RAVEN: Perception-aware Optimization of Power Consumption for Mobile Games

Chanyou Hwang (KAIST)*, Saumay Pushp (KAIST)*, Changyoung Koh (KAIST), Jungpil Yoon (KAIST), Yunxin Liu (Microsoft Research), Seungpyo Choi (KAIST), Junehwa Song (KAIST)

*Co-primary authors, order chosen alphabetically.

KAIST NC lab. Microsoft Research

Rapid Evolution of Mobile Devices

Larger screens, more powerful processors, and bigger batteries.

2008 **2010** **2014** **2017**

Mobile Games?

Battery Killer: Graphics Processing

- Use more power for better graphics.
- Draw **60 frames** per second **regularly**!

Graphics processing

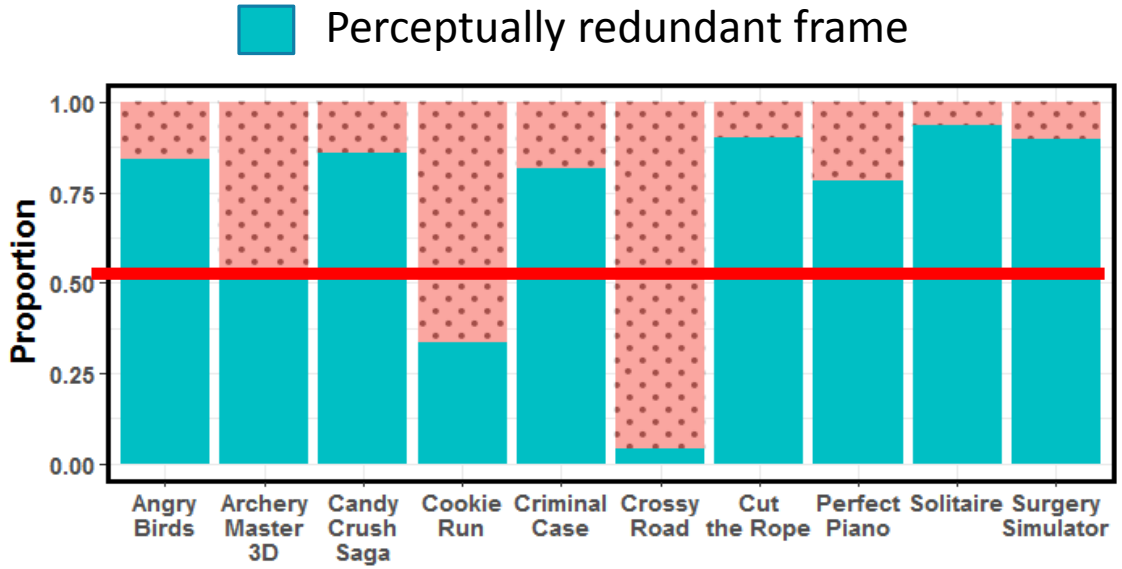
Perceptually Redundant Frames

- Drawing **perceptually redundant** frames

  → Make no change in **human eyes!**

  → **Useless**

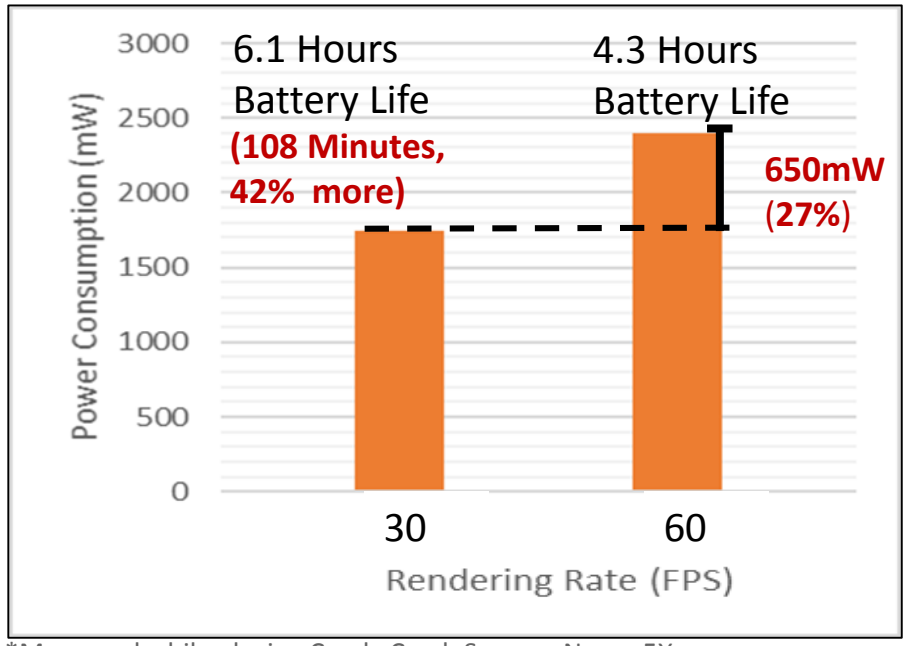
Perceptually Redundant Frames



- Perceptually redundant frames are **common** in mobile games.
  - **SSIM** (Structural Similarity)[1] > 0.975[2] → Perceptually redundant (similar enough)
- More than **50%** of frames are perceptually redundant in 8 of 10 mobile games.

[1] Wang, Zhou, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. "Image quality assessment: from error visibility to structural similarity." IEEE transactions on image processing (2004)
[2] Eduardo Cuervo, Alec Wolman, Landon P. Cox, Kiron Lebeck, Ali Razeen, Stefan Saroiu, and Madanlal Musuvathi. Kahawai: High-Quality Mobile Gaming Using GPU Offload. MobiSys '15

Expected Power Saving



*Measured while playing Candy Crush Saga on Nexus 5X

Reducing the **half** of **frame renderings** → **Extends** battery life **42%** more!

Research Problem

How can we reduce perceptually redundant frames in mobile games?

Previous Approaches

	Approaches
Static	• Limiting Frame Rate[3]
Dynamic	• **Content change rate** based frame rate scaling[4] • **Input-based** frame rate scaling[5]

[3] Samsung Game Tuner
[4] Kim, Dongwon, Nohyun Jung, and Hojung Cha. "Content-centric display energy management for mobile devices". DAC 2014.
[5] Yu Yan, Songtao He, Yunxin Liu, and Longbo Huang. "Optimizing Power Consumption of Mobile Games". HotPower '15.

Previous Approaches

Adjusting frame rate

- **Coarse-grained approach**
- Adjusts the frame rate in a period.

