CV Cube: A kit for teaching practical machine learning and computer vision

Abstract. In machine learning there has been strong evidence that data is more important than algorithms; still while teaching machine learning most of the focus is on teaching algorithms. Little to no focus is on teaching about data generation and collection. As a result of this students have no experience about challenges that arise while collecting data sets in real life and ways to overcome those challenges. In this paper, we present CV Cube - a simple kit for teaching machine learning and computer vision techniques that encourages students to work with data that they generate. Hence, students not only learn algorithms but also learn to tackle problems faced while working with new data; therefore gain experience in pre-processing, filtering and noise removal techniques also.

Keywords: Machine learning, computer vision, teaching, CV Cube

1 Introduction

Peter *et al.* (2009) stated that in machine learning data is more important than algorithms [1]. This is in contrast to the current teaching methods used for the said subject. Any machine learning course is more about learning to apply algorithms and little to no focus is on how data set is collected and pre-processed. As a result, students miss hands on experience about dealing with real world data. In this paper, we present CV Cube: a kit that can be used to teach machine learning and computer vision techniques. This not only makes the learning process practical but fun too. Similar approach was discussed by Azemi *et al.* regarding delivering introductory computer courses in order to make teaching engaging and more interactive [2].

CV Cube is a small/handy object feature detector. It is a combination of hardware and software kit which can analyze small/handy objects using a USB webcam and generate data about their features. The feature data include shape, relative size, number of edges, color, height of the object in view and absolute size. This data can be collected via CV Cube's API in real time and can be used as input for various machine learning algorithms. For a more advanced learner, kit can produce raw data that may be in form of unprocessed data or raw image frames that the camera sees. CV Cube is not only a learning platform for students of machine learning and computer vision but also acts as tool for researchers in these particular domains.

2 Hardware description

As mentioned before, CV Cube is combination of hardware and software kit. In this section we will discuss the hardware components of CV Cube. Table 1 shows the components in hardware of CV Cube.

Table 1. Hardware components in CV Cube.

Component	Brief Description
Frame	Acts as support structure for whole cube
USB Camera	To capture pictures of object in view
LED Lights	To create various lighting conditions
Control Unit	To control the lightening conditions

Figure 1 shows the CV Cube's hardware. The frame is simply a 30 cm x 30 cm cube structure made out of perforated MS angle. Perforated MS angle is used because they are easy to assemble using just nuts and bolts; hence making the CV cube easy to disassemble and easy to carry anywhere. The USB camera is assembled on top of CV Cube facing downwards where the object will be placed. The structure has four sided LED lightning parallel to upper four edges with a control unit to control the lightning conditions.

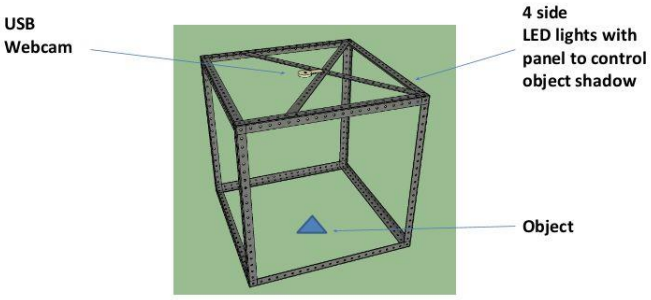

Fig. 1. Hardware of CV Cube.

3 CV Cube's API and feature extraction methodology

The software kit for the CV Cube has been written in Python using the Open CV library. It provides an API to get data from the CV Cube in real time and collect that data in any form for instance CSV. The API provides data about numerous features of the object in view. Table 2 lists the features that the API can provide. In this section, we will discuss about the feature extraction methodology.

Table 2. Features that can be extracted using API.

Feature name	Brief Description
Number of sides	Number of sides in the object
Shape	Prediction about the shape of the object
Color	Prediction about the color of the object
Relative size	Relative size of the object with respect to the size of full image canvas
Absolute size	Absolute size of object in inches
Height	Prediction about the approximate height of the image if the object is 3D

3.1 Detecting number of sides

A series of image transformations are applied in order to decide the number of the sides in the given object. Figure 2 below shows the step by step process.

