Detecting Display Energy Hotspots in Android Apps

Mian Wan, Yuchen Jin, Ding Li and William G. J. Halfond
Motivation

AVERAGE SCREEN SIZE OF NEW SMARTPHONE MODELS
4906 smartphone models, launched from Jan 2007 to May 2014
@somospostpc
Display Energy Optimization for OLED Screens

Nyx Color Transformation Technique (Li et al. ICSE2014)
Where to Apply Display Optimization Techniques?

• Apply to the whole app
  • Some UIs may already be energy-efficient
  • Don’t want to use automatically transformed colors

• Apply according to developers’ intuition
  • The judgement is subjective and error-prone
Goal of Our Approach

• Our goal – to identify the UIs that is not energy efficient
  • Display Energy Hotspot (DEH): a UI of a mobile app whose energy consumption is higher than an energy-optimized but functionally equivalent one
• Our approach uses color transformation to generate an energy efficient baseline, and estimates how much energy can be possibly saved through power modeling.
Overview of dLens

1. Target App
2. Replay and Capture
3. Establish Optimization Baseline
4. Predict Display Energy
5. Rank UIs

Workload
DEP
UI Rankings
1. Workload Replay and Screenshot Capture

- Workload
- <event, timestamp>
- APK
- Replay and Capture Mechanism
- Screenshots
- <screenshot, timestamp>
2. Establish Optimization Baseline

• To quantify the optimization potential for a UI, we need an optimization baseline

• How to generate it?
  • Give one possible and reasonably optimized version of the UI
  • Use this version of UI as a baseline
2. Establish Optimization Baseline

• Solution: Nyx – a color transformation technique for web applications

• Nyx exploits static analysis technique to generate color transformation scheme (CTS) for web pages
2. Establish Optimization Baseline

• Challenges to adapt Nyx:
  • More colors in a screenshot
  • More complex color relationship
3. Predict Display Energy

Step 1
Screenshots

< screenshot, timestamp >

Step 2
Transformed Screenshots

Prediction Module

Power & Energy of screenshots

DEP
3. Predict Display Energy

• For screenshot $s_i$, we get its energy estimate
  \[ E(s_i, t_i, t_{i+1}) = P(s_i) \times (t_i - t_{i+1}) \]

• As for power, its power is the sum of each pixel’s power:
  \[ P(s_i) = \sum_{k \in |s_i|} C(R_k, G_k, B_k) \]

• At the granularity of a pixel, its power model $C(R_k, G_k, B_k)$ is defined in a Display Energy Profile (DEP)
How to Construct a DEP

\[ C(R, G, B) = rR + gG + bB + c \]
4. Prioritize the User Interfaces

**inputs:** power and energy of original screenshot $s$ and its transformed one $s'$

\[
\Delta P = P_s - P_{s'}, \\
\Delta E = E_s - E_{s'}
\]

\[
\text{IsDEH}(s, p) = \begin{cases} 
\text{true}, & p > 0 \\
\text{false}, & p \leq 0 
\end{cases}, \quad p \in \{\Delta P, \Delta E\}
\]

Sort the screenshots in descending order based on the magnitude of $\Delta P$ and $\Delta E$
Example of the Output of dLens

<table>
<thead>
<tr>
<th>Rank</th>
<th>Screenshot</th>
<th>$\Delta P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image1" alt="Screenshot" /></td>
<td>155.10</td>
</tr>
<tr>
<td>2</td>
<td><img src="image2" alt="Screenshot" /></td>
<td>154.46</td>
</tr>
<tr>
<td>3</td>
<td><img src="image3" alt="Screenshot" /></td>
<td>153.37</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rank</th>
<th>Screenshot</th>
<th>$\Delta E$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image4" alt="Screenshot" /></td>
<td>2339.09</td>
</tr>
<tr>
<td>2</td>
<td><img src="image5" alt="Screenshot" /></td>
<td>2147.31</td>
</tr>
<tr>
<td>3</td>
<td><img src="image6" alt="Screenshot" /></td>
<td>1575.40</td>
</tr>
</tbody>
</table>
Evaluation

• **RQ 1**: How accurate is the dLens analysis?
• **RQ 2**: How generalizable are the dLens results across devices?
• **RQ 3**: How long does it take to perform the dLens analysis?
• **RQ 4**: What is the potential impact of the dLens analysis?
### Subject Applications and Devices