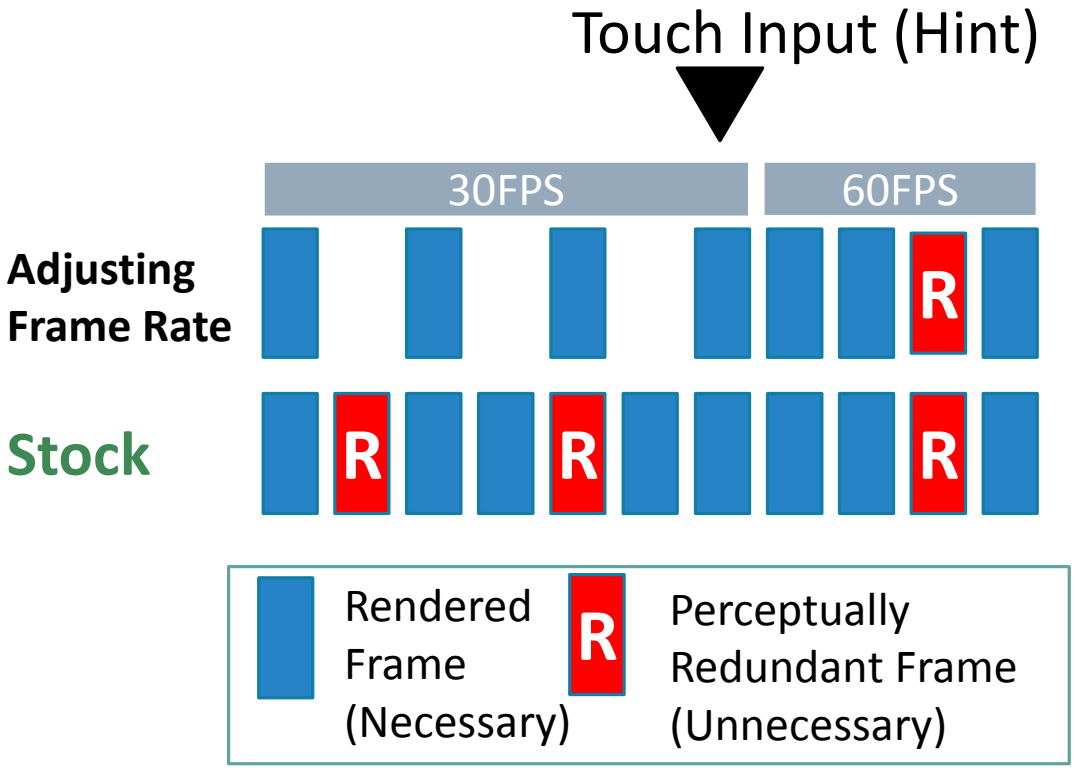


Table of Contents

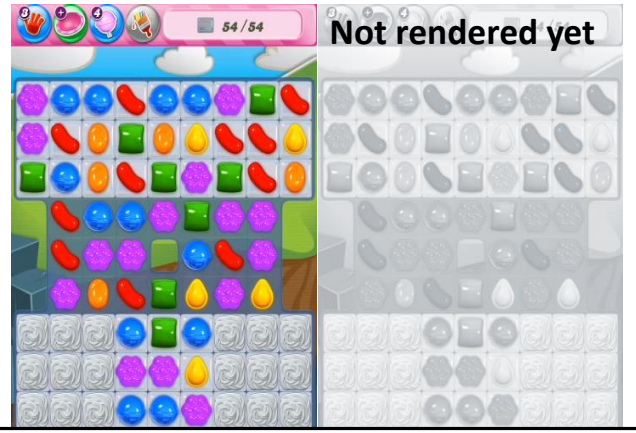
Solution Approach

RAVEN optimizes power consumption by reducing perceptually redundant frames

Perception Aware Scaling of Frame Rendering Rate(**PAS**)

① **Predict** the perceptual similarities with upcoming frames

② **Skip** rendering frames **if perceptually similar enough (= redundant)**



Predict the perceptual similarity

Skip rendering
(if perceptually redundant)

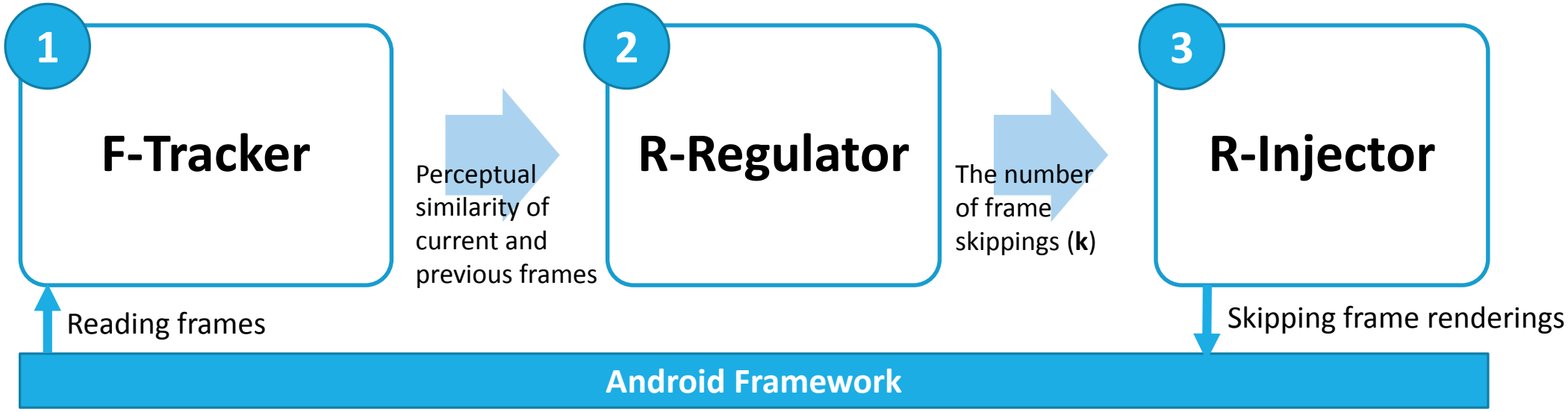
NC lab.


RAVEN-Enabled
Stock (60 FPS)
FPS
60
30
0
레벨 1
점수:
14220
다음
1 나
2 cool!!!
3 Heimar

KAIST

Table of Contents

System Overview



1. **Reads** current frame and **measures** perceptual similarity (to the previous frame)
2. **Predicts** the perceptual similarities to next frames and **decides** the number of frame skippings (using perceptual similarity thresholds)
3. **Skips** rendering next **k** frames (**k** is decided in R-Regulator)

Table of Contents

Technical Challenges

Processing high-resolution game graphics (> 1920x1080)

- in **hard real-time** (< 16.6 ms) with **low energy overhead**

Supporting **commercial devices** and **games**.

Y-Difference-based perceptual similarity prediction

Reading low resolution virtual display

Customizing Android graphics architecture

Y-Difference-based Perceptual Similarity Prediction

NC lab.

