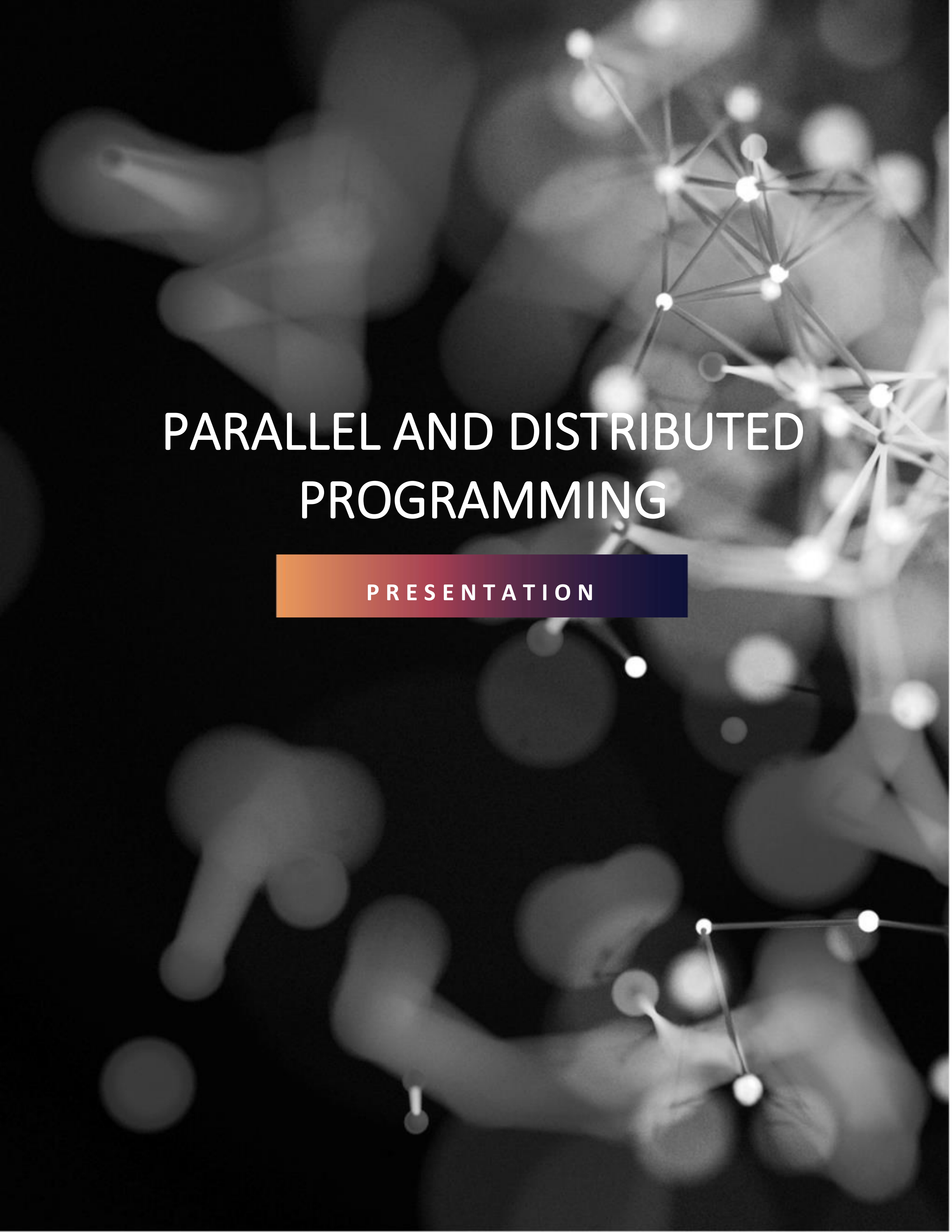


PARALLEL AND DISTRIBUTED PROGRAMMING

PRESENTATION

GROUP MEMBERS

NAME:	HASHIM MUNEEB
REG NO:	FA20-BCS-136
NAME:	TOOBA SYED
REG NO:	FA20-BCS-130
NAME:	WASEEM TASAWAR
REG NO:	FA20-BCS-133
NAME:	FAZEELA REHMAN
REG NO:	FA20-BCS-090

SUBMITTED TO:
SIR EZAZ MUSTAFA

EDGE DETECTION USING OPENMP

Edge detection is a technique used to identify the boundaries or edges within an image. The Sobel operator is a popular edge detection method that uses convolution with a pair of 3x3 kernels to approximate the gradient of the image intensity.

Steps in Edge Detection Using the Sobel Operator:

Grayscale Conversion:

- Convert the image to grayscale to simplify the processing by reducing the color channels to a single intensity channel.

Apply Sobel Kernels:

- Convolve the image with the Sobel kernels in the X and Y directions to detect horizontal and vertical edges.

Compute Gradient Magnitude:

- Combine the gradients from the X and Y directions to get the magnitude of the gradient. This is typically done using:

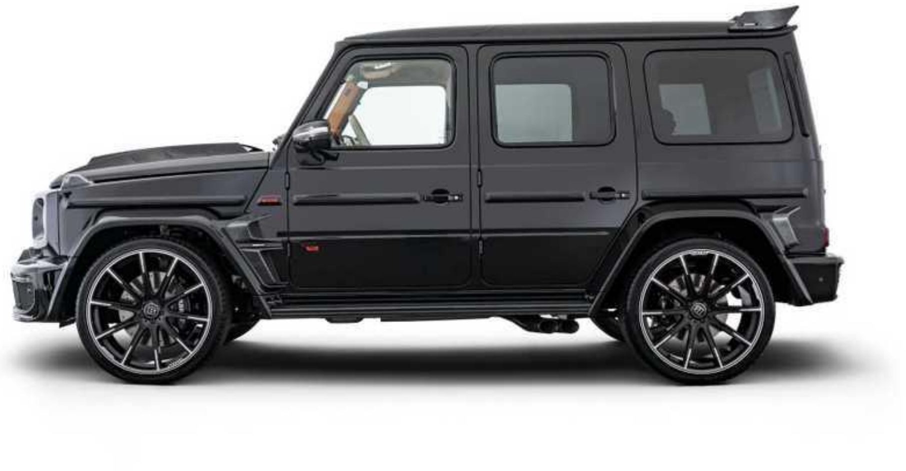
$$\text{Gradient Magnitude} = \sqrt{(G_x)^2 + (G_y)^2}$$

where Gx and Gy are the gradients in the X and Y directions, respectively.

