Detectron Object Detection & Manipulating Images Using Cartoonization

Team Members

ALLENA VENKATA SAI ABHISHEK
(VSKP MTECH DSS)
Regn. No. - 122021601009

SONALI KOTNI (VSKP MTECH DSS)
Regn. No. - 122021601014

DETECTRON

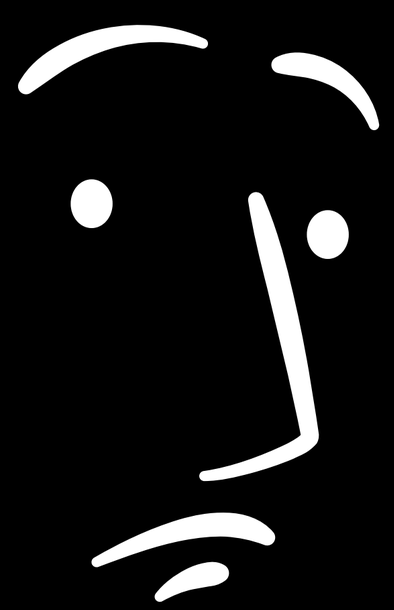
- Detectron is Facebook AI Research’s (FAIR) software system
- Implements state-of-the-art object detection algorithms, including Mask R-CNN.
- It is written in Python and powered by the Caffe2 deep learning framework.

GOAL OF DETECTRON

The goal of Detectron is to provide a

- high-quality,
- high-performance
- codebase for object detection research.

It is designed to be flexible in order to support rapid implementation & evaluation of novel research.



WHY DETECTRON

- High quality
- Industry standard codebase for object detection research
- Incredibly accurate
- Object related algorithms are embedded in Detectron
- Leverages the Caffee2 deep learning framework underneath.
- In a popular language, Python

"**Detectron** model is meant to advance object detection by offering speedy training and addressing the issues companies face when making the step from research to production "

— Mark Roosevelt

DETECTRON includes implementations of the following object detection **ALGORITHMS:**

- Mask R-CNN
- RetinaNet
- Faster R-CNN
- RPN
- Fast R-CNN
- R-FCN

RPN

- A type of region-based object detector
-
- Efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance.

- The method, called Mask R-CNN, extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition.

- Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5 fps.

- Mask R-CNN outperforms all existing, single-model entries on every task.

- Mask R-CNN is easy to generalize to other tasks.

FAST R-CNN

- Instead of feeding the region proposals to the CNN, we feed the input image to the CNN to generate a convolutional feature map.
- From the convolutional feature map, we identify the region of proposals and warp them into squares .
- By using a RoI pooling layer we reshape them into a fixed size so that it can be fed into a fully connected layer.
- From the RoI feature vector, we use a softmax layer to predict the class of the proposed region and also the offset values for the bounding box.
- Uses selective search algorithm on the feature map to identify the region proposals is used to predict the region proposals.

FASTER R-CNN

- Work well with dense & small scale objects.
- Uses convolution neural networks like
  a. YOLO (You Look Only Once)
  b. SSD (Single Shot Detector)
- You don't have to feed 2000 region proposals to the convolutional neural network every time.
- The predicted region proposals are then reshaped using a RoI pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes.

MASK R-CNN

- Conceptually simple & flexible framework for object instance segmentation.
- Efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance.
- The method, called Mask R-CNN, extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition.
- Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5 fps.
- Mask R-CNN outperforms all existing, single-model entries on every task.
- Mask R-CNN is easy to generalize to other tasks.

R-FCN

- A type of region-based object detector
-
- **Efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance.**

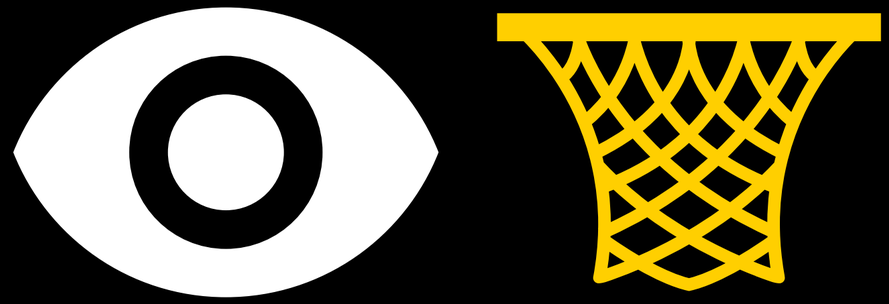
- The method, called Mask R-CNN, extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition.

