**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;

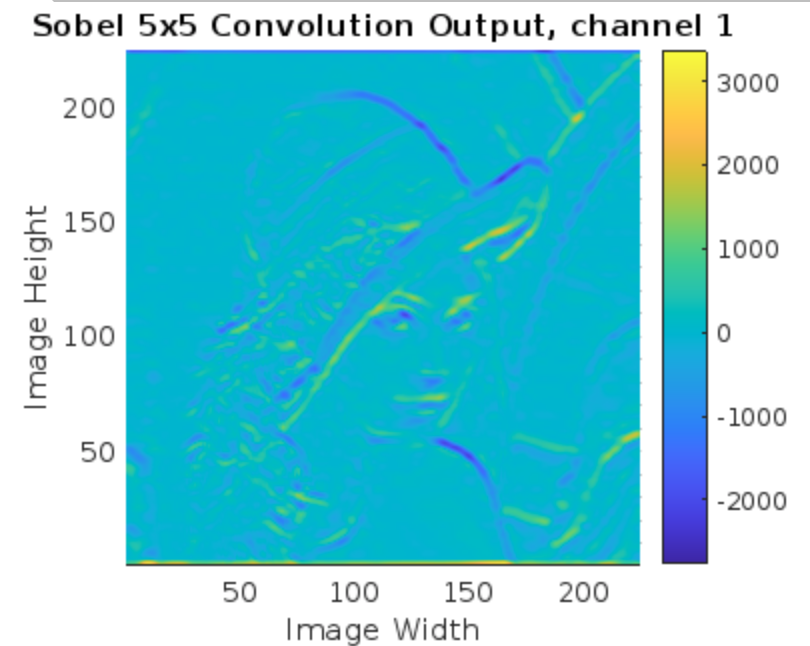
return 0;

}

**Original Image**

****

**3D Surface Plot**

****