123Dog muzzle detection based on a cascade of Haar-like classifier

Jerry Wang and Rebecca liu

|  |  |
| --- | --- |
| Institute of Information  & Mathematical Sciences  Massey University,  Albany, Auckland, New Zealand  Supervisor: Dr. Andre Barczak |  |

In this paper, we create a cascade of Haar-like classifier to detect and track dog muzzle area. Haar cascade features is the most common technology in computer vision for object detection. The proposed program can detect a dog from an image, video clip, or video camera. The training of the cascade classifier is conducted with two OpenCV utilities opencv\_traincascade and opencv\_createsamples.

Keywords: Computer Vision; Convolutional Neural Network (CNN); Haar-like Feature; Local Binary Patterns (LBP); Object Detection; Training Classifier; XML file;

# 1 Introduction

Machine Learning is a subfield of Artificial Intelligence (AI). Machine learning is an application of Artificial Intelligence that provides systems the ability to automatically learn and improve from experiences without being explicitly programmed (Wekipedia, 2018). **Machine Learning focuses on the development of computer programs**that can access data and use it to learn for themselves. Machine Learning can be categorized as supervised and unsupervised ones.

Haar-like cascade classifier is a supervised classifier that can use labeled examples to predict future events in a new dataset by learning from the past (Bradski & Kaehler, 2008). Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.

The Haar-like feature supported by OpenCV is an object detector initially proposed by Paul Viola and Michael Jones in “Rapid Object Detection using a Boosted Cascade of Simple Features”. (Bradski, 2008)

# 2 Background and literature review

Bradski (2008) indicated that the object detection technology implemented in OpenCV is commonly known as the Viola-Jones detector. The Viola-Jones detector uses a form of Adaboost algorithm, which showed in Figure 1 below.

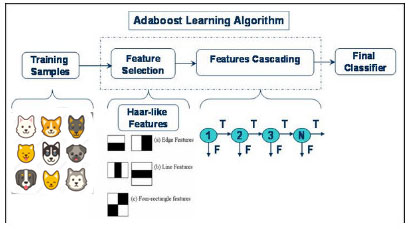


Fig 1. How Face Detection Works

(Internet source: https://www.quora.com/How-can-I-understand-Haar-like-feature-for-face-detection)

Figure 2 is the flowchart for a common Haar Cascade classifier training and testing.

Object Detection

Haar FeaturenCascade Classifier Training

**Fig 2 Flowchart for Haar cascade training and testing**

Ozhiganov (2017) indicated that Haar and LBP classifier were the best object detection algorithms before Convolutional Neural Network (CNN) was introduced. We have studied these three object detection algorithms and have made a comparation between these three methods, which are listed in Figure 3 below.

Fig 3 comparation of Haar, LBP and CNN

Base on Figure 3, we can conclude that both Haar and LBP classifiers are easy to train, have the ability to handle scaled objects with reasonable detection speed and accuracy. However, both classifiers are failed to detect titling objects. CNN, on the other hand, need a very extensive training, but it has a higher accuracy, especially on tilting objects and it is the ultimately fastest method in those three.

In this project, we chose Haar cascade to detect dogs because it’s accuracy can overcome the disadvantage of an extensive training. We would have chosen LBP if this project was proposed on an embedded system.

# 3 Methodology

In this project, we will collect certain amount of positive and negative dog face images. The images will be processed with the programs provided with OpenCV SDK. Once the final XML file is produced, we will have two experiments. The first test will be carried out with 50 test images. The experimental result and the accuracy rate will be calculated for further analysis. The second test is conducted with a video clip downloaded from the Internet. The ultimate goal is to have an acceptable accuracy rate at the speed of not less than 80 percent of the normal rate.

## 3.1 Collect dog muzzle samples (positive images)

Positive samples are those images that contain the training object. The more positive images we have, the more accurate classifier we get. Thanks to Khosla, Jayadevaprakash, Yao & Li (2011), we have downloaded the positive images from Stanford Dogs Dataset (SDD). The dataset contains 20580 images of 120 dog breeds from all over the world. For this project, we randomly pick 2000 positive images. Large number of positive images can improve accuracy rate. However, it also increases processing and training time.

A file POS.TXT that contains the positive images and the object annotations in the image have to be created manually, which is the most time-consuming job in this project. The file structure is shown as below.

File POS.TXT:

Img\_1.jpg 1 140 100 45 45

Img\_2.jpg 2 100 200 50 50 50 30 25 25

. . .

img\_n 1 130 100 50 50

where the coordinates are the bounding rectangle of the objects

## 3.2 Collect negative samples

Negative (background) samples are those images that do not contain the training object. General speaking, to train a highly accurate classifier, a lot of negative images that look exactly like the positive ones are needed. For this project, we need 4000 of them. We use 2 methods to obtain negative images. First, we crop out dog muzzle area from a positive image and save it to another image file as a negative image, because it does not contain the training object. By this way, we obtain 2000 negative images while processing the positive images. Second, we simplify this job by taking a 3 minutes video clip at Massey University Albany campus. We write a small OpenCV program to extract every other frame in the video and save as a JPEG image. By this method, we obtain another 2700 negative images.

