International Journal for Modern Trends in Science and Technology, 7(05): 63-71, 2021

ISSN: 2455-3778 online
DOI: https://doi.org/10.46501/IJMTST0705010
Available online at:http://www.ijmtst.com/vol7issue05.html

Image Cartoonization

Nikhil Nautiyal[1] | Veernarayan Sinha[1]| Savleen Kaur[2]

[1]Student Research Scholar, Department of Computer Science and Engineering,Lovely Professional University.
[2]Assistant Professor, Department of Computer Science and Engineering, Lovely Professional University.

To Cite this Article
Nikhil Nautiyal., Veernarayan Sinha & Savleen Kaur. Image Cartoonization. *International Journal for Modern Trends in Science and Technology* 7, 63-71 (2021).

Article Info
Received on 16-April-2021, Revised on 25-April-2021, Accepted on 02-May-2021, Published on 08-May-2021.

ABSTRACT

This paper presents an approach to cartoonize digital images into cartoon-like. The method used is different from as others used previously. This paper focuses on various techniques involved during the whole process, which, when used on layer by layer, gives an appropriately balanced output. We tend to explore different functions that can be integrated together in a particular pattern to get a filtered and composed output. The mathematical approach to different functions has also been explicated and the working is explained in detail. This system aims to use the filters' full functionality, which will help both in application and research, and can work as a framework to any computer vision-related systems and can be improved and embedded with other systems to work both as an independent module or as an integrated system.

KEYWORDS:*Computer vision, image cartoonization using python and OpenCV, using filters to cartoonize image in python, Bilateral, Gaussian, Pencil Edge, Pencil sketch, Laplacian, Median filters*

I. INTRODUCTION

Cartoons are illustrations of characters that can be imaginary or are based on a personality. They are semi-realistic or non-realistic, which are popular nowadays to depict a situation or a happening in a funny, satirical way.

One of the earliest examples of traditionally animated movies, Fantasmagorie (1908), in which all the movie frames were hand-drawn, and till years the animation culture with hand-drawn frames. Walt Disney caught the race with their remarkable cartoon series and took the animation cartoons to the next level.

Earlier, cartoonists used to draw these cartoons by hand, and when 'Anime' started catching the light, it became difficult for them to attract each movie frame by hand, which took a lot of time and was not reversible if met with a mistake.

With increasing technology, numerous software came into the market to digitally draw the illustrations, thus reducing human effort and were more efficient than being hand-drawn by artists. Which also, in time, improved with numerous features introduced. Toy Story (1995), the first fully computer-animated feature film, was a huge success. It portrayed the characters as very interactively stunning, and the excellent animation made it very life-like.

Automation technology is booming nowadays. Human effort is reduced to the minimum; with a set of programs doing our job for us, animation technology has taken a considerable leap. Our job is to make those programs. So, this is the project which converts regular camera clicked images into cartoonized form,

created using Python language and relative libraries. In times like today, where technology is taking giant leaps with great minds improving them day by day, the animation industry's main problem is the quality, with 4K, IMAX, and all other super high-definition movies catching popularity. Their camera being different exclusively, the

animation of these movies while maintaining quality is a huge task. Toy Story 4, The Garden of Words being an excellent example for being one of the most detailed animated movies ever made. Anime movies take much longer than regular movies in times of VFX effects and animations. The workforce, skillset, and resources in terms of equipment and investment are more than classic movies, so time plays an essential role in creating these movies. For saving time, automation is the best way to go; with the improvement in machine learning and artificial intelligence, every production process has improved significantly and improves the quality of the product.

Playing with images is also a 'fun' activity, and lately, image manipulation has been blowing up social sites, with many people making careers in image manipulation and enhancement. The OpenCV library works on the same principle, and here we get to know the backend procedures of image processing and understand how computer vision works. These filters and techniques are also used to improve lost details on older photos, and with a combination of some filters, a black and white image can also be converted into a colored one.

