Artur Zacniewski

Web-Based API as a Tool in Teaching Computer Vision Concepts

Edukacja - Technika - Informatyka nr 1(19), 297-304

2017

©Wydawnictwo UR 2017
ISSN 2080-9069
ISSN 2450-9221 online

„Edukacja – Technika – Informatyka” nr 1/19/2017
www.eti.rzeszow.pl
DOI: 10.15584/eti.2017.1.41

ARTUR ZACNIEWSKI

Web-Based API as a Tool in Teaching Computer Vision Concepts

Doktor inżynier, Akademia Marynarki Wojennej w Gdyni, Wydział Nawigacji i Uzbrojenia Okrętowego, Zakład Informatyki, Polska

Abstract

The article presents the stages of building Web-based API as a tool for teaching selected concepts of image processing server and shows a structure allowing for its implementation. During the implementation open source tools were used – graphical library OpenCV and Python programming language, which allows the implementation of both web part (by Django framework), as well as the cooperation with the above library.

Idea of the creation of new Web-based API endpoints and the ability to change the parameters library OpenCV algorithms by modifying the URL in your browser were shown. Stages of creating each endpoint has been depicted with practical example on the image processing. Advantages and limitations of the proposed solution were presented.

Keywords: Web-based API, Computer Vision, image processing

Introduction

An Application Programming Interface (API) is a set of specifications and rules that software programs can follow to communicate with each other. An API expresses a software component in terms of its operations, inputs, outputs, and underlying types. It also defines functionalities that are independent of their respective implementations, which allows definitions and implementations to vary without compromising the interface. A good API makes it easier to develop a program by providing all the building blocks. A programmer then puts the blocks together (Cataldo, Mockus, Roberts, Herbsleb, 2009).

A Web-API is just an API for either a web server or web browser. In this article server-side API is discussed. This kind of API is a programmatic interface to defined request-response message system. Most popular standard for representing data structures in such systems is JSON (Java Script Object Notation) and formats based on JSON. Nowadays APIs for web applications are moved towards collection of RESTful web resources. REST (Representational State Transfer) is a software architecture style consisting of guidelines and best practices for creating scalable web services. These RESTful Web APIs are accessible via standard HTTP methods (like GET, POST, PUT, DELETE etc.) by a variety of HTTP clients including browsers and mobile devices (Benslimane, Dustdar, S., Sheth, 2008; Cataldo i in., 2009). Designing of RESTful Web APIs is covered in details in (Richardson, Amundsen, 2013).

Creators of Computer Vision applications and libraries can face a problem with showing their software to the wider audience, i.a. as a tool in teaching chosen concepts. Is there a way to overcome this obstacle with a help of Web technologies? One of the most useful feature of Web-API is ability to share programming interface and resources of particular Web system to the client without necessity of installing anything. User doesn't need to have particular software installed on his machine, but thanks to Web-API he can check possibilities of interesting system using only his browser. Of course this API must be created before. In the article project with Web-API aimed at teaching concepts of Computer Vision with OpenCV is presented.

The idea of using Web-based API to interact with Computer Vision libraries isn't quite new. There are few promising portals like CloudCV.org, BetaFaceApi.com or PyImageSearch.com, but no relevant literature about this topic was submitted.

Materials and methods

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. The library has more than 2500 optimized algorithms, which can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has C++, C, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS (OpenCV, 2017). It's perfect tool to teach and learn concepts of Computer Vision.

Python is one of the aforementioned languages, which works with OpenCV. It also has expansive library of web frameworks and could be proper tool to build Web-based API. Figure 1 shows configuration of the server working with Python. It's a typical configuration for most server providers. The only recommended operating system for production Python web stack deployments is Linux.

A Web Server Gateway Interface (WSGI) server implements the web server side of the WSGI interface for running Python web applications (Makai, 2015).

