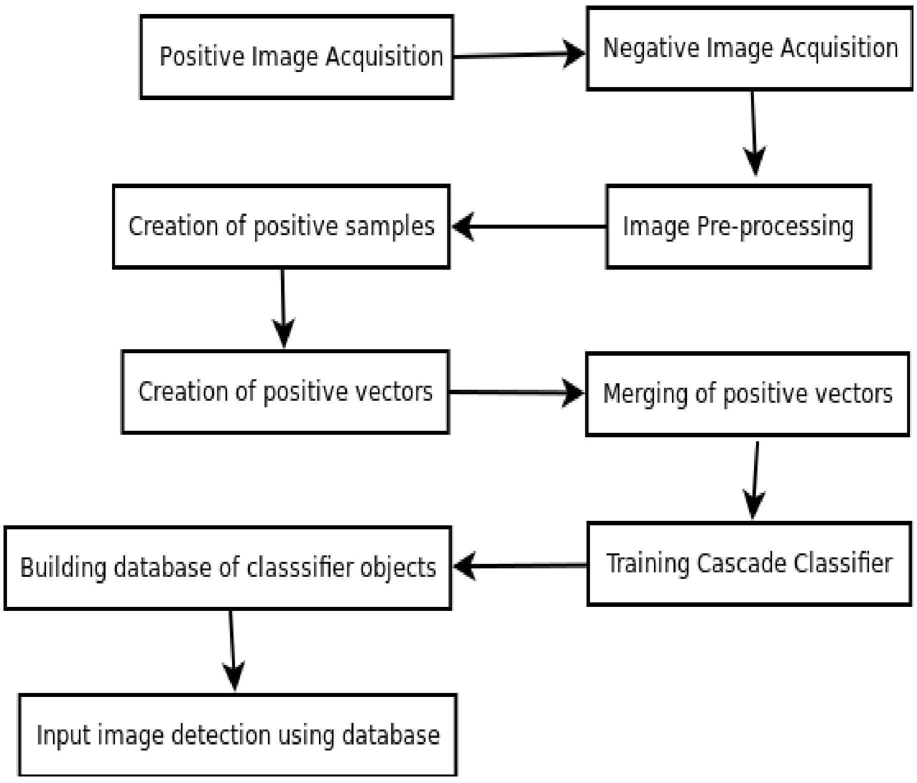
The classifier took an approximate of 30 minutes for each stage of the training on a laptop with 4GB of RAM, 256GB SSD and a mobile quad core AMD processor with clock speed of 2.3GHz.

Building the database:

The above process was repeated for every character that we wished to detect. For the purposes of this paper, we generated a database of 6 characters. The database uses a key-value store hosted locally. It sends a cascade file to a python script that loads it and checks it against the input image. If there is a match the detected character is highlighted and the TV series is printed. If there is no match the next cascade file is passed and the process is repeated.

Architecture:

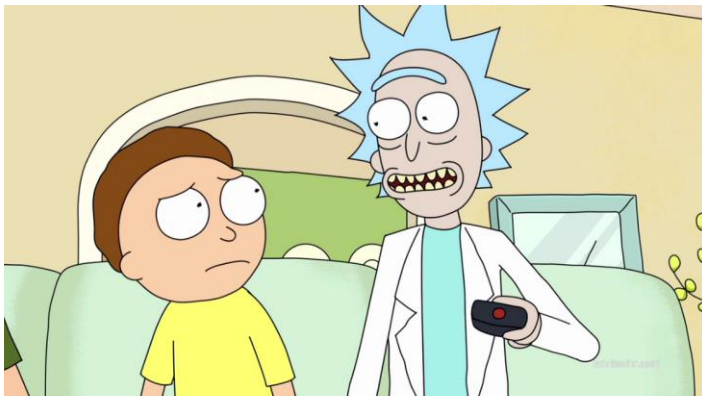
The working of the system can be described by the following flow diagram:


Positive Image Acquisition
Negative Image Acquisition
Image Pre-processing
Creation of positive samples
Creation of positive vectors
Merging of positive vectors
Training Cascade Classifier
Building database of classsifier objects
Input image detection using database

4 Results

The results discuss the working of the system. First, we have the source of the negative image sets – Satoshi's negative image sets [7].

