Perceived-Color Approximation Transforms for Programs that Draw

Phillip Stanley-Marbell
University of Cambridge

Martin Rinard
Massachusetts Institute of Technology

Human color perception acuity is not uniform across colors. This makes it possible to transform drawing programs to generate outputs whose colors are perceptually equivalent but numerically distinct. One benefit of such transformations is lower display power dissipation on organic light-emitting diode (OLED) displays. We introduce Ishihara, a language for 2D drawing that lets programs specify perceptual-color equivalence classes to use in drawing operations enabling compile-time and runtime transformations that trade perceived color accuracy for lower OLED display power dissipation.

Drawing languages and APIs are pervasive in modern software systems[1,2] and play an increasingly important role in modern programming languages.[3] The images that programs in these drawing languages and APIs generate are almost always intended for human consumption. The implementations of drawing languages should therefore take into account human visual perception, in particular the fact that colors that differ numerically may appear acceptably equivalent to many human observers.

In mobile platforms, applications such as web browsers, messaging apps, and productivity software that use 2D drawing APIs accounted for approximately 60 percent of mobile application use in 2013, rising to more than 70 percent in 2017.[4] As a result, in common use, mobile platforms spend most of their time displaying the result of 2D drawing operations.

ISHIHARA: 2D DRAWING WITH APPROXIMATE COLOR

This article presents Ishihara, a language for 2D graphics that implements the core operations in common 2D drawing APIs and languages and that lets programs specify perceptual equivalence classes of colors for use in drawing. Ishihara lets programs specify these equivalence classes statically at compile time or dynamically at runtime. This lets programmers tailor the perceptual

fidelity of colors in drawing programs to specific applications or even to dynamically changing application contexts. Ishihara uses this information on perceived color equivalence to transform colors in drawing programs, either at compile time or runtime, for lower power dissipation of drawn images when displayed on organic light-emitting diode (OLED) displays.

Because perceived colors compose in non-trivial ways (such as when mixed), Ishihara provides language-level operators that free programmers from having to manually manage the intersection of perceived-color equivalence classes.

Ishihara's Empirical Color-Matching Dataset

Ishihara's perceived-color semantics builds on a large dataset of empirical color-matching data from Specimen,[5] a popular mobile game in the iOS App Store with more than 220,000 unique users from a wide demographic. Specimen players repeatedly perform color identification tasks to maximize a color identification score. One of the goals of Specimen is to explore human color perception; to this end, Specimen anonymously collects color evaluation data from its players. Working with this data (which consists of more than 28 million color-matching evaluations), we develop a well-calibrated empirical model that characterizes equivalence classes of perceived colors based on the fraction of Specimen players that evaluate different colors as identical. The Specimen dataset, which we introduce and analyze for the first time in this work, is many orders of magnitude more extensive than any study of perceptual color acuity in the color-perception literature.[6]

Applications of Approximate Perceived Color

Because they enable thinner and lighter displays, wider color gamut, and more, OLED displays are growing in popularity over LCDs. Today, they are used in platforms ranging from the Apple Watch, Google Nexus 6P, and Google Pixel 2, to the MacBook Pro touch bar and the iPhone X.

The physics of generating light in semiconducting materials such as the organic semiconductors used in OLED displays results in fundamentally different energy costs for light of different wavelengths (and thus of different colors). Unlike legacy display technologies such as LCDs, OLED display power dissipation depends on the colors in the content they display.[7,8] OLED displays can therefore have significantly different power dissipation when displaying two images that many users may perceive to be acceptably equivalent in terms of color content.

Several research efforts[7-10] have explored ways to exploit OLED display properties for improved energy efficiency. Ishihara is the first to explore using a large crowdsourced dataset to empirically estimate perceptual equivalence classes and to apply these equivalence classes in transformations for a drawing language.