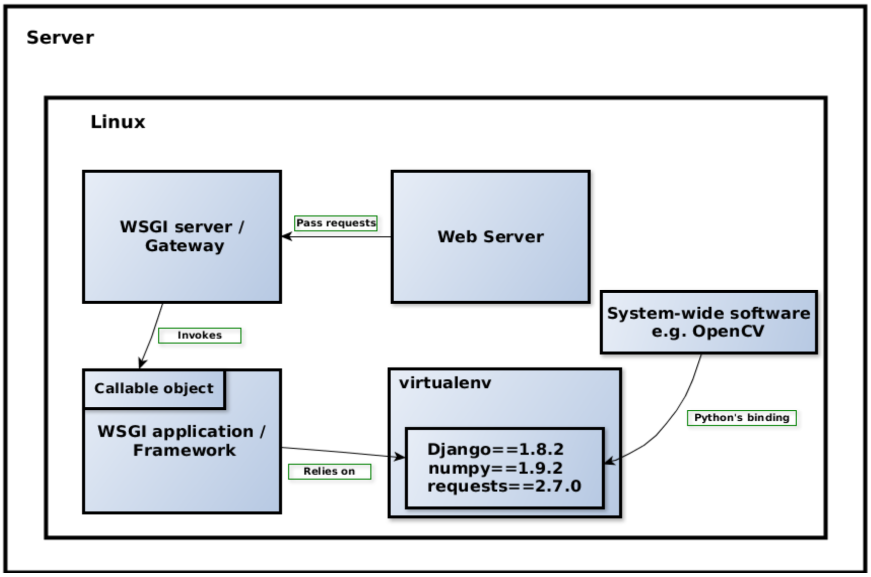


Fig. 1. Configuration of server working with Python (Makai, 2015)

Application dependencies are the libraries other than project code that are required to create and run your application. Dependencies are installed separately from system-level packages to prevent library version conflicts. The most common isolation method is using virtual environments. Each 'virtualenv' is its own copy of the Python interpreter and dependencies in the 'site-packages' directory. To use a 'virtualenv' it must first be created with the *virtualenv* command and then activated. The virtual environment stores dependencies in an isolated environment. The web application then relies only on that 'virtualenv' instance (Makai, 2015). In this particular case uWSGI, Django, Requests and NumPy were used.

Django is one of the most popular frameworks for Python (Django, 2017) and it allows for building Model-View-Controller (MVC) applications.

Requests is „an elegant and simple HTTP library for Python, built for human beings" (Requests). And finally NumPy is the fundamental package for scientific computing with Python (NumPy, 2017). NumPy is necessary, because OpenCV represents images as multi-dimensional NumPy's arrays.

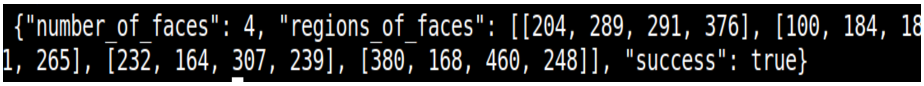
Experiments

Many popular areas of Computer Vision can be observed and learnt using Web-based API, for example face detection or simple thresholding. This article shows implementation of few possible algorithms that can be used, but the list of acceptable options is much longer.

Face detection

Haar feature-based cascade classifier for object detection has been initially proposed by Viola (Viola, Jones, 2001) and improved by Lienhart (Lienhart, Maydt, 2002). In OpenCV basic cascade classifier class for object detection is *CascadeClassifier* class. The *load* method of this class loads classifier from the XML file and the *detectMultiScale* method allows detecting objects of different sizes in the input image. The detected objects are returned as a list of rectangles (OpenCV, 2017). This class and its methods can be used in a Django's view to build given feature in Web-based API. View is a part of application that reacts on input and gives results dependent on used algorithm.

In this particular case 'haarcascade_frontalface_default.xml' file with trained Haar classifier was chosen, but there are few other different classifiers available in OpenCV. Using command line library like *cURL* allows to get information in JSON format. Figure 2 shows example of interacting with Web-based API. Response in JSON format consists of information about number of detected faces and their coordinates.



```
{"number_of_faces": 4, "regions_of_faces": [[204, 289, 291, 376], [100, 184, 18
1, 265], [232, 164, 307, 239], [380, 168, 460, 248]], "success": true}
```

Fig. 2. JSON response after detecting multiple faces

Due to some disadvantages of Viola-Jones algorithm only chosen images (frontal face, no face rotation, proper lightning conditions) were analysed. Detailed analysis and proposition of improvement for this algorithm is presented in (Yi-Qing Wang, 2014).

Additional parameters in the URL

Generally, passing an argument via URL could be performed in the following way:

```
address-of-the-server/view-name/parameters/
```

For the purposes of this article simple Web-based API was created on author's page (http://www.zacniewski.pl/vision/). User interacts with basic Computer Vision algorithms by submitting images from the Internet and watching results. Parameters can be changed directly in the browser and no additional software is needed. OpenCV (version 3.2.0) installed on VPS (Virtual Private Server) allows for active learning in the area of Computer Vision.

In the case of face detection the only argument that could be passed to the server is a path to the image (in this case URL of the image). But many algorithms require additional parameters, for example value of threshold in thresholding methods. Django allows for passing additional parameters in the URL. Values of these parameters can be used in particular views.

In OpenCV method dedicated to simple thresholding is called *threshold* and it requires fixed-value threshold value that is applied to the single-channel array.

