ISSN No:-2456-2165

Image Detection for Defence and Surveillance Using Machine Learning

Aashay Pawar
Research Intern, DRDO
B.E. Pune Institute of Computer Technology

Abstract:- Enlightening lives, growing connectivity around the creation, that's the boundless promise offered by data driven technology. Images and videos are a part of life, even at technology, at the stake of privacy. This is indeed required for application as well as security purpose. An image may consist a lot of information or sometimes absolutely nothing. Making use of high-speed machines such as computers can make things easy, but alone a computer can do nothing. A computer can achieve such a skill to extract features using a key technology named" Machine Learning". This paper specifically features a way in which defence systems can make use of this technology for security and surveillance purpose.

I. INTRODUCTION

Computer vision is an interdisciplinary arena that pacts with how computers can be made to advance high level understanding from digital images or videos. The idea is to systematize tasks that the human visual systems can do so as a computer should be able to diagnose that this is some object. We're going to see how a computer reads an image now. This is very interesting to notice the image that is there in front of your screen right, our normal human can easily tell that there is something in this image but can computers really see this? Well the answer is no. Computers see a matrix of numbers between 0 to 255 right. For a coloured image there will be firstly 3 channels namely red, green and blue and there'll be a matrix associated with each of these channels. Each element of this matrix represents the intensity of brightness of that pixel. All of these channels will have their separate matrices and these will be stacked on to each other to create a three-dimensional matrix. So, a computer will now be able to interpret a coloured image as a 3D matrix. One thing to know here that, for a black and white image, there is only single channel and image formed is a 2D matrix.

OpenCV is the library which is used for computer vision. It was first developed in the year 1999 at Intel by Gary Brad Sky and the initial announcement came out in 2000. OpenCV supposed extensive diversity of programming languages such as C++, Python, Java etc and also supports diverse platforms including Windows, Linux, etc. OpenCV python is nothing but a Python wrapper for the original OpenCV C++ execution. In OpenCV, all the images are converted to numpy array. This makes it calmer to assimilate it with supplementary libraries that practices numpy, for instance SciPy and matplotlib.

II. BASIC IMAGE DETECTION

A. Face Detection Using OpenCV

We first need to have an image loaded. Declare it with image file path. Create a cascade classifier which contain the features of the face. Using OpenCV, we will read the image and the features. It will look up for the row and column values for the face numpy ndarray, basically the face rectangle coordinates. Steps as below:

```
cascade_face = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
sampleimg = cv2.imread("photo.jpeg")
grayimg = cv2.cvtColor(sampleimg, cv2.COLOR_BGR2GRAY)
face = cascade_face.detectMultiScale(grayimg,ScaleFactor = 1.05,minNeighbors=5)
```

Now we need to add a rectangle shape box to the face. For this we need to call rectangle function for the image specifying the coordinates.

B. Capturing Video with OpenCV

A video is nothing but multiple images or multiple frames which are displayed very quickly so that it looks like a video. We will be using lutes to build a window where images will appear really fast so that we can see it as a video. now let us see how we can capture a video using OpenCV. the first thing we need to do is import OpenCV. Then we shall create a VideoCapture object and this number basically tells the computer to use the built-in camera. If we want to use an external camera, we just need to change the number inside VideoCapture(number). We can also give path to a video file if the file exists in the system.

```
samplevideo = cv2.VideoCapture(0)
check, frame = samplevideo.read()
```

Here, check is a bool data type that returns true if python is able to recite the video capture object otherwise it'll return false. Frame is a numpy array. It represents the first image that video captures. Since we saw that video is nothing but multiple images which appear is really fast and it looks like a video. So, what happened here Python was able to read the video capture object that's why we have got the output as true basically a check has returned true. To create a frame window, we want to generate a frame object which will deliver the images of the video capture object and we will recessively show each frame of the video being

ISSN No:-2456-2165

captured. Using imshow function we can show all the frames. In demand to capture a video, we shall be using a while loop. The state would be such that except check is true, python shall play the frame.