Contributions

This article makes four contributions to the state of the art:

1. We introduce Ishihara, a language for 2D drawing with constructs that allow programs to define colors with a specification of acceptable color approximation. Programmers can either specify the degree of approximation statically or delegate choosing the degree of approximation to the compiler at compile time or the operating system at runtime.
2. We introduce a program transformation that is designed to maintain perceived-color equivalence while reducing the OLED display power dissipation caused by programs. Unlike the Crayon system[8] by which it is inspired, Ishihara's transformations operate in the hue space independent of brightness and are built on a large multi-thousand-user dataset of real-world mobile data rather than a limited purpose-designed user study.
3. We analyze the Specimen dataset to derive color-equivalence classes. We use these color-equivalence classes in implementing the perceived-color semantics of imperative drawing programs.

4. We apply the program transformation based on the Specimen dataset and evaluate its effect in reducing the power dissipation caused by drawing programs on OLED displays.

DRAWING WITH APPROXIMATE COLORS

We use uppercase letters in a blackboard typeface (for example, $\mathbb{C}$) to represent sets and use lowercase letters to represent metavariables in the abstract syntax (for example, n). In both cases, we use the subscript π to denote quantities related to perception (for example, p_π) and use the subscript λ to denote program quantities that do not take into consideration perception (for example, i_λ).

We refer to the numeric value of a color (as represented in a machine's memory) as its nominal color. Each nominal color may be perceptually equivalent to a set of other nominal colors for some individual observer or group of observers. We refer to the sets of nominal colors perceived to be identical as perceptual color-equivalence classes or simply as perceptual colors.

To allow us to analyze color perception separately from brightness and saturation, we often need to refer to the hue associated with a given nominal or perceptual color. One practical means for quantifying the hue of a given color is to convert the color to the hue-saturation-brightness (HSB) color space. A 24-bit HSB color space uses 8 bits of resolution for each component of the color space. Each of the 2^8 hues can be rendered at 2^{16} different levels of saturation and brightness, to form one of 2^{24} colors. Intuitively, colors are less saturated when the hues from which they are formed have more white mixed in.

Defining Perceptual Colors

A fraction of at most p_π of observers perceive colors in an equivalence class $\mathbb{C}\,(p_\pi)$ to be identical when viewed on a computer display, even though the colors in the set $\mathbb{C}\,(p_\pi)$ may have different numeric values. In practice, smaller values of p_π lead to larger equivalence classes, providing more opportunity for perceptual-color-preserving program transformations.

To specify perceived colors in programs, we explicitly associate a nominal color, h, with an acceptability population fraction, p_π, to obtain a program-specified perceived color (h, p_π). In a program-specified perceived color, p_π serves as a color accuracy specification, analogous to precision and accuracy specifications in programming languages and systems for approximate computing.[11-14]

Ishihara Abstract Syntax

Figure 1 presents the abstract syntax for Ishihara. The syntactic categories and their corresponding meta-variables are: numerals (n), expressions (e), lists (l), and statements (s). These meta-variables may have subscripts (such as s_0 or e_c).

Variables in Ishihara represent images. Temporary image variables and the global image (denoted in the code as "image") represent the drawing state of an Ishihara program at any point in its execution. Statements in Ishihara assign image expressions to variables. The declare operator (:=) allocates a new variable name and memory location, associating the memory location with the variable name and assigning the value of the right side of the statement to the memory location. The assign operator (=) assigns values to memory locations associated to previously declared variables. Statements may generate new images from polygon expressions (polygon) and compositing operations (composite).

Colors used in Ishihara image expressions are perceived colors, denoted by the keyword "pcolor" followed by an acceptability population fraction and an RGB triplet. An acceptability population fraction is either a constant literal, the identifier "compilePercept" (which is resolved at compile time), or the identifier "dynamicPercept" (which is resolved at runtime). Ishihara thereby allows programmers and dynamic code generators to delay the decision on the degree of

color approximation until compile time using compilePercept, or to delay it to runtime using dynamicPercept. In the latter case, a compiler can make no assumptions about the value of p_π and must generate code to perform the dynamic API call to retrieve a system value for p_π at runtime. Example uses include drawing programs whose color output fidelity adapts to an operating system setting such as a low or critical battery level.

