DesignEye: A Tool for Design Teams to Analyze and Address Visual Clutter

by

Amal Kumar Dorai

S.B., Massachusetts Institute of Technology (2005)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2008

Author .....
Department of Electrical Engineering and Computer Science
May 23, 2008

Certified by .....
Dr. Ruth Rosenholtz
Principal Research Scientist
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

DesignEye: A Tool for Design Teams to Analyze and Address Visual Clutter

by

Amal Kumar Dorai

Submitted to the Department of Electrical Engineering and Computer Science
on May 23, 2008, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

User interface design is critical for the success of any information technology, from software packages to automobile dashboards. Colleagues from many different job functions often need to collaborate to produce these designs, in work envioronments which can often be strained. Collaborative design software can alleviate some of the problems of these teams, but current software is rarely tailored to the needs of a cross-functional group and does not actively guide users to create better designs.

In this thesis, I describe DesignEye, a tool that I have developed with the Perceptual Science Group at MIT. Our tool computes the clutter of images and identifies the most salient visual elements, and allows designers to work with the softare in a tight iterative feedback loop to ensure that the most important design elements garner the most end-user attention. DesignEye is tailored to the use cases we have observed in our studies of design teams, and facilitates side-by-side comparisons of multiple design candidates as interface designers test various ideas.

I conclude by demonstrating the extensibility of DesignEye, and its role not only as a tool but as a generalized platform to assist interface designers. Any vision model which quantifies some aspect of human cognitive response to a stimulus can be incorporated into DesignEye, allowing interface designers to create better and more effective user interfaces in an information technology world that is growing rapidly more complex.

Thesis Supervisor: Dr. Ruth Rosenholtz
Title: Principal Research Scientist

Acknowledgments

First and foremost, I would like to thank my advisor, Ruth Rosenholtz, for taking me on as a student on such short notice, and then helping me focus on the most important issues and see the broader implications of my work. I would also like to thank Will Schroeder of MATLAB for his enthusiasm for my project and his willingness to support our work through his company's resources. Finally, I would like to thank my mother, without whom I would not have had this opportunity.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

1.1 Vision

A user of a typical computer program today faces a dizzying array of menus and icons. An airline pilot needs years of training to be able to find the right instrument among dozens in the event of an emergency. Automobile drivers face a cluttered assortment of gauges which can distract them from hazards on the road.

The entire field of user interface design, and more narrowly human-computer interaction, seeks to address the challenges users face in interacting with and marshaling large amounts of information. There exists a large corpus of theoretical results and rules of thumb which can be applied to improve interface usability and learnability. In practice, however, applying these theoretical results is difficult because interface design is a thorny process involving many functional groups with different agendas. Designers, human factors specialists, engineers, marketers, and brand managers often find it difficult to coordinate and communicate when developing interfaces.

The difficulties in this coordination are twofold. First, there is no *a priori* metric of what constitutes a good interface, so the design process is frequently interrupted by slow user testing cycles which delay the iterative design process. Second, there are few interface design tools which are well-suited for collaboration between disparate functions within a company.

The Perceptual Science Group at MIT is dedicated to exploring the underlying

mechanisms of human visual perception, and has developed the Statistical Saliency Model [11], and the Feature Congestion measure of visual clutter [12], measures which describe which elements of an image are salient, the approximate difficulty of searching for an element within the image, and the ease with which an attention-grabbing element can be added to the display. These models allow interfaces to be optimized for visual clarity, which is almost a prerequisite for further optimiziations in interactive behavior.

The algorithms, however, are difficult to use on their own; in their raw form, they are better suited for mathematicians and scientists than for interdisciplinary design teams. The purpose of this project was to create a tool which facilitates this type of group interaction, drawing on the existing research in collaborative interface design, and optimized for iteratively creating progressively less cluttered interfaces.

As computing devices continue becoming more powerful and less expensive, the problem of interface clutter will only get worse if left unmanaged. Our design tool combines the power of the Statistical Saliency Model with intuitive functionality to allow modern design teams to combat this proliferation of clutter.

1.2 Organization

First, the specified requirements of our design tool, and the related research which informed our efforts. Chapter 2 reviews the Statistical Saliency Model and the relevance of our metrics in making design decisions. In Chapter 3 I discuss the features of my tool and the justifications for each, as well as an overview of the implementation details. In Chapter 4 I assess the tool against its design goals and outline potential applications. I conclude by listing the contributions of the project and likely avenues for future work.

