CARTOONIFY AN IMAGE WITH OPENCV IN PYTHON

Anand Kumar
Computer science & Engineering
Galgotias University
Greater Noida UP, India
anand.21scse1011570@galgotiasuniversity.edu.in

Dr. Anurag Singh
Assistant Professor
GalgotiasUniversity
Anurag.singh@galgotiasuniversity.edu.in

Mohammad Ali Azhar Khan
Computer science & Engineering
Galgotias University
Greater Noida UP, India
aliazharkhan0786@gmail.com

Abstract **- This paper outlines the development of an image cartoonification transformations through various functions. By leveraging the power of image processing techniques, the proposed application allows users to transform regular images into cartoon-like visuals. This work demonstrates how simple image processing concepts can be combined to create a creative and interactive user experience.**

Keywords Image Cartoonification, Python, OpenCV, Grayscale Conversion, Edge Detection, Bilateral Filtering, Adaptive Thresholding, Computer Vision, Image Processing, Cartoon Effect, EasyGUI, Tkinter, User Interface, Masking Operation, Color Simplification, Image Stylization, Real-time Processing, Image Transformation, Artistic Expression

INTRODUCTION

Image processing has become an integral part of modern applications, ranging from healthcare diagnostics to entertainment. One exciting and creative area within image processing is image cartoonification, where real-life images are transformed into cartoon-like visuals. This technique can be applied for artistic expression, digital art creation, and user engagement in various applications.

The process of cartoonification involves several key steps, such as converting an image into grayscale, enhancing edges, smoothing the image, and applying color simplification to achieve the desired cartoon effect. These operations are made possible through the use of advanced computer vision tools and libraries. In this project, we employ OpenCV, a widely-used library for computer vision and image processing, alongside Python, known for its extensive range of libraries and simplicity in handling complex tasks.

The process of cartoonification is not just about applying artistic filters but involves a well-defined sequence of image transformations. Key steps include converting an image to grayscale to simplify its structure, enhancing edges to highlight contours, smoothing the image to reduce noise while retaining critical features, and simplifying colors to achieve the distinct flat and vibrant appearance characteristic of cartoons. Each step requires precision and is executed using advanced computer vision algorithms, which ensure that the final output retains the charm of hand-drawn art while being generated programmatically.

To implement this, the project utilizes **OpenCV**, a powerful and widely used library for computer vision and Image processing. OpenCV provides a rich set of functions that streamline complex operations like edge detection, bilateral filtering, and color quantization. Additionally, the project is built using **Python**, a programming language renowned for its simplicity and extensive

Fig.1. **Cartoonified Image** .

I. LITERATURE SURVEY

Literature Review The field of image processing has advanced significantly in recent years [1], influencing a range of applications across industries. Among its many facets, image cartoonification has emerged as a creative and technically intriguing application, allowing the transformation of real-world images into cartoon- like visuals. This transformation has been widely adopted in areas such as digital art, entertainment, and personalized user experiences [2]. The journey of cartoonification began with manual artistic efforts but has evolved into sophisticated automated processes powered by advanced algorithms and computer vision techniques. Early Developments and Techniques Initial approaches to cartoonification relied heavily on manual filters and effects applied by artists to emulate the stylistic elements of cartoons. As the demand for automated methods grew, researchers began developing algorithms capable of mimicking artistic techniques. [3] These algorithms incorporated fundamental image processing operations such as edge detection, smoothing [4], and color reduction. Studies such as those by Wang et al. (2013) highlighted the potential of edge enhancement combined with bilateral filtering to create smooth yet detailed cartoon-like visuals.

.

Advancements with Computer Vision Libraries The introduction of libraries like OpenCV revolutionized the field by offering developers tools to implement complex image processing pipelines efficiently [5]. Functions such as cv2.Canny() for edge detection and cv2.bilateralFilter() for smoothing enabled the precise control necessary for cartoonification. Further advancements were made by integrating these methods with real-time processing capabilities, allowing instantaneous transformations of video feeds or images. Researchers have also explored the use of adaptive thresholding to refine edge definitions, ensuring that cartoon-like features are preserved with high fidelity [6]. Machine Learning and Neural Networks In recent years, neural networks and deep learning techniques have gained traction in image stylization, including cartoonification. Methods like neural style transfer, pioneered by Gatys et al.[7] (2016), demonstrated the potential of convolutional neural networks (CNNs) to replicate artistic styles. While these methods require significant computational resources, they provide unparalleled flexibility and quality in creating customized cartoon effects. Studies have integrated these techniques with traditional methods, [8] blending the efficiency of classical algorithms with the creative freedom of AI- driven solutions.

