Efficient Deep Racing via Environmental Constraining

Ki Young Bang, Brandon Lam, Jeffery Tsz Hang Wong, Leo Jingtao Zhang
Pooria M. Yaghini
Department of Electronic Engineering and Computer Science

Background

- Autonomous vehicles have grown increasingly complex and resource intensive.
- Autonomous vehicles today take inputs from numerous sensors and can handle events such as loss of traction, poor visibility, and pedestrian avoidance.
- The standard for decent performance has increased dramatically.
- It has been practically accepted that systems outside of the range of thousands of dollars can not come close to today's standard of performance.

Project Goal

Create a cost-efficient yet high-performance autonomous driving system by constraining the number of variables by using a controlled environment; Mario Kart is one of such possible domains. This provides three benefits:

1. Raw images are inputted directly through HDMI, bypassing sensor issues.
2. A limited number of unique obstacles reduces the complexity for object detection.
3. Controller inputs create deterministic movement, avoiding the need to handle details such traction and gear changes

References

Togelius, Julian, et al. Computational Intelligence in Racing Games. *Computational Intelligence in Racing Games,* Springer-Verlag Berlin Heidelberg, 2007.

"Nintendo Gamecube Controller Protocol." *Nintendo Gamecube Controller Pinout,* 8 Mar. 2004, www.int03.co.uk/crema/hardware/gamecube/gc-control.html.

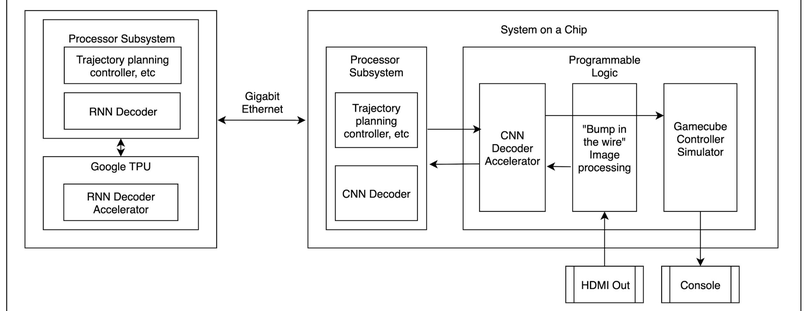
System Description/Materials

There system requires 4 components:

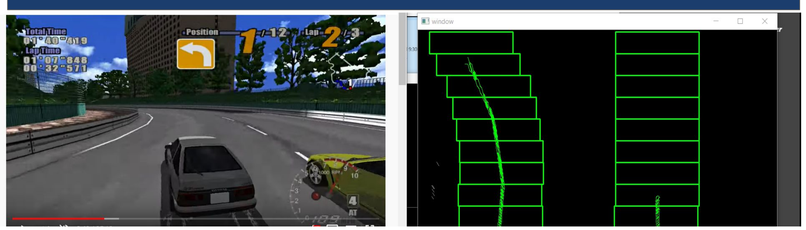
1. Processing element for vision
2. Interface to emulate controller input
3. Processing element for decisions
4. Processing element for trajectory planning

We will use a distributed system consisting of a Zynq-7000 SoC and Coral Devboard connected over gigabit ethernet. The Zynq provides a traditional CPU and an FPGA which we can use as the CNN encoder/image classification (1) and provide output controller signals using the gamecube controller protocol (2). Encoded images are sent to the Coral Dev Board over gigabit ethernet which will perform RNN decoding (3 & 4).

Block Diagram



Software Demo



Timeline/Milestones

- Quarter 1
  - Week 1~4: Become familiar with libraries, core, methodologies,etc
  - Week 5~7: Hardware/Software development
  - Week 8~9: Integration/Debugging
  - Week 10: First prototype
- Quarter 2
  - Week 1~5: Improve the driving algorithm
  - Week 6~8: Final debugging
  - Week 8~10: Final prototype
  - Ongoing: Collect training data

Quarterly Accomplishments

- Hardware
  - Learned to program and deploy code on SoC
  - Circuit designed for FPGA controller output logic & PCB
  - FPGA programming server has been setup
  - Finalized hardware system design
- Software
  - Developed a curved lane detection
    - Captures a laptop screen and detects the lane lines from a footage in real time
  - Challenge: calculate an accurate numerical value of instantaneous curvature for trajectory planning

Future Tasks

- Hardware
  - Accelerate the neural networks
  - Test controller simulation with real hardware
- Software
  - Implement TVM+VTA stack and PYNQ for CNN encoding
  - Further enhance the lane detection algorithm
  - Develop path/trajectory planning