CONTOURS AND COLORS: THE SCIENCE OF IMAGE SEGMENTATION

A project report submitted to
MALLA REDDY UNIVERSITY
in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLGY
in
COMPUTER SCIENCE & ENGINEERING (AI & ML)

Submitted by

R.SRILEKHA	:	2111CS020559
D.SRINATH	:	2111CS020560
G.SRINIDHI	:	2111CS020561
P.SRINIKA	:	2111CS020562
A.SRIPADA SAI VENKATA YUKTHESWAR	:	2111CS020564
SRIRAM MANIKUMAR	:	2111CS020565

Under the Guidance of

MR.K. Manoj sagar M.E.
Assistant Professor AI&ML

MALLA REDDY UNIVERSITY
(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI & ML)

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

COLLEGE CERTIFICATE

This is to certify that this is the bonafide record of the application development entitled,**"CONTOURS AND COLORS:THE SCIENCE OF IMAGE SEGMENTATION"** Submittedby**R.SRILEKHA**(2111CS020559), D.SRINATH(2111CS020560), G.SRINIDHI(2111CS020561) ,**P.SRINIKA(2111CS020562), A.SRIPADA SAI VENKATA YUKTHESWAR(2111 CS020564), SRIRAM MANIKUMAR (2111CS020565)** B.Tech III year I semester,Department of CSE (AI&ML) during the year 2022-23. The results embodied in the report have not been submitted to any other university or institute for the award of any degree or diploma.

PROJECT GUIDE	**HEAD OF THE DEPARTMENT**
K.MANOJ SAGAR, M.E.	**Dr. THAYYABA KHATOON**
ASSISIANT PROFESSOR	**CSE(AI&ML)**

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

I would like to express our gratitude to all those who extended their support and suggestions to come up with this application. Special Thanks to our mentor **Prof. K.MANOJ SAGAR** whose help and stimulating suggestions and encouragement helped us all time in the due course of project development.

I sincerely thank our **HEAD OF DEPARTMENT DR. THAYYABA KHATOON** for her constant support and motivation all the time. A special acknowledgement goes to a friend who enthused us from the back stage. Last but not the least our sincere appreciation goes to our family who has been tolerant understanding our moods, and extending timely support.

ABSTRACT

Image segmentation, a cornerstone of computer vision, has been revolutionized through the application of machine learning techniques. This abstract provides an encompassing overview of diverse segmentation methodologies within this framework. From region-based techniques like watershed segmentation, which accurately separates overlapping cells in microscopic images, to edge-based methods such as Canny edge detection that enhance object boundaries for pedestrian detection in surveillance, and further to clustering segmentation exemplified by K-means clustering, effectively categorizing land cover types in satellite imagery. The abstract culminates in the prominence of deep learning, embodied by Mask R- CNN, which excels in instance segmentation, enabling precise detection and pixel-level masking of diverse objects in complex scenes. These techniques collectively underscore the transformative impact of machine learning, enabling precise and contextually informed segmentation across various domains

CONTENTS

1. INTRODUCTION

1.1 PROBLEM DEFINITION:

The main challenge in image segmentation is to accurately divide the image into distinct and meaningful regions. This task is inherently complicated by variables, fuzzy boundaries, and the variety of image objects encountered in real-world situations The problem is compounded by the need for classification models function to match different lighting conditions, scale differences, and presence of noise so in images It can generalize well across data types and applications. Furthermore, the computational demands of the task, especially in real-time applications, introduce another dimension of the problem, requiring the search for efficient algorithm optimization techniques so problem definition in image classification revolves around these challenges which will be addressed to create a more versatile and accurate distribution system.

1.2 OBJECTIVES OF THE PROJECT:

The objective of this image segmentation project aim is to provide a robust and versatile system that can accurately classify images into logically consistent areas Using state-of-the-art techniques in computer vision and deep learning Adaptability of the segmentation model to different environments and application areas, including Lee Additionally, the project seeks to explore optimization methods for realtime presentation, ensuring system efficiency in visual data processing Finally, . aims to contribute to the advancement of image classification technology, providing reliable solutions for applications requiring accurate and meaningful field detection in visual context.

