Integrating Algorithmic Parameters into Benchmarking and Design Space Exploration in 3D Scene Understanding

Luigi Nardi, PhD
Software Performance Optimisation group
@PACT Haifa
September 12th 2016

In collaboration with:
Take away messages

1. Our community has to move from benchmarks that are used like unit tests, towards benchmarks that expose performance/energy tradeoffs against quantitative end-to-end quality of result.

2. Doing so opens up the scope for adaptivity, and helps ensure we are optimising at lower levels for the right profile of higher-level operations.

3. Some of the techniques we use for auto-tuning at the compiler/architecture level can be used at the algorithmic adaptation level - in fact achieving much bigger performance improvements.
Simultaneous localisation and mapping (SLAM)
Build a coherent world representation and localise the camera in real-time

Applications, e.g.:
- Robotics
- Autonomous driving
- 3D printing
- Augmented reality
What CV researchers say about KinectFusion performance

"KinectFusion cannot run in real-time on mobile"

"You need a fat GPU to run KFusion"
Holistic approach to SLAM performance:

**SLAMBench**

A publicly-available benchmarking framework for quantitative, comparable and validatable experimental research to investigate trade-offs in performance, accuracy and energy consumption of a SLAM system.

**Error metric:** absolute trajectory error (ATE) based on dataset ground truth

*Introducing SLAMBench, a performance and accuracy benchmarking methodology for SLAM (ICRA 2015)*
SLAMBench framework

SLAM benchmarks
- KinectFusion (Dense SLAM)
- LSD-SLAM (Semi-dense SLAM)
- ORB-SLAM (Sparse SLAM)

Implementation languages
- C++
- OpenMP
- OpenCL
- CUDA
- SYCL
- PENCIL

Desktop to embedded platforms
- ARM
- Intel
- NVIDIA

Datasets
- ICL-NUIM
- TUM RGB-D

Performance evaluation
- Frame rate
- Energy
- Accuracy
### Configuration parameters:

**Space 1**
- **Algorithmic:**
  - Application-specific parameters
  - Minimisation methods
  - Early exit condition values

**Space 2**
- **Compilation:**
  - opencl-params: -cl-mad-enable,-cl-fast-relaxed-math, etc.
  - LLVM flags: O1, O2, O3, vectorize-slp-aggressive, etc.
  - Local work group size: 16/32/64/96/112/128/256
  - Vectorisation: width (1/2/4/8), direction (x/y)
  - Thread coarsening: factor (1/2/4/8/16/32), stride (1/2/4/8/16/32), dimension (x/y)

**Space 3**
- **Architecture:**
  - GPU frequency: 177/266/350/420/480/543/600/DVFS
  - # of active big cores: 0/1/2/3/4
  - # of active LITTLE cores: 1/2/3/4

---

**What is the optimisation space?**

Warning: huge spaces, impossible to run exhaustively
# KinectFusion algorithmic features

<table>
<thead>
<tr>
<th>Features</th>
<th>Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume resolution</td>
<td>64x64x64, 128x128x128, 256x256x256, 512x512x512</td>
</tr>
<tr>
<td>$\mu$ distance</td>
<td>0 .. 0.5</td>
</tr>
<tr>
<td>Pyramid level iterations (3 levels)</td>
<td>0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11</td>
</tr>
<tr>
<td>Image resolution (image ratio)</td>
<td>1, 2, 4, 8</td>
</tr>
<tr>
<td>Tracking rate</td>
<td>1, 2, 3, 4, 5</td>
</tr>
<tr>
<td>ICP threshold</td>
<td>$10^{-6} .. 10^{2}$</td>
</tr>
<tr>
<td>Integration rate</td>
<td>1 .. 30</td>
</tr>
</tbody>
</table>

**Image resolution (image ratio):**
- 640x480
- 320x240
- 160x120
- 80x60
Incremental optimisation process is not the way to achieve an optimal result, would be better to explore the full product space.
Algo design-space exploration (DSE)

Samples

Algorithmic configuration parameters

Machine learning

Predictive model

Time

Power

Error

Active learning

New samples

Run
Machine learning methods used

Decision Tree

Random Forest
DSE on algorithmic parameters error/runtime

<table>
<thead>
<tr>
<th>Machine</th>
<th>CPU</th>
<th>CPU name</th>
<th>CPU GFLOPS</th>
<th>CPU cores</th>
<th>GPU</th>
<th>GPU name</th>
<th>GPU GFLOPS</th>
<th>TDP Watts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardkernel ODROID-XU3</td>
<td>ARM A15 + A7</td>
<td>Exynos 5422</td>
<td>80</td>
<td>4 + 4</td>
<td>ARM</td>
<td>Mali-T628</td>
<td>60 + 30</td>
<td>10</td>
</tr>
</tbody>
</table>

_accuracy limit = 0.05 m

- Default configuration
- Active learning
- Random sampling
DSE compiler parameters speedup
DSE architecture parameters power/runtime

- Configuration
- Default configuration
- Pareto front

Power consumption (W) vs Runtime (sec)
Luigi Nardi - Imperial College London

DSE final result

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Runtime (FPS)</th>
<th>Max ATE (cm)</th>
<th>Power (Watts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>6.03</td>
<td>4.41</td>
<td>2.77</td>
</tr>
<tr>
<td>Best runtime</td>
<td>39.85</td>
<td>4.47</td>
<td>1.47</td>
</tr>
<tr>
<td>Best accuracy</td>
<td>1.51</td>
<td>3.30</td>
<td>2.38</td>
</tr>
<tr>
<td>Best power</td>
<td>11.92</td>
<td>4.45</td>
<td>0.65</td>
</tr>
<tr>
<td>Power &lt; 1W</td>
<td>29.09</td>
<td>4.47</td>
<td>0.98</td>
</tr>
<tr>
<td>Power &lt; 2W</td>
<td>39.85</td>
<td>4.47</td>
<td>1.47</td>
</tr>
<tr>
<td>FPS &gt; 10</td>
<td>11.92</td>
<td>4.45</td>
<td>0.65</td>
</tr>
<tr>
<td>FPS &gt; 20</td>
<td>28.87</td>
<td>4.47</td>
<td>0.91</td>
</tr>
<tr>
<td>FPS &gt; 30</td>
<td>32.38</td>
<td>4.47</td>
<td>1.01</td>
</tr>
</tbody>
</table>

- Most of the improvement comes from the algorithmic space
- KinectFusion real-time on a popular embedded device
- Enabling auto-tuning at the domain-specific level
Crowdsourcing mobile Android SLAMBench

- SLAMBench OpenMP
- SLAMBench OpenCL

- It runs a set of Pareto configurations on the available languages on your device
- Then shows the best achieved result
Conclusion - take away messages

1. Our community has to move from benchmarks that are used like unit tests, towards benchmarks that expose performance/energy tradeoffs against quantitative end-to-end quality of result.

2. Doing so opens up the scope for adaptivity, and helps ensure we are optimising at lower levels for the right profile of higher-level operations.

3. Some of the techniques we use for auto-tuning at the compiler/architecture level can be used at the algorithmic adaptation level - in fact achieving much bigger performance improvements.
References I


Copyrights


• Author: Dyson Ltd. Dyson 360 Eye. [Video]. Retrieved from https://www.youtube.com/watch?v=OadhulCDAjk


• Author: Unknown. Compiler. [Image]. Retrieved from http://d3q6qq2zt8nhwv.cloudfront.net/107/large-icon.png