Moo-Ryong Ra, Anmol Sheth, Lily Mummertx, Padmanabhan Pillai, David Wetherallo, Ramesh Govindan

Odessa
Enabling Interactive Perception Applications on Mobile Devices

Miroslav Cupák

25/03/2013

Some slides borrowed from http://catnip.usc.edu/home/.

Outline

- Problem & Contributions
- Metrics
- Applications
- Sprout
- Application Performance
- Design & Implementation
- Evaluation
- Conclusions

Problem

- resource constrained mobile devices
- offloading and parallelism
- interactive perception applications

- requirements
  - crisp response
  - continuous processing
  - compute-intensive
  - performance depending on data

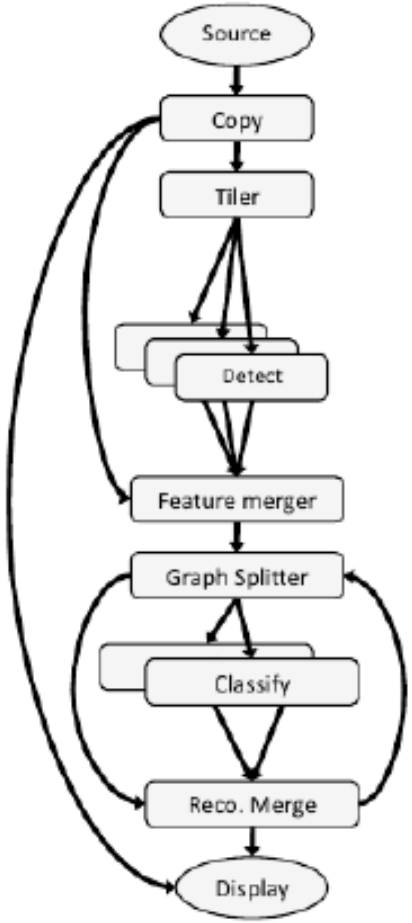
Contributions

- study of offloading & parallelism decisions
- Odessa
  - runtime automatically determining how best to use offloading and parallelism to achieve responsiveness and accuracy
- successful evaluation of the greedy approach

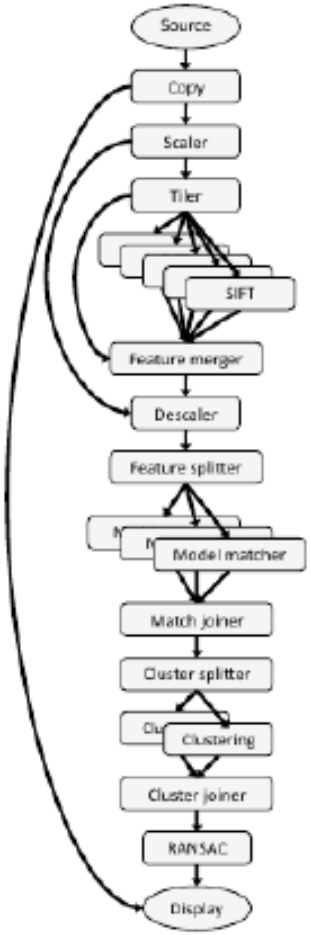
Metrics

- makespan
  - time to execute all stages of computation for a frame
- throughput
  - rate at which frames are processed
- techniques
  - code offloading
  - pipelining
  - data parallelism