1.3 LIMITATIONS OF THE PROJECT:

Object complexity: If an image contains object complexity or clutter, it can be difficult for segmentation algorithms to accurately identify and separate individual objects

Lighting conditions: Changing the lighting can affect the appearance of objects in a photo. Segmentation algorithms can struggle in handling different lighting conditions

Technical aspects: Image segmentation can be highly computational, especially for large images or real-time processing. This can limit the speed and efficiency of the segmentation algorithms, especially on low-power devices**.**

Blurred borders: Blurred images or objects with fuzzy borders cause complications. Algorithms can struggle to pinpoint where one thing ends and another begins.

2. ANALYSIS

2.1 INTRODUCTION:

Image segmentation analysis is a pivotal process in computer vision, involving the partitioning of an image into coherent segments to extract meaningful information for diverse applications. By assigning labels to pixels or delineating object boundaries, segmentation aids in object recognition, feature extraction, and overall image understanding. Its types range from semantic and instance segmentation to color-based and binary segmentation, each serving specific purposes. Despite challenges like appearance variability and computational complexity, image segmentation finds critical applications in medical imaging, autonomous vehicles, satellite imagery analysis, and augmented reality. Various techniques, including thresholding, clustering, edge-based methods, and deep learning approaches like Convolutional Neural Networks (CNNs), are employed to achieve accurate and efficient segmentation results.

2.1 SOFTWARE REQUIREMENT SPECIFICATION:

2.1.1 SOFTWARE REQUIREMENT:

- GOOGLE COLLAB
- Window 10 (or more)

2.1.2 HARDWARE REQUIREMENT:

- GPU (Graphic Processing Unit)
- 8 GB of RAM (or more)
- 20 GB of Free Disk Space
- Cuda compatibility

2.2 ARCHITECTURE:

The methodology of image segmentation involves a series of steps to divide an image into meaningful and distinct regions. Different segmentation methods may be employed based on the specific characteristics of the images and the desired outcome.

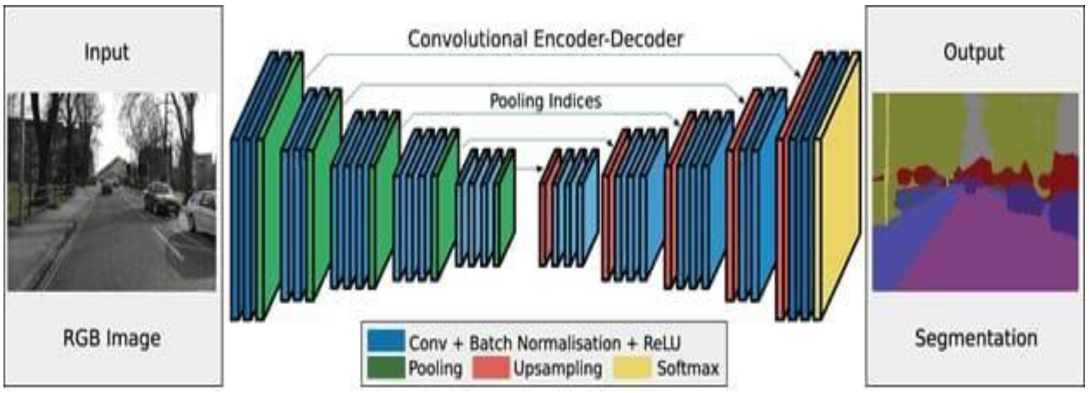

Image segmentation works by using encoders and decoders. Encoders take in input, which is a raw image, it then extracts features, and finally, decoders generate an output which is an image with segments. Above mentioned is the basic model of an image segmentation.

3. DESIGN

3.1 INTRODUCTION:

The design of image segmentation is a multifaceted process aimed at developing methodologies and algorithms to partition images into meaningful segments, thereby enabling efficient and accurate analysis in numerous domains. Image segmentation is crucial for tasks such as object recognition, medical diagnosis, and scene understanding. The design involves selecting appropriate segmentation techniques based on the specific requirements of the application, addressing challenges like variability in appearance and computational complexity. Researchers employ diverse approaches, including traditional methods such as thresholding, clustering, and edge-based techniques, as well as modern deep learning methods like Convolutional Neural Networks (CNNs). The design process considers the balance between precision and computational efficiency, and it often incorporates techniques for handling ambiguity and variability in real-world images. As image segmentation is integral to advancing fields like computer vision, the design continually evolves to meet the demands of emerging technologies and applications.