Linear regression using Y-Difference

- **Y-Difference** of **recent two frames**
- **+ Moving average** of recent **Y-Difference values**

→ **SSIM** of current and next frames

Estimated Perceptual Similarity of $\boldsymbol{f_N}$(current frame) and $\boldsymbol{f_{N+k}}$ (k^{th} next frame)

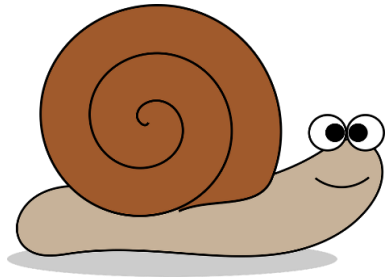
$$\mathbf{EPS}(\boldsymbol{f_N}, \boldsymbol{f_{N+k}}) = \mathbf{1} - \boldsymbol{c_{kl_1}} \times \boldsymbol{D_Y}(\boldsymbol{f_{N-l}}, \boldsymbol{f_N}) - \boldsymbol{c_{kl_2}} \times \boldsymbol{ma_w}$$

- $\boldsymbol{D_Y}(\boldsymbol{f_{N-l}}, \boldsymbol{f_N})$is the Y-Difference of the frames $\boldsymbol{f_{N-l}}\ and\ \boldsymbol{f_N}$.
- $\boldsymbol{ma_w}$ is the moving average of the recent Y-Diff scores over window size **w**

Why Y-Difference?

Why **RAVEN uses Y-Difference** to **predict SSIM?**

- **SSIM is too heavy** and **slow** for using in mobile devices.



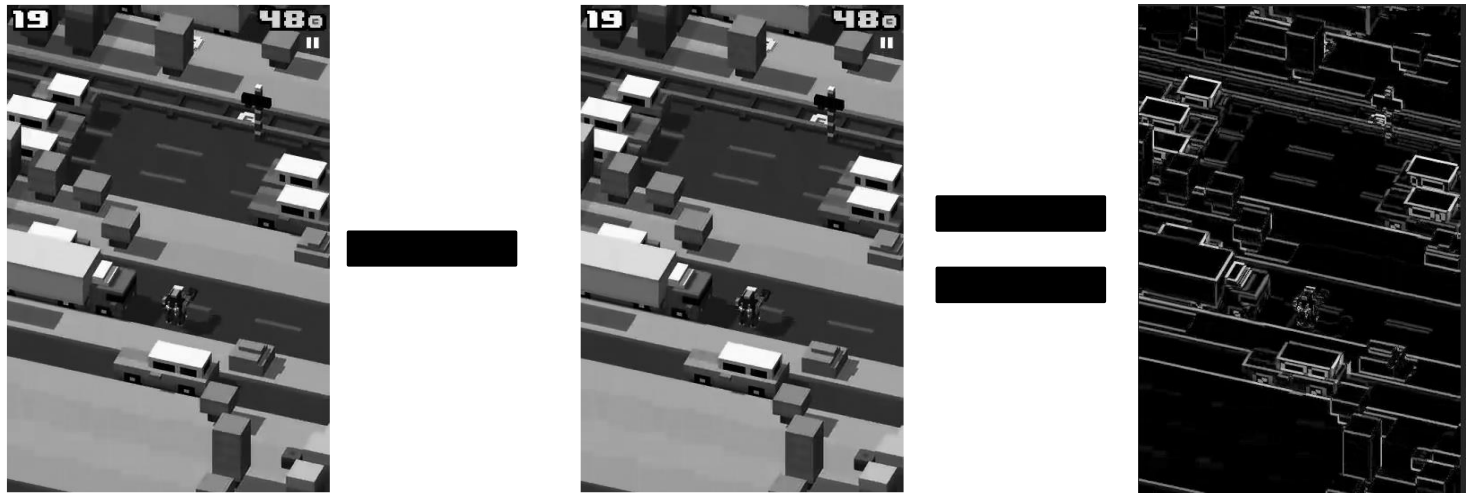
Slow computation
(< 13 FPS, > 80ms per frame)

Heavy energy overhead
(> 13% of total power consumption)

Y-Difference

Y-Difference (**Luminance** difference)

- Sum of the **luminance differences** for each pixel

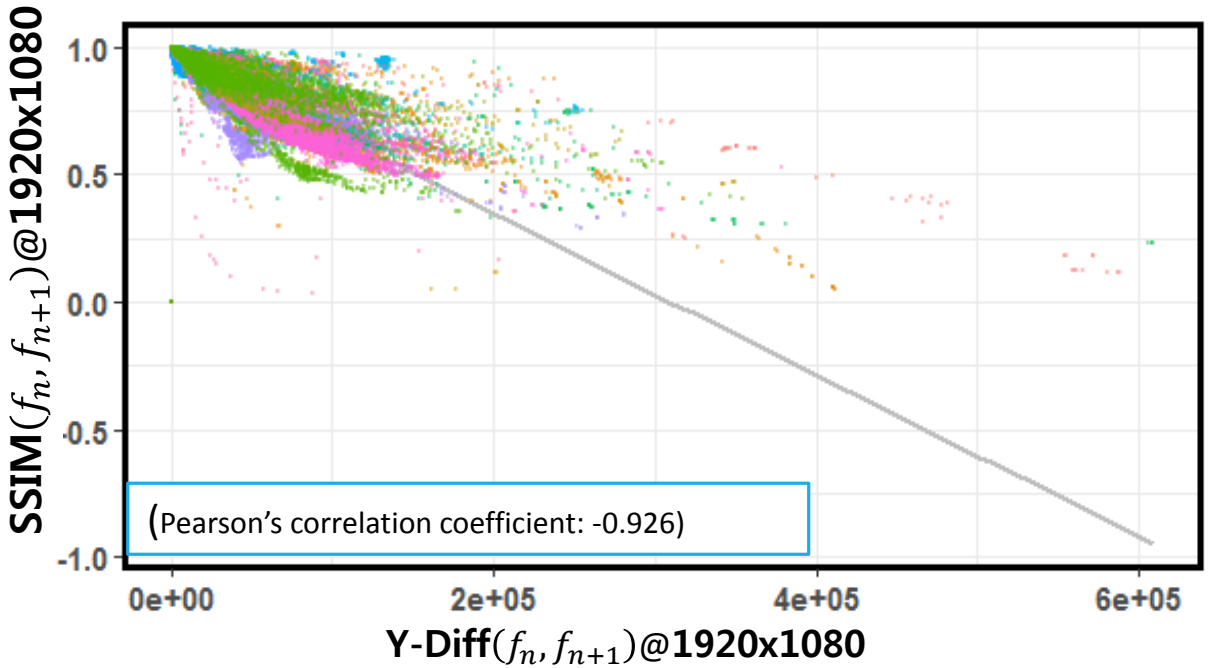


- Human eyes are sensitive to luminance changes in detecting motions.