Although digital animation has more audience, there are still millions of people who still purchase comic books. Therefore, there is still a vast market for paper comics because why not? The traditional way of enjoying cartoons is always fun.

There are numerous ways to convert a specific photo to a cartoon one, but the best way is to do manual with software like Adobe Photoshop, and not everyone has their hands on tools like these, so the users there are limited. Although the output obtained through the manual process is the best, it is time-consuming, and doing it for a large number of images is not efficient at all. Therefore, systems like these are a go-go for every field as it provides output in seconds with upgrades and everything, the scale will surely improve with even better results.

STRUCTURE OF PAPER

The paper is organized as follows: In Section 1, the introduction of the paper is provided along with the structure, important terms, objectives and overall description. In Section 2 we discuss related work. In Section 3 ,technical approach used in the system is explained. Section 4 states the problem statement, which was tried to unravel in the system, Section 5 explains the implementation of system, Section 6 shows the experiment done to test the system on various kinds of images, Section 7 shows the post implementation, Section 8 shows the flowchart and followed by Section 9 with the conclusion and future scope of the project.

OBJECTIVES

This project plans to create an application with a simple user interface allowing users to apply the cartoon filters to images of their choice. The filters are designed to provide artistically and comically appealing results on a wide range of pictures. The system heeds to focus on the program's simplicity as the code is written in Python language, which is considered the most' fun' and easy to understand yet with a wide range of applications in every field. The interface is not bound to be used by any age group and any system or service providers. It can be easily accessed from any device with a browser with an internet connection when hosted online.

II. RELATED WORK

Generative adversarial networks (GANs) [1,2,3] are primarily used in unsupervised machine learning. GANs can be used in different applications like image super-resolution, image inpainting [4], semantic image editing, image cartoonization [6], image synthesis, image colorization [7], classification, and style transfer [5]. GANs proposed a generative model for a natural image that evolves to generate more realistic-looking data.

Image Smoothing [8, 10, 9, 11] is used to produce an image with less pixel and reduce the noise of an image. Smoothing is also usually based on a single value representing the image, such as the average value of the image or the middle value.

Early used methods are optimization-based methods and filtering-based [12]. Xu and Fan [13, 14] proposed end-to-end networks for image smoothing. Bi et al. proposed an L1 transformation for the image flattening problem. Min et al. introduced image smoothing by reducing a quadratic energy function.

Rendering in computer graphics means the process by which a virtual scene is converted into an image. In photorealistic rendering, the pixels are imaged compared to non-photorealistic rendering, where the line art, outlining, shading, distortions, and other techniques are used. A variety of methods have been developed to create images with flat shading, mimicking cartoon styles. Such methods use either image filtering [15] or formulations in optimization problems [16]. However, turning existing photos or videos into

cartoons, such as the problem studied in this paper, is much more challenging. Non-photorealistic Rendering transforms a photograph into what looks like a painting. It represents image content with different artistic style like watercolor [22], pencil sketch [19, 18], paints [17, 20]. NPR methods are used in filtering-based method [23], image abstraction [24, 21], end-to-end neural network [6], use cases of videos [25], photos [6], and portraits [26].

Superpixel [27, 31, 30, 28] is a set of image pixels that share similar visual characteristics and perceptually meaningful. Features from superpixels may improve the performance of image quality assessment. Gradient ascent-based algorithms [29, 32] maximize a function. In a lot of optimization, we reformat maximizing a function as minimizing the negative of that function. Some superpixel algorithms are similar between pixels as edges, treating pixels as nodes, graph-based algorithm.

III. TECHNICAL APPROACH

Filtering an image is a necessary process done in image processing. It can be done for blur removal, noise removal, edge detection, etc. Linear and a non-linear algorithm are used for filtering. An appropriate filter should be selected for any specific purpose. If the input image has a high magnitude and the amount of noise is low, it is considered a non-linear filter. And if the input image has a low magnitude and the amount of noise is high, it is regarded as a linear filter. Linear filters are the most frequently used filters as it is most straightforward and fastest. The algorithms used for linear filters are Gaussian Filter, Laplacian Filter, and non-linear filters: Median filter, Bilateral filter.

