Teaching Parallel Programming Using Computer Vision and Image Processing Algorithms

Professor Dan Connors
email: Dan.Connors@ucdenver.edu
Department of Electrical Engineering
College of Engineering and Applied Science
University of Colorado Denver
Motivation

• Students encounter the challenges of programming multicore & GPU systems without the proper background

• Effective parallel programming requires concepts across computer architecture, compilers, and operating system

• Introduction of emerging PDC concepts
  • Any change in teaching traditional programming systems must be predicated on clear motivational examples and view of the benefits in real-world application and problem domains
  • The STEM [Science, Technology, Engineering, and Math] effort is focused on demonstrating the impact of the field. The concept of “programming computers” by itself may not be enough to sustain interest in our field.
Approach

- The focus of our approach is to integrate GPU-related programming concepts in distinct phases within multiple courses in the curriculum
- Adopt real applications/real systems as the learning motivation and use them in teaching related topics
- Provide project-based experiences and opportunities

1st Year
- Motivate Parallelism
- Expose students to the concept of high-level concepts of GPU parallelism
- Relate parallelism to performance
- Real-world application domains & demonstrations

2nd Year
- Code Parallelism
- Students deploy API interfaces and GPU templates
- Understand the scenarios for deploying GPGPU codes: CUDA/OpenCL

3rd Year
- Optimize Parallelism
- Students advance their understanding of GPU model
- Overcome performance bottlenecks with knowledge of computer organization concepts

4th Year
- Explore Parallelism
- Students independently leverage GPU systems
- Investigate open-ended projects with GPU acceleration
UC Denver – NSF/TCPP PDC Curriculum

**Digital Foundation**
- **8 hours**
  - ELEC 1201-1: Intro to Electrical Engineering
  - ELEC 1510-3: Logic Design
  - ELEC 1520-3: Embedded Systems I

**Digital Core**
- **6 hours**
  - ELEC 2520-3: Embedded Systems II
  - ELEC 3651-3: Digital Hardware Design

**Digital Specialty**
- **7 hours**
  - ELEC 4501-3: Microprocessor-Based Design (MBD)
  - ELEC 4521-1: MBD Lab
  - ELEC 4511-3: Hardware-Software Interface Design (HW-SW)
  - ELEC 4561-1: HW-SW Lab
  - ELEC 4723-3: Computer Architecture
  - ELEC 4727-3: Computer Vision Acceleration

---

**Notes:**
- NSF/TCPP PDC Curriculum Early Adopter Initiative
Real-World Applications

- Computer Vision is an excellent domain to showcase the emerging need for parallel execution and parallel programming concepts

Application Areas

- Navigation – CV is used to guide a car or robot along roads or paths while avoiding obstacles
- Captchas
- Autonomous Navigation
- Object Recognition & Augmented reality
Real-World Applications

- Computer Vision is an excellent domain to showcase the emerging need for parallel execution and parallel programming concepts.
Example: SIFT Keypoint Matching with GPUs

- CUDA/OpenCL GPU Execution of Keypoint Matching

```c
__kernel void SiftMatch()
{
  int gid = get_global_id(0);
  // Code for an individual workitem: scan all database keypoints
  int min0, min1, diff, match = -1;
  min0 = 1000000;
  min1 = 1000000;
  // SCAN all keypoints looking to generate best match
  for (int db_i = 0; db_i < DB_Keypoints; db_i++) {
    int sum = 0;
    for (int kp_i = 0; kp_i < 128; kp_i++) {
      float diff = (Frame0[gid*128 + kp_i] - Frame1[db_i*128 + kp_i]);
      sum += diff * diff;
    }
    if (sum < min0) {
      min1 = min0; min0 = sum; match = db_i;
    } else if (sum < min1) {
      min1 = sum;
    }
  }
  d_distance[gid] = min0;
  if (100 * min0 < threshold * min1)
    d_index[gid] = match;
  else
    d_index[gid] = -1; // Match doesn’t pass threshold test
}
```

- MnN Keypoint comparison
- Keypoint: 128 vector length
- Usually SAD – sum of differences or Vector Distance
- Computation: MnNx128x2

\[
Score_{ij} = \sum_{i=0}^{128} (Frame_0[i] - Frame_1[i])^2
\]

- For each Target Keypoint: find 2 lowest scores: \( min[0], min[1] \)
- Keypoint Match if: \( min[0] < a \cdot min[1] \)
- \( a \) typically evaluated at \( [60] \)
Example: SIFT Keypoint Matching with GPUs

- SIFT (Scale Invariant Feature Transform) – CPU versus GPU performance and scaling
Computer Vision Summary

- CV exercises integrated into OpenCV: data-parallel GPU execution
- Object recognition with histograms
- Edge detection (sobel filter)
- Parallel K-means clustering and KNN classification
- Glyph matching

- The NVIDIA Corporation funded our program with a CUDA Center of Teaching Excellence grant
  - Enabled the resource for an additional Teaching Assistant (TA) for one semester

- The NSF/TCPP Curriculum awarded an Early Adopter Award for developing core curriculum for CS/CE undergraduates related to parallel and distributed computing (PDC) topics. (2012)