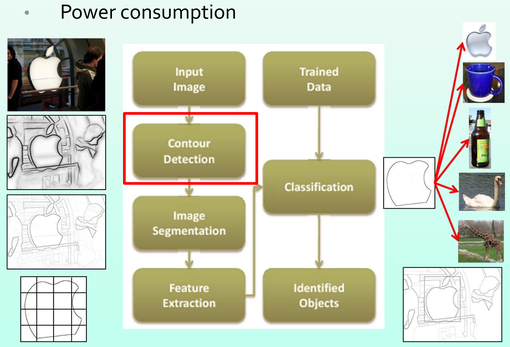
Contour Detection on Mobile Platforms

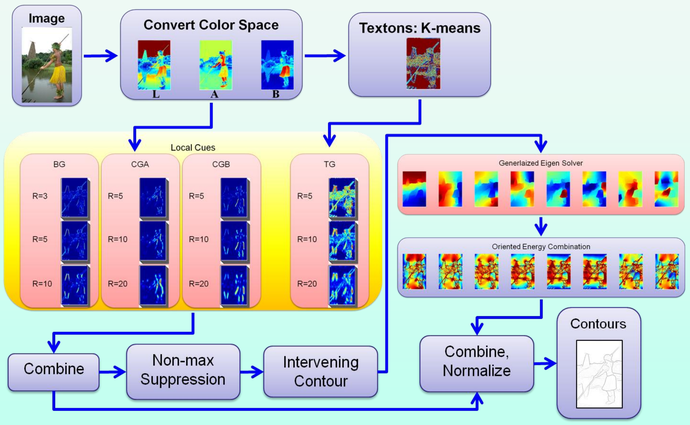
Bor-Yiing Su and Kurt Keutzer

CBIR on Mobile

- Using Damascene as a case study to understand the capability of modern mobile platforms in terms of:
  - Execution time
  - Power consumption



Damascene Computation Flow



Targeting Platforms

Cloud Platforms

Specifications	GTX 480
CUDA Cores	480
Processor Clock	1.4 GHz
SP FLOPS	1.35 T
Memory Bandwidth	177.4 GB/s
Memory Size	1.5 GB
L1 Cache + Shared Memory	64 kB per SM
L2 Cache	768 kB

Specifications	Core i7 920
Cores	4
Processor Clock	2.66 GHz
SP FLOPS	42.56 G
Memory Bandwidth	25.6 GB/s
L1 Cache	64 kB per core
L2 Cache	1MB per core
L3 Cache	8 MB

Mobile Platforms

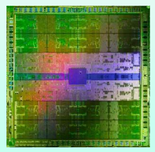
Specifications	Cortex A9
Cores	4
Processor Clock	800 MHz
SIMD Width (NEON)	128 bits
DMIPS	8000
L1 Cache	32 kB
L2 Cache	512 kB
Memory Bandwidth	N.A.

Specifications	MSM 8655
Cores	1
Processor Clock	1 GHz
SIMD Width (NEON)	128 bits
DMIPS	2100
L1 Cache	32 kB
L2 Cache	256 kB
Memory Bandwidth	N.A.

Performance on GTX 480

- The starting implementation
- Image size used for benchmarking: 321 x 481 pixels

Routine	Execution Time (ms)
Preprocess	0.789
Textons	129
Localcues	733
Intervening	14.12
Lanczos Solver	800
Poseprocess	7.96
Total	1.706



Performance on ARM Cortex A9

- Image size used for benchmarking : 205 x 154 pixels
  - Due to memory limitations
- Compiler flags
  - -O3 -ftree-vectorize -march=armv7-a -mfloat-abi=softfp -mfpu=neon
- Scalability is very close to linear on the number of threads

Routines	Number of Used Threads		
	1	2	4
texton	13.43	6.66	3.41
localcues	79.99	40.73	21.85
intervene	40.65	20.39	10.55
Lanczos	272.80	141.91	80.47
Total	410.26	213.11	119.70

Execution Time (s)