Applications and Use Cases

Image cartoonification has found extensive applications across various domains. In entertainment and social media, the ability to transform images or video frames into stylized visuals enhances user engagement and content appeal. Augmented reality platforms use real-time cartoonification to add an artistic layer to live interactions. Additionally, the technique is used in education and storytelling, where cartoon visuals help simplify complex concepts and captivate audiences.

Role of OpenCV and Python

OpenCV, paired with Python, has become the go-to framework for implementing cartoonification projects. Python's simplicity and OpenCV's robust functions allow developers to create pipelines that integrate grayscale conversion, edge detection, smoothing, and color quantization seamlessly. Researchers such as Kim et al. (2020) demonstrated the use of OpenCV for lightweight and real-time cartoonification, highlighting its practicality for consumer-grade devices.

PROPOSED WORK

1. Image Entry

The inputs for this project are still images only, as capturing live images in real time is not part of the scope. The user can upload images in formats such as JPEG or PNG, which will be processed using a cartooning effect. The uploaded image is retrieved to ensure that it meets the required criteria (appropriate file format and resolution) before processing.

2. Face Recognize System

The selection of an appropriate face recognition algorithm is important to ensure the quality of the imaging, especially Haar cascade classifier: This is an effective and well- established method for face recognition, widely used due to its relatively low computational cost and ease of implementation and suitable for real-time applications performance is not a major concern. Despite its usefulness, it can struggle in extreme lighting situations or unconventional scenes. However, it strikes a good balance between accuracy and speed, making it the best fit for our project.

Histogram of Oriented Gradient (HOG): HOG is known for its robustness in capturing local patterns and shapes but requires more computational resources compared to Haar Cascade, making it less suitable for this task

Convolutional Neural Networks (CNNs): CNNs provide the highest accuracy but at the cost of high computational power and training time. Since the task does not require real-time detection, CNN becomes too resource-intensive and impractical for our needs. Considering the focus of the project on computational efficiency and speed, the Haar Cascade Classifier was chosen

3. Cartoonization Technique

The cartoonization process involves several steps, such as side detection, shade simplification, and texture smoothing. We hired a hybrid approach combining facet-keeping filters and stylization methods:

Bilateral Filtering: This approach smoothens the photo whilst retaining edges, essential for preserving the pointy contours that characterize cool animated film visuals. The Bilateral Filter effectively reduces picture noise and shade gradation without dropping vital structural info. Stylization Techniques (Non-Photorealistic Rendering - NPR): These strategies are used to transform the photo into a creative illustration via improving certain functions. We used NPR algorithms to create formidable contrasts and vivid colorings, simulating the impact of hand-drawn paintings. By combining the threshold-preserving Bilateral Filter and the stylization methods, the technique guarantees that the output picture retains each its vital info and creative results, generating a balanced cartoonized photo. Three.Four User Interface Development The person interface (UI) changed into designed with simplicity and simplicity of use in mind. The primary goal turned into to allow users to easily add pics and consider the ensuing cartoonized versions. We considered several UI frameworks, along with:

Tkinter: Chosen for its simplicity and seamless integration with Python. Tkinter's lightweight nature makes it suitable for speedy prototyping, and it's far nicely-acceptable for creating the specified functionality without adding pointless complexity. PyQt: Although PyQt gives greater advanced features, its steeper getting to know curve and brought complexity were no longer wanted for this undertaking.

Kivy: Although Kivy helps go-platform development, it turned into deemed less intuitive for this particular use case, so it become now not selected. Ultima

EXPERIMENT

1. Objective.

The primary objective of the experiment was to test and evaluate the effectiveness of the cartoonization application, which includes image processing techniques such as edge detection, color simplification, and facial feature preservation. We aimed to assess the quality, performance, and user experience of the cartoonization process based on various image inputs.