n, v	$\in$	$\mathbb{N} \subset \{0\} \bigcup \mathbb{Z}^+$	(Finite-width Integers)
s_n	::=	q **=** e_i | q **:=** e_i |	
		s_0 **;** s_1 | **skip** .	(Statements)
q	::=	**image** | *identifier* .	(l-values)
e_x	::=	**(**n**,**n**)** .	(Point expressions)
l_c	::=	**{**e_x{**,**e_x**}}** .	(Polygon corners)
e_p	::=	**pen** e_h n .	(Pen expressions)
e_s	::=	**polygon** l_c e_p .	(Polygon expressions)
color	::=	**pcolor @** e_a .	(Perceived color)
e_a	::=	n |	
		compilePercept |	
		dynamicPercept .	(Acceptability Expressions)
e_h	::=	*color* n_r n_g n_b | e_m .	(Color expressions)
e_m	::=	**mix** e_h n_1 e_h n_2 **@** e_a .	(Color mixing)
e_c	::=	**composite** e_i **,** e_i .	(Compositing expression)
e_i	::=	**image** | **color2image** e_h |	
		e_c | e_s .	(Image expressions)

Figure 1. Abstract syntax for Ishihara.

COLOR APPROXIMATION USING EQUIVALENCE CLASSES FROM THE SPECIMEN DATASET

Figure 2 plots the power dissipation of an OLED display as a function of hue, for colors at constant saturation and brightness. The data in the plot are based on validated hardware measurements of OLED display power dissipation as a function of color, presented in the research literature.[8]

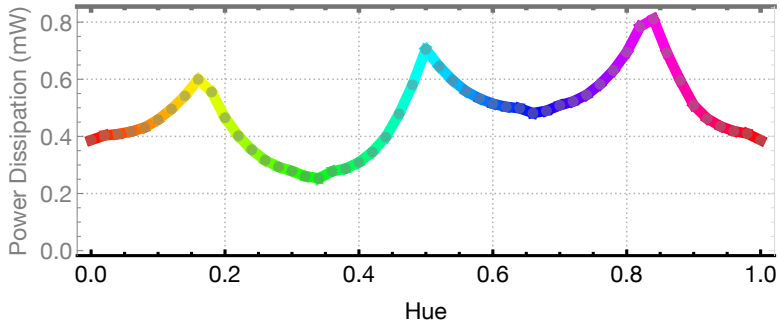


Figure 2. The power dissipation of OLED displays varies with hue (which wraps on [0, 1]). The data shown are from hardware measurements on a representative OLED display. Points in the plot represent power dissipation at a given hue. The line joining the points is colored to show the hue variation.

The data in Figure 2 show that OLED power dissipation is not monotonic in hue (at constant saturation and brightness). As a result, small changes in hue can lead to large changes in display

power dissipation. At the same time, these changes in hue might not be perceptible to many observers. We quantify perceptible hue change later in this article using the Specimen dataset.

Example

Figure 3 shows two Ishihara drawing programs and their outputs. When viewed on both LCDs and OLED displays, the two images look similar to many human observers (more than 10 percent of color comparisons in the Specimen dataset confused these two hues). Displaying Figure 3(b) on a representative OLED display requires 12.9 percent more power than displaying Figure 3(d). Because many mobile devices dissipate 30 to 50 percent of their power in their displays,[8] a 12.9-percent change in power dissipation could equate to almost 6-percent longer battery life.

```
green   := color2image pcolor @10 34 255 16;
shape0  := polygon{(0,0), (50,0),
           (50,50), (0,50), (0,0)} pen green 1;
image0  := composite shape0, green;
image    = composite image, image0;
```

(b)

(a) This program generates the image in (b).

```
green   := color2image pcolor @10 34 255 91;
shape0  := polygon{(0,0), (50,0),
           (50,50), (0,50), (0,0)} pen green 1;
image0  := composite shape0, green;
image    = composite image, image0;
```

(d)

(c) This program generates the image in (d).

Figure 3. Two Ishihara programs. Displaying the output image (b) dissipates 12.9 percent more power than displaying the image (d) on one representative OLED display. When viewed on standard LCDs and OLED displays, the two images look similar.