Gaussian Filter

Gaussian Filter - Gaussian Filter is a linear operation. However, it does not preserve edges in the input image - the value of sigma governs the degree of smoothing, and eventually, how the edges are preserved. It is used to blur the image and remove noise and detail. In one dimension, the Gaussian function is:

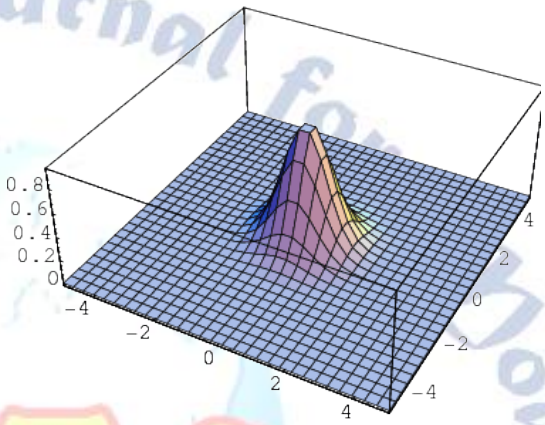
In 1D:

$$g_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

In 2D:

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

The distribution, σ is assumed to have a mean of 0 Shown graphically –.



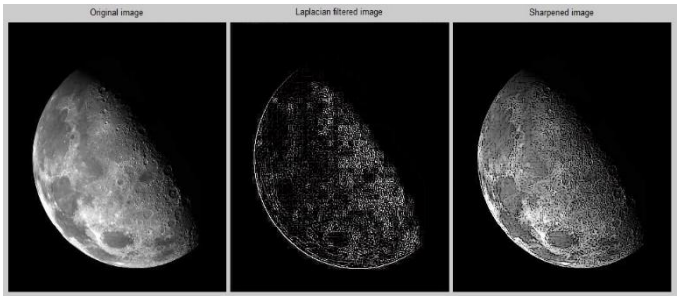
In pencil Sketch, We have used the gaussian blur technique with 25 x 25 pixels by default, and the default sigma values filter on the image to smoothen our image. By increasing the filter size, we can create thin lines for our sketch, and it is used to reduce the noise in the picture.pdftotext is part of the Xpdf software suite. Poppler, which is derived from Xpdf, also includes an implementation of pdftotext. On most Linux distributions, pdftotext is included as part of the poppler-utils package.

Laplacian Filter

A Laplacian filter is an edge detector used to compute the second derivatives of an image, measuring the first derivative's change rate. This determines if a change in adjacent pixel values is from an edge or continuous progression. The Laplacian method is used in a linear differential equation with the help of a second-order derivative.

$$\nabla \cdot \nabla\psi = \frac{\partial^2\psi\partial}{x^2} + \frac{\partial^2\psi\partial}{y^2}$$

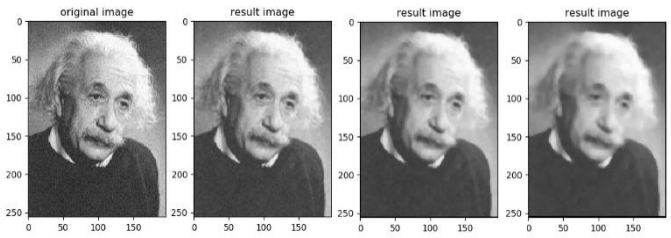
Where ψ denotes the image.



Laplacian filter kernels usually contain negative values in a cross pattern, cantered within the array. The corners are either zero or positive values. The center value can be either negative or positive. The following array is an example of a 3x3 kernel for a Laplacian filter.

