NAME : SAIFULLAH

NAME: HANZLA

```
!apt-get update
!apt-get install -y libopencv-dev
```

Show hidden output

```
!pkg-config --modversion opencv4
```

```
4.5.4
```

```
!ls
```

```
from google.colab import files
files.upload()
```

Show hidden output

```
%%writefile grayscale.cu
#include <opencv2/opencv.hpp>
#include <cuda_runtime.h>
#include <iostream>

using namespace cv;
using namespace std;

__global__ void grayscaleKernel(unsigned char* input,
                                unsigned char* output,
                                int width, int height)
{
    int x = blockIdx.x * blockDim.x + threadIdx.x;
    int y = blockIdx.y * blockDim.y + threadIdx.y;

    if (x < width && y < height)
    {
        int idx = (y * width + x) * 3; // BGR pixel
        unsigned char b = input[idx];
        unsigned char g = input[idx + 1];
        unsigned char r = input[idx + 2];

        output[y * width + x] = (r + g + b) / 3;
    }
}

int main()
{
    Mat img = imread("input.jpg");
    if (img.empty())
    {
        cout << "input.jpg not found" << endl;
        return -1;
    }

    int width = img.cols;
    int height = img.rows;

    Mat gray(height, width, CV_8UC1);

    unsigned char *d_input, *d_output;
    cudaMalloc(&d_input, width * height * 3);
    cudaMalloc(&d_output, width * height);

    cudaMemcpy(d_input, img.data,
               width * height * 3,
               cudaMemcpyHostToDevice);

    dim3 threads(16, 16);
```

```
    dim3 threads(16, 16);
    dim3 blocks((width + 15) / 16, (height + 15) / 16);

    grayscaleKernel<<<blocks, threads>>>(d_input, d_output, width, height);

    cudaMemcpy(gray.data,
               d_output,
               width * height,
               cudaMemcpyDeviceToHost);

    imwrite("gray_gpu.jpg", gray);

    cudaFree(d_input);
    cudaFree(d_output);

    cout << "Grayscale DONE" << endl;
    return 0;
}
```

```
Writing grayscale.cu
```

```
!nvcc grayscale.cu -o grayscale `pkg-config --cflags --libs opencv4`
```

```
/usr/include/opencv4/opencv2/stitching/detail/warpers.hpp(235): warning #611-D: overloaded virtual function "cv::detail::PlaneWa
  class AffineWarper : public PlaneWarper
        ^

Remark: The warnings can be suppressed with "-diag-suppress <warning-number>"

/usr/include/opencv4/opencv2/stitching/detail/warpers.hpp(235): warning #611-D: overloaded virtual function "cv::detail::PlaneWa
  class AffineWarper : public PlaneWarper
        ^

/usr/include/opencv4/opencv2/stitching/detail/blenders.hpp(100): warning #611-D: overloaded virtual function "cv::detail::Blende
  class FeatherBlender : public Blender
        ^

/usr/include/opencv4/opencv2/stitching/detail/blenders.hpp(127): warning #611-D: overloaded virtual function "cv::detail::Blende
  class MultiBandBlender : public Blender
        ^

/usr/include/opencv4/opencv2/stitching/detail/warpers.hpp(235): warning #611-D: overloaded virtual function "cv::detail::PlaneWa
  class AffineWarper : public PlaneWarper
        ^

Remark: The warnings can be suppressed with "-diag-suppress <warning-number>"

/usr/include/opencv4/opencv2/stitching/detail/warpers.hpp(235): warning #611-D: overloaded virtual function "cv::detail::PlaneWa
  class AffineWarper : public PlaneWarper
        ^

/usr/include/opencv4/opencv2/stitching/detail/blenders.hpp(100): warning #611-D: overloaded virtual function "cv::detail::Blende
  class FeatherBlender : public Blender
        ^

/usr/include/opencv4/opencv2/stitching/detail/blenders.hpp(127): warning #611-D: overloaded virtual function "cv::detail::Blende
  class MultiBandBlender : public Blender
        ^
```