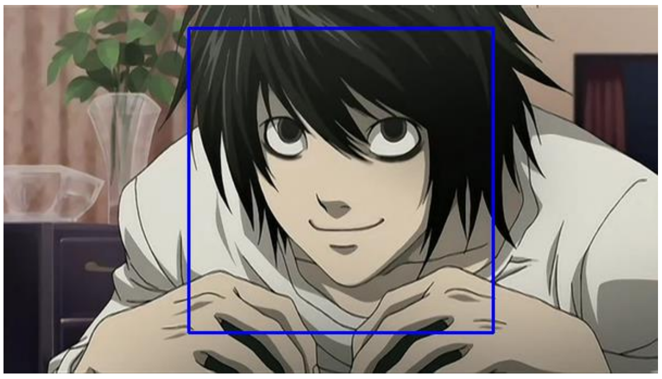
The following are the input images that we tested against the system-

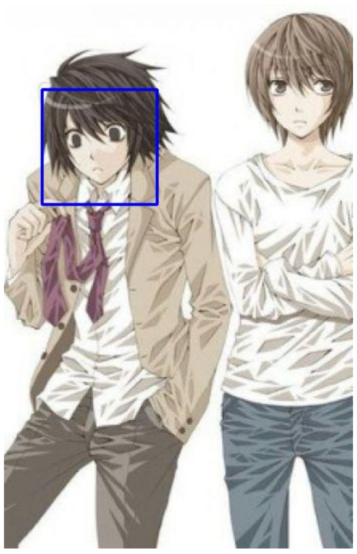


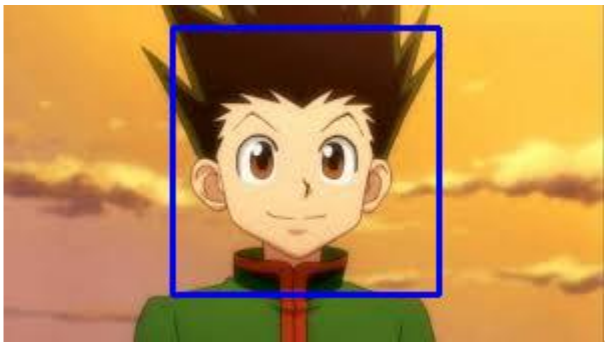
The output for each image from top to bottom left to right are as follows-

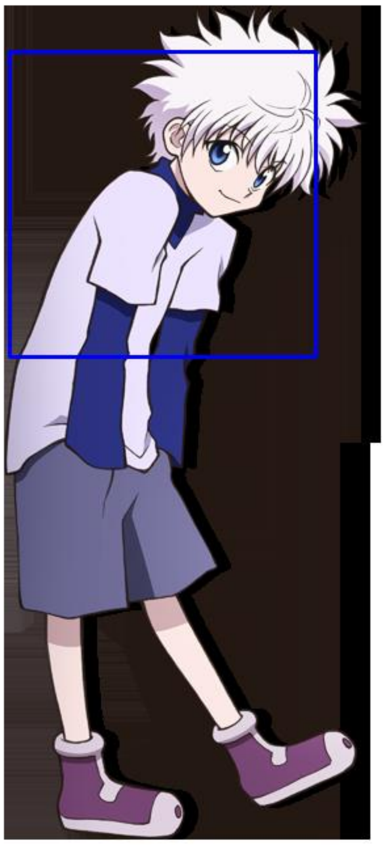
```
>Series: Death Note

>Series: Death Note

>Series: Hunter x Hunter

>Series: Hunter x Hunter

>Series: Rick and Morty
```