1.3 Requirements and Usage Scenarios

In creating DesignEye, we solved a **more general problem** than analyzing clutter and saliency alone. We attacked the higher-level problem of creating a tool that extends beyond our particular vision models, and can be modified to use any model that predicts cognitive responses to a given design. By creating a general-purpose architecture for receiving automated feedback for a design, other models that analyze different aspects of visual perception can benefit from being encapsulated in a DesignEye environment. Because the functionality of DesignEye is completely abstracted from the underlying clutter and saliency transforms, any other mapping transform can be substituted in its place.

When developing the requirements for DesignEye, we worked primarily from pilot studies in the Perceptual Science Group focusing on the collaborative design process. We also took into account the achievements and limitations of prior research in the field, especially other design tools. Our pilot studies suggested the following use cases for such a design tool. Details on the specific implementation of the requirements is in Chapter 3.

1.3.1 Single-user

Any design tool focused on collaboration must be effective for a single user; users should view DesignEye as a proactive step to ensuring that their interfaces are visually uncluttered, rather than as simply rhetorical tool to trump the superiority of their ideas. As single-user desktop software, DesignEye will probably spend most of its time in this use case, and we developed the following requirements

- Correctness: The most basic requirement of the tool, of course, is that it correctly implement the Statistical Saliency Model algorithm and that it allows users to process any valid image. The core clutter and saliency algorithms "attach" at just one point in the code, so that they can be swapped out for any other algorithms as necessary.

- Performance: The tool should be as fast as possible given the limitations of working within MATLAB. We precompute all image transforms to the extent possible, and compile the software into native x86 for better performance.

- Flexibility: The user must be able to select parts of images to analyze, view multiple images side-by-side, etc.

1.3.2 Single-function use

When multiple users from one group are working on a single project, the technical behavior is very similar to the single-user use case but a few additional features faciliate group collaboration. DesignEye sessions can be saved, sent to colleagues, and restored, so that team can share ideas and thought processes with each other in the context of the predicted visual perception of the design.

1.3.3 Multidisciplinary use

At the multidisciplinary level, the political and organizational implications of the tool start to outweigh the technical merits of the computations. Providing an intuitive interface and common language becomes critical, and the outputs must be as resistant as possible to misinterpretation.

The side-by-side comparison functionality of DesignEye aims to reduce the possibility for misinterpretation. The concept of locked scaling, discussed in Chapter 3, ensures that image adjustments are always kept consistent to avoid misinterpretation (see section 3.4.1 for more details). Thus, designers can use the tool to clearly articulate the justifications for their decisions to managers/engineers/executives with less experience in human factors and visual perception.

Furthermore, when saving a session and restoring it, the user can recompute the saliency and clutter metrics, avoiding the possibility of one party willfully misleading another (as might be the case when an advertising agency is working with a client, etc.) Because the tool is a compiled program, tampering with it will be very difficult.

1.3.4 Dynamic Interfaces

Much interface complexity comes not from the static layout itself, but from the logical flow through various modes. The visual dynamics of mode transitions (animations, etc.) were outside the scope of this project and may be addressed in future work.

1.4 Implications of Related Research

Although the Statistical Saliency Model is a recent contribution of the Perceptual Science Group, my work on this tool was informed by prior work in two broad categories: software tools to assist interface design, and the organization and communication frameworks for multidisciplinary design teams. There is of course a large corpus of prior work on visual search and clutter which pertain to the Statistical Saliency Model itself, but which did not directly inform the design of the tool itself.