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (MB)</th>
<th>Screenshots</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook</td>
<td>23.7</td>
<td>116</td>
<td>554</td>
</tr>
<tr>
<td>Facebook Messenger</td>
<td>12.9</td>
<td>55</td>
<td>268</td>
</tr>
<tr>
<td>FaceQ</td>
<td>17.9</td>
<td>96</td>
<td>470</td>
</tr>
<tr>
<td>Instagram</td>
<td>9.7</td>
<td>93</td>
<td>429</td>
</tr>
<tr>
<td>Pandora internet radio</td>
<td>8.0</td>
<td>75</td>
<td>278</td>
</tr>
<tr>
<td>Skype</td>
<td>19.9</td>
<td>65</td>
<td>254</td>
</tr>
<tr>
<td>Snapchat</td>
<td>8.8</td>
<td>142</td>
<td>465</td>
</tr>
<tr>
<td>Super-Bright LED Flashlight</td>
<td>5.1</td>
<td>20</td>
<td>51</td>
</tr>
<tr>
<td>Twitter</td>
<td>13.7</td>
<td>101</td>
<td>388</td>
</tr>
<tr>
<td>WhatsApp Messenger</td>
<td>15.3</td>
<td>65</td>
<td>242</td>
</tr>
</tbody>
</table>
Workload Replay and Screen Capture

• We manually generate the workloads that traverse almost all of the functionality of each app
• We used RERAN tool to replay workloads
• We used AShot tool to capture the screenshots
RQ1: Accuracy of Power Model

The average estimation error rate varied from 5% to 8% across these 3 devices.
RQ2: Generalizability

DEH results for one device can typically represent the results for many other similar devices.

The rankings are almost identical ($\bar{R} = 0.9929$)
### RQ3: Analysis Time

<table>
<thead>
<tr>
<th>Name</th>
<th>Time for Color Transformation (s)</th>
<th>Time for Estimation (s)</th>
<th>Overall (s)</th>
<th>Per UI(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook</td>
<td>1,470</td>
<td>7</td>
<td>1,477</td>
<td>12</td>
</tr>
<tr>
<td>Facebook Messenger</td>
<td>997</td>
<td>3</td>
<td>1,001</td>
<td>18</td>
</tr>
<tr>
<td>FaceQ</td>
<td>1,145</td>
<td>5</td>
<td>1,151</td>
<td>12</td>
</tr>
<tr>
<td>Instagram</td>
<td>2,799</td>
<td>6</td>
<td>2,806</td>
<td>30</td>
</tr>
<tr>
<td>Pandora internet radio</td>
<td>1,418</td>
<td>4</td>
<td>1,423</td>
<td>19</td>
</tr>
<tr>
<td>Skype</td>
<td>871</td>
<td>3</td>
<td>875</td>
<td>13</td>
</tr>
<tr>
<td>Snapchat</td>
<td>1,444</td>
<td>8</td>
<td>1,453</td>
<td>10</td>
</tr>
<tr>
<td>Super-Bright LED Flashlight</td>
<td>863</td>
<td>1</td>
<td>865</td>
<td>43</td>
</tr>
<tr>
<td>Twitter</td>
<td>1,316</td>
<td>6</td>
<td>1,323</td>
<td>13</td>
</tr>
<tr>
<td>WhatsApp</td>
<td>897</td>
<td>3</td>
<td>901</td>
<td>13</td>
</tr>
</tbody>
</table>
RQ4: Potential Impact

• We searched for DEHs in a large set of Android apps from Google Play
• After automatically taking screenshots, we manually checked all screenshots and removed invalid screenshots
  • In total, we collected screenshots of 962 apps
• We used *dLens* to analyze these apps’ initial pages
RQ4 Results

398 apps contain DEHs

Some app consumes 101% more energy
Top 10 Offenders of Energy Efficiency
Summary

• Present a new technique for detecting DEHs in mobile apps
• Combine color transformation and power modeling
• Our evaluation shows our tool is accurate, within 8% of ground truth
• The results of our tool can be generalized across devices
• The DEH problem is common: we detected DEHs in 398 (41%) apps of 962 Android apps
Thank you!

Detecting Display Energy Hotspots in Android Apps

Mian Wan, Yuchen Jin, Ding Li and William G. J. Halfond
Color Patterns

Color Ratio in Apps without DEHs

- Black: 93%
- Dark Gray: 4%
- Gray: 1%
- White: 1%
- Dim Gray: 1%

Color Ratio in Apps with DEHs

- White: [CELLRANGE]
- Dim Gray: [CELLRANGE]
- White Smoke: [CELLRANGE]
- Others: [CELLRANGE]
Difference in Building DEP

• Dong et al. didn’t isolate the display power, thus in their model $c > 0$, which is the constant power for displaying black.

• In order to isolate the display power, we calculate the power difference with and without connecting cable linking screen and CPU, thus in our model $c = 0$. 
Our limitations:

• The screenshot contains other elements (e.g. Android status bar) not belonging to an app’s UI

• Color Transformation is also applied to dynamic elements (e.g. images)
• 60% choose transformed app for general usage
• 97% choose transformed app for battery critical
Invert Colors