Y-Difference and SSIM

Y-Difference is a good approximation of **SSIM**

- It shows high correlation with SSIM



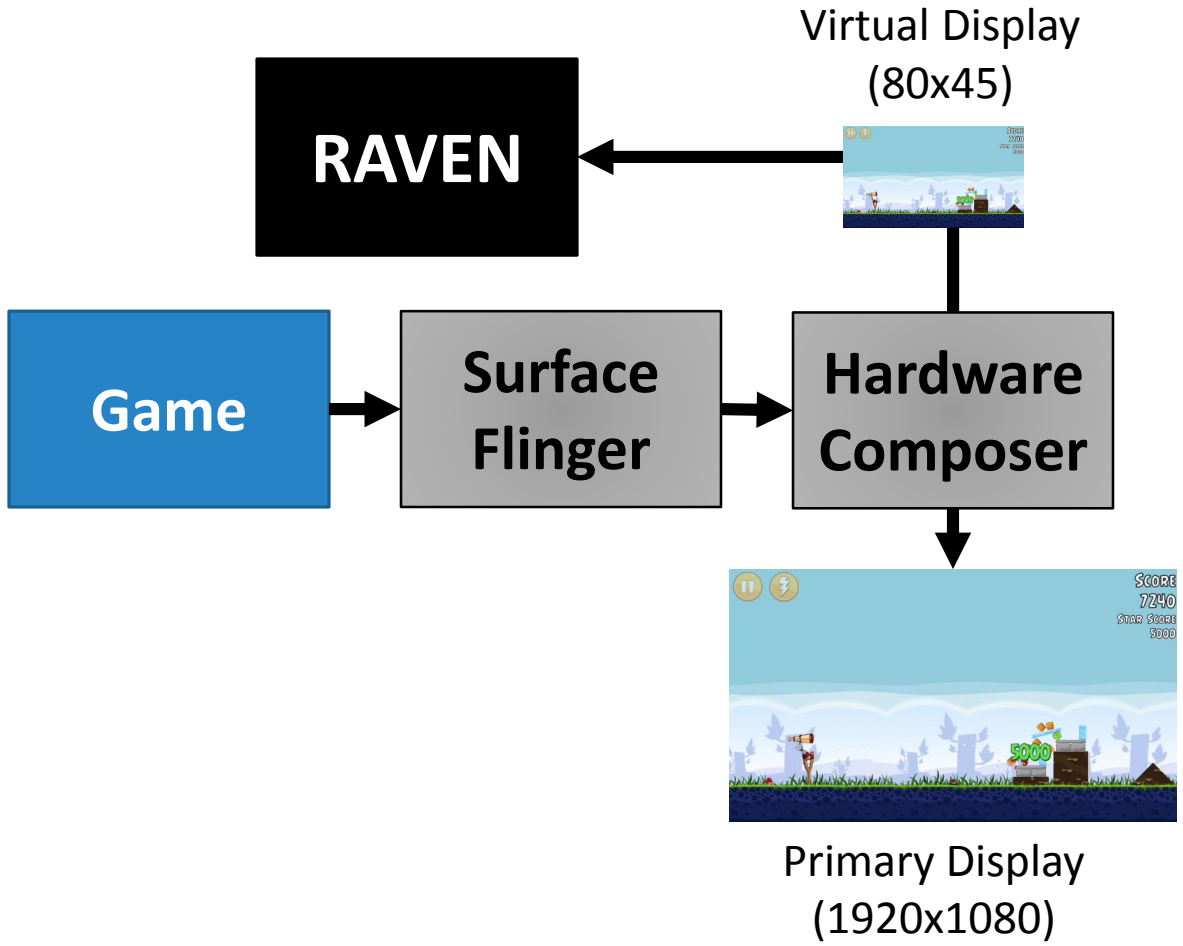
Reading Low Resolution Virtual Display

Virtual Display

- Provided by Android SurfaceFlinger
- Efficient frame cloning aided by

  Hardware Composer

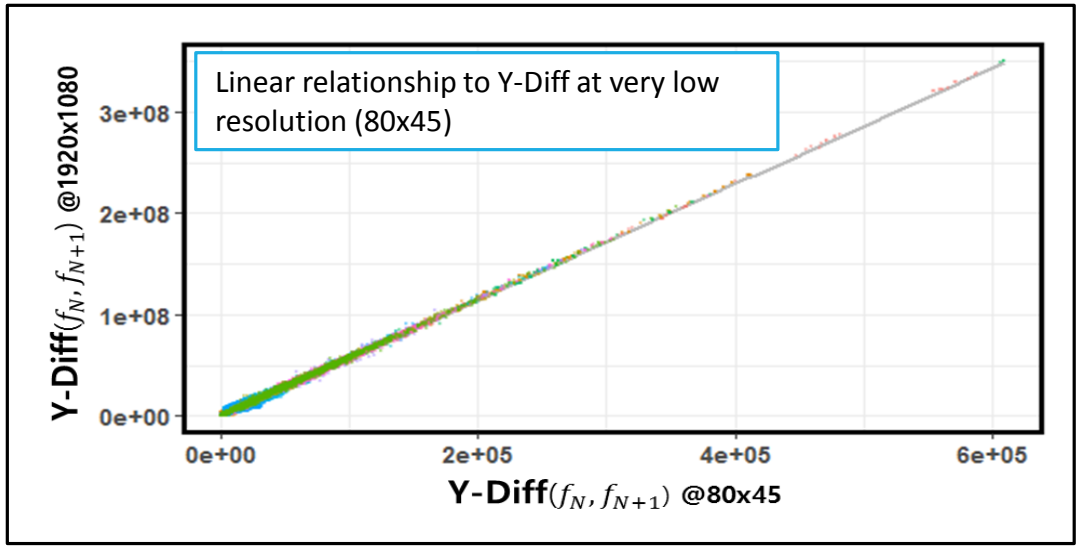
  - Specialized hardware for composing frames from multiple applications.