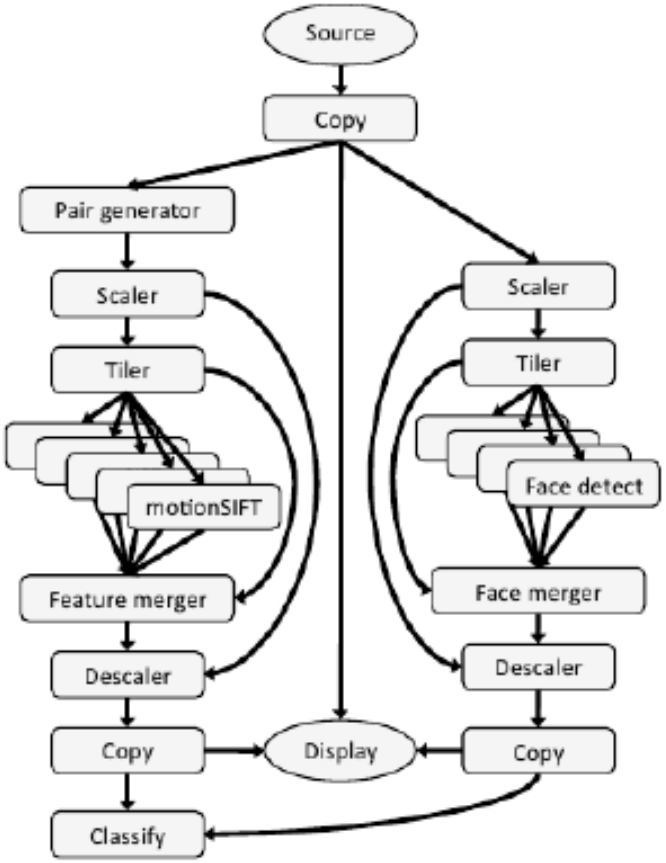
Applications



(a) Face Recognition



(b) Object and Pose Recognition



(c) Gesture Recognition

Sprout

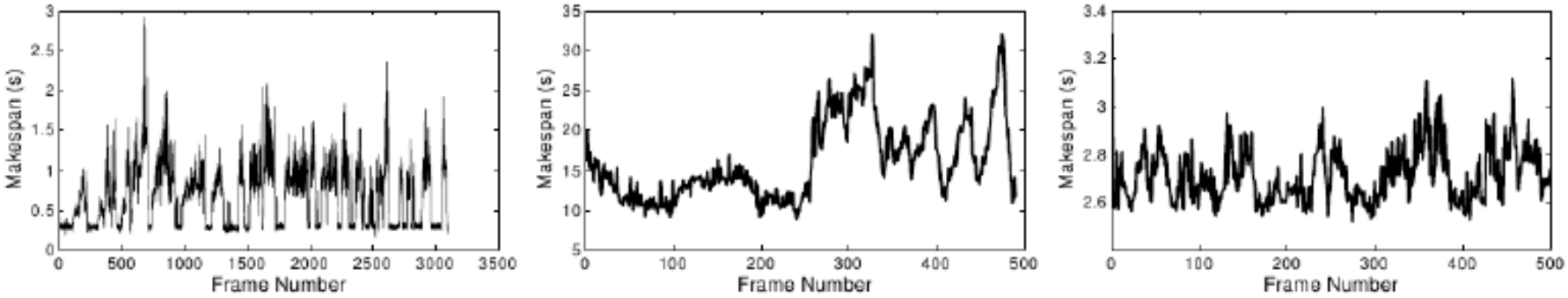
- distributed stream processing system
- uses a data flow model to dynamically distribute processing of an application
- takes care of data transfer, coarse-grained data parallelism, pipeline parallelism

- Odessa
  - what/when/how to offload on top of Sprout

Application Performance

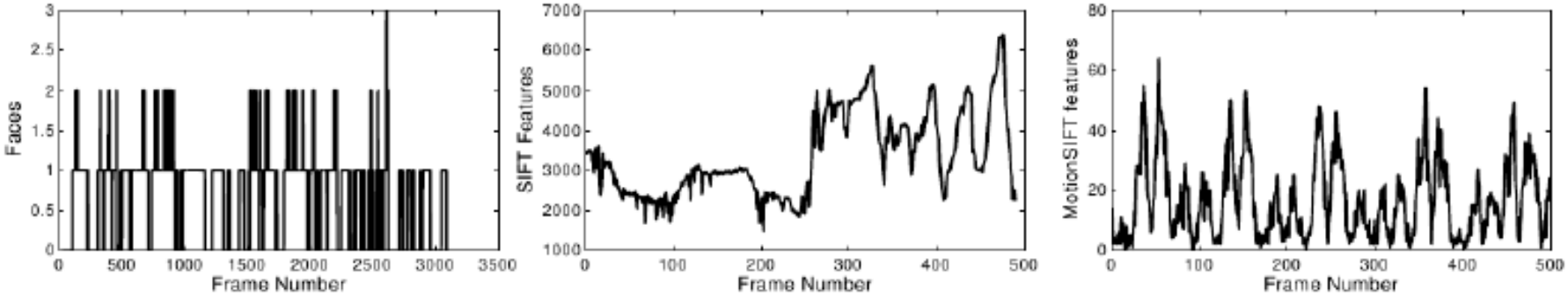
- observations
  - offloading must be made dynamically (input variability, network bandwidth, device parameters)
  - data parallelism cannot be determined apriori
  - static choice of pipeline parallelism is not optimal
- setup
  - netbook (1 core), laptop (2 cores), server (8 cores)
  - 30 fps, 640x480 px video, 3000 (500) frames
  - network bandwidth emulation

Input Data Variability



(a) Face recognition (b) Object and pose recognition (c) Gesture recognition

Figure 3: Variation in the per frame makespan.