In the definition of function responsible for handling view in Django, additional parameters should be declared. Results of applying different values of threshold for binary type (THRESH_BINARY) of thresholding are presented on figure 3. In this example additional parameter is just a fixed-value of threshold (OpenCV, 2017).

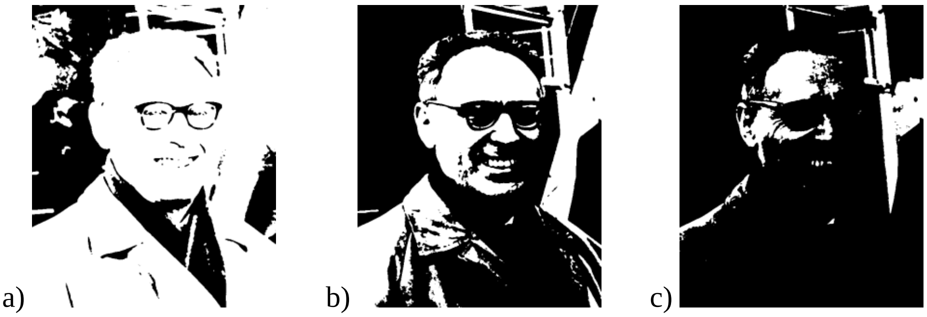


Fig. 3. Simple thresholding with THRESH_BINARY:
a) threshold = 50, b) threshold = 127, c) threshold = 170

Global value set for the threshold is not always right solution. It may be not good in all situations where image has different lightning conditions in different areas. Using adaptive thresholding allows to overcome this problem by getting different thresholds for different regions of the image. Function *adaptiveThresh -old* from OpenCV can be used in the Django's view. Changing one of the parameters of this function can be done via Web-based API. ADAPTIVE_THRESH_GAUSSIAN_C option was used – threshold value is the weighted sum of neighbourhood values where weights are a Gaussian window (OpenCV, 2017).

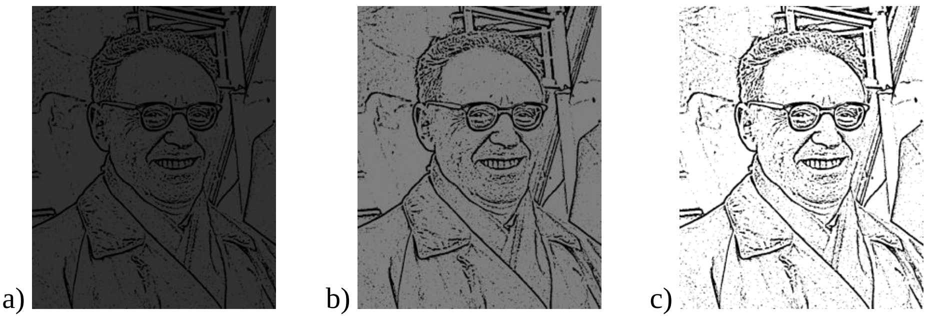


Fig. 4. Adaptive thresholding with ADAPTIVE_THRESH_GAUSSIAN_C:
a) threshold = 50, b) threshold = 127, c) threshold = 170

Results of applying adaptive thresholding are presented on figure 4. In this example additional parameter is non-zero value assigned to the pixels for which the thresholding condition is satisfied (OpenCV, 2017). Passing additional parameter to the API is identical like in the simple thresholding case, but the view must be appropriately modified.

Some of Computer Vision algorithms require two or more parameters. The problem with passing them in the URL may arise. Using *named groups* in Django's URL dispatcher solves easily this issue. The main rule is that names of the parameters in URL dispatcher should be the same as these in corresponding views. Part of URL dispatcher and part of the corresponding view are presented on figure 5. Expression starting with '?P' is *named group* and the name in angle bracket is a parameter that is used in the *canny* view. In this particular example *thr1* and *thr2* names are used both in the dispatcher and in the view.

a)
```
url(r'^vision/canny-detector/(?P<thr1>[0-9]+)/(?P<thr2>[0-9]+)/$',
                     'face_detector.views.canny'),
```

b)
```
def canny(request, thr1="100", thr2="200"):
    edges = cv2.Canny(image, float(thr1), float(thr2))
```

Fig. 5. Using named groups in Django: a) part of the URL dispatcher, b) part of the view