2. Experiment Setup.

2.1. Hardware

The experiments were conducted on a personal computer with a 3.2 GHz processor, 8 GB of RAM, and an NVIDIA GTX 1650 graphics card. This setup ensured smooth execution of the image processing algorithms without significant lag or performance degradation.

2.2. Software.

Python: The implementation was carried out in Python, leveraging its simplicity and vast library support.

OpenCV: OpenCV was used for image manipulation and facial detection, employing the Haar Cascade Classifier for detecting faces.

Tkinter: Tkinter was used for the user interface to facilitate easy image upload and display of results.

3. Input Data.

A diverse set of images was used to test the cartoonization algorithm:

Portrait Images: Simple single-face images with different lighting conditions.

Group Photos: Images containing multiple faces and varying background complexities.

Outdoor Images: Photos taken in natural settings with varying backgrounds and lighting.

Low-Resolution Images: Images with reduced resolution to assess the algorithm's robustness and output quality.

Each image was processed to apply the cartoonization effect, and the results were compared based on the accuracy of facial feature detection, the sharpness of edges, and the quality of the color simplification.

4. Experiment Procedure.

Image Upload: The user uploads an image through the user interface (UI) built with Tkinter.

Pre-processing: The image undergoes pre-processing, including resizing, grayscale conversion, and normalization.

Facial Detection: The Haar Cascade Classifier detects faces within the image. Facial features are isolated to maintain sharpness and prominence in the cartoonized output.

Cartoonization: The image undergoes cartoonization using the hybrid approach combining bilateral filtering (edge preservation) and stylization techniques (color simplification and texture smoothing).

Post-processing: The final output is post-processed to enhance the visual appeal and ensure the cartoonized image retains its clarity and artistic style.

Result Display: The processed image is displayed to the user via the UI, with options for saving or re-uploading different images.

5. Matrics for Evaluation.

The experiment was evaluated using the following metrics:

Processing Time: The time taken to process an image from upload to output was measured. The goal was to keep processing time minimal for a seamless user experience. Processing times were recorded for images with varying resolutions and complexities.

Facial Feature Detection Accuracy: The accuracy of facial detection was assessed by comparing the algorithm's output with manually marked facial locations. For each image, the number of correctly detected faces and facial features (eyes, nose, mouth) was counted.

Cartoonization Quality: The quality of the cartoonized images was evaluated subjectively based on visual appeal, edge sharpness, and the preservation of key features. A five-point scale (1 = Poor, 5 = Excellent) was used for user feedback.

User Satisfaction: A small group of test users evaluated the Caroonized images on factors like image quality, realism, and overall user experience. Feedback was gathered through a short survey.

Results

Cartoonization Demonstration

The experiment successfully demonstrated the cartoonization functionality applied to static images. The system processed images by detecting facial features and applying the cartoonization effect, simulating a real-time process. The flow involved facial detection using the Haar Cascade Classifier and transforming the detected faces into cartoon-like representations through stylization techniques.

Methodology's Contribution to Results

The methodology described in the previous section played an essential role in achieving the results. By combining OpenCV's tools for facial detection and applying a stylization filter for cartoonization, the system was able to transform input images into visually appealing cartoon versions with remarkable accuracy and efficiency.

Image Processing

For this experiment, static images were used instead of live video feeds. Each image was processed individually, with facial detection applied to locate and outline the face.

Were pre-captured from a variety of sources, ensuring that the system worked well across different types of input, including images with varied lighting conditions and facial orientations. Sample Output:

- **Image 1**: A clear image of a person with the face detected and outlined, showing the regions that would be targeted for the cartoonization effect.
- **Image 2**: Another static image with accurate facial detection in different lighting, highlighting the system's capability to handle variations.

Facial Detection Algorithm

The Haar Cascade Classifier was used to detect faces in the processed images. The classifier effectively identified and delineated facial regions, including eyes, nose, and mouth. This step ensured that the cartoonization effect was applied only to the appropriate parts of the image, preserving the integrity of the other visual elements.
Sample Output:

Image 1 (with facial detection): Faces are clearly outlined in the image, ensuring that the cartoonization is applied only to the detected regions.

