**Task1:**

For task one, to perform cartoon effect, I referred and followed few methods mentioned in “Toonify: Cartoon Photo Effect Application” by Kevin Dade.

Method Implemented:

1. Bilateral filter: First, I implemented a Bilateral filter on input image, filtering is important to perform image processing as it remove noise from the given image. Apart from smoothing the image bilateral filter also preserves the edges of the image. It uses both gaussian and range filter to do that.
2. Colour Quantization: In this technique colours of an image are grouped in different regions as per the RGB and then replace them with colour which represent the region. To perform this Uniform quantization has been used. I divided the image colour in four regions [0-63], [64,127], [128,191], [191,255].
   1. Divide RGB component in equal parts.
   2. For every pixel in image assign it to its RGB region.
   3. Determine the colour by taking the mean of all values in that region.
   4. Iterate over the image pixels and change the colour by representative of that region.

Final output after colour quantization:



\*\*If more regions are selected, we will see the change in effect of the output.

**Tasks2:**

Parametric equation of a circle :

X = a+r\*cos(t)

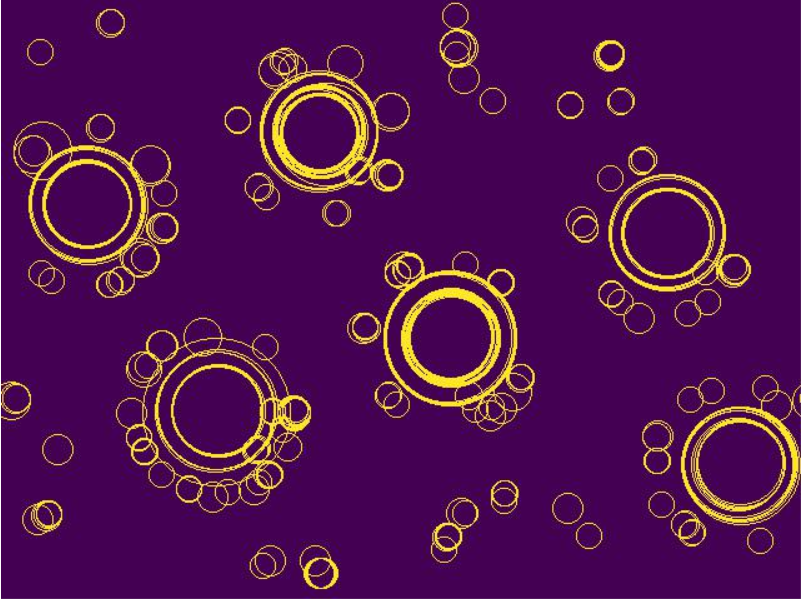
Y = b+r\*sin(t) where t∈[0,2π), r is radius and (a,b) are centre.

Steps for Implementation:

1. All possible circles are defined by values of a,b,r as per the parametric equation of circle.
2. Pixels for each circle can be determined by looping over t (steps). Only those circles will be detected which are between rmin (minimum radius) and rmax (maximum radius). Taking higher value of radius makes the system computationally expensive.
3. First convert the image to grayscale and through canny edge detector create an edge of the input image.
4. Use Hough transformation to determine the centre of the circles. Centre will be chosen where sin and cos value intersects each other.
5. Once edge is created, iterate over only strong pixels, and find the coordinates of centre of the circle that pass by the point (as per the parametric equation).
6. Set a threshold of 40% for circles which are good enough. We need to make sure that circle with common centre are not rejected.
7. Finally to detect or draw the circle, create an empty matrix of the shape of the image and pass the coordinates and radius to skimage.draw.circle\_perimeter and assign each pixel a value of 1.



**Input image**



**Circle detected**

**References:**

<https://stackoverflow.com/questions/58889908/implementing-a-bilateral-filter>

<https://towardsdatascience.com/building-an-image-cartoonization-web-app-with-python-382c7c143b0d>

<https://au.mathworks.com/help/images/ref/imbilatfilt.html>

<https://www.geeksforgeeks.org/python-bilateral-filtering/>

<https://staff.fnwi.uva.nl/r.vandenboomgaard/IPCV20172018/LectureNotes/IP/LocalOperators/bilateralfilter.html>

<https://github.com/adityaintwala/Hough-Circle-Detection>

<https://scikit-image.org/docs/0.14.x/api/skimage.draw.html#skimage.draw.circle_perimeter>

<https://muthu.co/reducing-the-number-of-colors-of-an-image-using-median-cut-algorithm/>

<https://stacks.stanford.edu/file/druid:yt916dh6570/Dade_Toonify.pdf>

<https://stackoverflow.com/questions/47566093/computer-vision-counting-small-circles-in-an-image>

<https://stackoverflow.com/questions/38619382/how-to-count-objects-in-image-using-python>