Code (PA7.cu)

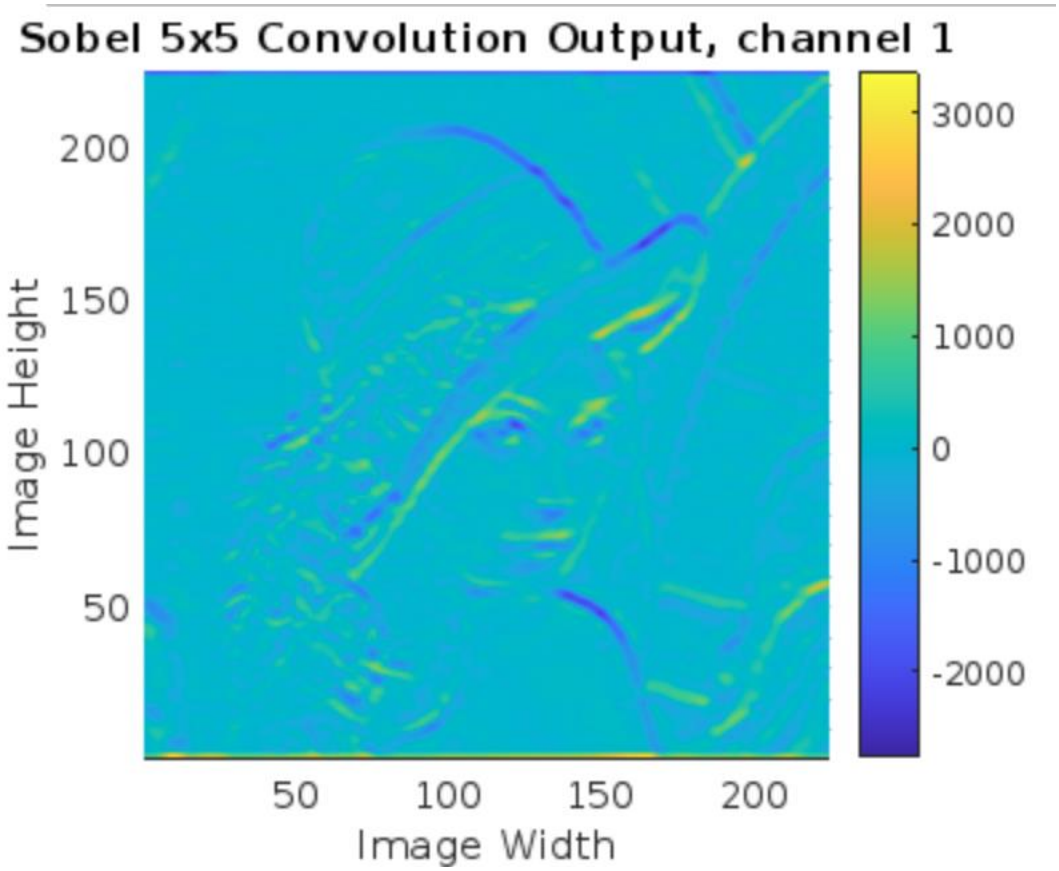
```
#include <stdlib.h>
#include <iostream>
#include <cuda_runtime.h>
#include <cstdlib>
#include <ctime>
#include <fstream>

using namespace std;
#define TILE_WIDTH 16
__global__ void convolution2D(unsigned int* inputImage, int* filter, int* output, int height, int
width,int channels, int filterSize){
        int row = blockIdx.y * blockDim.y + threadIdx.y;
        int col = blockIdx.x * blockDim.x + threadIdx.x;
        if(row < height && col < width){
          for (int c = 0; c < channels; ++c) {
                int pixelVal = 0;
                int start_col = col - (filterSize/2);
                int start_row = row - (filterSize/2);
                for(int i = 0; i<filterSize; i++){
                        for(int j = 0; j<filterSize; j++){
                           int row_idx = start_row + i;
                           int col_idx = start_col + j;
                           if(row_idx >= 0 && row_idx <= height-1 && col_idx >= 0 && col_idx <=
width -1){
                                pixelVal += inputImage[(row_idx*width+col_idx)*channels + c] *
filter[i*filterSize+j];
                           }
                        }
                }
                output[(row*width+col)*channels + c] = pixelVal;
            }
        }
}

int main(int argc, char* argv[]){
   unsigned int* hostInputImage;
   int* hostOutputImage;
   unsigned int inputLength = 225 * 225 * 3;

   hostInputImage = new unsigned int[inputLength];
   hostOutputImage = new int[inputLength];
```

```
    std::ifstream infile("image1.dat");
    unsigned int pixelValue;
    unsigned int i = 0;

    while (infile >> pixelValue && i < inputLength) {
        hostInputImage[i++] = pixelValue;
    }
    infile.close();

    int maskRows = 5;
    int maskColumns = 5;
    int imageChannels = 3;
    int imageWidth = 225;
    int imageHeight = 225;

    int hostMask[5][5] = {
        {2, 2, 4, 2, 2},
        {1, 1, 2, 1, 1},
        {0, 0, 0, 0, 0},
        {-1, -1, -2, -1, -1},
        {-2, -2, -4, -2, -2}
    };

    unsigned int* deviceInputImage;
    int* deviceOutputImage;
    int* deviceMask;

    size_t imageSize = imageWidth * imageHeight * imageChannels;

    cudaMalloc((void**)&deviceInputImage, imageSize * sizeof(unsigned int));
    cudaMalloc((void**)&deviceOutputImage, imageSize * sizeof(int));
    cudaMalloc((void**)&deviceMask, maskRows * maskColumns * sizeof(int));

    cudaMemcpy(deviceInputImage, hostInputImage, imageSize * sizeof(unsigned int),
cudaMemcpyHostToDevice);
    cudaMemcpy(deviceMask, hostMask, maskRows * maskColumns * sizeof(int),
cudaMemcpyHostToDevice);

    dim3 dimGrid(ceil((float)imageWidth / TILE_WIDTH), ceil((float)imageHeight / TILE_WIDTH));
    dim3 dimBlock(TILE_WIDTH, TILE_WIDTH, 1);

    convolution2D<<<dimGrid, dimBlock>>>(deviceInputImage, deviceMask,
deviceOutputImage,imageHeight,imageWidth,imageChannels,5);
```

```
    cudaDeviceSynchronize();

    cudaMemcpy(hostOutputImage, deviceOutputImage, imageSize * sizeof(int),
cudaMemcpyDeviceToHost);
   std::cout << "Writing output file..." << std::endl;
    std::ofstream outfile("image1.out");
    for (unsigned int i = 0; i < inputLength; ++i) {
       outfile << hostOutputImage[i] << "\n";
    }
    outfile.close();

    cudaFree(deviceInputImage);
    cudaFree(deviceOutputImage);
    cudaFree(deviceMask);
    delete[] hostInputImage;
    delete[] hostOutputImage;

    return 0;
}
```

Original Image

3D Surface Plot


Sobel 5x5 Convolution Output, channel 1
Image Height
200
150
100
50
50
100
150
200
Image Width
3000
2000
1000
0
-1000
-2000