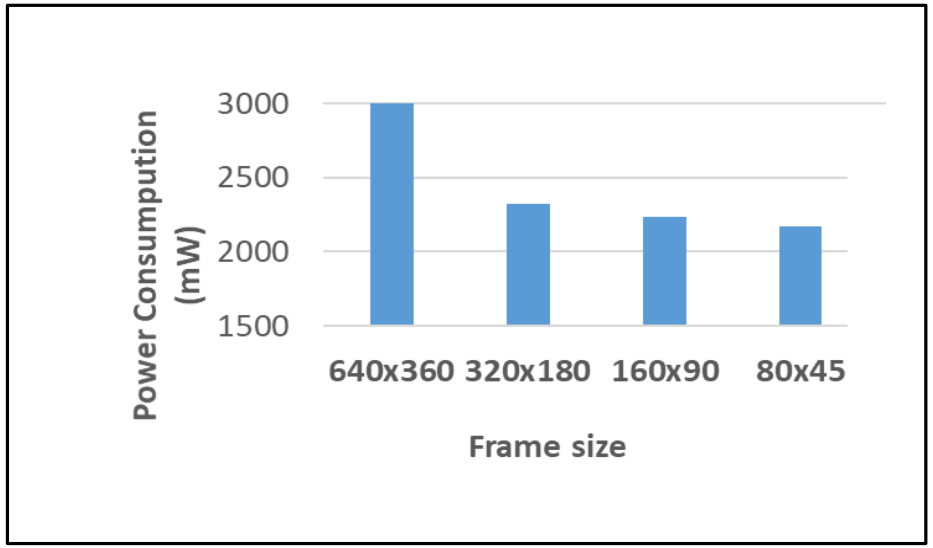


Y-Difference in Low Resolution Frames

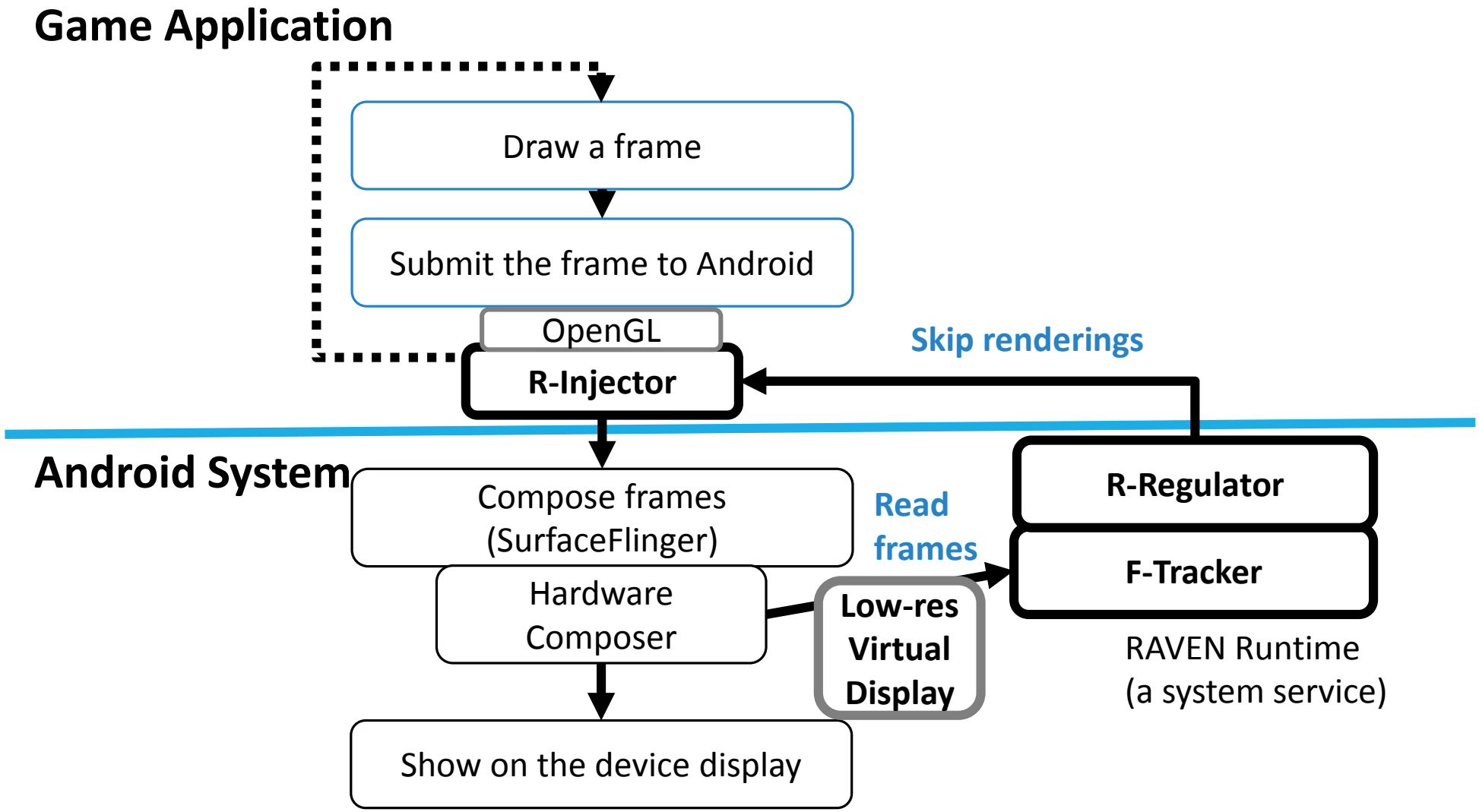
By reading low-resolution frame, **RAVEN** uses less energy!

- Y-Diff is working at a very low resolution (80x45)