Brightness-Independent Approximation in the Hue Space

We develop an algorithm to minimize display power dissipation modulo perceived-color equivalence in Ishihara programs. The algorithm takes as input perceptual colors and a display power model, such as the model in Figure 2. Hues in the power model of Figure 2 are represented with values between 0 and 1. The hue space wraps around, with the same hue represented by 0 and 1. Ishihara's color transformation algorithm exploits the form of the power dissipation in hue space as follows.

We divide the hue space to match the three convex regions of Figure 2. Because the hue space corresponds to different ratios of the three primaries (red, green, and blue), and because of the observed relationship between power dissipation and intensity for each of these primaries in OLED displays, we expect to always be able to perform such a partitioning. For each region of the hue space, we identify the hue, h^*, of minimum power dissipation. Let $h \in [0, 1]$ be a hue value, and let α_n, β_n, and γ_n be constants for $n \in \{1, 2, 3\}$. Then, we fit the hue power model to three convex functions:

$$P = \alpha_n h^2 + \beta_n h + \gamma_n \ , \ n \in \{1,2,3\}. \tag{1}$$

To minimize the power P as a function of hue h, we take the derivative of P per region, equate to zero, and solve for h^*:

$$h^* = -\frac{\beta_n}{2\alpha_n}. \tag{2}$$

The algorithm for minimizing power dissipation modulo perceived-color equivalence selects the hue from the equivalence class that is closest to h^*.

Algorithm 1 shows the procedure for minimizing display power dissipation modulo perceived-color equivalence, for colors in an Ishihara program. The algorithm first uses the perceptual equivalence classes for the specified acceptability population fraction p_π to obtain a sub-range of

the hue space. For a sub-range that lies within a single region, the algorithm selects the hue in that region that is closest to the region's h^*. Finally, the algorithm converts the input color to the HSB space, replaces the input hue with the hue selected in the previous step while keeping saturation and brightness the same, and converts the color back to the RGB color space. For example, for the RGB value (0.43, 1.0, 0.0) and acceptability population fraction $p_\pi = 0.10$, the equivalence class is over the hue range [0.1667, 0.5]. From Equation 2, the minimum h^* in that hue range is 0.3277, and the optimum power-minimizing color for the Ishihara expression "pcolor @10 0.43 1.0 0.0" is the RGB value (0.1, 1.0, 0.0).

Data: Ishihara color expression (p_π, i_r, i_g, i_b).
Result: A color (o_r, o_g, o_b) in the RGB color space such that (o_r, o_g, o_b) is in the visual equivalence class of (i_r, i_g, i_b) modulo p_π and is closest entry to h^*.

/* Obtain the equivalence class set for the input color. */
1 $\mathbb{E} \leftarrow \textbf{equivClass}(p_\pi, i_r, i_g, i_b)$

/* Determine the regions of the hue space that $\mathbb{E}$ maps to. The function **hueRegions** returns the set of contiguous hue regions that do not straddle convex regions of Figure 2, of the equivalence class to which it is applied. The result could be multiple disjoint regions $[r_{\text{hmin}}, r_{\text{hmax}}]$. */
2 $\{[r_{\text{hmin}}, r_{\text{hmax}}]\} \leftarrow \textbf{hueRegions}(\mathbb{E})$

/* For each $[r_{\text{hmin}}, r_{\text{hmax}}]$ in the set above, determine which of the convex regions of Figure 2 it corresponds to. Pick the value in the range that is closest to h^*. Then, use the power model of Equation 1 to select the hue o_h from these single hues selected from all the hue ranges $\{[r_{\text{hmin}}, r_{\text{hmax}}]\}$ that has the lowest power dissipation. */
3 $o_h \leftarrow \arg\min_{\{[r_{\text{hmin}}, r_{\text{hmax}}]\}}(P(\{[r_{\text{hmin}}, r_{\text{hmax}}]\}))$

/* Convert (i_r, i_g, i_b) to HSB color space and replace its hue with that computed above. */
4 $(i_h, i_s, i_b) \leftarrow \textbf{rgb2hsb}(i_r, i_g, i_b)$
5 $(o_r, o_g, o_b) \leftarrow \textbf{hsb2rgb}(o_h, i_s, i_b)$

6 **return** (o_r, o_g, o_b)

Algorithm 1. Color replacement modulo visual equivalence.