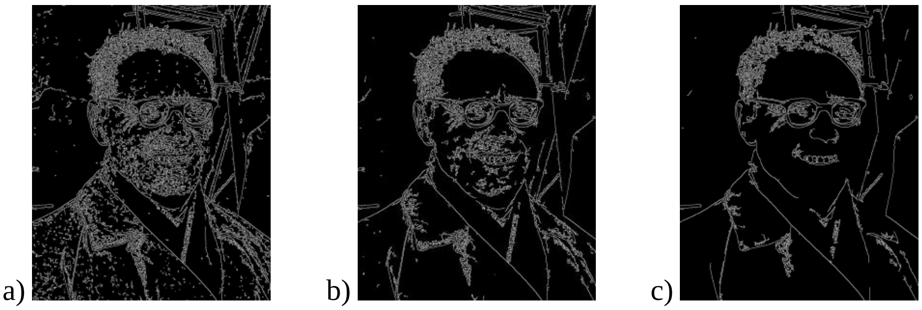


Fig. 6. Canny's edge detection with different values of thresholds: a) lower = 50, upper = 100; b) lower = 50, upper = 150; c) lower = 50, upper = 250

In OpenCV one of the algorithms that requires two parameters is Canny's edge detection algorithm. Aforementioned parameters are upper and lower threshold for the hysteresis procedure (Canny, 1986; OpenCV, 2017). Results of applying Canny's algorithm through Web-based API are presented on figure 6. It is worth mentioning that Canny recommended an*upper*/*lower* ratio between 2:1 and 3:1 (Canny, 1986).

Table 1 presents endpoints of Web-based API presented in the article. There is no limitation in the number of arguments passed to the server in the URL, so theoretically many more algorithms can be observed and analysed using Web browser, without necessity of installing any Computer Vision library. Student can check and try given algorithm by typing URL in the browser and watch the results.

Table 1. Endpoints of Web-based API

Algorithm	Example of URL usage	Explanation
`Face detection with Haar classifier`	`zacniewski.pl/vision /detect-faces/`	No parameters needed
`Thresholding`	`zacniewski.pl/vision /thresholding/127/`	**127** is a value of threshold (binary thresholding and adaptive thresholding)
`Canny's edge detection`	`zacniewski.pl/vision /canny-detector/100/200/`	**100** is a value for lower and **200** for upper threshold in hysteresis procedure of Canny's algorithm

One can also pass parameters of chosen methods, change them to spot the differences. All without installing Computer Vision library.

Discussion

Web-based API may serve as practical tool in learning Computer Vision concepts, but some requirements to the server must be met. Level of privileges and possibility of installing packages and libraries (both system-wide and in virtual environments) plays key role. For example cheaper servers don't have GUI (Graphical User Interface) libraries, data is transferred using JSON, which is not always sufficient. Better option is VPS, where one can install and modify software (also GUI libraries) for teaching purposes.

The process of creating Web-based API consists of creating proper URL in URL dispatcher and implementing view for this URL. Parameters passed from the URL are being used in the view function or class. Modern frameworks facilitate this process in significant way.

Presented approach allows expanding created API in easy way. Gaining from using this kind of learning are obvious. 'Learning-by-doing' concept is achieved by experimenting with different algorithms, modifying them and watching the results, without necessity of buying or installing software. One can try to learn new things using only his browser, it can be done with PC or smartphone, what makes this method very versatile.

Disclaimer

All pictures used in this article were labelled in Google for reuse in non-‑commercial purposes.

Literature

Benslimane, D., Dustdar, S., Sheth, A. (2008). Services Mashups: The New Generation of Web Applications. *IEEE Internet Computing*, *12*, 13–15.

Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *8* (6), 679–698.

Cataldo, M., Mockus, A., Roberts, J., Herbsleb, J. (2009). *Software Dependencies, Work Dependencies, and Their Impact on Failures*. *IEEE Transactions on Software Engineering, 35*, 864–878.

Django (2017). Retrieved from: https://www.djangoproject.com/ (1.2017).

Lienhart, R., Maydt, J. (2002). An Extended Set of Haar-like Features for Rapid Object Detection. *IEEE ICIP*, *1*, 900–903.

Makai, M. (2015). *The Full Stack Python Guide to Deployments*. San Francisco: Gumroad.

NumPy (2017). Retrieved from: http://www.numpy.org/ (1.2017).

OpenCV (2017). Retrieved from: http://docs.opencv.org/ (1.2017).

Requests, Retrieved from: http://docs.python-requests.org/en/latest/index.html (8.2015).

Richardson, L., Amundsen, M. (2013). *RESTful Web APIs*. Sebastopol: O'Reilly.

Viola, P., Jones, M. (2001). *Rapid Object Detection using a Boosted Cascade of Simple Features*. IEEE CVPR.

Wikipedia about Mikhail Botvinnik. Retrieved from: https://en.wikipedia.org/wiki/Mikhail_Botvinnik (8.2015).

Yi-Qing Wang (2014). An Analysis of the Viola-Jones Face Detection Algorithm. *Image Processing On Line*, *4*, 128–148.