SVEUČILIŠTE U SPLITU  
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

SEMINARSKI RAD

Napredne računalne arhitekture

SOBEL FILTER

Ivan Akrapović

Jozo Krstanović

Dominik Mišadin

Split, lipanj 2025.

**SADRŽAJ**

[1. UVOD 1](#_Toc200472349)

[2. CUDA 2](#_Toc200472350)

[3. Sobel filter 3](#_Toc200472351)

[4. Implementacija 4](#_Toc200472352)

[5. Zaključak 6](#_Toc200472353)

[LITERATURA 7](#_Toc200472354)

# UVOD

Uvod

# CUDA

Opisat Nvidia CUDA

# Sobel filter

Opisati što je, kako funkcionira (kernel 3x3 matrica, x i y prolazi)

# Implementacija

Pokazati sitnice u kodu, možete screenshotove umjesto ovakog teksta

|  |
| --- |
| **Isječak koda 3.1.** Zaključavanje threada pomoću mutex biblioteke |
| // g++ -o mutex mutex.cpp -lpthread -std=c++11  #include <iostream>  #include <thread>  #include <vector>  #include <mutex>  #include <chrono>  using namespace std;  using namespace std::chrono;  const int INCREMENT\_COUNT = 100000000;  int counter = 0;  std::mutex mtx;  void increment\_task(int iterations)  {      for (int i = 0; i < iterations; ++i)      {          ++counter;      }  }  void increment\_task\_mutex(int iterations)  {      for (int i = 0; i < iterations; ++i)      {          std::lock\_guard<std::mutex> lock(mtx); // automatski unlock          ++counter;      }  }  void run\_threads(int num\_threads, bool print\_time = true)  {      counter = 0;      auto start = high\_resolution\_clock::now();      vector<thread> threads;      int per\_thread = INCREMENT\_COUNT / num\_threads;      for (int i = 0; i < num\_threads; ++i)      {          threads.emplace\_back(increment\_task\_mutex, per\_thread);      }      for (auto &t : threads)      {          t.join();      }      auto stop = high\_resolution\_clock::now();      auto duration = duration\_cast<milliseconds>(stop - start);      if (print\_time)      {          cout << "Threads: " << num\_threads << "\t| Final counter: " <<   counter << " | Time: " << duration.count() << " ms" << endl;      }  }  void run\_sequential()  {      counter = 0;      auto start = high\_resolution\_clock::now();      increment\_task(INCREMENT\_COUNT);      auto stop = high\_resolution\_clock::now();      auto duration = duration\_cast<milliseconds>(stop - start);      cout << "Sequential \t| Final counter: " << counter           << " | Time: " << duration.count() << " ms" << endl;  }  int main()  {      run\_sequential();      run\_threads(2);      run\_threads(4);      run\_threads(8);      run\_threads(12);      return 0;  } |

# Zaključak

LITERATURA

1. Mitchel, T. M.: „Machine Learning“, McGraw-Hill Science/Engineering/Math, 1997.