Cartoonification of Human in an Image

Shariq Suhail .Md[1], *Likhith Lanka*[2], *Giri Pranay*[3] and *Bhupathi Badhavath*[4]
Indian Institute Of Information Technology Chittoor, Sri City, A.P., India

***Abstract*— With the advent of social media and the number of users increasing rapidly, cartoonizer softwares have gained a lot of popularity. Cartoonish generally images have more saturated colours and are sharp. We have implemented a basic algorithm which upon taking an input image detects bounding boxes around humans and then applies the cartoonizer algorithm. To detect bounding boxes, we have used pre-trained YOLO object detection network. The basic cartoonizer algorithm takes use of adaptive filtering and bilateral filtering to produce a sketch like image.**

Keywords: **Adaptive filter, Bilateral filter, YOLO detector.**

I. Introduction

In the recent years we have seen a number of cartoonizer softwares been released. People have started using such softwares to obtain a cartoonish effect for their images and no doubt that such softwares gained a lot of attraction within a few span. Getting cartoonish effect will mesmerize the users, but what goes behind is some basic yet effective image processing techniques.

The cartoonish images have the following two properties. First is that, they have more saturated colours than any real life image. The second property being that they have more sharper edges. The quality of a cartoonish image is defined by looking at how well an image meets the above mentioned criteria. The aim of this project is to cartoonify human in an image. Hence, detecting the humans in a given image becomes the preliminary task. There has been a great deal of study in the object detection methods. Many algorithms have been proposed to effectively detect objects in real time scenarios. One of such deep learning based object detection network is YOLO which stands for You Look Only Once. Some of the earlier methods like Fast RCNN, Faster RCNN employ an additional regional proposal network which makes them slower. Though the accuracy of YOLO detection network is somewhat less than SSD, it is much faster which is the reason for us to choose YOLO over the other netrworks. Since our project is mainly focused on image cartoonification, we have used a pre-trained YOLO network.

Once bounding boxes are detected, we apply our cartoonification algorithm on each of the bounding boxes to generate desired output. The cartoonification algorithm has the following two sub tasks. 1) Detecting and boldening of the edges from the given image. 2) Smoothing and quantizing the colors in the image. After performing the above two steps, we combine the results produced by each of them to generate the final output.

The algorithm which is proposed is able to produce artistically and comically appealing results on wide variety of input pictures. However, not all the images would result in equally satisfying results. The paper is divided into sections as follows. The following section will describe in detail about the cartoonification algorithm. In section III we will describe briefly about the YOLO architecture. Section IV will show the results achieved on various inputs. Finally, section V will conclude the paper.

II. Cartoonification Algorithm

The cartoonification algorithm is divided into two sub-tasks. Firstly, we have to detect and bold the edges. Secondly, we have to smooth the image by preserving the edges to form homogeneous colour regions.

A. Detection and boldening of Edges

One of the characteristic of cartoonish images is that they are sharper than ordinary image. In this part of the algorithm, we will detect edges and thick their edges.

1) Median Filter: The input image we want to cartoonify may have some salt an pepper noise. In order to reduce such noise we are applying median filter. The purpose of the median filter is to replace the centroid value with the median value. Performing median filter essentially forms a pre-processing and smooths out any extreme specks if present. It helps in improving the results of the later processing. We have chosen the filter size to *7X7*. Hence RGB pixel values at the centroid are replaced with the median of the 49 RGB pixel values from the local neighborhood. Special care has to be taken in selecting the filter size because too large filter size will fade out the edges.

2) Adaptive Filtering: A system with a linear filter which has a transfer function controlled by variable parameters and those parameters can be adjusted according to an optimisation algorithm is called as adaptive filter.Almost all adaptive filters are digital filters since the difficulty in implementation of optimisation algorithm is very high. Adaptive filters are used for applications where the parameters of the transfer function are not known in advance or are changing.Due to rapid development in digital signal processors,adaptive filters have been used everywhere and are now commonly seen in devices like mobile phones,digital cameras and medical monitoring element.

