|  |
| --- |
| Real-time OBject detection with Python and OpenCV  Machine Learning individual delivery |
| Antonio Jesús Leal Martín |

**Antonio Jesús Leal Martín**  
Data Science  
The Bridge

General Vision

Object detection is a computer vision process that combines two different tasks:

* Predicting the class of one object in an image.
* Locate (and somehow marking it) that object within the image.

This is an example of an input image (right) and the output after object detection process (left):

 

In this case, the Haar Cascade Classifier (HCC) model has been used in order to achieve this goal. Tools for running HCC can be found in OpenCV library repositories.

Specifications

**Software**

The minimum software requirements to run the attached code are the following:

* Python 3, VSCode with Python and Jupyter extensions installed.
* Python modules: opencv
* OBS + virtualCam plugin
* Unity3D application

**Hardware**

The software has been ran using the following computer specs:

* **CPU:** Intel(R) Core(TM) i7-6700 (at least Inte(R) Core(TM) i5 recommended)
* **RAM:** 16Gb (more than 8Gb recommended)
* **GPU:** AMD Radeon R9 390
* **OS:** Windows 10

**Requirements**

In order to achieve object detection, we need data. I created the training dataset by myself taking screenshots from the game itself. The dataset can be found in the folder ‘data/Man’ in the file ‘images.rar’. This file must be uncompressed at the same folder.

HCC works with positive and negative data, being the negative those images where the object to be detected does not appear, and the positive images where the object appears at least once. Also, openCV includes a set of useful console applications for the training process (they will be explained in mor detail later):

* Opencv\_annotations.exe
* Opencv\_createsamples.exe
* Opencv\_traincascade.exe

These tools can be downloaded from the following link:

<https://sourceforge.net/projects/opencvlibrary/files/3.4.13/>

Finally, we will need the input, in this case, it comes from an Unity3D application that can be downloaded from

<https://we.tl/t-5Jw6U4CoCX>

Steps

1. **Making the 3D application**

I used Unity3D in order to create a small city to place the objects to detect. I programmed a script to control the camera within that environment too. (C#)

1. **Generating the data**

Once I was able to move the camera I programmed a script (C#) to get screenshots from the game (images from the camera view). With it, I generated 200 negative images and around 150 positive images.

1. **Generating positive samples and negative file list**

Using opencv\_annotations.exe, I created the positive samples from the positive images. This application shows every positive image and let you draw rectangles to locate every instance of the object you want to detect. After this process, you get a text file with the path of every image and the data of the rectangles containing the samples. The console command used is:

opencv\_annotation.exe --annotations=pos.txt --images=Positive/

Using this text file as input, I used opencv\_createsamples.exe to generate the vector file for the samples. This is a binary file that cannot be read with a text reader software, but it is what the trainer needs to train the program. The command used for this application is the following:

opencv\_createsamples.exe -info pos.txt -w 24 -h 24 -num 1000 -vec pos.vec

For the negative images, I wrote a python program (‘cascadeutils.py’) to list the path to every image in a txt file.

1. **Training HCC**

Using the .vec file for positive samples and the .txt file for the negative images, I trained the HCC model using the console application opencv\_traincascade.exe. The command used for this task is below:

opencv\_traincascade.exe -data Cascade/ -vec pos.vec -bg neg.txt -w 24 -h 24 -precalcValBufSize 6000 -precalcIdxBufSize 6000 -numPos 150 -numNeg 1000 -numStages 12 -maxFalseAlarmRate 0.3 -minHitRate 0.999

This training results in a file named ‘cascade.xml’ which is the file with the model trained parameters. We will feed the cascade classifier with this file in order to make the object detection.

1. **Getting video input and detecting objects**

In the ‘main.py’ file, I programmed the video reading and the application of the cascade model using methods from the opencv library. Before running the program, we need to have the Unity app running and OBS capturing that window as a virtual webcam.

Having all these things ready we can now run ‘main.py’, this will show the processed video marking the object to detect.

Conclusions

I successfully achieved object detection with a low rate of false positives. However, some problems were found in object detection:

* It is hard for the model to detect objects in shadowed areas.
* Also, the model fails to detect the object in certain points of the animation.
* Some false positives were found in areas with patterns like the object.

The firsts two problems can be fixed augmenting the number of positive samples, gathering data from both shadowed/sunny areas and through all the animation. The last one can be improved training the model with more negative images.

Project scope

Apart of being improved as mentioned before, this program has been made to detect objects in a game demonstration, but it can be used for other tasks within/outside videogames field. I mention below the two topics that would interest me the most:

* Programming bots that automatically play games
* Using object detection for artificial vision (autonomous cars, security systems…)