Routines	Number of Threads		
	1	2	4
texton	1	2.02	3.94
localcues	1	1.96	3.66
intervene	1	1.99	3.85
Lanczos	1	1.92	3.39
Total	1	1.93	3.43

Scalability on Number of Threads

Execution Time Analysis

- The performance gap between snapdragon and Core i7 920 is about 50x
  - Frequency difference: 1G vs. 2.66G
  - Memory bandwidth difference: 1G vs 25.6 G Bytes/s
  - A quad-core snapdragon with perfect scalability will close the gap to 12x
- Platform choice of computation
  - Transferring the entire image to the cloud and using the CUDA implementation will be faster than performing the computation locally on the mobile platform

Routine	Snapdragon	Cortex A9	Nehalem	Fermi
texton	5.19	3.41	0.15	0.03
localcues	52.15	21.85	1.22	0.16
intervene	6.55	10.55	0.17	0.003
Lanczos	151.75	80.47	2.93	0.68
Total	217.04	119.70	4.68	0.88

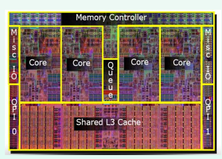
Execution Time (s)

Routine	Snapdragon	Cortex A9	Nehalem	Fermi
texton	1	1.52	34.60	162.19
localcues	1	2.39	42.75	317.99
intervene	1	0.62	38.53	2183.33
Lanczos	1	1.89	51.75	224.81
Total	1	1.81	46.35	247.48

Speedup Ratio

Performance on Core i7 920

- Porting the original CUDA implementation to serial C implementation
  - Using our revised algorithms from the CUDA implementation
- Using pthread to parallelize routines that execute more than 1 second on a 321 x 481 image
  - Exploring coarse grained parallelism on CPU instead of fine grained parallelism on GPU
- Image size used for benchmarking: 321 x 481 pixels



Routines	Number of Used Threads		
	1	2	4
texton	2.99	1.45	0.75
localcues	18.93	9.44	5.26
intervene	3.01	1.54	1.01
Lanczos	11.10	6.13	4.71
Total	38.32	20.27	13.48

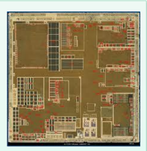
Execution Time (s)

Routines	Number of Threads		
	1	2	4
texton	1	2.06	4.00
localcues	1	2.01	3.60
intervene	1	1.95	2.97
Lanczos	1	1.81	2.36
Total	1	1.89	2.84

Scalability on Number of Threads

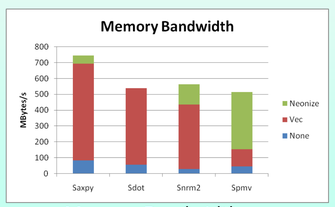
Performance on Snapdragon

- Image size used for benchmarking : 205 x 154 pixels
  - Due to memory limitations
- Setting all compiler flags delivers a 6x speedup on the contour detection computation
  - It is essential to get the compiler flags right on floating point computations



Routine	Flag_None	Flag_All	Ratio
texton	106.45	5.19	20.51
localcues	395.81	52.15	7.59
intervene	13.81	6.55	2.11
Lanczos	830.48	151.75	5.47
Total	1358.89	217.04	6.26

Execution Time (s)
Compiler Flags None vs. All



Memory Bandwidth on Lanczos Subroutines

Power Analysis

- Race to halt principle is well applied on the desktop platforms
- Mobile platforms are about 400x more power efficient than desktop platforms
- Although the execution time on the mobile platform is about 50x (compared to CPU) to 250x (compared to GPU) slower, it still consumes the least amount of energy
- As long as the image uploading and the results downloading energy is smaller than the computation energy, we should keep the computation on the cloud

Routine	Snapdragon	Nehalem (1 thread)	Nehalem (2 threads)	Nehalem (4 threads)	Fermi
Max Power (W)	0.47	183	203	217	264
Min Power (W)	0.17	172	183	186	242
Total Energy (Joule)	68.18	2039.03	1230.79	943.52	221.88
Ratio	1.00	29.91	18.05	13.84	3.25