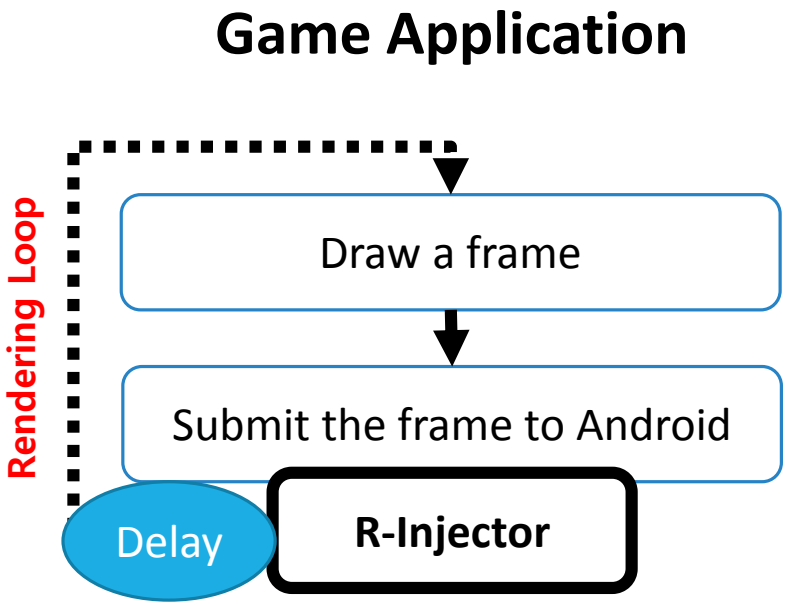




Customizing Android Graphics Architecture



Customizing Android Graphics Architecture



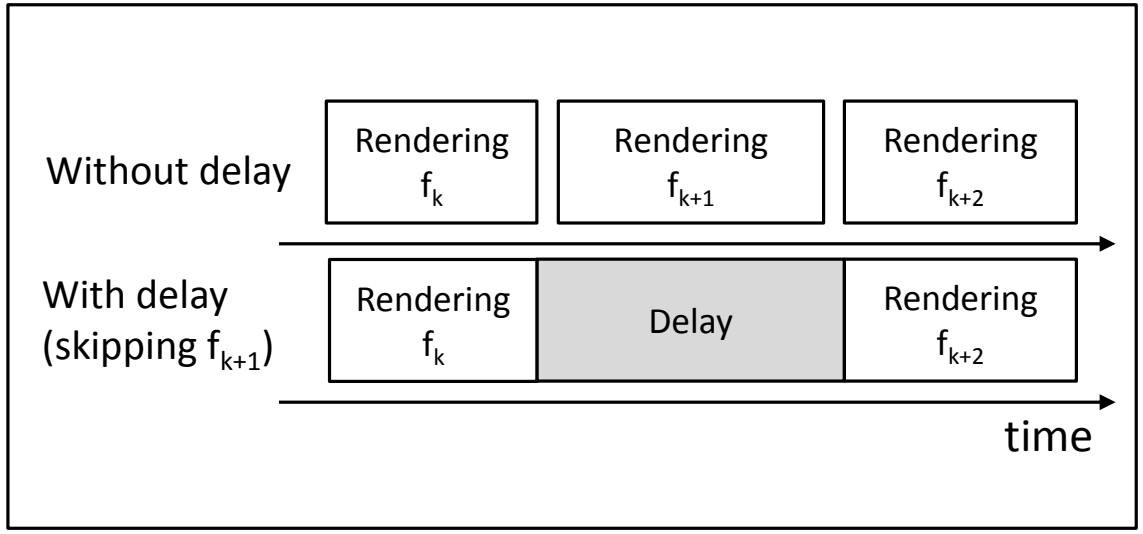


Table of Contents

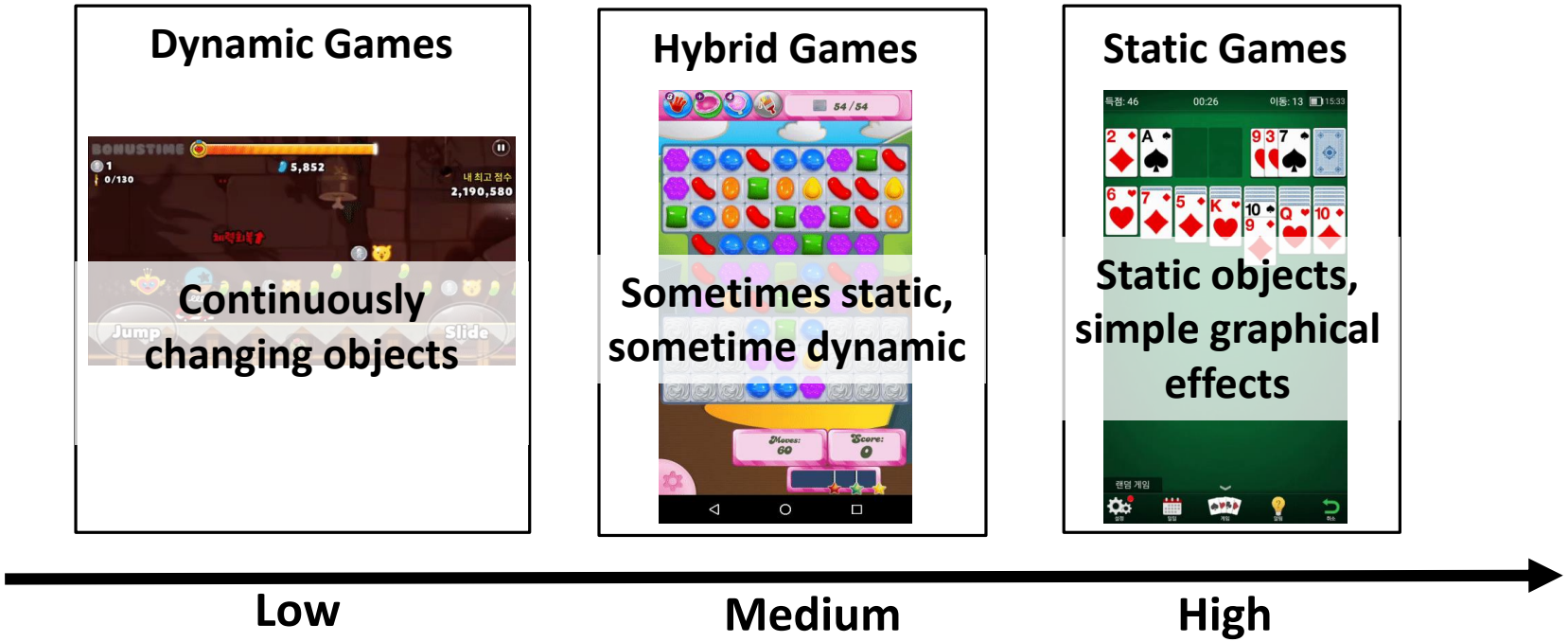
Evaluation

Experiment Methods / Metrics

Method	 Energy Measurement	 Video-based Study	 User Assessment
Metric	Power consumption	Video quality score, # of frame skippings	User assessment score (UX)

Target Workloads

Three categories of workloads



Comparisons

	Better user experience	More power saving
Baseline	**60 FPS** Original, Best user experience	**30 FPS** Conventional approach (Limiting frame rate)
RAVEN-Enabled	**U-PAS** **U**ser-friendly **P**erception **A**ware **S**caling (PAS in the paper)	**E-PAS** **E**nergy-friendly **P**erception **A**ware **S**caling (PAS++ in the paper)