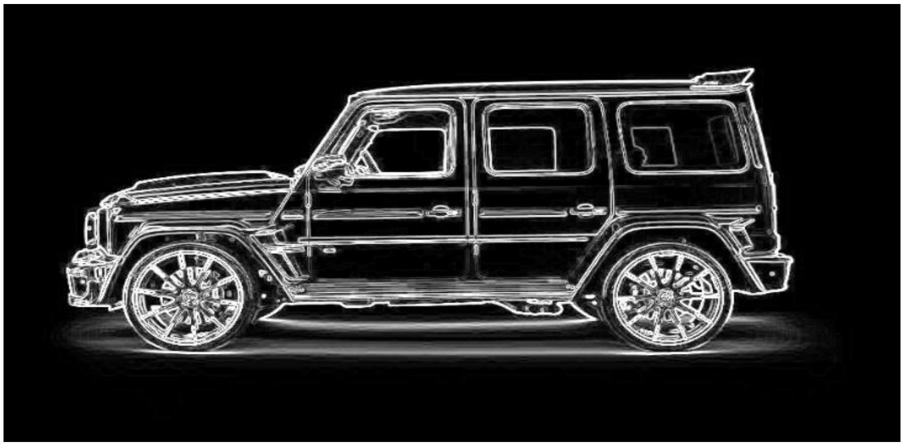
CODE:

```
#include <iostream>
#include <opencv2/opencv.hpp>
#include <omp.h>

int main() {

    cv::Mat image = cv::imread("C:\\Users\\user\\Desktop\\car.jpg", cv::IMREAD_GRAYSCALE);

    if (image.empty()) {
        std::cerr << "Error: Could not open or find the image!" << std::endl;
        return -1;
    }


    cv::Mat edges(image.size(), CV_8U);

    int Gx[3][3] = {
        {-1, 0, 1},
        {-2, 0, 2},
        {-1, 0, 1}
    };
    int Gy[3][3] = {
        {-1, -2, -1},
        { 0,  0,  0},
        { 1,  2,  1}
    };




// PARALLEL IMPLEMENTATION OF SOBEL KERNEL



    #pragma omp parallel for collapse(2)
    for (int y = 1; y < image.rows - 1; y++) {
        for (int x = 1; x < image.cols - 1; x++) {
            int sumX = 0;
            int sumY = 0;


            for (int i = -1; i <= 1; i++) {
                for (int j = -1; j <= 1; j++) {
                    sumX += Gx[i + 1][j + 1] * image.at<uchar>(y + i, x + j);
                    sumY += Gy[i + 1][j + 1] * image.at<uchar>(y + i, x + j);
                }
            }
            int magnitude = sqrt(sumX * sumX + sumY * sumY);
            edges.at<uchar>(y, x) = cv::saturate_cast<uchar>(magnitude);
        }
    }


    cv::imwrite("edges.jpg", edges);

    std::cout << "Edge detection completed and saved as edges.jpg" << std::endl;
    return 0;
}
```

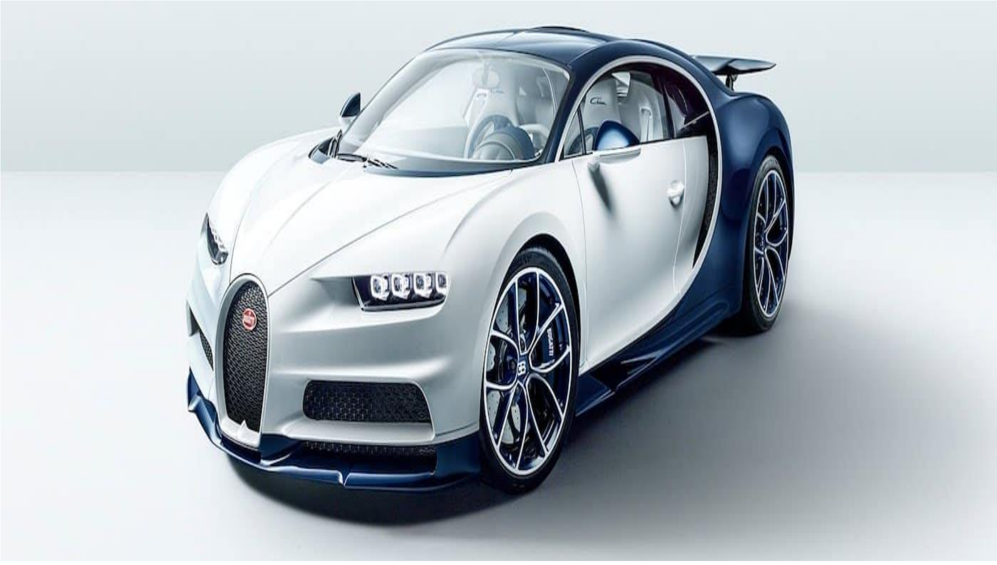
INPUTS



OUTPUT



INPUT



OUTPUT