Moose Blazers

Exploring GPGPU Processing on Embedded Systems

Members: Jordan Pringle, Melissa Anewalt, Scott Walstead, Peter Gaede

Special Thanks: Ronald Scrofano, Chandra Krintz, Tim Sherwood, Janet Kayfetz, Kyle Jorgensen

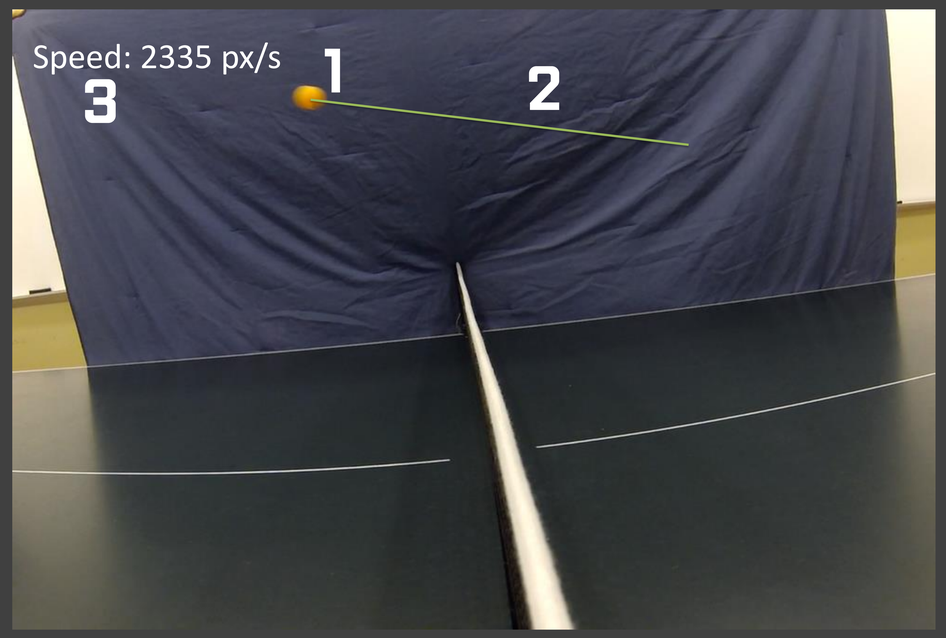
Our **Mission**...

General purpose computing on graphics processing units (GPGPU), in which graphics processing units (GPUs) are used as application accelerators, has been a mainstay of high-performance computing for several years. Recently, GPGPU has come to be supported in embedded systems, where there is an emphasis on low power consumption like the Nvidia Jetson TK1. The CPU in the system is not suitable for computationally intensive computer vision applications. To overcome this limitation, we used Nvidia's CUDA parallel computing architecture to distribute the computational load across all 192 cores of the integrated GPU. This implementation increases performance while still retaining the low power characteristics of the Jetson development kit.

Implemented on **NVIDIA Jetson TK1**...

- **nVidia** 2.32GHz Quad-Core ARM CPU
- **nVidia** Kepler GK20a GPU
- 192 CUDA Cores (~326 GFLOPS)
- 2GB DDR3L Memory
- <5W Power Usage Under Load
- HDMI Out
- USB 3.0