3.2 PROJECT STATE DIAGRAM:

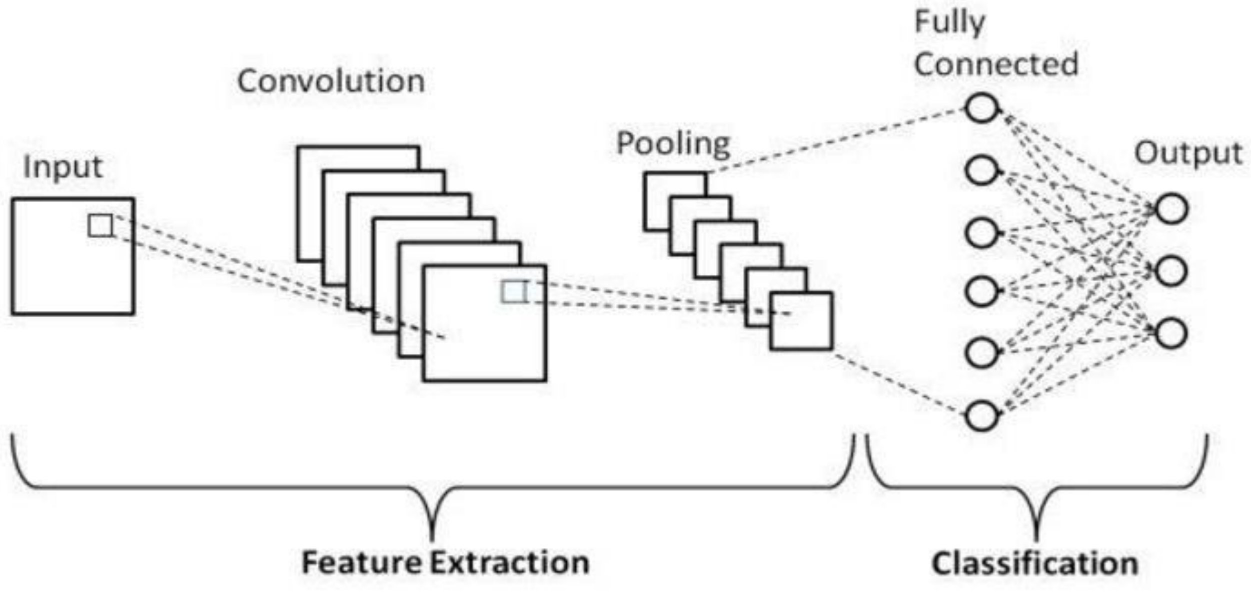

Fig 3.2.1 State Diagram

3.3 DATA SET DESCRIPTION:

Our dataset, SA-1B, consists of 11M diverse, high-resolution, licensed, and privacy protecting images and 1.1B high-quality segmentation masks collected with our data engine. We compare SA-1B with existing datasets and analyze mask quality and properties. We are releasing SA-1B to aid future development of foundation models for computer vision.

Images: We licensed a new set of 11M images from a provider that works directly with photographers. These images are high resolution (3300×4950 pixels on average), and the resulting data size can present accessibility and storage challenges. Therefore, we are releasing downsampled images with their shortest side set to 1500 pixels.

Masks: Our data engine produced 1.1B masks, 99.1% of which were generated fully automatically. Therefore, the quality of the automatic masks is centrally important. We compare them directly to professional annotations and look at how various mask properties compare to prominent segmentation datasets.

3.4 DATA PREPROCESSING TECHNIQUES:

Preprocessing is a crucial step in image segmentation as it helps enhance the quality of the image and prepares it for more effective segmentation. The main goals of preprocessing in image segmentation include noise reduction, contrast enhancement, and the removal of unwanted artifacts. Here are some common preprocessing steps in image segmentation:

Image Acquisition:

Obtain the digital image through a camera, scanner, or other imaging devices. Ensure that the acquisition process captures relevant details and minimizes distortions.