Median Filter

It is a non-linear filtering technique used to remove noise and improve the edge detection result of an image. And preserve the edges of an image during noise removal. Median filters are suitable for handling extreme outlier, which would skew an average. A classic use of the median filter is for removing salt and pepper noise. Intensity calculation of median filter is done by replacing the central pixel of an image with the median value from a sorted window. In this, a positive odd integer is used for the kernel size.



Picture III.(1) Application of median blur on an image

Bilateral Filter

It is a non-linear filter technique used for noise-reducing and edge-preserving of an image. It preserves the sharp edges of an image. This calculates the weighted average with the help of Gaussian distribution, which depends on Euclidean distance and radiometric differences. And weighted average replaces the intensity of each pixel value. The cv2.bilateralFilter() operation is a little slower compared to other filters.

Original

Bilateral

IV. PROBLEM STATEMENT

The cartoonization of an image depends on the type of method used in the system. Several ways are used for the same process. The most often use methods are– GAN (Generative Adversarial Networks), a machine learning framework that generated new data based on training data, and OpenCV library, which we are using on this system.

The current system works on OpenCV library and corresponding filters that are applied in the input image. Many filters can be used independently or can be added with others to create a custom filter. The most common and widely known filters are medianBlur(), GaussianBlur(), Laplacian(), BilateralFilter() and many more. These filters are best productive when used collectively with each other. Implementing a single filter with limited features will give output but certainly not an acceptable one.

In this system, there are certain combinations of filters used to carry out the cartoonization process.

What is new in the system to be developed?

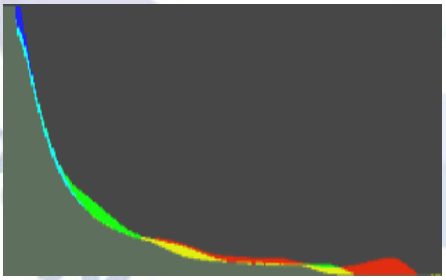
The project, Image Cartoonization is creative, more flexible, and customizable in a sense. This web app provides users to make their personalized cartoonized images as per their choice based on different filters provided. Here, a user can upload or click a picture of his own and then manipulate various adjustments with given sliders before opting for the final image. Each filter has 2-3 sliders with clearinstructions to have a clear understanding.

V. IMPLEMENTATION

The primary function is starting of the system. In this function, there is a sidebar. Streamlit provides you sidebar select box function to easily create one. In the sidebar, there are some values, and with each of the values, there is a function bundled. When a user clicks on any one of them, the corresponding process is triggered. By default, the first value is the Pencil Sketch string is selected, and the 'if' statement calls the PencilSketch() function with st.title, we can create a bold title. And with st.image we can display any image on our streamlit app.

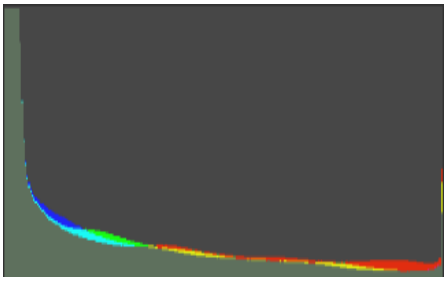
Then, the section containing elements for processing image here, Pencil Sketch, Detail Enhancement, Pencil Edge, and Bilateral filter, are displayed. There is a slider to change the value of the different filters as per user convenience. For an interactive slider st.slider function is used in Streamlit. Now there is a widget to be added where user can select their own photos from their local system, streamlit.file_uploader() function is used here; this widget allows user to select their image by browsing or dragging and dropping the image into the box-covered area around the button. For adding texts like messages or any important point to the app streamlit.write() function is used. Once the user is done with selecting the image, the next step is to display the image, and streamlit.image() function is used here for the same with corresponding parameters.

Below is a sample of some filters and their work explained.



Picture 1. (a) Picture 1. (b)

Picture 1.(a) shows the original image with the histogram shown. As we can observe, the image here is a dark tone photograph with not many details on the darker portion of it. The RGB histogram (Picture 1.(b)) shows no pointers on the right side, on any color except a tad bit of red.