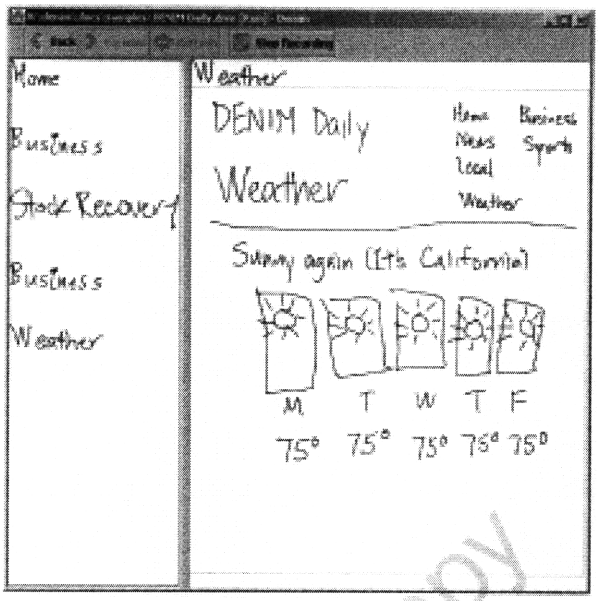
1.4.1 Interface design tools

Since DesignEye is an interface design tool, the most relevant prior efforts are other design tools themselves. While there are many such tools, described below, which aim to simplify especially the early design and prototyping process, no existing tools actually evaluate designs. The user has no benchmarking metric to see whether their design is improving along a defined parameter; this is one of the gaps that DesignEye addresses.

Newman and Lin et al. [10] developed the Denim Web site design tool, which allows Web designers to sketch a Web site and pathways describing links between Web entities. The tool then translates the handwritten sketch to a live site map, allowing the designer to get a feel for the navigation dynamics and make adjustments. It is focused on the early design process, not the camera-ready final adjustments, and is an effective enhancement of the standard paper prototyping process commonly used by Web designers.

Landay and Myers describe the SILK interface prototyping tool [9], which is a

Figure 1-1: Example website created with Denim – handwritten prototype has link dynamics embedded.



more general tool than Denim in that it is not limited to Web pages, but allows instant prototyping and walkthroughs of any visual interface. Currently used for shared meeting records and similar collaboratively produced documents, SILK could provide an API interface for our tool, where designers could use SILK to develop the dynamics of an interface and our tool to work on the saliency of specific static elements.

Hammond and Davis attack a even higher-level metaproblem with LADDER [7], a language that specifies how sketches are translated into domain specific descriptions. One can use a LADDER specification in a relatively constrained design environment where the overall design space is relatively constrained. One can envision an application of our tool which uses a weakly specified LADDER transform to compare the clutter of two sketched interfaces; this would be difficult for raw sketches because variations in the sketch itself would overwhelm the higher-level structure in the clutter measure.

Figure 1-2: SILK is a powerful tool for quickly prototyping an actual graphical user interface.

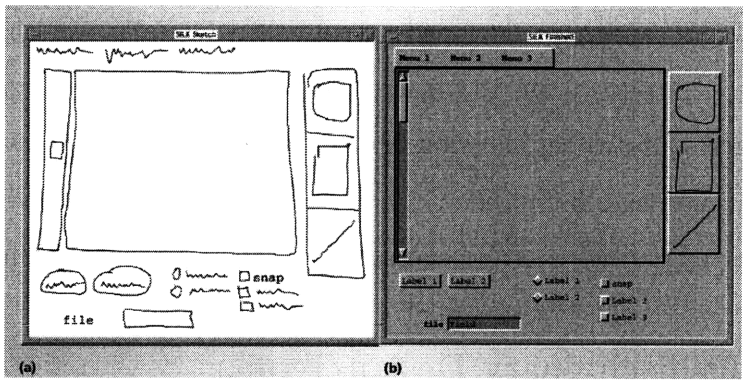
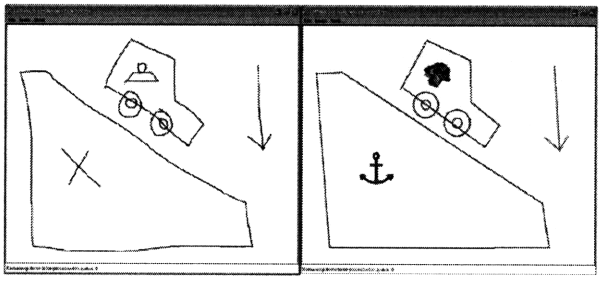


Figure 1-3: Example LADDER session in a physics/engineering context.



Adler and Davis [2] extend the sketch recognition aspect of their work to multimodal conversations about design, such as speech which captures the *rationale* behind team-driven design decisions. They are building inherently mutimodal design tools such as interactive whiteboards, and interface this with the existing LADDER functionality.

Sinha and Landay have extended the design of visual interfaces into the multimodal world [13], and have devloped a tool called CrossWeaver that is useful for interfaces where transitions can be triggered by keystrokes, gestures, voice commands, etc. This suggests a future application of our tool; by evaluating the visual clutter of an multimodal interface, a tool like CrossWeaver can suggest moving some functionality into other modalities to avoid excessively cluttering the visual interface.