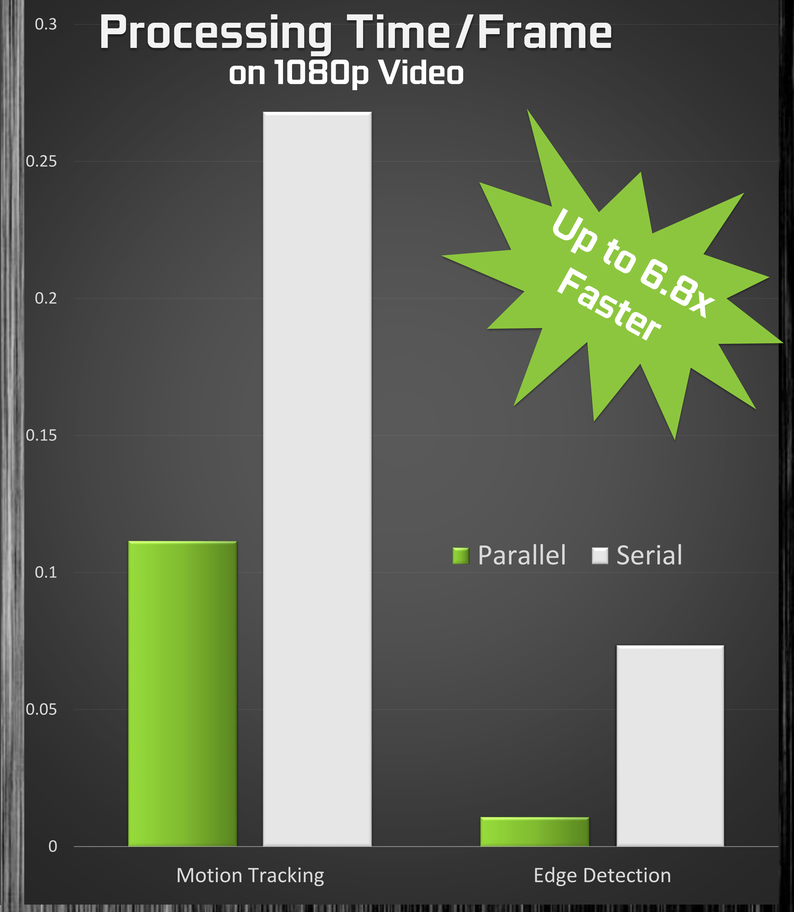
On top of **nVidia's CUDA Parallel Computing Architecture**, **OpenCV**'s Computer Vision libraries enable us to efficiently process the data within each video frame.



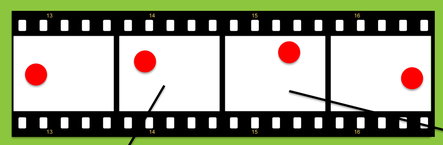
The Main Screen:

Our Prediction engine takes into consideration the speed and direction of the ball to decide if the ping pong ball will make it over the net or not. The output video including speed and vectors can be saved at the end.

1. The Ball being tracked
2. Velocity vector showing speed and angle
3. The ball's current velocity



How it all **works**...

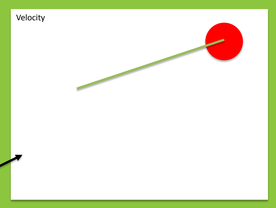


We start with an **input** file that we want to analyze.

Capture the Frame

Color Recognition finds the specific color of the ball and rejects the background

Bring in the **Next Frame**



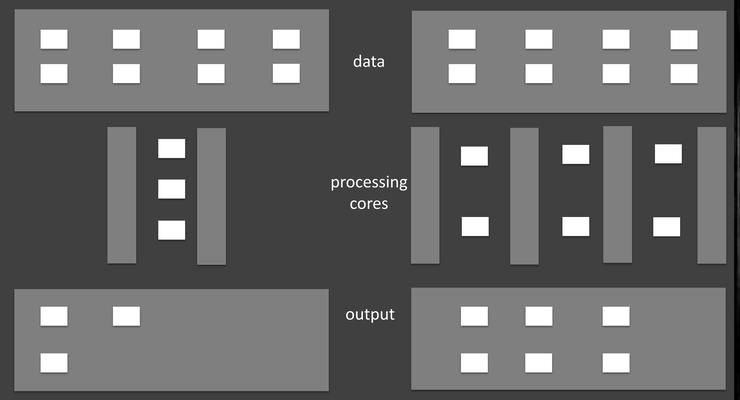
Display on Screen
The output displays the velocity of the ball along with a green line tracking the ball from frame to frame.

Edge Detection finds the ball in the Frame

Calculate the **Center** of the Ball

Calculate **Velocity** and **Trajectory** from the Previous Position

Serial Processing vs. Parallel Processing



These images depict the differences between serial and parallel processing. The serial version on the left can only process one pixel at a time but it can do it quickly. The parallel processor can process many pixels at once but at a slower rate.