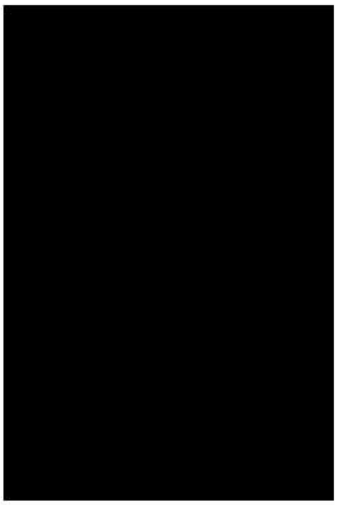
```
!./grayscale
```

```
Grayscale DONE
```

```
from PIL import Image
import matplotlib.pyplot as plt

plt.imshow(Image.open("gray_gpu.jpg"), cmap="gray")
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(4023.5), np.float64(6047.5), np.float64(-0.5))
```



```
%%writefile grayscale.cu
#include <opencv2/opencv.hpp>
#include <cuda_runtime.h>
#include <iostream>

using namespace cv;
using namespace std;

__global__ void grayKernel(unsigned char* input,
                           unsigned char* output,
                           int width, int height)
{
    int x = blockIdx.x * blockDim.x + threadIdx.x;
    int y = blockIdx.y * blockDim.y + threadIdx.y;

    if (x < width && y < height)
    {
        int i = y * width + x;
        int idx = i * 3;

        unsigned char b = input[idx + 0];
        unsigned char g = input[idx + 1];
        unsigned char r = input[idx + 2];

        output[i] = (r + g + b) / 3;
    }
}

int main()
{
    Mat img = imread("input.jpg", IMREAD_COLOR);
    if (img.empty())
    {
        cout << "input.jpg not found\n";
        return -1;
    }

    // 🔑 FORCE continuous memory
    Mat img_cont = img.clone();

    int width  = img.cols;
    int height = img.rows;
    int pixels = width * height;

    Mat gray(height, width, CV_8UC1);

    unsigned char *d_in, *d_out;
    cudaMalloc(&d_in, pixels * 3);
    cudaMalloc(&d_out, pixels);

    cudaMemcpy(d_in,
               img_cont.data,
               pixels * 3,
```

```
                cudaMemcpyHostToDevice);

    dim3 threads(16,16);
    dim3 blocks((width+15)/16, (height+15)/16);

    grayKernel<<<blocks, threads>>>(d_in, d_out, width, height);

    cudaMemcpy(gray.data,
                d_out,
                pixels,
                cudaMemcpyDeviceToHost);

    imwrite("gray_gpu.jpg", gray);

    cudaFree(d_in);
    cudaFree(d_out);

    cout << "Grayscale OK\n";
    return 0;
}
```

```
Overwriting grayscale.cu
```

```
!nvcc grayscale.cu -o grayscale `pkg-config --cflags --libs opencv4`
!./grayscale
```

```
/usr/include/opencv4/opencv2/stitching/detail/warpers.hpp(235): warning #611-D: overloaded virtual function "cv::detail::PlaneWa
  class AffineWarper : public PlaneWarper
        ^

Remark: The warnings can be suppressed with "-diag-suppress <warning-number>"

/usr/include/opencv4/opencv2/stitching/detail/warpers.hpp(235): warning #611-D: overloaded virtual function "cv::detail::PlaneWa
  class AffineWarper : public PlaneWarper
        ^

/usr/include/opencv4/opencv2/stitching/detail/blenders.hpp(100): warning #611-D: overloaded virtual function "cv::detail::Blende
  class FeatherBlender : public Blender
        ^

/usr/include/opencv4/opencv2/stitching/detail/blenders.hpp(127): warning #611-D: overloaded virtual function "cv::detail::Blende
  class MultiBandBlender : public Blender
        ^

/usr/include/opencv4/opencv2/stitching/detail/warpers.hpp(235): warning #611-D: overloaded virtual function "cv::detail::PlaneWa
  class AffineWarper : public PlaneWarper
        ^

Remark: The warnings can be suppressed with "-diag-suppress <warning-number>"

/usr/include/opencv4/opencv2/stitching/detail/warpers.hpp(235): warning #611-D: overloaded virtual function "cv::detail::PlaneWa
  class AffineWarper : public PlaneWarper
        ^

/usr/include/opencv4/opencv2/stitching/detail/blenders.hpp(100): warning #611-D: overloaded virtual function "cv::detail::Blende
  class FeatherBlender : public Blender
        ^

/usr/include/opencv4/opencv2/stitching/detail/blenders.hpp(127): warning #611-D: overloaded virtual function "cv::detail::Blende
  class MultiBandBlender : public Blender
        ^

Grayscale OK
```

```
from PIL import Image
import numpy as np

img = np.array(Image.open("gray_gpu.jpg"))
print("MIN:", img.min(), "MAX:", img.max())
```

```
MIN: 0 MAX: 0
```

```
%%writefile grayscale.cpp
#include <opencv2/opencv.hpp>
#include <iostream>
```

```
using namespace cv;
using namespace std;

int main()
{
    Mat img = imread("input.jpg");
    if (img.empty())
    {
        cout << "input.jpg not found\n";
        return -1;
    }

    Mat gray;
    cvtColor(img, gray, COLOR_BGR2GRAY);

    imwrite("gray_gpu.jpg", gray);
    cout << "Grayscale OK (OpenCV)\n";

    return 0;
}
```

```
Writing grayscale.cpp
```

```
!g++ grayscale.cpp -o grayscale `pkg-config --cflags --libs opencv4`
!./grayscale
```

```
Grayscale OK (OpenCV)
```

```
from PIL import Image
import numpy as np

img = np.array(Image.open("gray_gpu.jpg"))
print("MIN:", img.min(), "MAX:", img.max())
```

```
MIN: 0 MAX: 255
```

```
import matplotlib.pyplot as plt
plt.imshow(img, cmap="gray")
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(4023.5), np.float64(6047.5), np.float64(-0.5))
```