1.4.2 Multidisciplinary Design Teams

Another goal of DesignEye is to faciliate the often fractious multidisciplinary design process. Adamczyk and Twidale [1] explain why historically, collaborative design tools fail to achieve widespread adoption among multidisciplinary teams. He points out that many tools constrain what should be naturally an open, "playful" process, and that the tools do not have the flexibility to allow teams to find a common ground to start the design process in earnest.

Adamcyzk's argument is reflected in both DesignEye's structure and how we position it for use in teams; early in the design process, it must be a casual-use tool that is easy to use and does not impose any rigid guidelines. For visual tweaks later in the process, the tool can be more constrained as it will probably be the domain of a single individual or function. DesignEye also reflects existing research in communication among design team members.

Hendry [8] and Borchers [4] claim that one of the biggest impediments to collaborative design is the lack of a common language or representation for design concepts. This is highly relevant to users of our tool; without a common definition of "saliency" and "clutter," and without a threshold value of what constitutes a "good" design, it may be difficult to see the relevance of our metric. This is why we invested effort

in side-by-side comparison functionality for our tool; instead of explicitly defining "clutter" as a bad property of designs, we offer it as an inevitable parameter of any design and encourage users to apply judgment in trading off between clutter and information.

Bodker's work [3] focuses on stimulating creativity in the collaborative design process, and her work does throw a caution flag for evangelizing our tool too heavily. Our tool, if used very early in the design process, will tend to be constraining in nature as people naturally see a quantiative clutter metric and try to minimize it. Thus, they may be less creative in thinking of novel layouts or interface dynamics that could ultimately produce a better interface. For this reason, it may be better to avoid our tool in the earliest stages of design prototyping, and Bodker's lessons imply that a clutter/saliency metric for hand-drawn sketches may foster stifling behavior in the collaborative design process.

Burns and Vicente describe a framework [5] for describing communication in multidisciplinary design teams, and analyze their framework in the context of a real-world design process for a nuclear power plant control room. They use three levels of abstraction: "Objectives," "Functional Processes," and "Physical Components" for design situations where the physical properties of the components are not infinitely modifiable by the engineers. While the clutter of an individual component may be fixed, our tool can definitely help such teams achieve visual saliency for the right elements by modifying the layout and background of each component, such as panel dimensions, room configuration, etc.

Maia and Garcia have developed a framework for more systematically analyzing videotaped sessions of design meetings [6], specifically in the domain of collaborative software design. While this is more focused on observing the design process itself, the representational language that they have constructed could be used in interdisciplinary communication with our tool. As a design evolves and various designers try to "de-clutter" the interface, their language could serve as a tagging language to explain actions taken by each designer.

Sonnenwald has approached design teams from the interpersonal dynamics viewpoint[14],

and has categorized design team roles into 13 "communication roles" and specified the types of interactions likely to occur. While hers is more of a descriptive than a prescriptive analysis, her research suggests the types of people who are likely to become the champions of our tool and its primary advocates within design teams. An "Intragroup" star, who facilitates interaction among group members, would be interested in our tool as a way of avoiding baseless bickering between group members with no factual basis. The intratask star would be more focused on the actual science behind the tool, and would probably serve as the "brain in the loop" making the inevitable tradeoffs between information content and visual clutter.

Chapter 2

Saliency and Clutter Measures

This chapter explains the theory behind the algorithms that our tool uses to provide iterative design feedback. Based on the Statistical Saliency Model, these algorithms help designers identify which elements of their design will garner the highest user attention, and how they can change their design to draw focus elsewhere or add new elements without increasing confusion.

2.1 The Statistical Saliency Model

The Statistical Saliency Model is based on the notion that the visual system notices items that are "unusual," and the development of the algorithms have been an effort to quantify this measure of "unusual" in a way that correlates with observed search times in subject testing.

Framed graphically, if the centroid of the local set of features is plotted in the feature space, the most salient features will be those that are relatively distant, and the least salient features will cluster around the existing features.

The following example highlights how the Statistical Saliency model captures actual perceived differences in feature saliency; if an item can be placed in a way that gains attention, it is likely that the existing features have low variance along some parameter, which must be varied strongly in the newly-placed feature.

Figure 2-1: (a) In this feature space, the item represnted by the black dot will be much less salient than that represented by the white dot. From Rosenholtz et al. 2008.