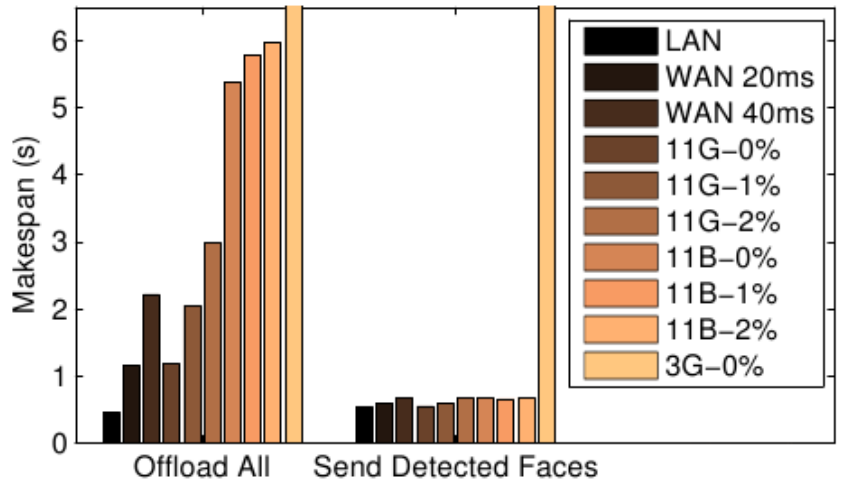
(a) Face recognition (b) Object and pose recognition (c) Gesture recognition

Variability Across Mobile Platforms

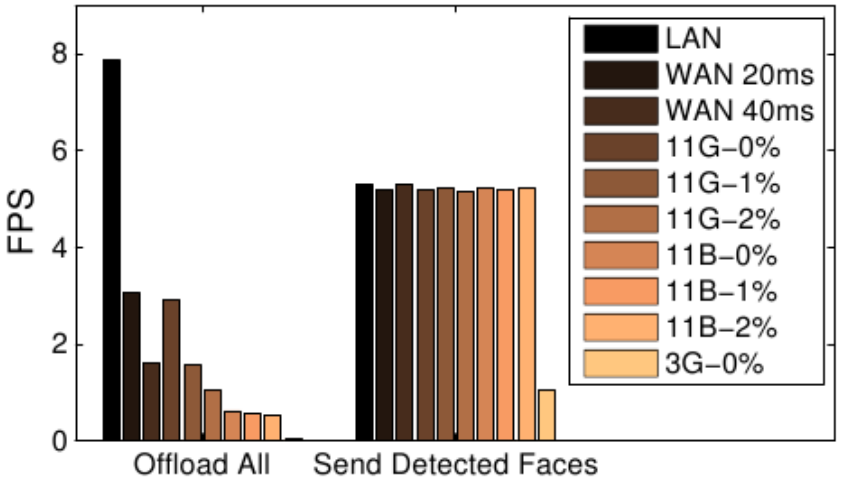
Application	Makespan (s) Laptop	Makespan (s) Netbook	Speedup
Face Recognition	0.078	0.20	2.94
Object and Pose Rec.	1.67	9.17	5.47
Gesture Recognition	0.54	2.34	4.31

Table 3: Median speedup in the overall application performance across the two devices.

Network Performance



(a) Makespan



(b) Frame Rate

Data & Pipeline Parallelism

# of Threads	% Frames with faces	Mean exec. time (ms)
1	61.66	149.0
2	24.87	15.6
3	38.11	18.0

Table 4: The accuracy and mean execution time of face detection with increasing number of worker threads.