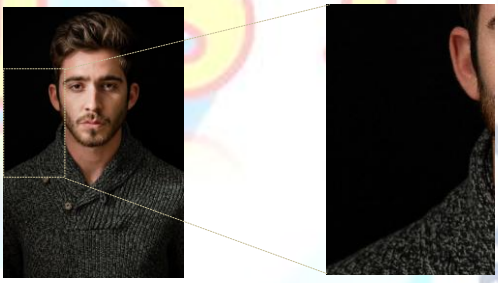


Picture 2. (a) Picture 2. (b)

The Detail Enhancement filter has been used in the image (Picture 2. (a)) to sharpen out the tiniest bit of details hence bringing a grainy image overall, which is a starting step for creating a final cartoonized image output.

The histogram (Picture 2.(b)) here is clearly more balanced here than the previous one, with details improved on the darker side.

There are certain things to keep in mind about to keep in mind while using the detail enhancement filter. A perfect photo is one that has the subject clearly separated from the background and has all the lights and metadata balanced and not undermined or over-toned. The slider must be used carefully as it is the small details that affect the final product.



Picture 3. (a) Picture 3. (b)

Picture 3.(a) shows the input image, and 3.(b) shows an edge of the image. A high-quality image is chosen for this example.

The output clearly depends on the input image as specified earlier; the higher the resolution, the more powerful will be the edge detection power of the filters.

Picture 3. (c) Picture 3. (d)

In Picture 3.(c) the Pencil Edge filter has been applied with values modified from the slider. Picture 3.(d) shows the edge catching accuracy of

the application where it perfectly catches even the edge with smallest bit of detail. The power of edge detection can be increased or decreased with the filter. But the noise factor is to be kept in mind while adjusting the filter because higher the power, more unnecessary edges will show up.

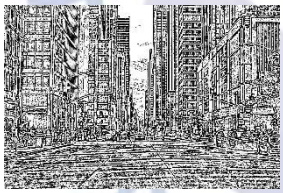
Streamlit app applies some lossless conversions to image but the quality of image is not affected at a large extent as shown in Picture 3.(d). The compression applied to image is around -2.0 to -3.0 times for larger size image either portrait or landscape.

VI. EXPERIMENT

1. The first testing is of a landscape image. The image has a lot of objects, but the composition of buildings and perspective creating a horizon and the leading lines makes the picture look very eye catchy and dynamic. The system works best for cityscape photos as concrete objects go very well with the filters, which makes the output look much cleaner and more 'cartoonified.'


Original Image

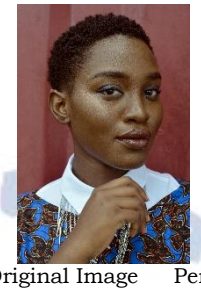

Pencil Sketch


Pencil Edge


Detail Enhancement


Bilateral

2. For the second test, a portrait picture is chosen for variety testing of the system. The subject in this image is of a darker complexion, and there are few light areas in the image as compared to the darker ones. The portrait image is a challenge for the system as it contains a single subject with very great details, and the background also poses as a distraction while tweaking the detail enhancement filter as unexpected noise swells up and bugs the output image (example can be seen in Picture 3.(d)).


Original Image


Pencil Sketch


Pencil Edge

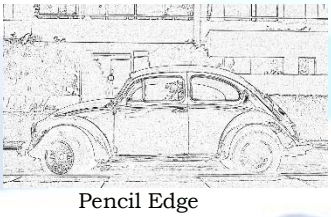

Detail Enhancement


Bilateral

3. The third test image was an abstract object with an abstract background with no dynamic subject and a lesser number of elements, and the system gave a brilliant output on every filter.