- **Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5 fps.**

- Mask R-CNN outperforms all existing, single-model entries on every task.

- **Mask R-CNN is easy to generalize to other tasks.**

RETINANET

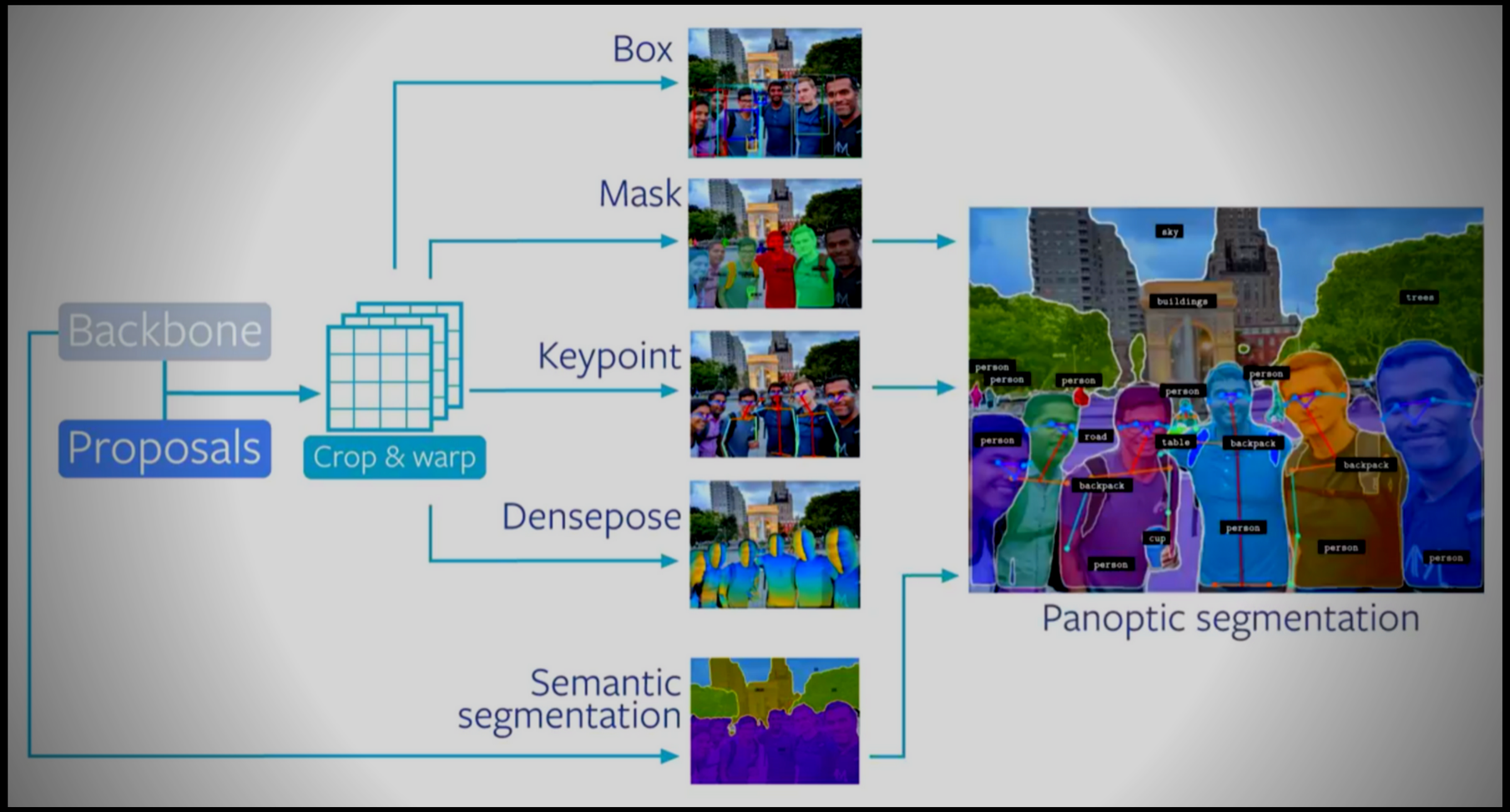


- One of the best, one-stage object detection models.
- Work well with dense and small scale objects.
- Formed by making two improvements over existing single stage object detection models -
  a. Feature Pyramid Networks (FPN) .
  b. Focal Loss.
- Using larger scales allows RetinaNet to surpass the accuracy of all two-stage approaches, while still being faster.