Grayscale Conversion:

Convert the image to grayscale if it is initially in color. This simplifies the segmentation process, as many segmentation algorithms are designed to work on grayscale images.

Noise Reduction:

Apply filtering techniques to reduce noise in the image. Common filters include:

- **Gaussian Filter**: Smoothens the image by applying a Gaussian distribution to the pixel values.
- **Median Filter**: Replaces each pixel's value with the median value of its neighborhood, effectively reducing impulse noise.

Contrast Enhancement: Improve the contrast of the image to highlight important features. Common methods include:

Histogram Equalization: Adjust the distribution of pixel intensities to enhance contrast.

Contrast Stretching: Expand the range of pixel intensities to cover the full dynamic range.

Normalization:

Normalize the intensity values to a common scale. This ensures that the pixel values have a consistent range, which can be important for certain segmentation algorithms.

Resizing:

Resize the image to a standard resolution or scale if needed. This can be useful for speeding up computation and ensuring consistent processing across images.

3.5. MODEL EVALUATION METRICS:

We evaluated SAM on a variety of metrics based on the downstream task in our experiments

- mIoU: We used the mean intersection-over-union after a given number of prompts to evaluate the segmentation quality of a mask when prompted with points.
- Human evaluation: We compared the masks generated by SAM to a baseline state-of-the-art interactive segmentation model, using a perceptual quality scale from 1 to 10.
- AP: We used average precision to evaluate instance segmentation for a edge detection.
- AR@1000: We used average recall to evaluate object proposal generation.
- ODS, OIS, AP, R50: We used the standard edge detection evaluation metrics from BSDS500 [72, 3].

4. DEPLOYMENT AND RESULT

4.1 INTRODUCTION:

The deployment of image segmentation models is a critical phase in bringing the technology from research and development to practical applications. After training a model to accurately segment images, the next step involves deploying the model to real-world scenarios and analyzing its performance on new, unseen data. This deployment phase is integral to ensuring the model's effectiveness and reliability in diverse environments. Following deployment, the evaluation of segmentation results becomes paramount. This involves assessing how well the model generalizes to new images, its robustness to variations in lighting and scene complexity, and its overall adherence to the segmentation task objectives. In this context, result analysis entails a comprehensive examination of metrics such as Intersection over Union (IoU), Dice Coefficient, and class-wise performance, providing insights into the model's strengths and areas for improvement. This introduction sets the stage for a deeper exploration of the deployment process and the critical evaluation of image segmentation results in practical applications.

4.2 SOURCE CODE:

Gray scale image:

```
from skimage.color import rgb2gray import numpy as np import cv2 import
matplotlib.pyplot as plt from scipy import ndimage image =
cv2.imread('dogs.jpg')# Load the image using OpenCV gray_image =
rgb2gray(image)# Convert the image to grayscale using scikit-image
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)# Convert BGR image to RGB format
for matplotlib display plt.figure(figsize=(10, 5))# Display the original and grayscale images using
matplotlib plt.subplot(1, 2, 1) plt.title('Original Image') plt.imshow(image_rgb)
plt.axis('off')
plt.subplot(1, 2, 2)
plt.title('Grayscale
Image')
plt.imshow(gray_image,
cmap='gray')
```

```
plt.axis('off')
plt.tight_layout()
plt.show()
```