Original Image


Pencil Sketch


Pencil Edge


Detail Enhacncement

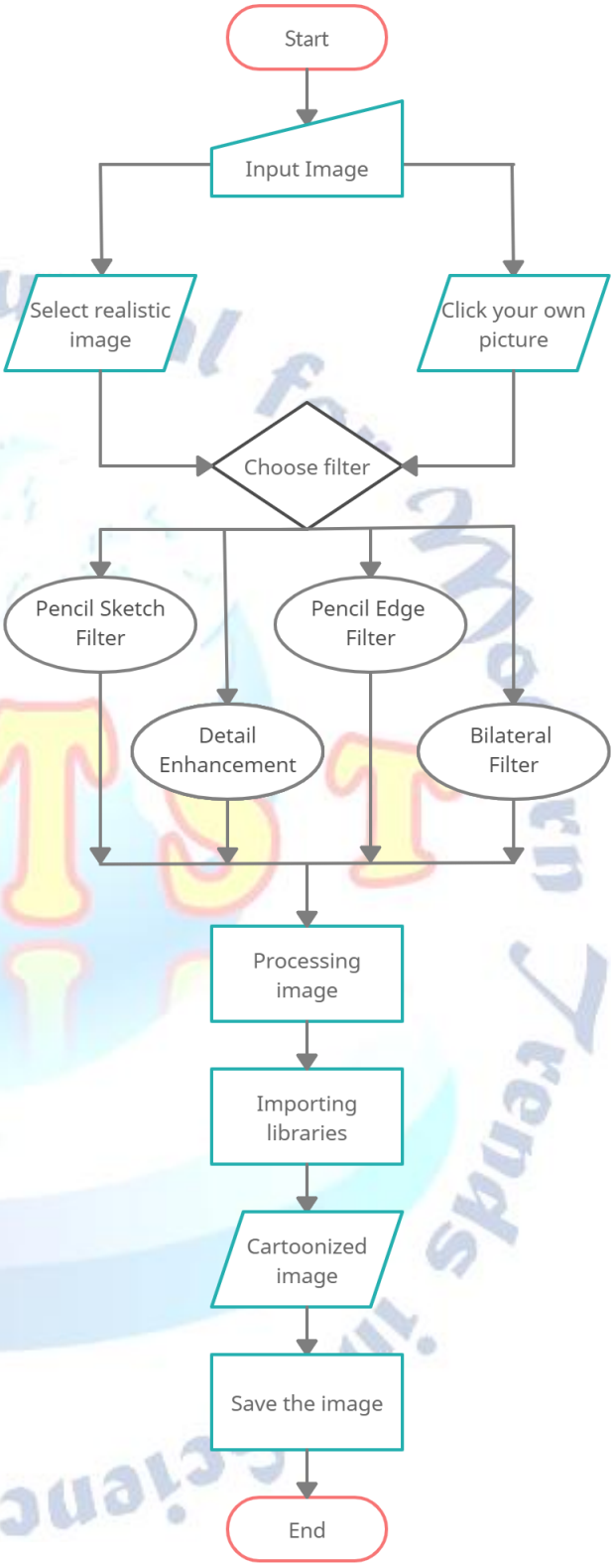

Bilateral

VII. POST IMPLEMENTATION AND SOFTWARE MAINTENANCE

The project objectives are examined and tested, and it is running successfully and converting a clicked photo or uploaded a photo in cartoonized image. All the filters and sliders are working

successfully. We tried to ensure that the user gets the greatest possible benefit from the project.
Software maintenance is focused on upgrading an application to ensure it remains productive and cost-effective. In other words, maintenance is essential to work that needs to be carried out on a web application so it can continue to function reliably and securely.
The best results are always with a realistic RGB labeled picture, the higher the input quality, the higher the output quality.
For example, taking pictures 1.(a) into consideration, for conversion of this particular image, there are certain things to keep in mind, as we can see the subject is light with a dark background, and to get this image as a cartoon, the first thing would smoothen the image up, so the softness slider goes first with acceptable value when we think the image is done with the softness slider the slider to 3-4 points ahead of it because the next step is Detail enhancement filter where we add grains to the image, which will bring out the details in it, hence covering up the extra 3-4 points of softness added earlier. This brings the image look more dynamic and eye-catchy. The output generated is not lossless if the filters are used with proper values, and a high-quality result is provided. The filters and functions used for conversion do not mingle with the quality of the image. Although there are certain compression algorithms that Streamlit uses, but they do not affect the overall quality of the image as the compression is high quality. An example of lossless compression can be Picture 3.(d), as the system is clearly able to identify the edges of contents of the image.