### 3.3 Build a vector output file of the positive samples

Positive samples can be created with opencv\_createsamples.exe application provided in OpenCV SDK.

Open command line window and type the content below at the prompt.

e:\ cascade\opencv\_createsamples.exe -vec POS.VEC -info positives\POS.TXT -bg negatives\NEG.TXT -w 40 -h 40 -num 2000

The program will create a file POS.VEC in the current folder.

## 3.4 Training Haar classifier

We train our own classifier using opencv\_traincascade.exe program provided in OpenCV SDK. Open command line window and type the content below at the prompt.

e:\cascade\opencv\_traincascade.exe -data XML -vec POS.VEC -bg NEG.TXT -numPos 2000 -numNeg 4000 -numStages 12 -precalcValBufSize 1024 -precalcIdBufSize 1024 -w 40 -h 40 -mem 1024 -nonsym -mode ALL -maxFalseAlarmRate 0.5 -featureType HAAR

A file pos.vec will be created after this program has finished successfully.

Note 1: Feature set size 40x40 pixels

Note2: We use "-nonsym" option for this classifier training. It will take more time, but the trained classifier will effectively detect both frontal and profile dog muzzles.

Note 3: We use “-mode ALL” option which will process upright Haar features and rotated features.

Note 4: “-numStages” is set to 12. The more stages you set, the more accurate your cascade.

After the application has finished, a trained cascade will be saved in CASCADE.XML file in the folder “XML”.

## 3.5 Coding

In OpenCV, we use CascadeCalssifier::detectMultiScale() function to detect objects.

void CascadeClassifier::detectMultiScale(InputArray image, vector& objects, double scaleFactor=1.1, int minNeighbors=3, int flags=0, Size minSize=Size(), Size maxSize=Size())

The code below will find all of the objects and draw an ellipse on each of them.

Mat frame,frame\_gray;

std::vector<Rect> muzzles;

cvtColor( frame, frame\_gray, CV\_BGR2GRAY );

equalizeHist( frame\_gray, frame\_gray );

dog\_cascade.detectMultiScale(frame\_gray,muzzles,1.1,5,0|CV\_HAAR\_SCALE\_IMAGE,Size(30,30));

for( size\_t i = 0; i < muzzles.size(); i++ ){

Point center(muzzles[i].x + muzzles[i].width\*0.5,muzzles[i].y+muzzles[i].height\*0.5);

ellipse(frame,center,Size(muzzles[i].width\*0.5,muzzles[i].height\*0.5),0,0,360, Scalar(255,255,0),4,8,0);

}

Note 1: Objects (muzzles) is the vector of rectangles where each rectangle contains one detected object.

Note 2: scaleFactor is the parameter specifying how much the image size is reduced at each image scale. Default is 1.1

Note 3: minNeighbors is the parameter specifying how many neighbors each candidate rectangle should have to detain it. Default is 3, we use 5 in here.

## 3.6 Testing setup

Experiment one

# 4 Results

# 5 Discuss

# 6 Conclusion

# Acknowledgements

We thank Dr. Andre Barczak, senior lecturer – Institute of Natural and Mathematical Sciences, Massey University for the great computer vision lectures.

# References

Bradski G. and Kaehler A. (2008) Learning OpenCV. Published by O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Khosla Aditya, Nityananda Jayadevaprakash, Bangpeng Yao & Li Fei-Fei. (2011) Novel dataset for Fine-Grained Image Categorization. First Workshop on Fine-Grained Visual Categorization (FGVC), IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Website:<http://vision.stanford.edu/aditya86/ImageNetDogs/main.html>. Retrieved 18:00pm, April 19,2018.

OpenCV Documentation. (2018). Cascade Classifier. Website: <https://docs.opencv.org/2.4/doc/tutorials/objdetect/cascade_classifier/cascade_classifier.html>. Retrieved 18:45, April 19, 2018.

OpenCV Documentation. (2018). Cascade Classifier Training. Website: <https://docs.opencv.org/3.1.0/dc/d88/tutorial_traincascade.html>. Retrieved 18:50pm, April 19, 2018.

OpenCV Documentation. (2018). Face Detection Using Haar Cascades. Website: <https://docs.opencv.org/3.4.1/d7/d8b/tutorial_py_face_detection.html>. Retrieved at18:46pm, April 19,2018.

Ozhiganov I. (2017). Convolutional Neural Networks vs. Cascade Classifiers for Object Detection. Big Data Zone. Website: <https://dzone.com/articles/cnn-vs-cascade-classifiers-for-object-detection>. Retrieved 18:25pm, April 19,2018.

The CIFAR-10 dataset. (2018). Website: <https://www.cs.toronto.edu/~kriz/cifar.html>

Wikipedia. (2018). Website: <https://en.wikipedia.org/wiki/Machine_learning>. Retrieved at 19:57pm, 01/May,2018.

Wikipedia. (2018). List of datasets for machine learning. Website: <https://en.wikipedia.org/wiki/List_of_datasets_for_machine_learning_research>. Retrieved at 20:12pm, 01/May,2018.