In our scenario,the main aim is detecting edges and boldening them.We use adaptive filter to retain edges present in an image.Detection of edges has been done as of now. Now,to thicken the edges we use adaptive thresholding.Adaptive thresholding is different from conventional thresholding.Usual thresholding algorithm scans an image and makes all the pixels whose intensity values are above a threshold are set

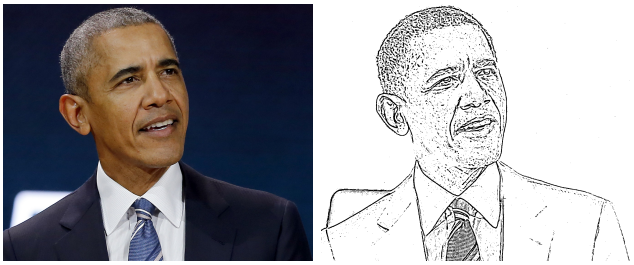

Figure 1. Original Image, Image Produced After Adaptive Filtering

to high and all the remaining pixels to a low value.Adaptive thresholding does not use global threshold for all the pixels, instead it changes threshold dynamically over the image.The input of adaptive thresholding is a grayscale or colour image and output is a binary image that is segmented.A threshold has to be calculated for all the pixels in an image.There are two main approaches for finding threshold:(i) the Chow and Kaneko approach and (ii) local thresholding.The approach of both the ideas is that smaller regions of image are likely to have approximately uniform illumination,thus helpful for thresholding. Chow and Kaneko divide an image into an array of overlapping subimages and then find the optimum threshold for each subimage by investigating its histogram. The threshold for each single pixel is found by interpolating the results of the subimages. The drawback of this method is that it is computational expensive and, therefore, is not appropriate for real-time applications.An alternative approach for finding local threshold is to statistically examine the intensity values of the local neighbourhood. Simple and fast functions include the *mean* of the local intensity distribution or median of the local distribution or the average of minimum and maximum values.

B. Homogeneous Color Regions

The other important aspect of cartoonizing images is to create homogeneous colour regions. In this part of the algorithm, we will smooth the image continuously to create homogeneous color regions.

1) Bilateral Filtering: To create homogeneous colour regions, we have to smooth the image. If we use smoothing filters like Gaussian blurr for the purpose of creating homogeneous colour regions, we would end up blurring the edges also. Since when the the center of the filter is on the edge, the center value is replaced with weighted average of surrounding pixel values. The best filter for this kind of purpose is bilateral filtering. The drawback of using bilateral filtering is that, it is extremely slower than Gaussian filtering. As the filter size increases, time taken by bilateral filtering also increases rapidly. One way to overcome this is to down sample the image both in x and y direction and then apply bilateral filtering with smaller filter size. In our approach we down sample a given image using gaussian pyramid by a factor of 4. Another advantage of bilateral filter is that, it preserves the edges even if applied multiple times. So, instead of applying larger filters for fewer iterations we applied filter of size *9x9* for more number of iterations. Generally the number of iterations has to be tuned properly for each image. The bilateral filter equation is given below

$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}\big(\,\|p-q\|\,\big) G_{\sigma_r}\big(|I_p - I_q|\big) I_q \qquad (1)$$

In the above equation, suppose the filter center is at pixel p and S be the filter coordinates. G_{σ_s} gives space weight, G_{σ_r} accounts for range weight and W_p is a normalization factor. Here instead of assigning a weighted sum of pixel values as done in gaussian filter, we adjust the weight by not only considering how far a pixel is from center coordinate but also include how different the pixel is in terms of intensity. If a pixel which is close in colour with the centroid pixel will have more weight than a pixel which is at the same distance but having different intensity. Performing this additional step plays an important role and helps in preserving the edges in the image. After the filtering is done, the image is again restored to its original size.

Suppose the filter is applied on a pixel of an edge. The space weight term gives some weight by considering the distance between the center pixel and the current pixel. As we are additionally multiplying by range space term, which gives a very less weight because all the surrounding pixels will have high intensity as compared to edge pixel. This brings down the overall weight and hence will not smooth out edges.

If the weighting functions are gaussian functions then the overall weight assigned will be calculated as given below

$$w(i,j,k,l) = \exp\left(-\frac{(i-k)^2+(j-l)^2}{2\sigma_s^2} - \frac{|I(i,j)-I(k,l)|^2}{2\sigma_r^2}\right) \qquad (2)$$

In the above equation, (i,j) are pixel coordinates of the filter center p and (j,l) be coordinates of neighbouring pixel q. σ_s, σ_r be guassian filter parameters of space weight and range weight respectively.

