Collision Avoidance on NVIDIA Tegra®

Dr. Christoph Lauer, Audi AG
Dr. Richard Membarth, DFKI
The Digitalization of the Automobile
The Digitalization of the Automobile
The Digitalization of the Automobile
New Technology Enables New Functionality

- Audis are already driving supercomputers
- Sensors are getting better and more accurate
- New functions will emerge
  - Gesture/voice recognition
  - Piloted driving
  - Safety functions (focus today)
360° Context Analysis is Computationally Expensive
Prediction Approach to Environment Analysis
Prediction Approach to Environment Analysis

- Objects are described by their physical properties
Prediction Approach to Environment Analysis

- Objects are described by their physical properties
- Movement is sampled and extrapolated
Prediction Approach to Environment Analysis

- Objects are described by their physical properties
- Movement is sampled and extrapolated
- All object hypotheses are combined with each other
Prediction Approach to Environment Analysis

- Objects are described by their physical properties
- Movement is sampled and extrapolated
- All object hypotheses are combined with each other
- In one example scene up to 10 million scenarios per iteration
AnyDSL

- A compiler framework for domain-specific libraries
  - Handle complexity through high-level code
  - Provide efficient real-time performance

- Separation of concerns through code refinement
  - Higher-order functions
  - Partial evaluation
  - Triggered code generation

http://anydsl.github.io
AnyDSL Compiler Framework

- Impala language (Rust dialect)
  - Functional & imperative language
- Thorin compiler [GPCE’15 *best paper award*]
  - Higher-order functional IR [CGO’15]
    - Special optimization passes
    - No overhead during runtime
- Whole-function vectorizer [CGO’11]
High-level Algorithm Description

- Trajectory computation
- Captures essence of algorithm

```python
for idx_t, object, hypothesis in iterate_hypotheses(num_objects, num_hypotheses) {
    // ...
    for t in range(0, num_iterations) {
        // compute object position at time t
        // ...
        x += time_step * (vel * cos(...));
        y += time_step * (vel * sin(...));
        pos_x(idx_t(t)) = x;
        pos_y(idx_t(t)) = y;
    }
}
```
High-level Algorithm Description

- Trajectory computation
- Captures essence of algorithm
  - *for* syntax: syntactic sugar for lambda function as last argument

```c
iterate_hypotheses(num_objects, num_hypotheses, lidx_t, object, hypothesis) -> () {
    // ...
    for t in range(0, num_iterations) {
        // compute object position at time t
        // ...
        x += time_step * (vel * cos(...));
        y += time_step * (vel * sin(...));
        pos_x(idx_t(t)) = x;
        pos_y(idx_t(t)) = y;
    }
}
```
Algorithm Mapping to CPU

- Iteration logic & scheduling specific for CPU

```rust
fn iterate_hypotheses(num_objects: int, num_hypotheses: int,
                      body: fn(fn(int) -> int, int, int) -> ()
                    ) -> () {
    for hyp in range(0, num_hypotheses) {
        for obj in range(0, num_objects) {
            let idx = hyp * num_objects + obj;
            fn idx_t(t: int) -> int { t * num_objects * num_hypotheses + idx }
            body(idx_t, obj, hyp);
        }
    }
}
```
Algorithm Mapping to GPU

- Memory layout & iteration logic specific for GPU

```javascript
fn iterate_hypotheses(num_objects: int, num_hypotheses: int,
                      body: fn(fn(int) -> int, int, int) -> ()
                ) -> () {
  let grid = (num_hypotheses, num_objects, 1);
  let block = (num_hypotheses, 1, 1);

  with cuda(grid, block) {
    let obj = threadIdx.x() + blockDim.x() * blockIdx.x();
    let hyp = threadIdx.y() + blockDim.y() * blockIdx.x();
    let idx = obj * num_hypotheses + hyp;
    fn idx_t(t: int) -> int { t * num_objects * num_hypotheses + idx }
    body(idx_t, obj, hyp);
  }
}
```
Hardware-specific Optimizations

- Instruction mapping
- Intrinsics
- libdevice
- Memory mapping
  - global, shared, ldg
- Memory transfer
- Zero-copy
- Managed memory (unified memory)
Optimization through Partial Evaluation

- Matlab: CPU timing on Intel i5-4288U: ~ 6 minutes
- Partial evaluation triggered through `@-operator`

```java
for idx_t, object, hypothesis in @iterate_hypotheses(num_objects, num_hypotheses) {
    // ...
}
```

- AnyDSL: CPU timing on Tegra K1/X1: ~ 2 seconds
- AnyDSL: GPU timing on Tegra K1: 15 milliseconds
- AnyDSL: GPU timing on Tegra X1: 7 milliseconds
Key Message

- High-performance computing will push the digitalization of the automobile and enable new safety functions
- AnyDSL helps Audi to manage development complexity and boosts computing performance
Thank You