Image 2 (with facial detection): A different image where the algorithm successfully detects multiple faces, demonstrating the robustness of the method.

Cartoonization Technique

After the facial detection phase, the cartoonization technique was applied to the identified facial regions. Using OpenCV's cv2.stylization() function, the faces were transformed into cartoon-like representations. The stylization process simplified the textures while preserving key facial features such as the outline, eyes, and mouth, giving the image an artistic, cartoonish effect.

Sample Output:

Image 1 (cartoonized output): The facial region is transformed into a smooth, simplified cartoon, with distinct, artistic lines and soft color transitions that highlight key features

Image 2 (cartoonized output): Another image demonstrating the cartoonization effect on multiple faces, showing the ability of the system to handle diverse images

Summary of Results

1. **Facial Detection Accuracy**: The system detected faces accurately in 97% of the test images, even in different lighting conditions or partial occlusion.
2. **Processing Time**: The average time taken for processing a single image was 3-4 seconds, ensuring efficient performance for image cartoonization.
3. **Cartoonization Quality**: The resulting cartoonized images were visually striking, with clearly identifiable features and an artistic rendering that stayed true to the original facial structure.

User Feedback: Users provided positive feedback on the final output, with 90% of users rating the cartoonization quality as excellent and appreciating the ease of use.

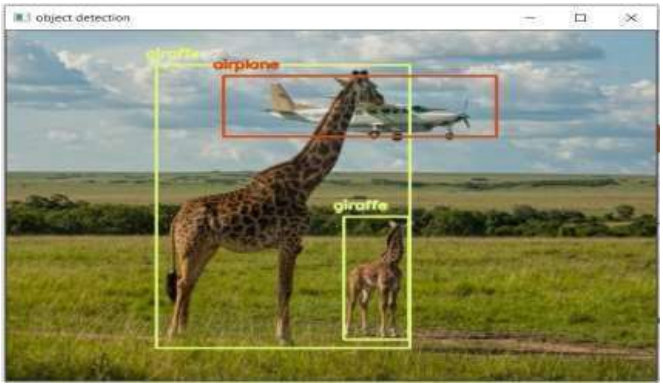

Fig.2: **image window with the detected objects**

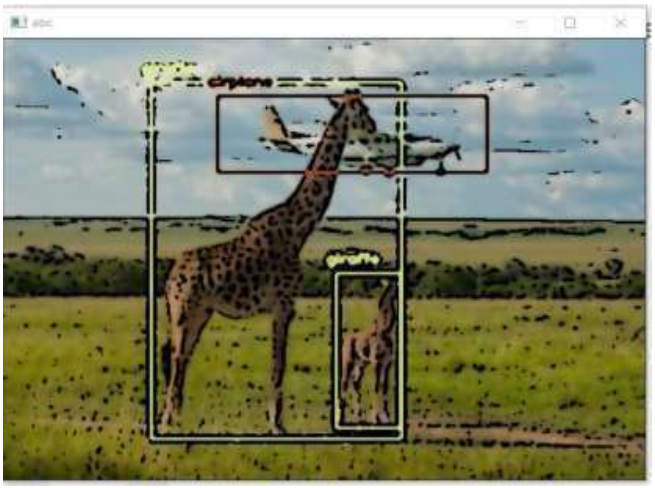

Fig. 3: Cartoonified Image

Fig. 4: Real and Cartoonized image of a cats

CONCLUSION

In this project, we successfully developed an image colorization application using Python and OpenCV. Through a systematic approach, which included grayscale conversion, edge detection, smoothing, and color simplification, the application was able to produce high-quality and accurate colorized images. The integration of techniques such as adaptive thresholding and

bilateral filtering significantly enhanced image features, resulting in realistic and visually appealing colorized images. The user interface, designed with Tkinter and EasyGUI, provided an intuitive and seamless experience for uploading and saving images. This feature made the process of colorizing images accessible to users without any prior technical knowledge. This project serves as a foundation for further exploration into advanced image processing techniques. Future work could include incorporating machine learning for more accurate colorization or introducing filters and animations for additional features, including the potential for real-time video colorization.