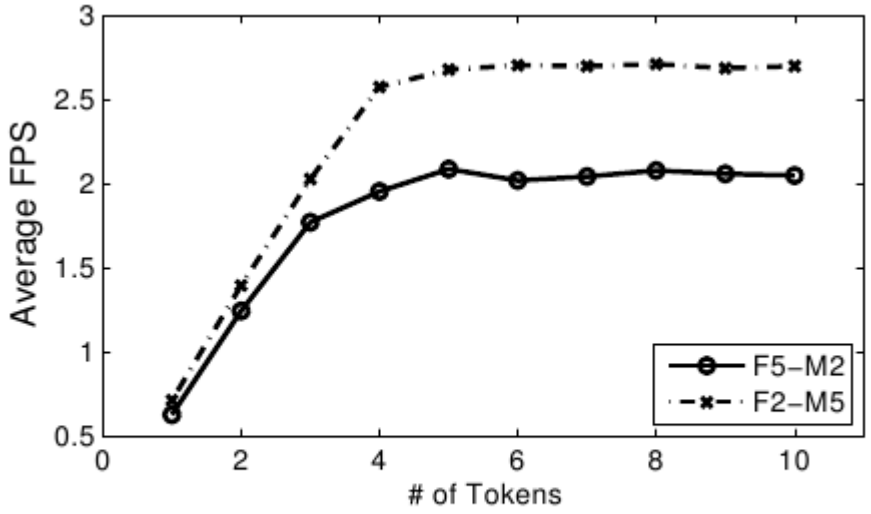


Figure 7: Frame rate with increasing number of tokens.

Design

- goals
  - minimize makespan and maximize throughput
  - quick response to environment changes
  - low computation and communication overhead
- parts
  - application profiler
  - decision engine

Profiler

- lightweight, online
- piggybacking approach
- function
  - maintain application performance profile for the decision engine
- data
  - execution time of each processing stage
  - wait time on connectors
  - transfer time

Decision Engine

- runs periodically
- greedy incremental approach
- estimates impact of offloading and parallelism based on processor frequencies and history of network measurements
- dynamic pipelining based on tokens

Decision Engine

bottleneck := pick the first entry from the priority heap
if bottleneck is a compute stage **then**
  a. estimate the cost of offloading the stage
  b. estimate the cost of spawning more workers
else
  if bottleneck is a network stage **then**
    a. estimate the cost of offloading the source stage
    b. estimate the cost of offloading the destination stage
  end if
end if
take the best choice among a., b. or do-nothing
sleep