- Except YOLOv2 (which targets on extremely high frame rate), RetinaNet outperforms SSD, DSSD, R-FCN and FPN.
- Built on top of ResNet and is responisble for computing convolutional feature maps of an entire.

USING THE FOLLOWING BACKBONE NETWORK ARCHITECTURES:

- ResNeXt{50,101,152}
- ResNet{50,101,152}
- Feature Pyramid Networks (with ResNet / ResNeXt)
- VGG16

FRAMEWORK OF DETECTRON

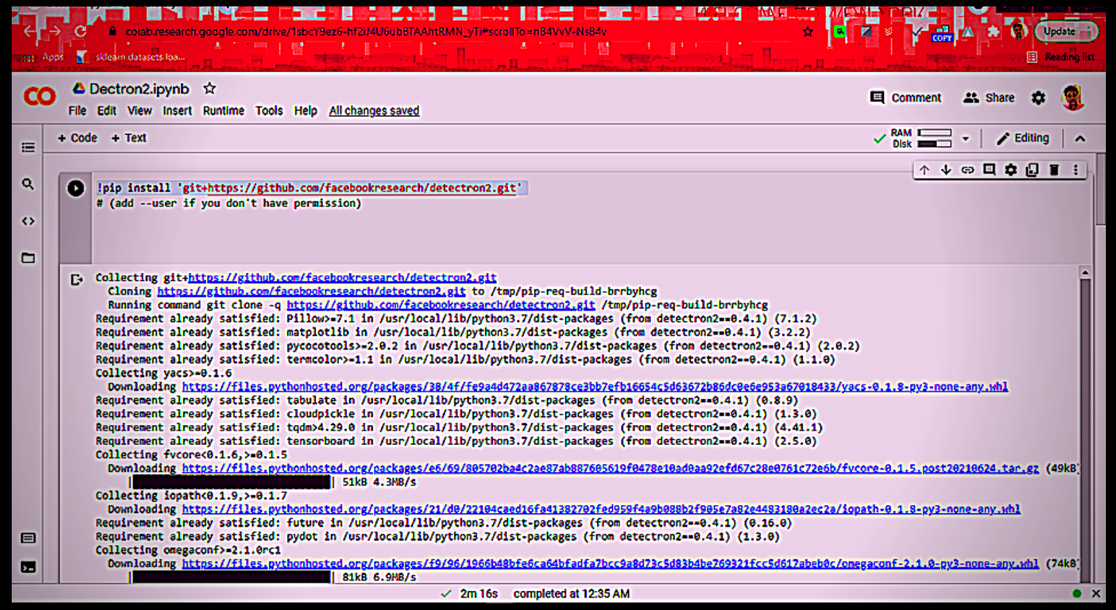



WANT TO USE
DETECTRON?

Lets make your dream come true !!!

- **STEP -1 -OPEN** your python IDE & use the below command -

```
!pip install 'git+https://github.com/facebookresearch/detectron2.git'
```



YOU WILL SEE YOUR IDE GOING CRAZY !!!
BUT DON'T GET SCARED !!