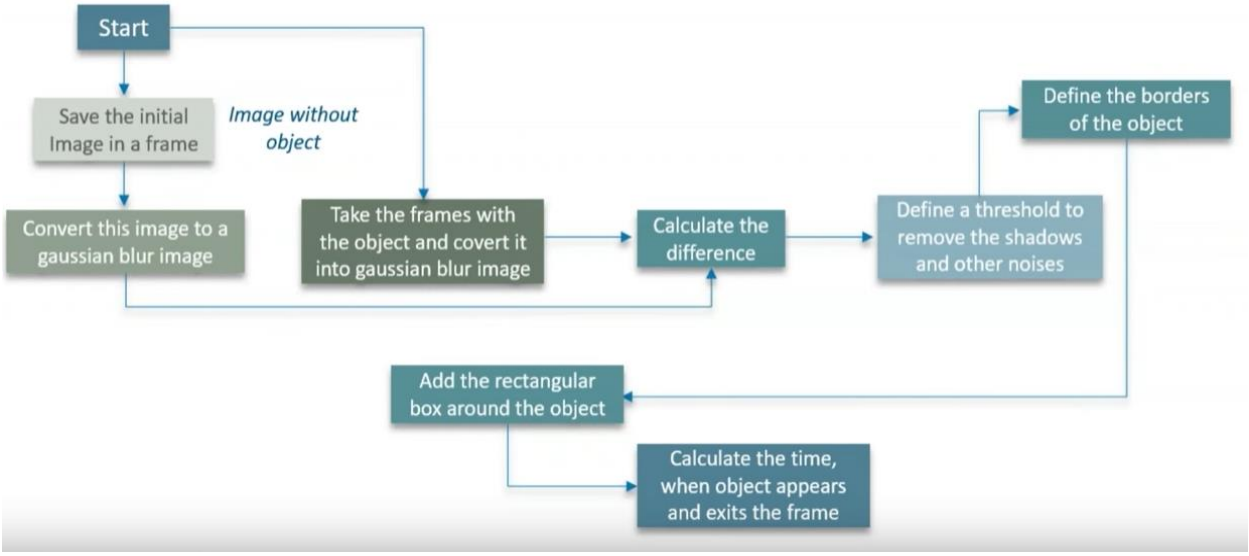

Fig 1:- Process Flow

First of all, we need to save the initial frame. The moment we switch on the camera the first frame the first image that will appear, we will save it. Then we'll convert that image to a Gaussian blur image. we'll take the frames with the object and converted into Gaussian blur image so basically this is done to give us the accurate results. we'll estimate the variance between the initial frame and the frames that will appear after the first frame since the first frame is stored already. then we are going to calculate the time an object appears and exits the frame. we'll save that in a data frame and we are going to visualize the data frame.

III. DEFINING A FILTER

A. Defining a Target

A Target is defined as an object inside the image, information of which we wish to retrieve. In this case our Target is the tank, refer the below image. A target can be anything, it could be a vehicle, or a fixed base. Comparing grayscale images or defining a filter that matches any particular object is a better option to detect something that's unwilling.

B. Creating a Target Filter

A target filer is a replica or a duplicate image of our desired target. As said, it can be anything. This must be something that we want to detect in our entire operation. Taking numpy ndarray of our target filter and numpy ndarray of the image taken can help us find our target.

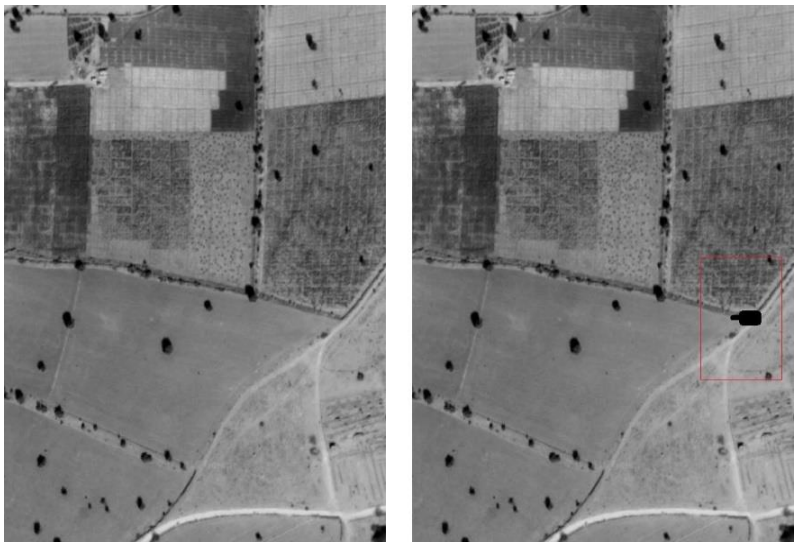

Fig 2:- Shows a Target Being Detected by the Camera

After successfully conversion of our arrays, one way to find our target is to compare both the arrays. Since this technique does not completely gives a desired output, other method is dimensionality reduction.