Threshold image

```
import numpy as np import cv2 import matplotlib.pyplot as plt from skimage.color
import rgb2gray from scipy import ndimage image = cv2.imread('dogs.jpg')# Load
the image using OpenCV gray_image = rgb2gray(image)# Convert the image to
grayscale using scikit-image
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)# Convert BGR image to RGB format for
    matplotlib display

gray_r = gray_image.reshape(gray_image.shape[0] * gray_image.shape[1])# Reshape the grayscale
image for thresholding mean_value = gray_r.mean()# Reshape the grayscale image for
thresholding thresholded = np.where(gray_r > mean_value, 1, 0)# Apply thresholding based on
mean value
thresholded_image = thresholded.reshape(gray_image.shape[0], gray_image.shape[1])# Reshape the
thresholded array back to the original shape plt.imshow(thresholded_image, cmap='gray')# Display
the thresholded image using matplotlib plt.title('Thresholded Binary Image') plt.axis('off')
plt.show()
```

```
import numpy as np import cv2 import matplotlib.pyplot as
plt original_image = cv2.imread('players.jpeg')
plt.imshow(original_image) img =
cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB)
plt.imshow(img) vectorized = img.reshape((-1,3)) vectorized
= np.float32(vectorized)
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0) K = 2
attempts=10
ret,label,center=cv2.kmeans(vectorized,K,None,criteria,attempts,cv2.KMEANS_PP_CENTERS
) center = np.uint8(center)
```

```
res = center[label.flatten()] result_image = res.reshape((img.shape))
figure_size = 15 plt.figure(figsize=(figure_size,figure_size))
plt.subplot(1,2,1),plt.imshow(img) plt.title('Original Image'),
plt.xticks([]), plt.yticks([]) plt.subplot(1,2,2),plt.imshow(result_image)
plt.title('Segmented Image when K = %i' % K), plt.xticks([]),
plt.yticks([])
plt.show() import
torch import
torchvision
print("PyTorch version:", torch.__version__)
print("Torchvision version:", torchvision.__version__)
print("CUDA is available:", torch.cuda.is_available())
import numpy as np import torch import
matplotlib.pyplot as plt import cv2 import sys
sys.path.append("..") from segment_anything import sam_model_registry,
SamAutomaticMaskGenerator, SamPredictor #torch.cuda.set_per_process_memory_fraction(0.5)

image = cv2.imread('/content/drive/MyDrive/Application Development-2/houses.jpg')  #Try houses.jpg
    or neurons.jpg
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)


plt.figure(figsize=(10,10)) plt.imshow(image)
plt.axis('off') plt.show()


sam_checkpoint = "/content/drive/MyDrive/Application Development-2/sam_vit_h_4b8939.pth"
    #sam_vit_h_4b8939.pth model_type
= "vit_h"


device = "cuda"


sam = sam_model_registry[model_type](checkpoint=sam_checkpoint) sam.to(device=device)
#There are several tunable parameters in automatic mask generation that control
```

```
# how densely points are sampled and what the thresholds are for removing low
# quality or duplicate masks. Additionally, generation can be automatically #
run on crops of the image to get improved performance on smaller objects, #
and post-processing can remove stray pixels and holes.
# Here is an example configuration that samples more masks:
#https://github.com/facebookresearch/segmentanything/blob/9e1eb9fdbc4bca4cd0d948b8ae7fe505d9f
    4ebc7/segment_anything/automatic_mask_ generator.py#L35

#Rerun the following with a few settings, ex. 0.86 & 0.9 for iou_thresh #
and 0.92 and 0.96 for score_thresh

mask_generator_ = SamAutomaticMaskGenerator(     model=sam,
points_per_side=32,     pred_iou_thresh=0.9,
stability_score_thresh=0.96,     crop_n_layers=1,
crop_n_points_downscale_factor=2,     min_mask_region_area=100,  #
Requires open-cv to run post-processing
)


masks = mask_generator_.generate(image)
print(len(masks)) def
show_anns(anns):
if len(anns) == 0:
     return
   sorted_anns = sorted(anns, key=(lambda x: x['area']), reverse=True)
ax = plt.gca()     ax.set_autoscale_on(False)     polygons = []     color =
[]     for ann in sorted_anns:         m = ann['segmentation']         img =
np.ones((m.shape[0], m.shape[1], 3))         color_mask =
np.random.random((1, 3)).tolist()[0]         for i in range(3):
         img[:,:,i] = color_mask[i]
ax.imshow(np.dstack((img, m*0.35)))
plt.figure(figsize=(10,10))
plt.imshow(image) show_anns(masks)
plt.axis('off') plt.show()
```

4.3 FINAL RESULT:

Gray Scale image:

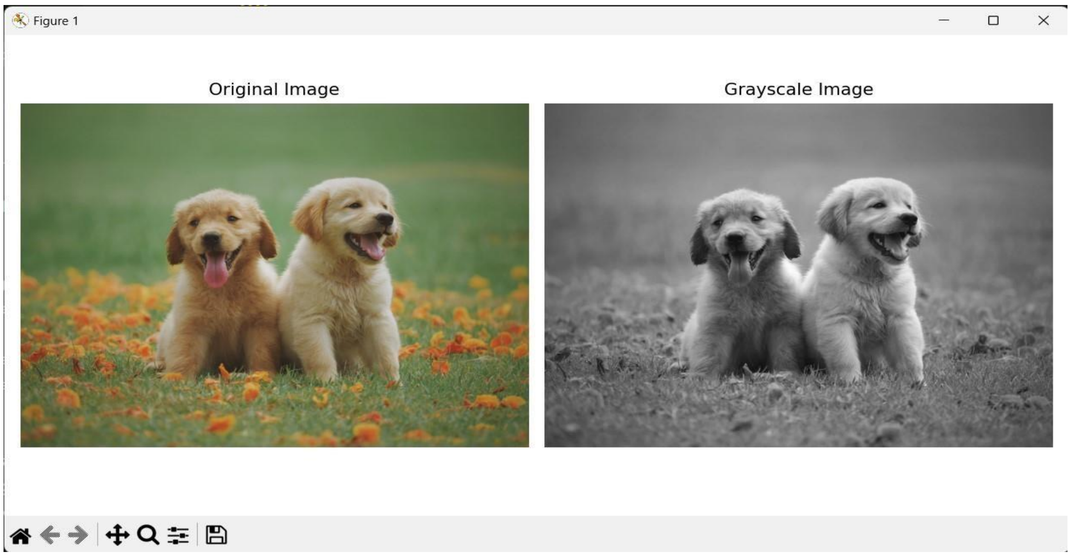

Fig no (1)

Threshold image:

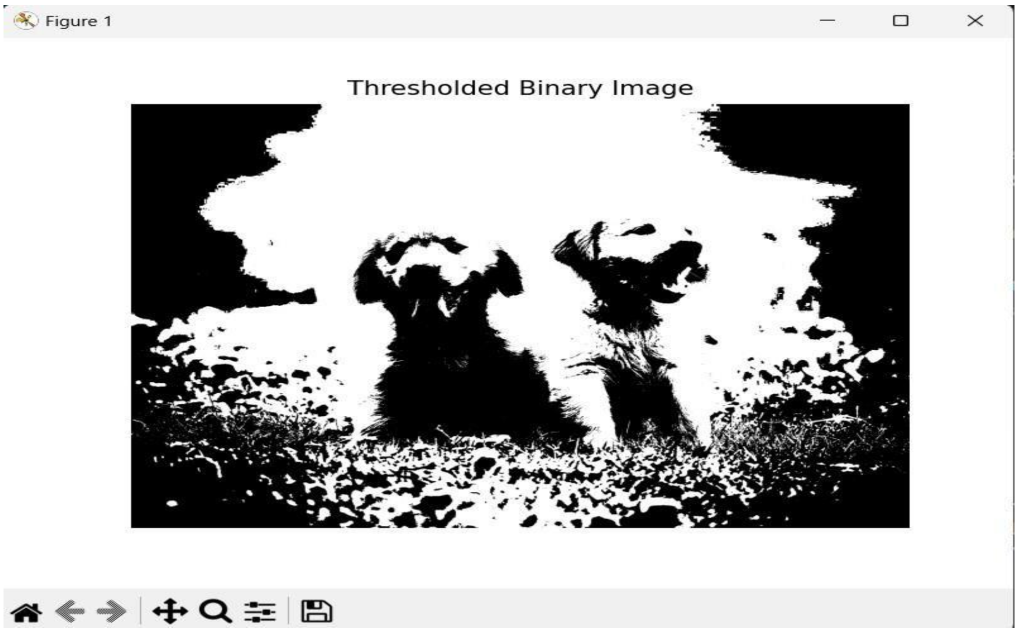

Fig no (2)