C. Recombine

The outputs generated after adaptive thresholding and adaptive filtering are recombined to give the final output. This is done by performing bitwise AND operation on the two outputs.

Figure 2. Original Image, Image Produced After Bilateral Filtering

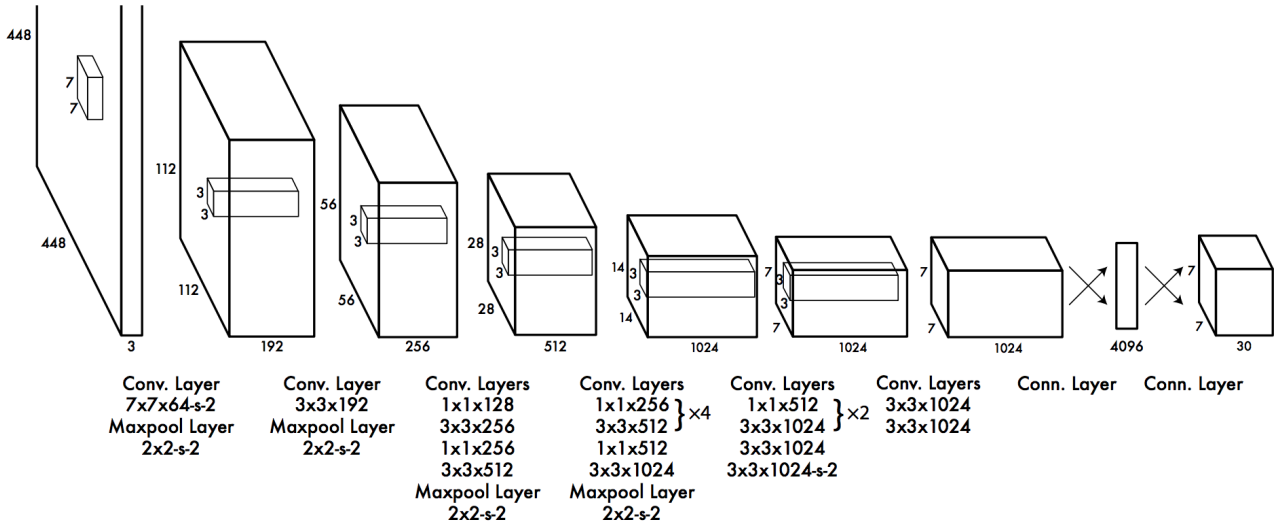

Figure 3. Original Image

III. YOLO Architecture

YOLO (You Only Look Once): Redmon presented for the first time a single stage method for object detection where raw image pixels were converted to bounding box coordinates and class probabilities and can be optimized end-to-end directly. One-shot detection allows to directly predict boxes in a single feed-forward pass without reusing any component of the neural network and also avoids region proposal generators which require huge computational resources, thus speeding up the detector.

YOLO considers a small set of candidate regions and directly predicts detections. Unlike region-based approaches, e.g. Faster RCNN, that predict detections based on features from the local region, YOLO uses the features from the entire image globally. In particular, YOLO divides an input image into a S S grid. Each grid predicts C class probabilities, B bounding box locations and confidences scores for those boxes. These predictions are encoded as a SS(5B+C) tensor. By throwing out the region proposal generation step entirely, YOLO is fast by design, running in real time at 45 FPS and a fast version, i.e. Fast YOLO, running at 155FPS. Since YOLO sees the entire image when making predictions, it implicitly encodes contextual information about object classes and is less likely to predict false positives on a background.

YOLO makes more localization errors because of the coarse division of scale, bounding box location and aspect ratio. YOLO may fail to localize some objects, especially small ones, possibly because the grid division is quite coarse, and because by construction, each grid cell can only contain one object. YOLOv2: Redmon and Farhadi proposed YOLOv2, an improved version of YOLO, in which the custom GoogLeNet network is replaced with a simpler DarkNet19, plus utilizing a number of strategies drawn from existing work, such as batch normalization, removing the fully connected layers, and using good anchor boxes learned with kmeans and multiscale training. YOLO predictions are made on a 13x13 feature map. Due to its small feature map size, it fails to detect smaller objects. In YOLOv2 higher layers features are concatenated lower level features through a passthrough layer. After concatenation detections are made on 26x26 feature map. So, YOLOv2 is able to detect much smaller objects. YOLOv2 achieved state of the art on standard detection tasks, like PASCAL VOC and MS COCO.