VIII. FLOWCHART



IX. CONCLUSION AND FUTURE SCOPE

In this paper, we proposed an image cartoonization web app that transforms real-world photos into high-quality cartoon-style images. The animation industry is not going to stop now, and the

standards are getting higher by the day, so this system provides room for the addition of features and is easily adjustable to any other source code required for larger modules. Our system can easily be patched up with an automation system and will give expected results image less pixelated.

There are four filters applied: -

1. Pencil Sketch – Coverts the contents of an image as if it were drawn from a pencil.
2. Detailed enhancement – Sharpens the image, adds detailed noise in the image to improve details.
3. Bilateral filter – Smoothen the image by removing noise while keeping the edges sharp.
4. Pencil edge – Converts the image into one which has only significant edges and fills the insides with white color.

The system is a success overall as it gives satisfactory results with a large number of images and with further development the support will increase as we ought to keep upgrading the system to give best results.

The output is largely dependent on the input and experimented and tested; the system works satisfactorily with almost all images. There are certain things to improve in our system:

- The bilateral filter needs improvement. On a good number of images, the filter gives black dots on certain parts of the image.
- The output of the system currently is completely manual, and input is required by the user. We ought to implement machine learning algorithms to automate some actions in the converting process to get better results.
- Currently, the best filters to cartoonize an image are added in the system, but with time, these filters will upgrade, and so will the system.

REFERENCES

[1] https://docs.streamlit.io/en/stable/api.html
[2] http://datahacker.rs/002-opencv-projects-how-to-cartoonize-an-image-with-opencv-in-python/
[3] https://discuss.streamlit.io/t/change-backgroud/5653/6
[4] https://towardsdatascience.com/image-processing-using-streamlit-d650fb0ccf8
[5] https://docs.streamlit.io/en/stable/main_concepts.html
[6] https://medium.com/total-data-science/streamlit-build-your-first-streamlit-machine-learning-web-app-for-your-data-science-projects-e2a6fec99702
[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, SherjilOzair, Aaron Courville, and YoshuaBengio. Generative adversarial nets. In Advances in Neural Information Processing Systems, pages 2672–2680, 2014.
[8] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In Advances in Neural Information Processing Systems, pages 82–90, 2016.
[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Advances in Neural Information Processing Systems 27, pages 2672–2680. 2014.
[10] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2536–2544, 2016.
[11] ArtsiomSanakoyeu, Dmytro Kotovenko, Sabine Lang, and Bjorn Ommer. A style-aware content loss for real-time HD style transfer. In Proceedings of the European Conference on Computer Vision (ECCV), pages 698–714, 2018.
[12] Yang Chen, Yu-Kun Lai, and Yong-Jin Liu. Cartoongan: Generative adversarial networks for photo cartoonization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 9465–9474, 2018.
[13] Lvmin Zhang, Chengze Li, Tien-Tsin Wong, Yi Ji, and Chunping Liu. Two-stage sketch colorization. In SIGGRAPH Asia 2018 Technical Papers, page 261.
[14] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. In European Conference on Computer Vision, pages 1–14. Springer, 2010.
[15] Dongbo Min, Sunghwan Choi, Jiangbo Lu, Bumsub Ham, Kwanghoon Sohn, and Minh N Do. Fast global image smoothing based on weighted least squares. IEEE Transactions on Image Processing, 23(12):5638–5653, 2014.
[16] Zeev Farbman, RaananFattal, Dani Lischinski, and Richard Szeliski. Edge-preserving decompositions for multi-scale tone and detail manipulation. In ACM Transactions on Graphics (TOG), volume 27, page 67. ACM, 2008.
[17] Sai Bi, Xiaoguang Han, and Yizhou Yu. An l 1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition. ACM Transactions on Graphics (TOG), 34(4):78, 2015.
[18] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In International Journal of Computer Vision, volume 98, page 2, 1998.
[19] Li Xu, Jimmy Ren, Qiong Yan, Renjie Liao, and Jiaya Jia. Deep edge-aware filters. In International Conference on Machine Learning, pages 1669–1678, 2015.
[20] Qingnan Fan, Jiaolong Yang, David Wipf, Baoquan Chen, and Xin Tong. Image smoothing via unsupervised learning. In SIGGRAPH Asia 2018 Technical Papers, page 259. ACM, 2018.
[21] H. Winnemoller, S. C. Olsen, and B. Gooch. Real-time video abstraction. ACM Transactions on Graphics, 25(3):1221–1226, 2006.
[22] L. Xu, C. Lu, Y. Xu, and J. Jia. Image smoothing via L0 gradient minimization. ACM Transactions on Graphics, 30(6):174, 2011.
[23] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576, 2015.
[24] Cewu Lu, Li Xu, and Jiaya Jia. Combining sketch and tone for pencil drawing production. In Proceedings of the