Evaluation: Performance & Overhead

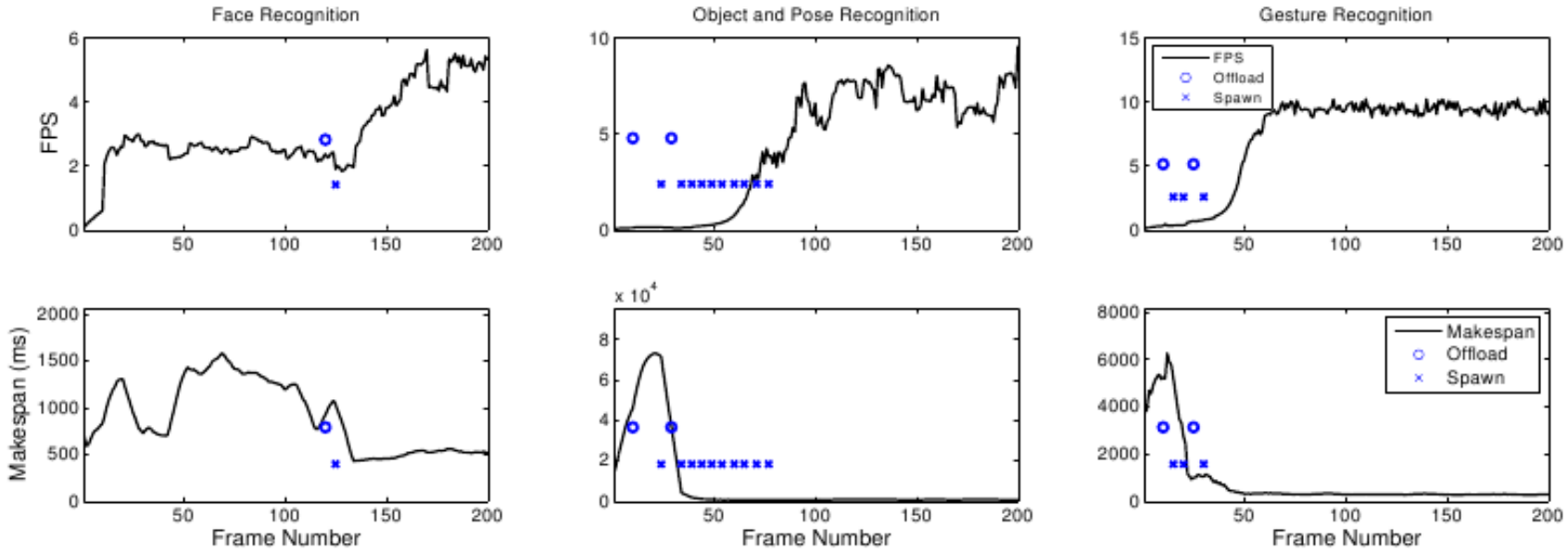
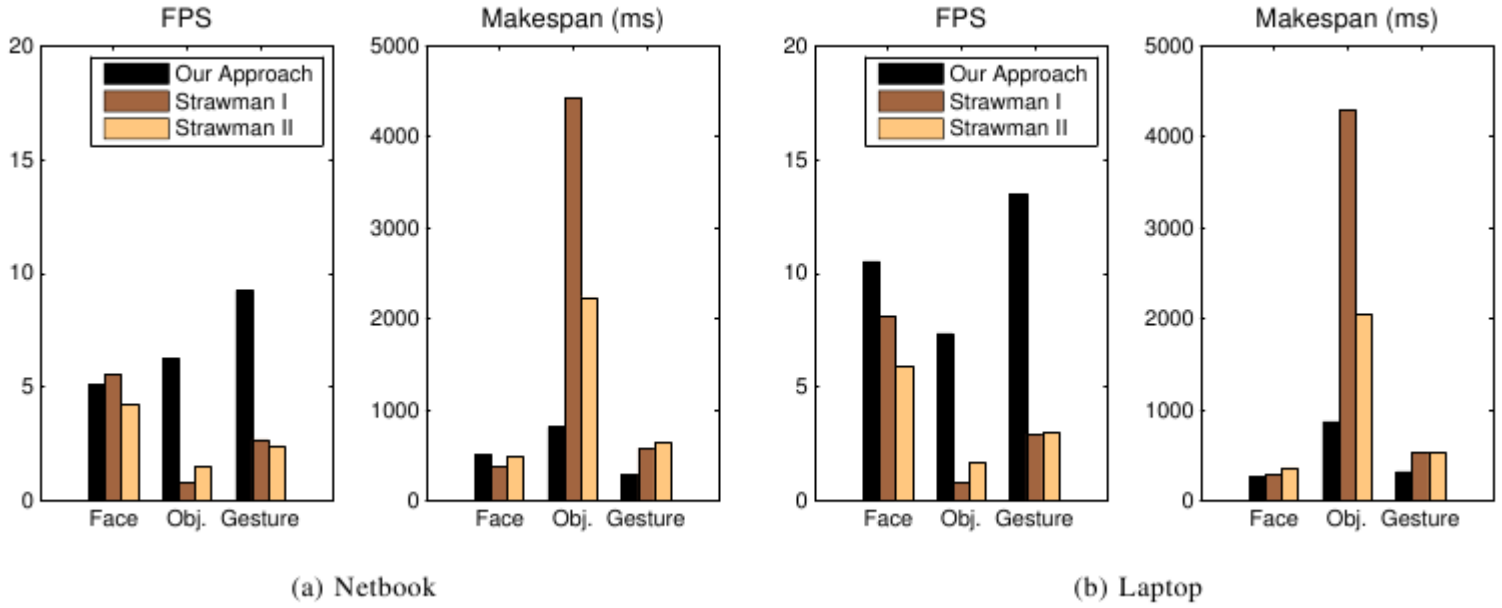


Figure 10: Figure shows the decisions made by *Odessa* across the first 200 frames along with the impact on the makespan and frame rate for the three applications on the netbook.