Day 2

```
%%writefile edges.cpp
#include <opencv2/opencv.hpp>
#include <iostream>
```

```
using namespace cv;
using namespace std;

int main()
{
    Mat gray = imread("gray_gpu.jpg", IMREAD_GRAYSCALE);
    if (gray.empty())
    {
        cout << "gray_gpu.jpg not found\n";
        return -1;
    }

    // 🔧 Noise kam karne ke liye blur
    GaussianBlur(gray, gray, Size(9,9), 2.0);

    Mat edges;
   Canny(gray, edges, 160, 320);  // strong edges only

    imwrite("edges_gpu.jpg", edges);
    cout << "Edges generated\n";

    return 0;
}
```

```
Overwriting edges.cpp
```

```
!g++ edges.cpp -o edges `pkg-config --cflags --libs opencv4`
!./edges
```

```
Edges generated
```

```
from PIL import Image
import matplotlib.pyplot as plt

plt.imshow(Image.open("edges_gpu.jpg"), cmap="gray")
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(4023.5), np.float64(6047.5), np.float64(-0.5))
```

```
%%writefile detect.cpp
#include <opencv2/opencv.hpp>
#include <iostream>

using namespace cv;
using namespace std;

int main()
{
    Mat original = imread("input.jpg");
    Mat edges = imread("edges_gpu.jpg", IMREAD_GRAYSCALE);

    if (original.empty() || edges.empty())
```

```
    {
        cout << "input or edges image missing\n";
        return -1;
    }

    // Binary edges
    threshold(edges, edges, 50, 255, THRESH_BINARY);

    vector<vector<Point>> contours;
    findContours(edges, contours, RETR_EXTERNAL, CHAIN_APPROX_SIMPLE);

    for (auto &c : contours)
    {
        Rect box = boundingRect(c);

        // 🔕 noise ignore
        if (box.area() < 2000) continue;
        if (box.width < 40 || box.height < 40) continue;

        // 🏢 Building (large)
        if (box.area() > 45000)
        {
           rectangle(original, box, Scalar(255,0,0), 4);
            putText(original, "Building",
                    Point(box.x, box.y-5),
                    FONT_HERSHEY_SIMPLEX, 0.6,
                    Scalar(255,0,0), 2);
        }
        // 🚗 Vehicle
        else
        {
            rectangle(original, box, Scalar(0,0,255), 3);
            putText(original, "Vehicle",
                    Point(box.x, box.y-5),
                    FONT_HERSHEY_SIMPLEX, 0.5,
                    Scalar(0,0,255), 1);
        }
    }

    imwrite("output_boxes.jpg", original);
    cout << "Detection complete\n";

    return 0;
}
```

```
Overwriting detect.cpp
```

```
!g++ detect.cpp -o detect `pkg-config --cflags --libs opencv4`
!./detect
```

```
Detection complete
```

```
plt.imshow(Image.open("output_boxes.jpg"))
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(4023.5), np.float64(6047.5), np.float64(-0.5))
```

```
!g++ edges.cpp -o edges `pkg-config --cflags --libs opencv4`
!./edges

!g++ detect.cpp -o detect `pkg-config --cflags --libs opencv4`
!./detect
```

```
Edges generated
Detect.cpp compile
```

```
from PIL import Image
import matplotlib.pyplot as plt

plt.imshow(Image.open("output_boxes.jpg"))
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(4023.5), np.float64(6047.5), np.float64(-0.5))
```