Symposium on Non-Photorealistic Animation and Rendering, pages 65–73. Eurographics Association, 2012.
[25] Li Xu, Cewu Lu, Yi Xu, and Jiaya Jia. Image smoothing via l 0 gradient minimization. In ACM Transactions on Graphics (TOG), volume 30, page 174. ACM, 2011.
[26] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In European Conference on Computer Vision, pages 694–711. Springer, 2016.
[27] Henry Kang, Seungyong Lee, and Charles K Chui. Flowbased image abstraction. IEEE transactions on visualization and computer graphics, 15(1):62–76, 2008.
[28] Tom Van Laerhoven, Jori Liesenborgs, and Frank Van Reeth. Real-time watercolor painting on a distributed paper model. In Proceedings Computer Graphics International, 2004., pages 640–643. IEEE, 2004.
[29] Zoya Shahcheraghi and John See. On the effects of pre-and post-processing in video cartoonization with bilateral filters. In Proceedings of IEEE International Conference on Signal and Image Processing Applications, pages 37–42. IEEE, 2013. 3
[30] Jan Eric Kyprianidis and Jurgen D ¨ ollner. Image abstraction ¨ by structure adaptive filtering. In TPCG, pages 51–58, 2008.
[31] Jue Wang, Yingqing Xu, Heung-Yeung Shum, and Michael F Cohen. Video tooning. In ACM Transactions on Graphics (ToG), volume 23, pages 574–583. ACM, 2004.
[32] Ming Yang, Shu Lin, Ping Luo, Liang Lin, and Hongyang Chao. Semantics-driven portrait cartoon stylization. In Proceedings of IEEE International Conference on Image Processing, pages 1805–1808. IEEE, 2010.
[33] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. International Journal of Computer Vision, 59(2):167–181, 2004.
[34] Radhakrishna Achanta, AppuShaji, Kevin Smith, AurelienLucchi, Pascal Fua, and Sabine Susstrunk. Slicsuperpix- ¨ els compared to state-of-the-art superpixel methods. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(11):2274–2282, 2012.
[35] DorinComaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis & Machine Intelligence, (5):603–619, 2002.
[36] Alastair P Moore, Simon JD Prince, Jonathan Warrell, Umar Mohammed, and Graham Jones. Superpixel lattices. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8. Citeseer, 2008.
[37] Greg Mori. Guiding model search using segmentation. In Proceedings of IEEE International Conference on Computer Vision, volume 2, pages 1417–1423. IEEE, 2005.
[38] Andrea Vedaldi and Stefano Soatto. Quick shift and kernel methods for mode seeking. In European Conference on Computer Vision, pages 705–718. Springer, 2008.