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

B. Boding, L. Nardi, M.Z. Zia et al. [1]

LSDPO (2017/2018) Paper Presentation
Tudor Tiplea (tpt26)

Problem

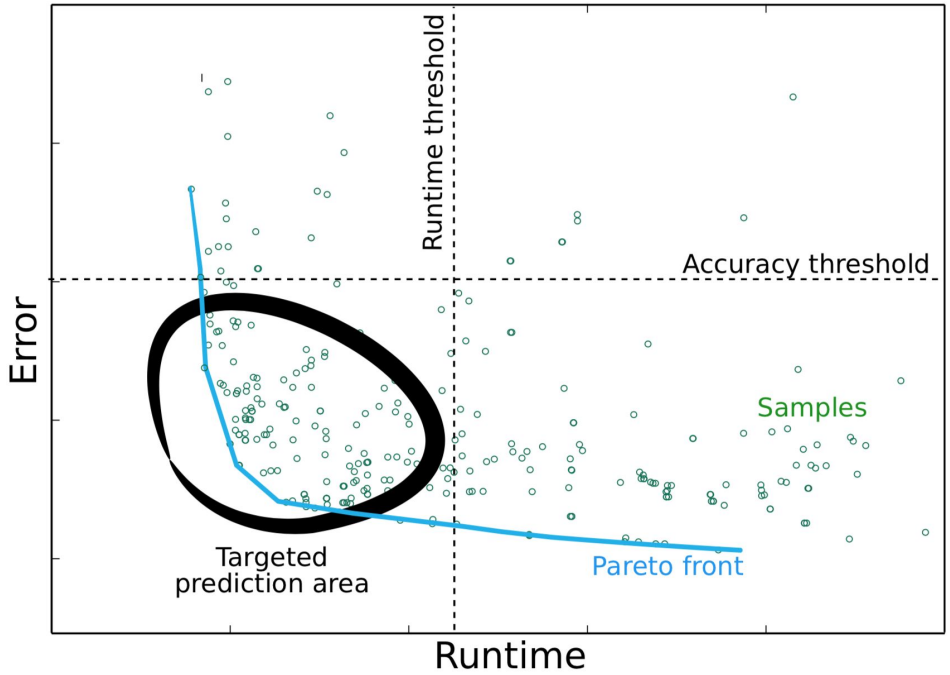
- Many modern systems must operate under increasingly more severe constraints
  - E.g. tight power consumption and thermal footprint constraints for mobile systems
- How can we help system designers make **informed trade-off decisions**?
  - E.g. balance performance/accuracy of a system under a power consumption < 1W constraint
- And how can we **automatically optimise** the system as much as possible?

Specific problem

- Demonstrate on a concrete application, a 3D scene understanding algorithm
  - High computational demands
- We can configure the system at the **algorithmic**, **compiler** and **architectural level**
  - Usual approaches only focus on the last two
- Measure performance in terms of **power consumption**, **runtime** (FPS) and **accuracy of computation**

Goal

- We want to identify the **Pareto optimal front** in the optimisation space
- These are the solutions that cannot be improved in any optimisation objective without degrading at least another objective



Performance model

- **1,800,000** possible configurations
- Cannot explore exhaustively
- Therefore, a **model predicting the performance** of a configuration must be built

	Parameters	Values
Algorithmic	**Volume resolution**	64x64x64, 128x128x128 256x256x256, 512x512x512
	μ distance	0.025, 0.075, 0.1, 0.2
	Pyramid level iterations	
	Level 1	3, 5, 7, 9, 11
	Level 2	3, 5, 7, 9, 11
	Level 3	3, 5, 7, 9, 11
	Compute size ratio	1, 2, 4, 8
	Tracking rate	1, 3, 5, 7, 9
	ICP threshold	$0, 10^{-4}, 10^{-5}, 10^{-6}, 1$
	Integration rate	1, 5, 10, 20, 30
Compiler	**OpenCL flags**	cl-mad-enable, cl-fast-relaxed-math, ...
	LLVM flags	O1, O2, O3, vectorize-slp-aggressive, ...
	Local work gr oup size	16, 32, 64, 96, 112, 128, 256
	Vectorization	
	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
Architecture	**GPU processor frequency**	177, 266, 350, 420, 480, 543, 600, DVFS
	Number of active big cores	0, 1, 2, 3, 4
	Number of active little cores	1, 2, 3, 4