EQUIVALENCE CLASSES OF PERCEIVED COLOR

We build equivalence classes of perceived colors using crowdsourced color-matching data.

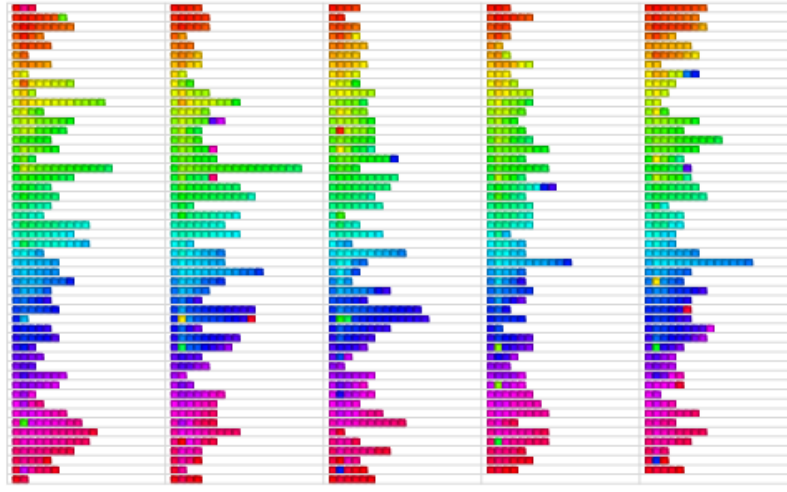
Methodology

We construct the perceptual equivalence classes using data collected from a popular mobile game, Specimen, from the iOS App Store. Specimen players repeatedly perform color-matching evaluations, which the game records anonymously. Players of the game select the colored object, or specimen, in the central circular region of the screen that they consider to be the closest color match to the square background behind the circular region. For each color selection, the game records the selected color. When an incorrect specimen is selected, the game also records both the selected color and the correct color. All of these recorded color selections from each player are gathered continuously and anonymously using a standard API, Flurry,[15] for anonymous in-

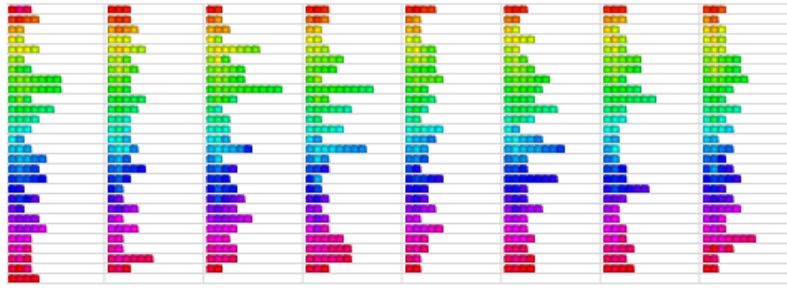
app analytics and stored on a central server. As of November 2016, this data collection system had recorded more than 220,000 users with more than 28 million specimen selection events.

Results

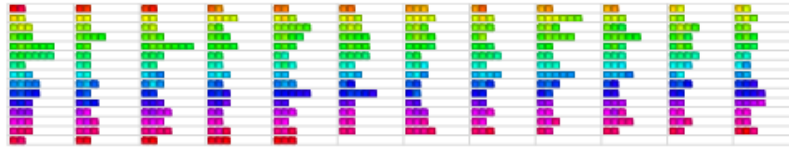
Figure 4 shows the equivalence classes of perceived hues for a given fraction, p_π, of the players that confuse a given hue. Figure 4(a) shows the equivalence classes for $p_\pi = 0.01$. The results in Figure 4(a) show that 1 percent of the color evaluations in the dataset confuses hues over a fairly large equivalence class—up to 16 hues in a hue discretization space of 256 hues (up to 6.25 percent).