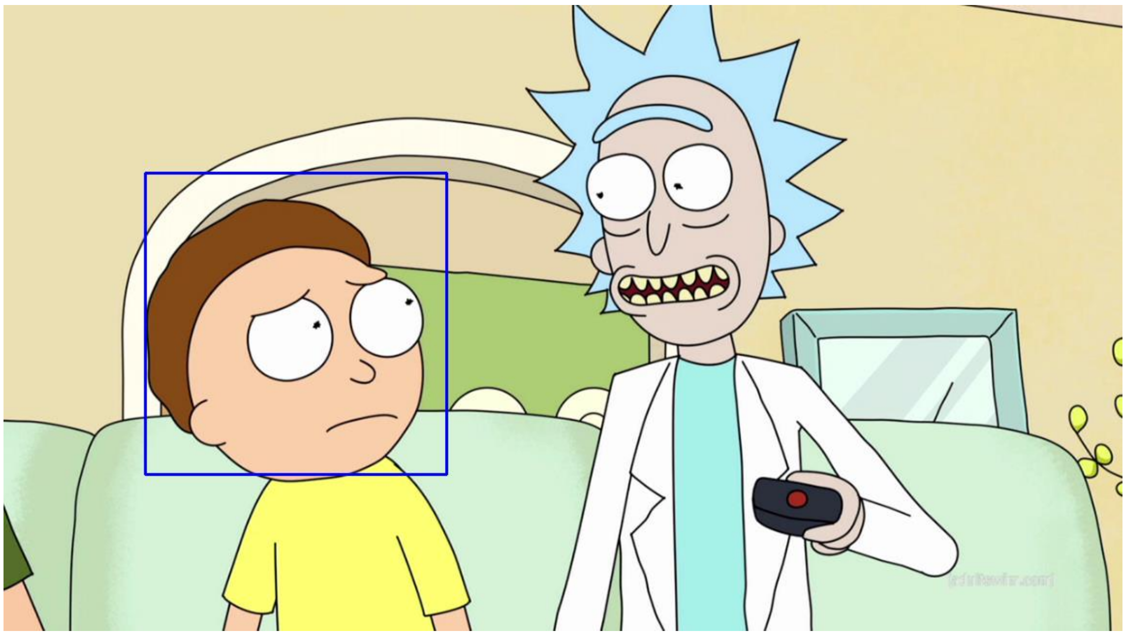
Also, these are the detected characters from the images-











Analysis-

The following table shows the performance of the system after testing it against more than 50 different test samples collected from the internet. The result is broken down according to the accuracy of each individual cascade file and then the accuracy of the entire system.

The test images had equal number of images with the character present and not present.

False positives are when the system detects the character where the character is not present and in images where the character is not present at all.

The accuracy is determined by the number of images where the character is detected correctly when he/she is present and when not detected when he/she is not present.

Cascade	***Character***	***Number of stages***	***Number of samples***	***Region***	***False positives***	***Accuracy***
1	Gon	15	6	Frontal, Neck, Side View	12%	86.71%
2	Killua	13	4	Frontal, Side View	15%	84.33%
3	L	17	9	Frontal, Side View	7%	90.09%
4	Morty	15	5	Frontal, Side View, Various Expressions	12%	83.67%
5	Donald Duck	14	5	Frontal	19%	74.31%
6	Light	16	7	Frontal, Side View	10%	88.90%

The overall performance of the system when tested against 75 images where one of the characters were present and 33 images where none of the characters were present is as follows. In this case, if the detected character is wrong it is counted as a false positive.

False positives- 17.54%

False negatives- 2.15%

Overall accuracy of the system- 82.3374%

Average time to process the image- 350.69 ms

5 Conclusion

From the above we can conclude that using Haar cascade classifiers to detect cartoon characters from an image holds promise as a method to be extended for use in an image search engine. the advantages of this method include the fast method to detect the images and the high accuracy of the system. We have used low quality classifiers because of resource constraints. We believe that the accuracy of the current system can be dramatically increased by using better quality classifiers. The pitfalls of this method include the high overhead required to train the classifier correctly. Our future work includes the research of a method to automate the database building process and to find a method to increase the quality of the classifier.

References

[1] BOUSSAA, Mohamed, et al. "ECG image classification in real time based on the haar-like features and artificial neural networks." (2015).

[2] Kuo, Ying-Che, Zih-Yi Yang, and Chih-Hung Yen. "Fast pedestrian detection system with a two layer cascade of classifiers." Computers & Mathematics with Applications 64.5 (2012): 1311-1323.

[3] Asi, Salina Mohd, et al. "Automatic craniofacial anthropometry landmarks detection and measurements for the orbital region." Procedia Computer Science 42 (2014): 372-377.

[4] Zheng, Lihong, et al. "Accuracy enhancement for license plate recognition." Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on. IEEE, 2010.

[5] Padilla, R., C. F. F. Costa Filho, and M. G. F. Costa. "Evaluation of haar cascade classifiers designed for face detection." World Academy of Science, Engineering and Technology 64 (2012).

[6] Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. Vol. 1. IEEE, 2001.

[7] https://github.com/sonots/tutorial-haartraining/tree/master/data/negatives

[8] http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html

[9]https://pythonprogramming.net/haar-cascade-object-detection-python-opencv-tutorial/

[10] http://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html

[11]http://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Image_Object_Detection_Face_Detection_Haar_Cascade_Classifiers.php