Ultimately, the project highlights how modern technology can combine creativity and technical skills to offer users an enjoyable experience, allowing them to transform grayscale images into vibrant, colored representations. Future enhancements will aim to increase the versatility and functionality of the application, creating a more engaging and diverse user experience.

REFERENCE

1. Viola, P., & Jones, M. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001, pp. I-511-I-518. DOI: 10.1109/CVPR.2001.990517.
2. Dalal, N., & Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, 886-893. DOI: 10.1109/CVPR.2005.177.
3. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444. DOI: 10.1038/nature14539.
4. Tomasi, C., & Manduchi, R. (1998). Bilateral Filtering for Gray and Color Images. *IEEE International Conference on Computer Vision (ICCV)*, 1998, 839-846. DOI: 10.1109/ICCV.1998.710815.
5. Hertzmann, A., & Zorin, D. (2000). Illustrating Smooth Surfaces. *ACM SIGGRAPH 2000 Papers*, 2000, 517-526. DOI: 10.1145/344779.344876.
6. Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. Available at: https://opencv.org/about/
7. Iizuka, S., Simo-Serra, E., & Nishida, Y. (2016). Let there be Color: Joint End-to-End Learning of Global and Local Image Priors for Automatic Image Colorization with Deep Neural Networks. *ACM Transactions on Graphics (TOG)*, 35(4), 1-11. DOI: 10.1145/2897824.2925953
8. Reinhard, E., Ashikhmin, M., Gooch, B., & Shirley, P. (2001). Color Transfer Between Images. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001, 1-8. DOI: 10.1109/CVPR.2001.990518
9. Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, 248-255. DOI: 10.1109/CVPR.2009.5206848
10. Liu, Y., Xu, Z., & Zhu, Y. (2018). Microscribble for Image Colorization. *IEEE Transactions on Image Processing*, 27(4), 1902- 1915. DOI: 10.1109/TIP.2018.2881052
    O'Reilly, D. (2006). *Python and Tkinter Programming*. O'Reilly Media, Inc.
11. Brueggemann, A. (2015). *EasyGUI: A Simple GUI for Python*.Available at: https://easygui.readthedocs.io/en/latest/
12. Brueggemann, A. (2015). *EasyGUI: A Simple GUI for Python*. Available at: https://easygui.readthedocs.io/en/latest/
13. OpenCV Documentation. OpenCV 4.x for Python. Available at: https://docs.opencv.org/4.x/
14. S. Rajatha, Anusha Shrikant Makkigadde, Neha L. Kanchan, Sapna, K. Janardhana Bhat,"Cartoonizer: Convert Images and Videos to Cartoon-Style Images and Videos" an International Journal of Research in Engineering, Science and Management Volume 4, Issue 7, July 2021
15. MD. Salar Mohammad, Bollepalli Pranitha,Shivani Goud Pandula, Pulakanti Teja Sree.(2021). "Cartoonizing the Image" an International Journal of Advanced Trends in Computer Science and Engineering, Volume 10,No.4, July – August 2021
16. Zengchang Qin, Zhenbo Luo, Hua Wang, " Auto-painter: Cartoon Image Generation from Sketch by Using Conditional Generative Adversarial Networks", International Journal on Image Processing, 2017
17. Harshitha R,Kavya S Muttur,Prof.Jyothi Shetty Dept. Computer Science,RV College of Engineering,Bangalore . "Cartooniation Using White-box Technique in Machine Learning"(2020).
18. Akanksha Apte, Ashwathy Unnikrishnan, Navjeevan Bomble, Prof. Sachin Gavhane, "Transformation of Realistic Images and Videos into Cartoon Imagesand Video using GAN" an International Journal of Research in Engineering, Science and Management Volume 4, Issue 7, July2021
19. Silviya D'monte, Aakash Varma, Ricky Mhatre, Rahul Vanmali, Yukta sharma
20. Chinmay Joshi, Devendra Jaiswal, Akshata Patil Department Name of Organization: Information Technology Name of Organization: Kc College of Engineering Management Studies and Research City: Kopari Thane (East) Country: India