(b) $p_\pi = 0.01$.



(b) $p_\pi = 0.02$.



(c) $p_\pi = 0.03$.

(d) $p_\pi = 0.05$.

Figure 4. Perceived equivalence classes of hues. All hues shown in each group are different. A fraction p_π of evaluations confused the first hue in each group with the remaining hues in the group.

We construct a table of the equivalence classes of hues from these results. The function "equivClass()" in Algorithm 1 uses this table. Each entry in the table is a pair (p_π, $\mathbb{S}_h$) consisting of a fraction p_π and a set $\mathbb{S}_h$ of hues such that fraction p_π of color evaluations confuse hues in the set $\mathbb{S}_h$.

Discussion

The data show that hues close to green are the most consistently difficult to discern and therefore have the largest equivalence class of perceived colors. Smaller values of p_π lead to larger equivalence classes (see Figure 4(a) and Figure 4(b)).

We evaluate the range of power dissipation that colors in each equivalence class cause on OLED displays, based on the hardware-measurement-based model presented earlier in the article. The results show that colors indistinguishable by 1 percent of the population ($p_\pi = 0.01$, Figure 4(a)) differ in the power dissipation they cause on OLED displays, by as much as 51.1 percent. Similarly, colors indistinguishable by 5 percent of the population ($p_\pi = 0.05$, Figure 4(d)) differ in the OLED display power dissipation they cause, by up to 23.2 percent.

These results show that it is worthwhile to transform color-related operations in Ishihara drawing programs. The results show that such program transformations can significantly improve how energy-efficient drawing programs use OLED displays while at the same time being acceptably equivalent for some fraction of observers.

EVALUATION

We evaluate Ishihara by implementing 15 benchmarks with non-zero acceptability population fractions. We transformed the programs based on the techniques described earlier in this article and executed them to obtain output images that are equivalent modulo the specified acceptability population fraction. We then evaluated the display power savings for the output of the transformed Ishihara programs compared to the untransformed programs.

Methodology

We created 11 Ishihara programs that draw images of national, regional, or transnational flags. We created four Ishihara programs that load bitmaps of natural scenes. For the bitmaps, we obtained five standard images used in the image processing literature.[16] We converted the Ishihara programs to Crayon and used Crayon's toolchain to compile, optimize, and execute the drawing programs. We obtained the image output and estimated the power dissipation that these outputs cause for OLED displays. We implemented the Ishihara transform algorithm (Algorithm 1), constructing tables for the function equivClass() from the Specimen dataset. We expect perceived-color equivalence transforms to be tuned in practice to individuals or sections of the population with similar color acuity. For the purpose of illustration, we applied the color transform algorithm with $p_\pi = 0.01$.

Results

The second image in each of the image pairs of Figure 5 shows the results of applying the Ishihara transform algorithm for $p_\pi = 0.01$. The power minimization modulo perceived-color equivalence transform reduces OLED display power dissipation caused by the images. The results show up to 15-percent improvement in OLED display power dissipation.

The results show that Ishihara's acceptability population fractions are effective in evaluating perceived-color equivalence. As the results in Figure 5 show, perceived-color equivalence classes of images allow Algorithm 1 to reduce the power that OLED displays dissipate when they display the output of drawing programs.

Figure 5. Outputs of Ishihara programs before transforms (left image in pair) and after perceived-color program transforms with p_{π} = 0.01 (right image in pair). The captions within the figure show the difference in OLED display power dissipation between the two images in each pair. The pairs look similar to many observers when viewed on a standard LCD or OLED display.