IV. DEEP LEARNING & CONVOLUTIONAL NEURAL NETWORK

A. Dropout Architecture of ConvoNet

In deep learning, CNN or ConvoNet is a feed forward artificial neural network, mostly pragmatic to analysing filmic imagery. CNN uses a difference of multilayer perceptron intended to entail minimal processing also acknowledged as Space Invariant Artificial Neural Networks (SIANN). CNNs use fairly a minute pre-processing algorithm as compared to other image sorting algorithms. These resources that the network studies the filters that in old-style algorithms were hand plotted. This is a major advantage. We want three straightforward components to outline a basic convolutional network:

- Convolutional Layer
- Pooling Layer
- Output Layer

- *Convolutional Layer*

Assume we have an image; we describe a weight matrix which extracts certain features from the images. This weight shall now run across the image such that all the pixels are visited at least once to give a convolved output. The weight matrix acts like a filter in an image extracting exact information from the original image matrix. A weight combination might be extracting edges, another one might a certain colour while another one might just blur the undesirable noise. The weights are cultured such that the loss function is minimised comparable to an MLP. Therefore, weights are cultured to extract features from the original image which support the network in accurate prediction. When we have several convolutional layers the original layer extract more generic features whereas as when the network gets deeper, the features extracted by the weight matrices are more and more composite and more suitable to the problem at hand.

- *Pooling Layer*

Every so often when the images are too huge, we need to diminish the number of trainable parameters. It is then anticipated to occasionally introduce pooling layers between succeeding convolutional layers. Pooling is done for the only purpose of reducing the spatial size of the image. Pooling is done autonomously on each depth dimension; therefore, the depth of the image remains unaffected. The most public form of pooling layer commonly applied is the max pooling.

- *Output Layer*

After multiple layers of convolution and padding, we need the output in the form of a class. The convolutional and pooling layer would solitarily be able to extract features and reduce the number of parameters from the original images. However, to produce the final output we need to apply a fully connected layer to generate an output equal to the number of classes we need. It turns out to be tough to reach that number with just the convolutional layers. Convolutional layer generates 3D activation maps, while we just need the output as whether or not an image belongs to a particular class. The output layer acts as a loss function like categorial cross entropy, to compute the fault in prediction. Once the forward pass is complete the back-propagation commences to apprise the weight and biases for mistake and loss reduction.

V. MOTION ANALYSIS

Numerous errands related to gesture approximation where an image sequence is managed to harvest and estimation of rapidity either at each point in the image 3D scene or even of the camera that produces the images.

- Egomotion: Defining the 3D rigid gesture of the camera from an image arrangement formed by the camera.
- Tracking: Following the actions of a reduced set of interest points or items in the image sequence.
- Optical flow: To govern each point how that point is moving absolute to the image plane. This motion is a outcome of both how the equivalent 3D point is moving in the act and how the camera is moving absolute to the scene.

VI. DECLARATIONS

These targets are set solely for the purpose of research and study. The aim of the project is to find new ways of getting the same solution is a better way and achieve excellent accuracy. The cameras are either fitter on a drone or a helicopter which takes images continuously and are available at ground in our datasets. There are quite a few things we need to take care of for entire process to run successfully. One such factor is weather conditions. This project was carried out in clear weather and in day time. For night time detections, night vision cameras were added to the prototype. Though accuracy achieved highest was during day time, it was merely less in dark.

VII. CONCLUSIONS

With this approach of Machine Learning, images can be easily classified and information be extracted. Similar models can be built to achieve more accuracy with enhancement in parameters and hardware. The above experiment requires a computer with good specifications for making things faster.

REFERENCES

[1]. J. K. Aggarwal and R. 0. Duda, Eds., "Special issue on digital filtering and image processing," IEEE Trans. Circuits Syst., vol. CAS-2 pp. 161-304,1975.
[2]. G. J. Agin and T. 0. Binford, "Computer description of curved objects," in Proc. 3rd Int. Joint Conf. Artificial Intelligence, 1973, pp. 629-640.
[3]. G. J. Agin and R. 0. Duda, "SRI vision research for advanced industrial automatic," in Proc. 2nd USA-Japan Computer. Conf., Aug. 26-28, 1975, Tokyo, Japan.
[4]. H. C. Andrews, Computer Techniques in Image Processing. New York: Academic, 1970.
[5]. Introduction to Mathematical Techniques in Pattern 1343 IEEE TRANSACTIONS ON COMPUTERS, DECEMBER 1976 Recognition, New York: Wiley, 1972.
[6]. H. C. Andrews, Ed., "Special issue on digital picture processing," Computer., vol. 7, pp. 17-87, May 1974.
[7]. H. C. Andrews and L. H. Enloe, Fjds., "Special issue on digital picture processing," Proc. IEEE, vol. 60, pp. 766-898, July 1972.
[8]. A. G. Arkadev and E. M. Braverman, Learning in Pattern Classification Machines. Moscow: Nauka, 1971.
[9]. D. I. Barnea and H. F. Silverman, "A class of algorithms for fast digital image registration," IEEE Trans. Computer., vol. C-21, pp. 179-186, 1972.
[10]. B. G. Batchelor, Practical Approach to Pattern Classification. New York: Plenum, 1974.