Input:

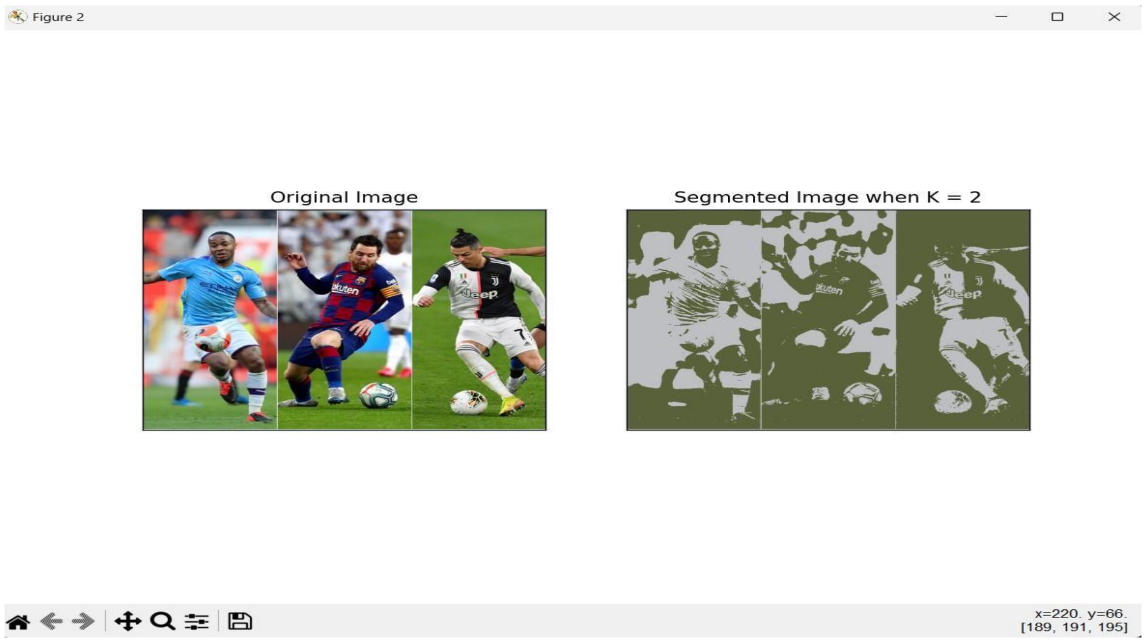

Fig no (3)

Input:

Fig no (4)

Output :