REFERENCES

1. *PDF Reference Version 1.6 (5th Edition)*, Adobe Press, 2004.
2. "Skia Graphics Engine," Android, October 2015; https://skia.org/.
3. M.B. McLaughlin, H. Sutter, and J. Zink, *A Proposal to Add 2D Graphics Rendering and Display to C++*, Technical Report N3888, ISO/IEC JTC1/SC22/WG21 Programming Language C++/SG13 Graphics, January 2014.
4. S. Khalaf, "U.S. Consumers Time-Spent on Mobile Crosses 5 Hours a Day," March 2017; http://flurrymobile.tumblr.com/post/157921590345/us-consumers-time-spent-on-mobile-crosses-5.
5. "Specimen: A Game About Color," PepRally, 2015; https://itunes.apple.com/us/app/specimen-a-game-about-color/id999930535?mt=8.
6. G. Wyszecki and W. Stiles, *Color Science: Concepts and Methods, Quantitative Data and Formulae, 2nd Edition*, Wiley-VCH, 2000.
7. M. Dong, Y.-S.K. Choi, and L. Zhong, "Power-saving color transformation of mobile graphical user interfaces on oled-based displays," *Proceedings of the 2009 ACM/IEEE International Symposium on Low Power Electronics and Design* (ISLPED '09), 2009, pp. 339–342.
8. P. Stanley-Marbel, V. Estellers, and M. Rinard, "Crayon: Saving power through shape and color approximation on next-generation displays," *Proceedings of the Eleventh European Conference on Computer Systems* (EuroSys '16), 2016, p. 11:1.
9. D. Li, A.H. Tran, and W.G.J. Halfond, "Making web applications more energy efficient for oled smartphones," *Proceedings of the 36th International Conference on Software Engineering* (ICSE 2014), 2014, pp. 527–538.

10. M. Linares-Vásquez et al., “Optimizing energy consumption of GUIs in Android apps: A multi-objective approach,” *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering* (ESEC/FSE 2015), 2015, pp. 143–154.
11. M. Carbin, S. Misailovic, and M. Rinard, “Verifying quantitative reliability for programs that execute on unreliable hardware,” *Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications* (OOPSLA ’13), 2013, pp. 33–52.
12. H. Hoffmann et al., “Dynamic knobs for responsive power-aware computing,” *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems* (ASPLOS XVI), 2011, pp. 199–212.
13. A. Sampson et al., “Enerj: Approximate data types for safe and general low-power computation,” *Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation* (PLDI ’11), 2011, pp. 164–174.
14. P. Stanley-Marbell and D. Marculescu, “A programming model and language implementation for concurrent failure-prone hardware,” *2nd Workshop on Programming Models for Ubiquitous Parallelism* (PMUP'06), 2006, pp. 44–49.
15. “Flurry Analytics,” Yahoo, 2016; https://developer.yahoo.com/analytics/.
16. *The USC-SIPI Image Database*, The University of Southern California Signal and Image Processing Institute, 2016; http://sipi.usc.edu/database/.

ABOUT THE AUTHORS

Phillip Stanley-Marbell is an assistant professor in the Department of Engineering at the University of Cambridge. His research focus is on exploiting an understanding of properties of the physical world and the physiology of human perception to make computing systems more efficient. Prior to joining Cambridge, he was a researcher at MIT, from 2014 to 2017. He received his Ph.D. from CMU in 2007, was a postdoc at TU Eindhoven until 2008, was a permanent Research Staff Member at IBM Research—Zurich from 2008 to 2012, and then an engineer at Apple until 2014. Prior to completing his Ph.D., he held positions at Bell-Labs Research (1995, 1996), Lucent Technologies and Philips (1999), and NEC Research Labs (2005). He is a senior member of the IEEE. Contact him at phillip.stanley-marbell@eng.cam.ac.uk.

Martin Rinard is a Professor in the MIT Department of Electrical Engineering and Computer Science and a member of the MIT Computer Science and Artificial Intelligence Laboratory at the Massachusetts Institute of Technology. His research interests have included programming languages, computer security, program analysis, program verification, software engineering, and distributed and parallel computing. Prominent results have included automatic techniques that enable applications to survive otherwise fatal errors and security attacks and techniques that trade off accuracy of end-to-end results in return for increased performance and resilience. He holds a PhD in Computer Science from Stanford University. He is an ACM Fellow and has received many awards including an Alfred P. Sloan Research Fellowship and Distinguished and Best Paper awards from a variety of publication venues. Contact him at rinard@csail.mit.edu.