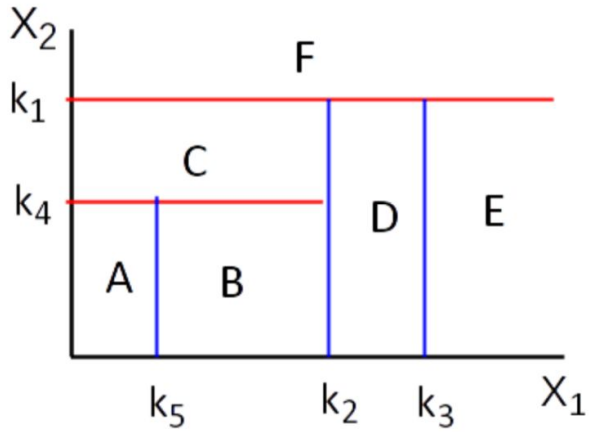
Active learning

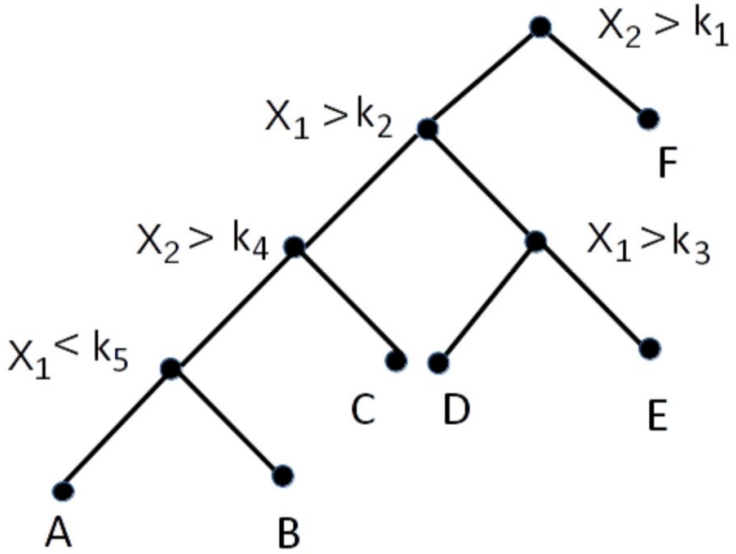
- Can bootstrap a predictive model using **active learning:**
  - Start with a random sample of configurations
  - Run the system with the sampled configurations
  - Measure the runtime, accuracy and power consumption
  - Train predictor using all the datapoints we've evaluated so far
  - Estimate Pareto optimal front using current predictor
  - Sample a new set of configurations localised in this new estimated area
  - Iterate
- In other words, use predictor to pick training examples that **improve its accuracy the most**

Randomised Decision Forest

- In this case, a better predictor than neural networks, SVMs and nearest neighbour
- A **decision tree** is a recursive binary partitioning of the input space:
  - A simple decision (1D threshold) at each **internal node**
  - **Output** of a **leaf** is average of training samples that reached that leaf
- A **randomised decision forest** is a collection of decision trees:
  - Output is average of outputs from each decision tree
  - Introduce **randomness** to remove variance in training:
    - Train each tree on random subset of training data
    - For each node, pick decision input variable randomly (e.g. volume resolution parameter)

Example: Binary Decision Tree





Step back - Co-design space exploration

- Follow an incremental, top-down approach:
  - Start with random sample of configurations
  - Estimate Pareto optimal front in the algorithmic level
  - Refine that at the compiler level
  - Refine even further at the architectural level

Results

- Greatest improvements gained at **algorithmic level** (6.35x improvement in execution time, 23.5% reduction in power consumption)
- Further improvements at lower levels, but of **smaller magnitude**
- Reached goal of running the 3D mapping in **real time**, on an **embedded device** with a **1W** power budget
- 4.8x execution time and 2.8x power consumption reductions over **hand-tuned, state-of-the-art implementations** of the 3D mapping algorithm

Opinion

- Shown that exploring algorithmic level is worth it for optimising a system
- But the approach doesn’t give the same impressive results at the lower levels
- This methodology was developed with this application in mind, no guarantee it would work well out of the box for other applications

Questions

Thank you!

References

[1] B. Bodin, L. Nardi, M. Z. Zia et al. '*Integrating Algorithmic Parameters into Benchmarking and Design Space Exploration in 3D Scene Understanding*'

All figures, plots and tables in this presentation are extracted from the paper above.