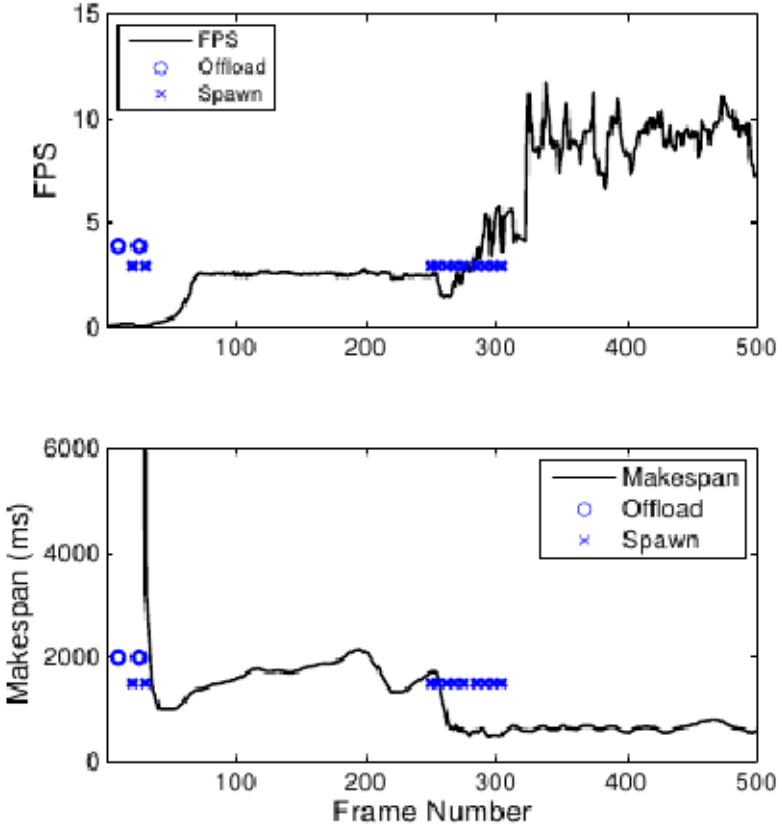
- minimum overhead

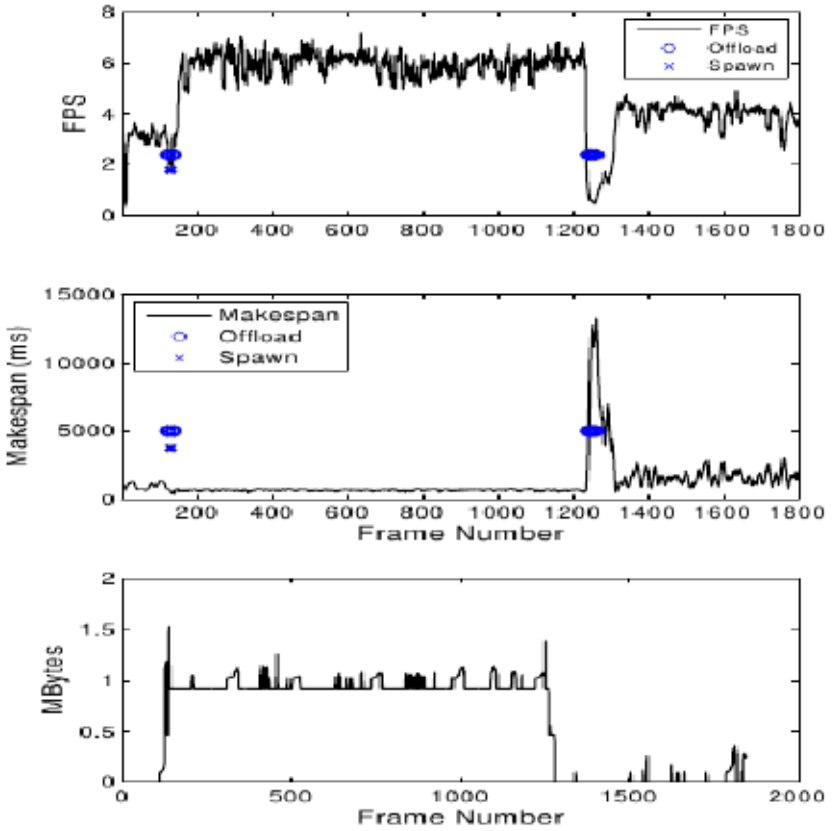
Evaluation: Other Strategies

- 3 strategies : offload all, domain-specific, offline optimizer



Evaluation: Context Adaptation





- taking fidelity into account left for future work

Related Work

- energy saving: MAUI
- graph-based partitioning: Coign
- static partitioning: Wishbone
- a set of pre-specified partitions: CloneCloud, Chroma, Spectra
- parallel processing: MapReduce

Conclusions

- understanding of the factors which contribute to offloading and parallelism decisions
- Odessa
  - runtime automatically determining how best to use offloading and parallelism to achieve responsiveness and accuracy
- extensive evaluation

Thank you! Questions?

Discussion

- Could static analysis be used to improve the approach?
- The paper focuses on computer vision problems. Would this approach work with other problems as well?
- Could this approach be altered to minimize energy consumption instead?
- How could the implementation be improved?