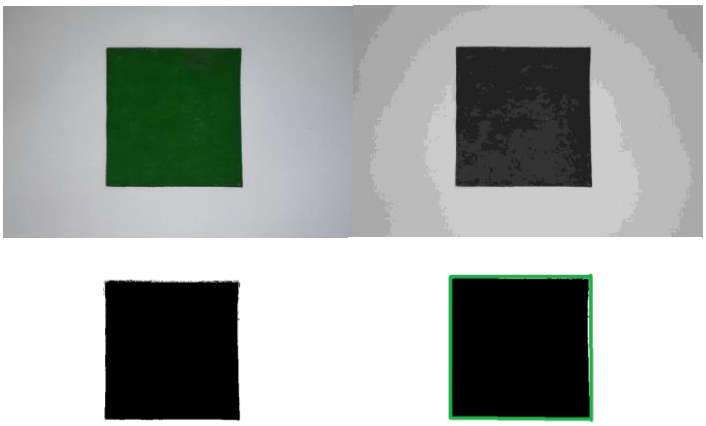

Fig. 2. Transformations applied to find the number of the sides.

First the image of the object is captured using the webcam. The initial image is 3-channel image obtained via the webcam. This image is converted to the 1-channel or

gray scaled image. Next a binary threshold is applied to the image in order to get a clear boundary of the object. At last the Canny Edge detection algorithm is applied to the image. Figure 2 shows the detected lines being drawn using green color. The number of edges detected is returned via the API.

3.2 Detecting shape

To detect the shape of the object the edges detected using the Canny algorithm is one of the major factors [3]. The number of sides is noted down and ratio of sides with one another is calculated. A simple algorithm to decide the shape is followed as given below.

Algorithm showing how decision about shape is made using number of sides obtained by applying Canny edge detection algorithm

```
if number of sides == 1 or number of sides == 2
   return Line
If number of sides == 3
   return Triangle
If number of sides == 4
   calculate ratio of non-diagonal sides
   if ratio is approximately equal to 1
       return Square
   else
       return Rectangle
If number of sides > 5 and number of sides < 15
  Let n be number of side
  return n-sided figure
else
  return round shape
```

3.3 Detecting color

Statistical methods have been used to detect the color of the object in view. Three frequency distribution plots one for each channel in RGB is created.

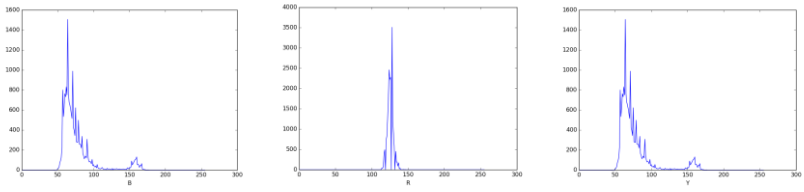

Fig. 3. Frequency plots for RGB.

These plots describes number of pixels and corresponding pixel values between 0 and 255 (both inclusive) for any particular channel. Currently, the system can predict

only three colors based on the maxima for the RBG plots. This frequency plot can also be accessed by the API and can be used to predict many more shades. Figure 3 below shows the frequency plots for RGB channels in object's image.

3.4 Detecting relative size

Relative size is just the ratio of the number of pixels that form the object with respect to the total number of pixels in the whole image. Figure 4 shows the image which is obtained after applying threshold to the single channel image.

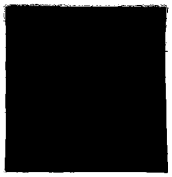

Fig. 4. Threshold image is used to calculate relative size.

From the image, it is very much clear that to calculate relative size; we just need to count the number of black pixels in image and total number of pixels in the image. Then, relative size can be obtained by using equation 1 where P_B is number of pixels that are black in threshold image and P_N is total number of pixels in the image.

$$\text{Relative size} = (P_B) / (P_N). \quad (1)$$

Relative size is important as it can be used to compare two objects that have been seen by the CV Cube. For instance, comparing the relative size of a coin and a wallet, the system can reach a conclusion that coin is smaller than a wallet.

3.5 Detecting absolute size

In order to calculate the absolute size, the system is first calibrated using an image showing a line and giving information about its exact length. Figure 5 ahead show the process of calibration.