KEEP CALM & TRUST THE DETECTRON

PS: Just installing detectron2

```
WARNING: The following packages were previously imported in this runtime:
  [pydevd_plugins]
You must restart the runtime in order to use newly installed versions.
```

RESTART RUNTIME

Restart your runtime, and run it, you will get the below message

```
Successfully built detectron2
```

- **STEP -2 -USE** the below command & import all the libraries required -

```
# install dependencies:
!pip install pyyaml==5.1 pycocotools>=2.0.1
import torch, torchvision
print(torch.__version__, torch.cuda.is_available())
!gcc - version
# opencv is pre-installed on colab

# install detectron2: (colab has CUDA 10.1 + torch 1.6):
# See https://detectron2.readthedocs.io/tutorials/install.html for instructions
!pip install detectron2 -f https://dl.fbaipublicfiles.com/detectron2/wheels/cu101/torch1.6/index.html
```

```
# Some basic setup:
# Setup detectron2 logger
import detectron2
from detectron2.utils.logger import setup_logger
setup_logger()


# import some common libraries


import numpy as np
import os, json, cv2, random
from google.colab.patches import cv2_imshow


# import some common detectron2 utilities


from detectron2 import model_zoo
from detectron2.engine import DefaultPredictor
from detectron2.config import get_cfg
from detectron2.utils.visualizer import Visualizer
from detectron2.data import MetadataCatalog, DatasetCatalog
```

Libraries Required

- CUDA
- PyTorch
- Torchvision
- detectron2
- Numpy
- JSON
- OpenCV
- random

- **STEP -3 -Detetcting & Labelling of Image -**

```
#getting Image from COCO dataset
!wget http://images.cocodataset.org/val2017/000000439715.jpg -q -O input.jpg
im = cv2.imread("/content/image.jpg")
cv2_imshow(im)

#Creating a detectron2 config and a detectron2 `DefaultPredictor` to run inference on this image.
cfg = get_cfg()
# add project-specific config (e.g., TensorMask) here if you're not running a model in detectron2's core library
cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml"))
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5  # set threshold for this model
# Find a model from detectron2's model zoo. You can use the https://dl.fbaipublicfiles... url as well
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml")
predictor = DefaultPredictor(cfg)
outputs = predictor(im)


# look at the outputs. See https://detectron2.readthedocs.io/tutorials/models.html#model-output-format for specification
print(outputs["instances"].pred_classes)
print(outputs["instances"].pred_boxes)

#Last and final step is to visualize our processed image.
v = Visualizer(im[:, :, ::-1], MetadataCatalog.get(cfg.DATASETS.TRAIN[0]), scale=1.2)
out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
cv2_imshow(out.get_image()[:, :, ::-1])
```

REFERENCE LINKS FOR YOU

Facebook's AI team Releases Detectron – A Platform for Object Detection Research

www.analyticsvidhya.com/blog/2018/01/facebook-launched-detectron-platform-object-detection-research/

Image Labelling Using Facebook's Detectron

https://towardsdatascience.com/image-labelling-using-facebooks-detectron-4931e30c4d0c

CARTOONIG AN IMAGE

- Now a days, we can see that people use many applications and software in order to edit the pictures .
- Haven't you all dreamt of your self as a cartoon? Your dream can become true now itself. Here in machine learning ,we use image processing techniques inorder to convert an image into a cartoon.