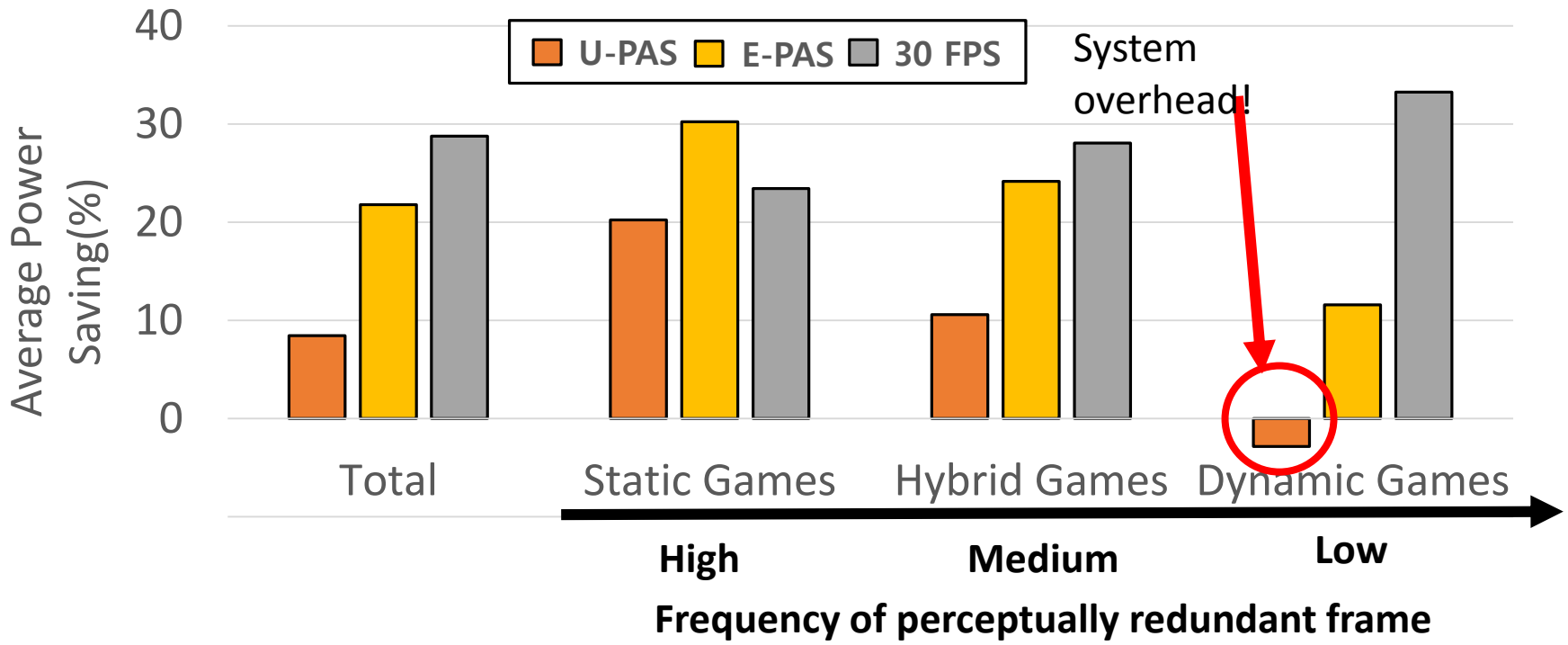
Energy Measurement

How much can **RAVEN** save energy?

- Measured with 8 games
  - 3 minutes with 5 repetitions for each setting

Power Saving

RAVEN saved up-to **30%** of total power consumption.
E-PAS saved **22%** and **U-PAS** saved **8%** in average.

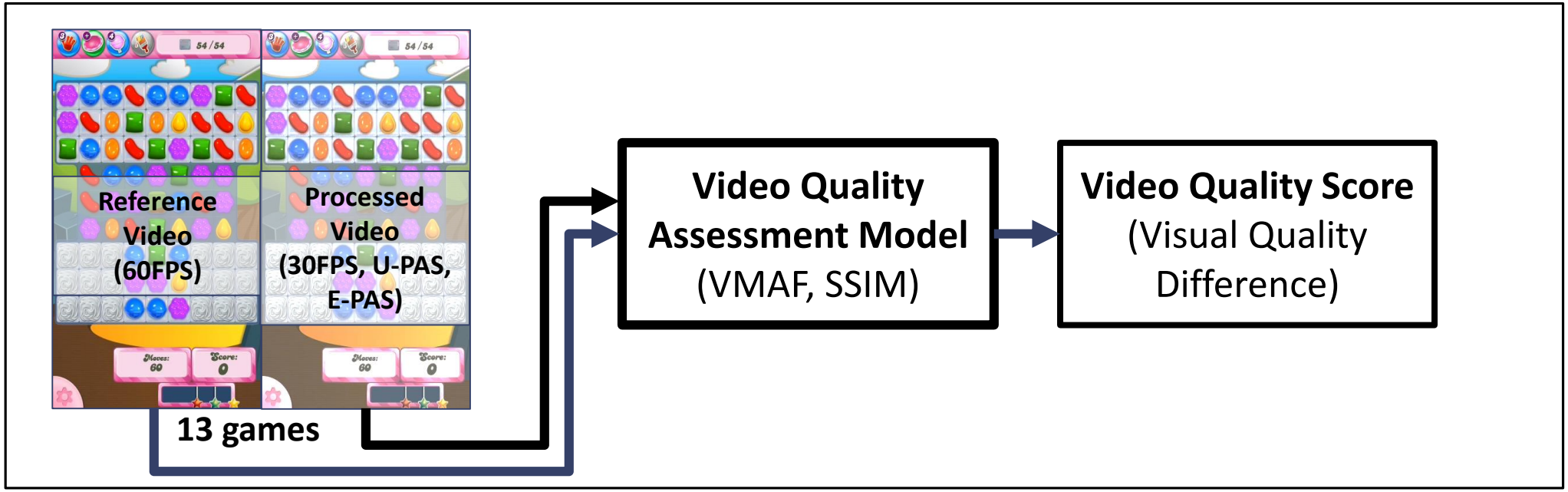


Video-based Study

How RAVEN affects the visual quality of mobile games?

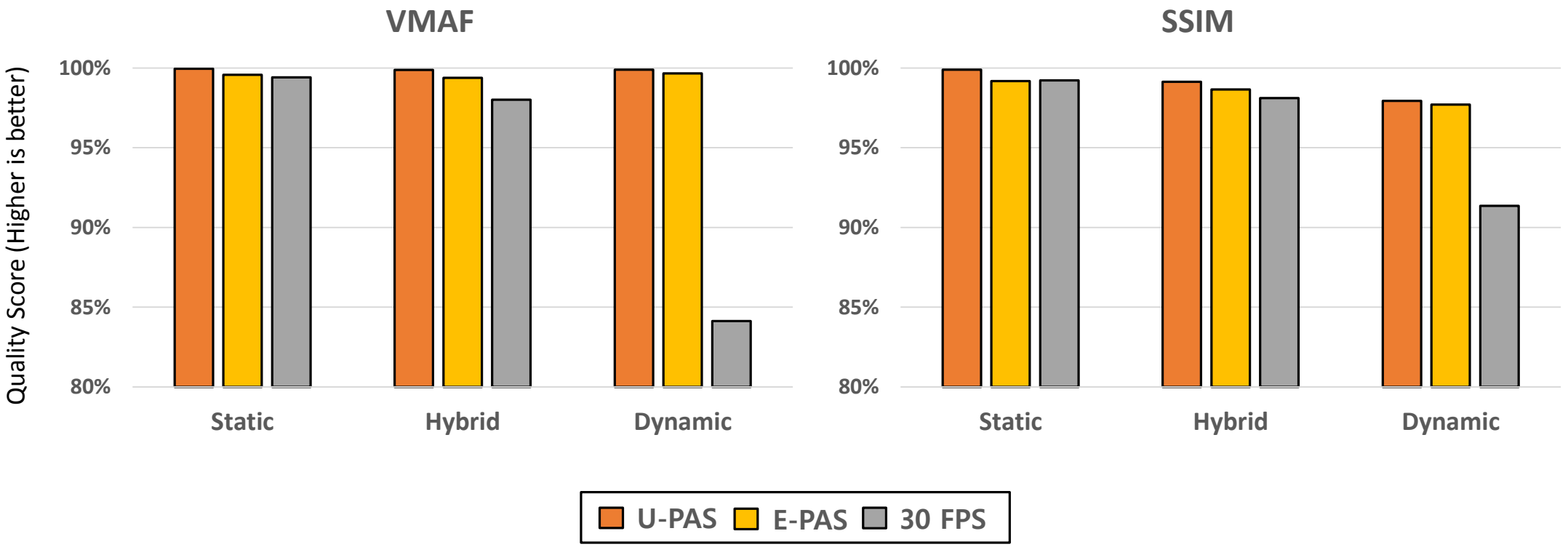
Objective video quality assessment models (VMAF, SSIM)

Result in the **quality difference** between the original video and target (Processed) videos.