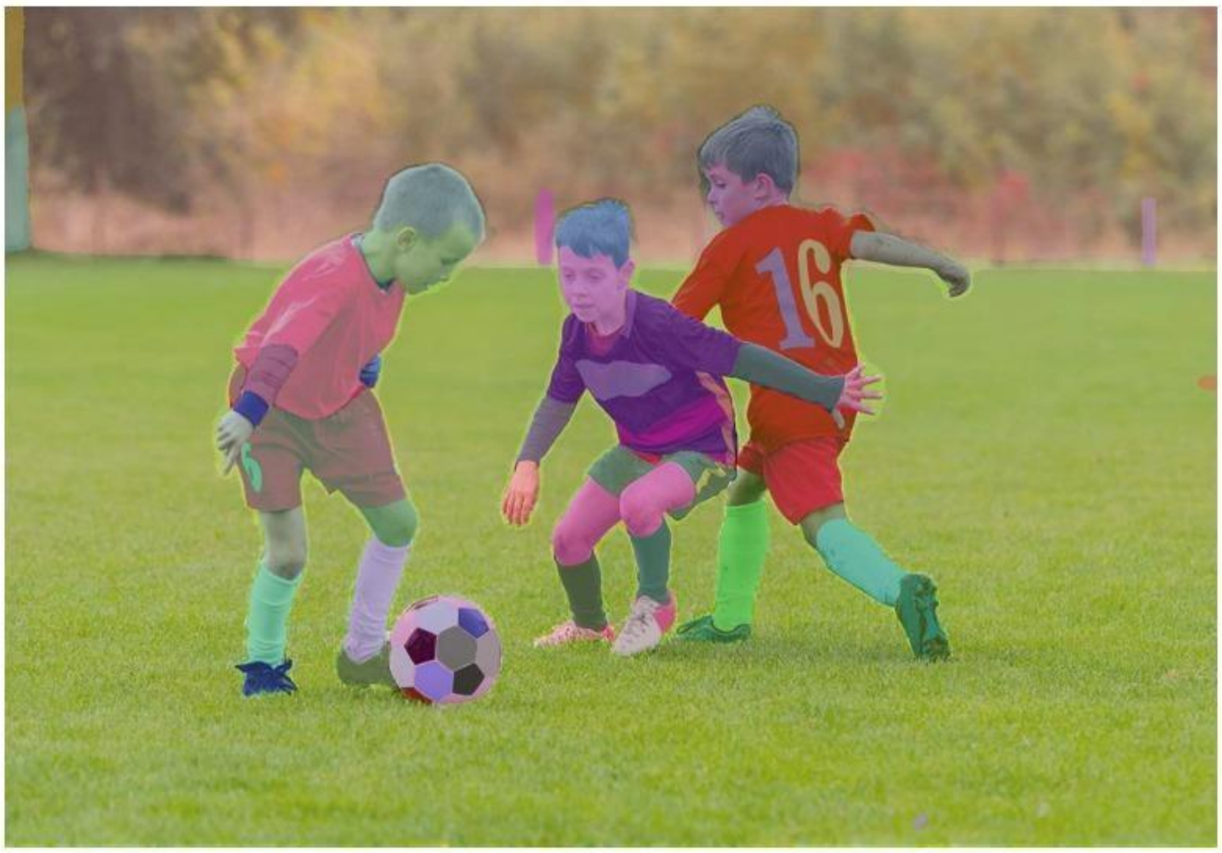

Fig no (5)

5 CONCLUSION

5.1 PROJECT CONCLUSION:

In conclusion, image segmentation stands as a pivotal task in computer vision, serving as the foundation for diverse applications such as object recognition, medical imaging, and autonomous systems. The methodologies explored, including Gray scale edge detection and clustering-based segmentation, exemplify the nuanced strategies employed to delineate meaningful regions within an image. The comprehensive segmentation methodology encompasses preprocessing steps for noise reduction and contrast enhancement, followed by advanced segmentation techniques that leverage edge information and clustering algorithms. Additionally, the integration of data augmentation techniques further fortifies segmentation models by exposing them to a broader spectrum of image variations. As an iterative process, image segmentation undergoes continuous refinement, validation, and post-segmentation analysis, embodying a dynamic field at the forefront of computer vision research and application development. The ongoing evolution of segmentation approaches and their adaptability to complex realworld scenarios underscore the importance of image segmentation in advancing the capabilities of computer vision systems.

5.2 FUTURE ENHANCEMENT:

Future work in image segmentation is poised to address several promising avenues to enhance the accuracy, efficiency, and applicability of segmentation methods across diverse domains. Firstly, the integration of deep learning techniques, especially convolutional neural networks (CNNs), continues to be an area of active research. The exploration of novel network architectures, attention mechanisms, and transfer learning strategies holds potential for further improving the performance of segmentation models. Moreover, the development of real-time and resource-efficient segmentation algorithms remains a priority, especially in applications such as robotics, autonomous vehicles, and augmented reality. Additionally, there is a growing need for domain-specific customization of segmentation models. Tailoring algorithms to specific industries, such as healthcare or agriculture, and addressing challenges posed by varied imaging conditions, resolutions, and modalities will be crucial. The incorporation of multi-modal information, including the fusion of RGB and depth data, or the integration of contextual information through graph-based models, presents exciting opportunities for more holistic and accurate segmentation.

REFERENCES

[1] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. What is an object? CVPR, 2010. 4, 10

[2] Edward H Adelson. On seeing stuff: the perception of materials by humans and machines. Human vision and electronic imaging VI, 2001. 5

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton.Layer normalization. arXiv:1607.06450, 2016. 16

[4] Pablo Arbel aez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation.TPAMI, 2010. 4, 10, 21, 28