1. IMPORTING LIBRARIES AND LOADING INPUT IMAGE

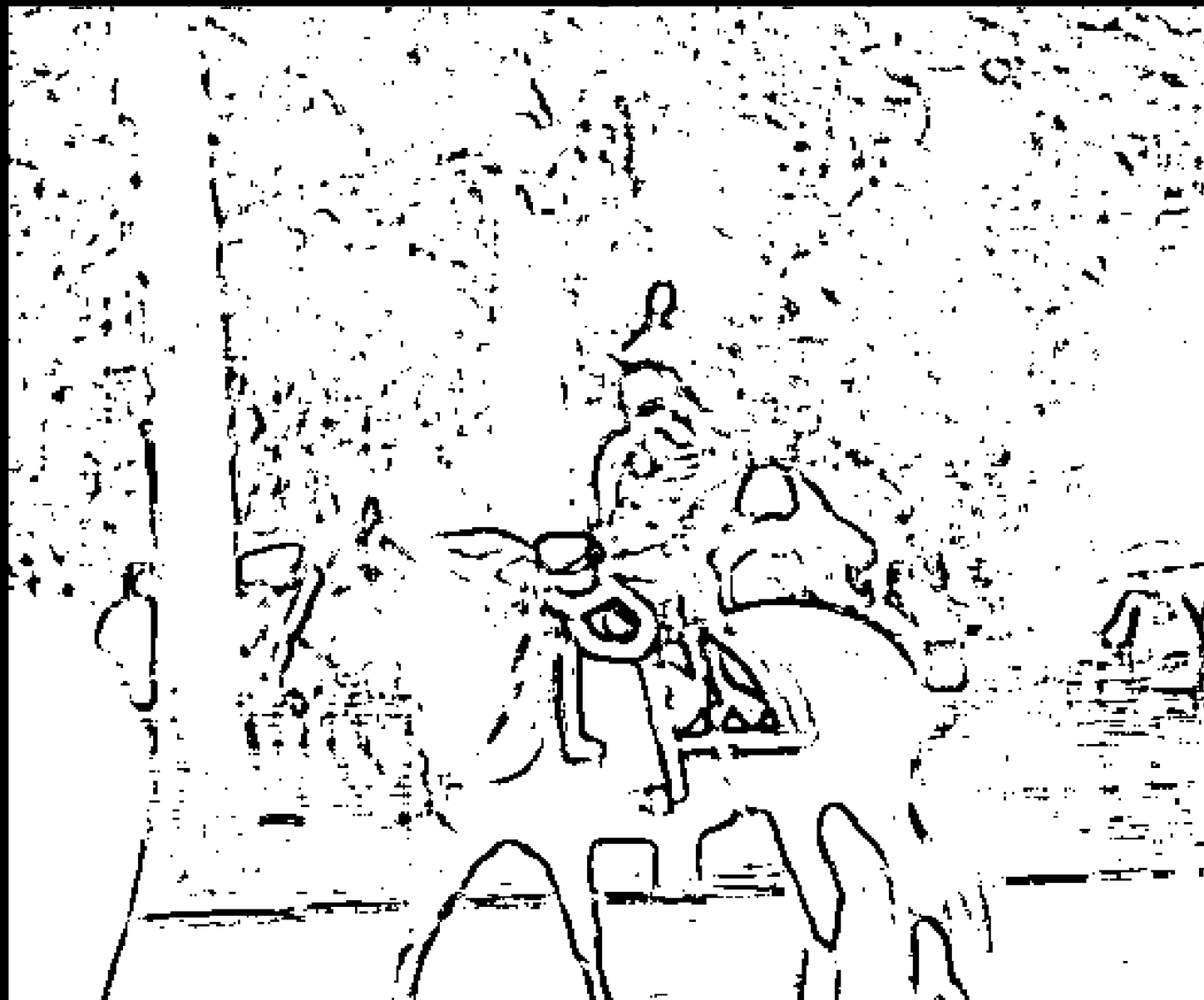
REQUIRED LIBRARIES:

- Open CV
- Numpy

2. CREATING AN EDGE MASK

When creating an edge mask, the thickness of the edges in an image is given the first priority. We get different thresholds for different regions of the same image. It will emphasize black edges around objects in the image thickness of the edges in an image is given the first priority.