IV. Results

We have tried giving various inputs to the cartoonification algorithm having one person and more than one person in it. Specific goals were set like; the generated image must have solid contours and also to achieve large regions of homogneous colours with a reduced colour palette. Bounding box is given by the yolo object detector for a given input. Cartoonification algorithm is applied only on the bounding box region

Following are the results produced when input image contains only one person.

The proposed algorithm is also able to cartoonify if more than one person is present in the given image. We also tried to check performance on such images. Following were the results produced for an image which has more than one person.

V. Conclusion

The cartoonification algorithm is able to produce artistically and comically appealing results as shown in the above section. We have tested the performance on wide range of input images and got satisfactory results. However, we can not expect the algorithm to perform well on every input image. It was observed that for most the input images which the algorithm failed had high amount of local variation. There are a lot of parameters like kernel sizes of median filter, adaptive filter, bilateral filter and sigmaColour, sigmaSpace, etc that have to be tuned to generate best results. Hence it

Figure 4. Original Image

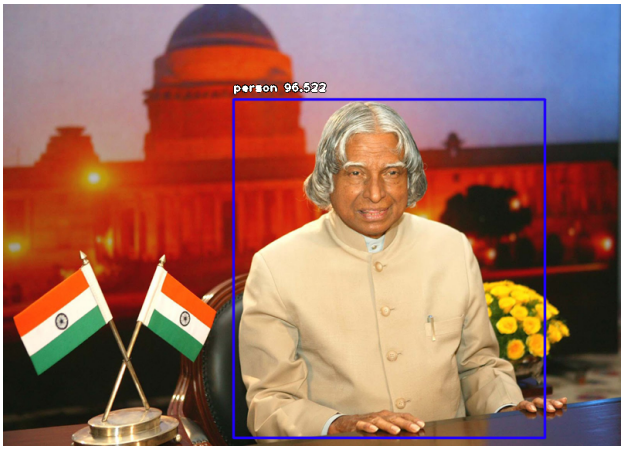

Figure 5. Bounding box detected by YOLO network

Figure 6. Cartoonified Image

would be unrealistic to expect a on-size-fits-all approach to produce satisfying results. The another problem is that there is no proper metric to evaluate the generated output with the input image. Hence most of the time, generated images were evaluated purely based on the gut of the user. We checked if

Figure 7. Original Image

Figure 8. Bounding box detected by YOLO network

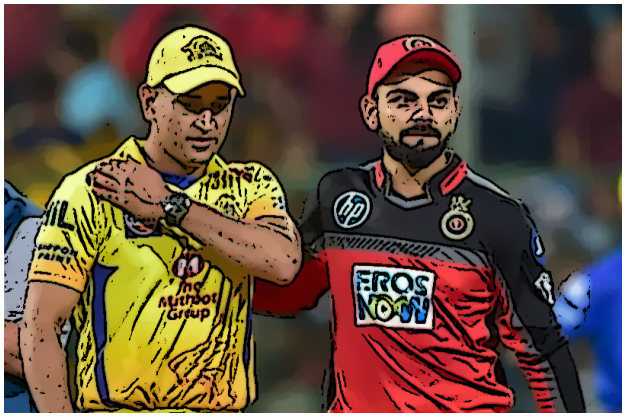

Figure 9. Cartoonified Image

the output image is meeting all the criteria. If the output was not good enough, we would tune some of the above mentioned parameters till the output seemed artistically appealing.

REFERENCES

[1] Kevin Dade, "Toonify: Cartoon Photo Effect Application", *Department of Electrical Engineering, Stanford University.*

[2] Baggio, Daniel. L, *Mastering OpenCV with Practical Computer Vision Projects Packt Publishing Ltd, 2012.*

[3] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi , "You Only Look Once: Unified, Real-Time Object Detection", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.*