A line which is exactly of 1 inch is shown to the CV Cube and the cube calibrates itself with a relation that length 1 inch is equal to the number of pixels used to draw this line. Next, time whenever the system wants to know the length of any edge, it uses equation 2. Here, P_E is number of pixels used to draw the edge and P_C is the number of pixels in the calibration line.

$$\text{Length of edge} = (P_E) / (P_C). \quad (2)$$

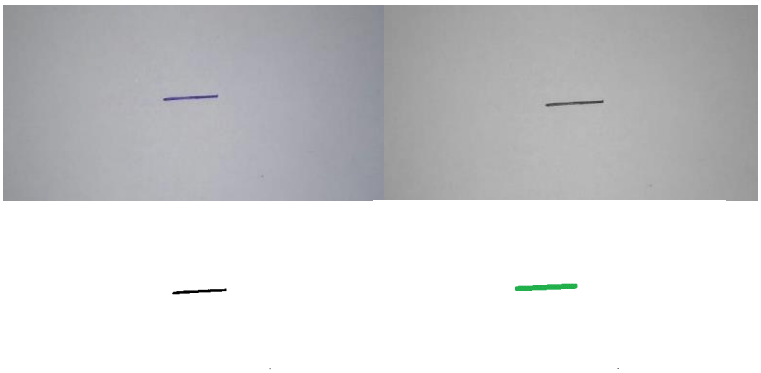

Fig. 5. Calibration process using 1 inch line.

3.6 Detecting height

To detect height of any object the system uses the shadow created using the LED lighting. Figure 6 shows the whole process. CV cube being a perfect cube makes exactly 45^{o} angle with the opposite-diagonal side. The length of shadow holds a relation with angle *BAD* as shown in image. As the angle *BAD* will increase the length of shadow will decrease. This correspondence can be calculated using properties of similar triangles and trigonometry will give the approximate height of the object in view.

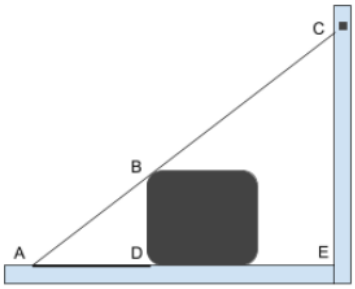

Fig. 6. Height calculation of a 3D object using its shadow.

4 A brief case study with tree leaves

In this section, we have shown how data collected by CV Cube can be used effectively. We have used leaves from two different trees to collect data namely: Mango and Guava tree. Samples of both have been shown below in Figure 7.

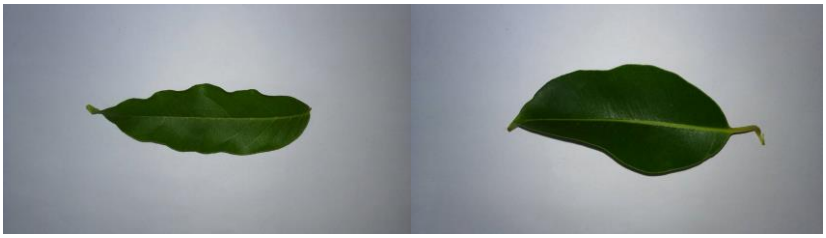

Fig. 7. Sample leaves: Guava and Mango.

In order to collect data, the leaves are kept in view of CV Cube and information regarding length and width of object in view is queried using API. Table 3 shows some sample values as generated by CV Cube.

Table 3. Leaves data generated by CV Cube (in inches).

Length	Width	Leaf Category
5.5	1.5	Mango
5.2	1.3	Mango
5.7	1.8	Mango
7	2.5	Guava
6.7	2.2	Guava
7.3	2.8	Guava

The data thus generated can be used as input for classification algorithms with labels or for clustering algorithm without labels.

The above study shows how useful CV Cube can be in collecting data regarding small/handy objects. We have selected only length and width data to be generated but as shown before CV Cube can extract data about many other features also.

5 Conclusions

We have not only shown effectiveness of CV Cube in data collection process but using CV cube in a semester long class of machine learning, we have seen students come up with interesting projects that not only enhance their learning but give them hands on experience to try and apply skills that they learn in classroom. Some noteworthy projects by students include: identification of common stationery items using CV Cube and Neural network layer build on top of CV Cube in order to detect complex shapes and objects. Moreover, doing so students learn about the most important part of machine learning; the data itself.

References

1. Halevy, A., Norvig, P. and Pereira, F.: The unreasonable effectiveness of data. IEEE Intelligent Systems, 24(2), pp.8-12, (2009)
2. Azemi A., D'Imperio N.: Improved approach for delivering an introductory computer science course. 39th IEEE Frontiers in Education Conference, (2009)
3. Kaur, Pahulpreet, and Bikrampal Kaur.: 2-D geometric shape recognition using canny edge detection technique. IEEE Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference, pp. 3161-3164, (2016).
4. Open CV Documentation: http://www.docs.opencv.org.