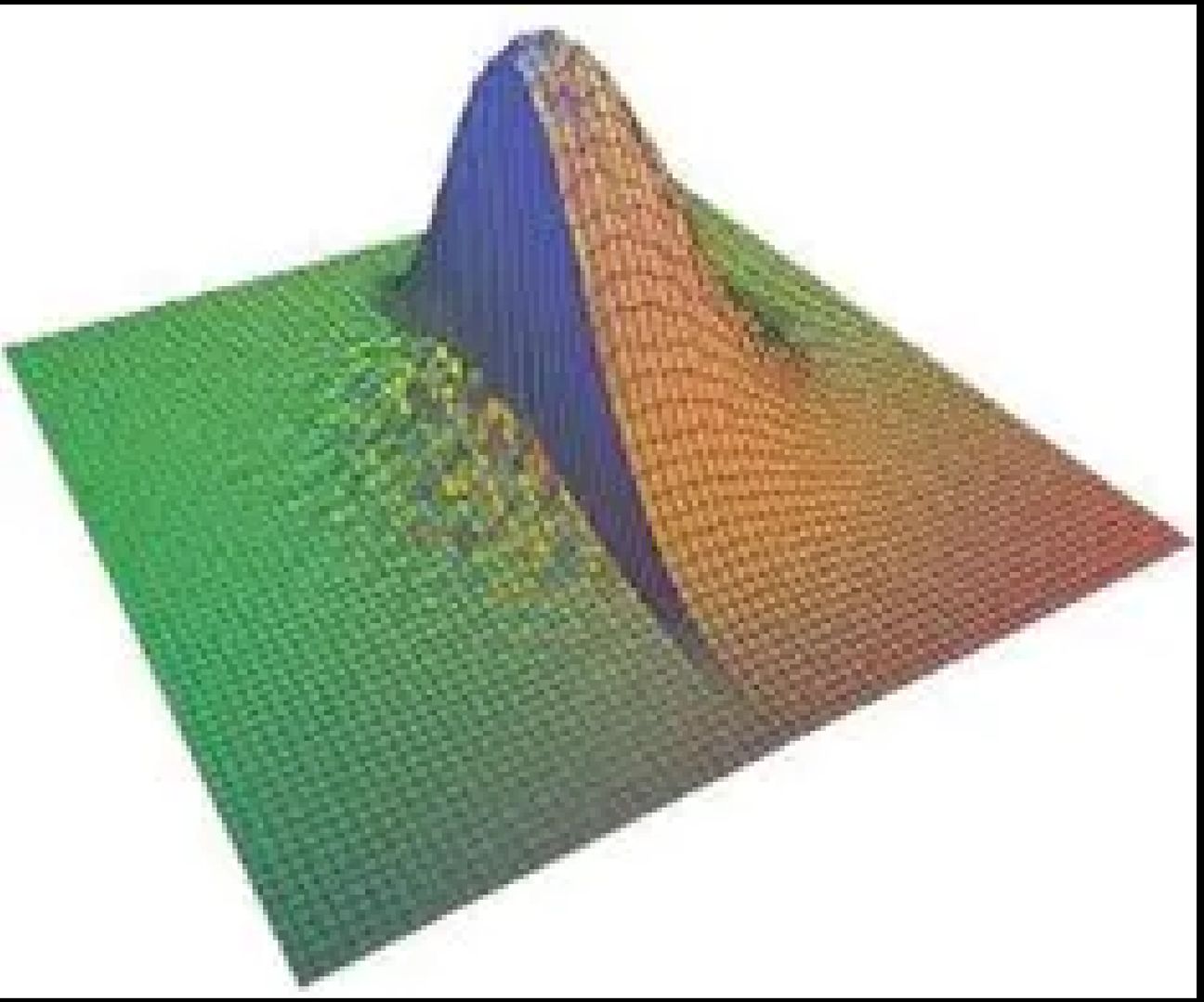
Now the image is converted into grayscale. The noise is compressed from the image to reduce the number of detected edges that are not required .



3. REDUCING COLOUR PALETTE

COLOUR QUANTIZATION:

Color quantization is performed by using the K-means clustering algorithm for displaying output with a limited number of colors.



BILATERAL FILTER:

There are three specifications which are important for bilateral filtering. They are:

- d :Diameter of each pixel neighborhood
- sigmaColor : A larger value of the parameter means that farther colors within the pixel neighborhood will be mixed together, resulting in larger areas of semi-equal colour

- **sigmaSpace : the standard deviation of the filter in the coordinate space. A larger value of the parameter means that farther pixels will influence each other as long as their colors are close enough.**

4. COMBINING EDGE MASK WITH THE COLORED IMAGE

Finally the edge mask is combined with the color-processed image.

Bitwise operations are performed on the image to get the output.

THANK YOU