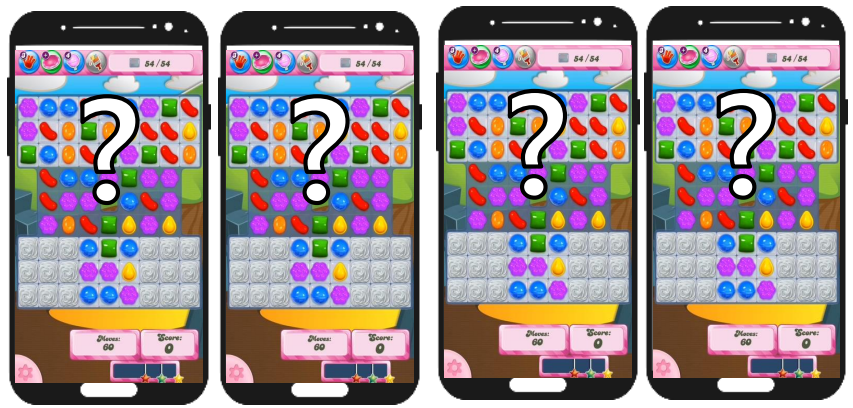
Visual Quality Difference

U-PAS and **E-PAS** provide **almost same visual quality to 60FPS**



User Assessment

How RAVEN affects UX of mobile games?



Blind test

Comparative test

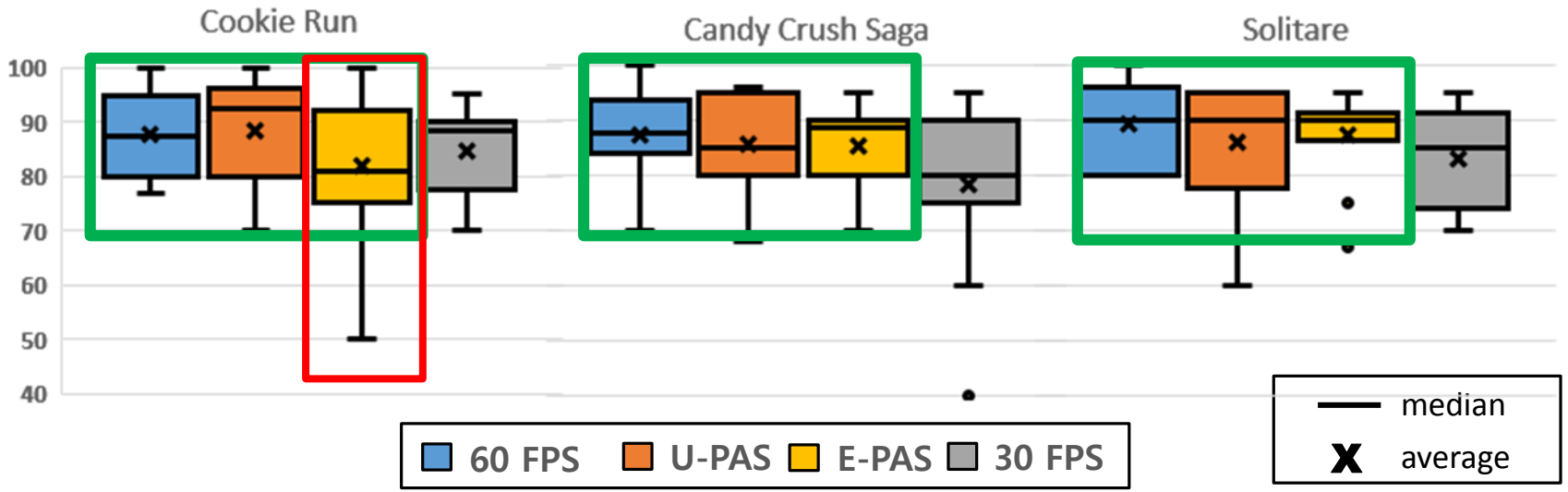
12 Participants played **3** games

- **Cookie Run** (Dynamic), **Candy Crush Saga** (Hybrid), **Solitaire** (Static)

Blind Test

Assessing user experiences without informing the setting of each task

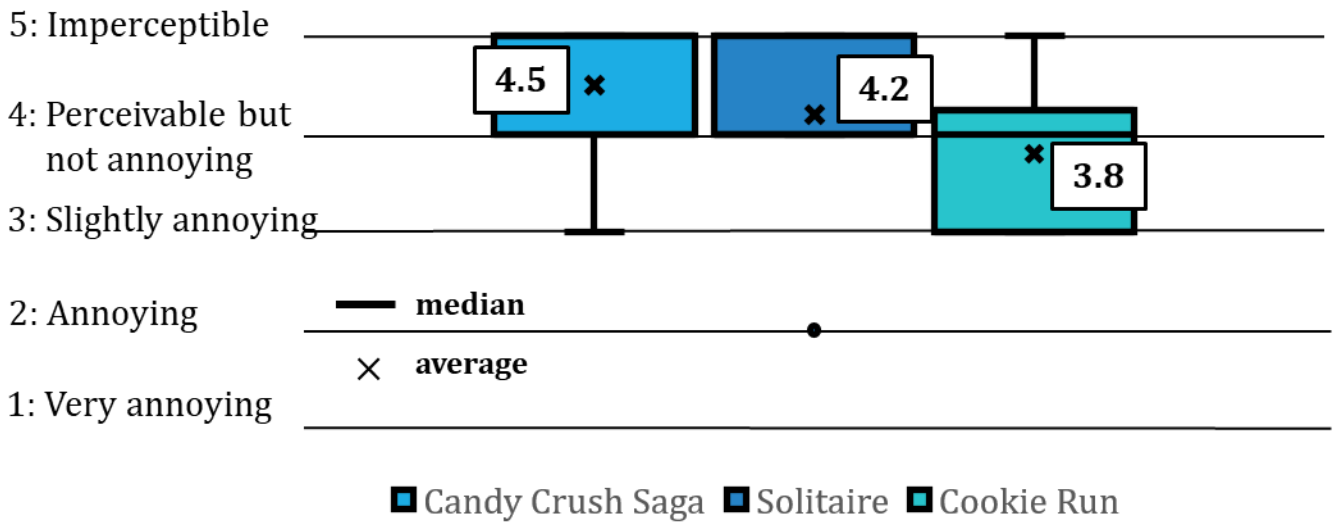
- **U-PAS** and **60 FPS** are hard to discriminate.
- **E-PAS** also shows quality user experience except in Cookie Run



Comparative Test

Direct comparison between **E-PAS** and **60 FPS**

- Most of participants scored either “**imperceptible**” or “**perceptible but not annoying**” **(except Cookie Run)**



Conclusion

RAVEN: Perception-aware Optimization of Power Consumption for Mobile Games

- **Reduces perceptually redundant frame renderings**
  - By using the PAS (Perception-Aware Scaling of frame rendering rate) method
- **Three key ideas** in the design and implementation
  - Y-Difference-based perceptual similarity prediction
  - Reading low resolution virtual display
  - Customizing Android graphics architecture
- **Saves energy** while maintaining **quality user experience**

Thanks!

NC lab.

Questions?