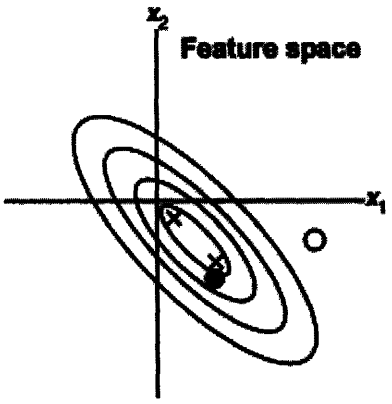
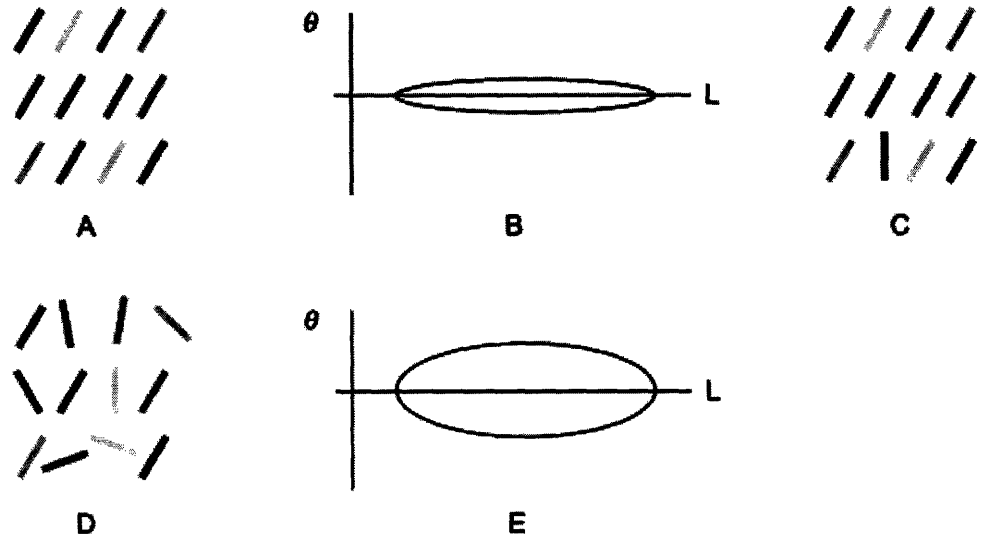


Figure 2-2: (a) High luminance variability, low orientation variability, as indicated in Panel (b). (c) One easily notices an item with an unusual orientation. (d) High variability in both luminance and orientation, as indicated in Panel (e). It would be difficult to draw attention to an item in this display using only the features of luminance and orientation. From Rosenholtz et al. 2008.



2.2 The Feature Congestion Clutter Measure

We use the term *saliency* to mean the extent to which a given visual element draws a viewer's attention, and our saliency maps are a transform that when applied to a target image highlight those areas most likely to draw attention. *Clutter* is the extent to which it is *difficult* to add a salient feature to an image; when an image's clutter map is computed, the strongest points will have a combination of feature orientations, colors, and constrasts/edges which make it very difficult to grab a viewer's attention at this spot. More specifically, when clutter is high, an excess of items has caused a degradation of user performance at visual search tasks. Our Feature Congestion measure of clutter is thus based directly on the Statistical Saliency Model, which in fact correlates well with other bottom-up measures of clutter, including those as simple as JPEG compressed file size (which is directly propotional to low-level image entropy).

The Feature Congestion measure of clutter is only one of several possible clutter metrics. For example, the Subband Entropy clutter measure is derived from the efficiency of compressing the image; lower-efficiency compression means there was less regularity and thus more clutter in the original image. The Edge Density measure works well on maps and other images with sharply delineated element boundaries.

Yet, after extensive user studies, we decided to use Feature Congestion as our canonical clutter measure. This is the only clutter measure that can properly deal with color congestion, which is a major issue especially in software design where color proliferation dramatically hurts object saliency. Furthermore, the Feature Congestion measure tracks quite similarly to other measures in cases not involving color, and incorporates concepts like edge detection (in the contrast clutter map).

Perhaps the most important advantage of the Feature Congestion measure over other clutter measures is the fact that it is a pointwise calculation and thus outputs not only a clutter value, but a *clutter map.* In a large interface with multiple components, a single clutter value may not offer enough feedback to guide a user; the clutter map allows users to quickly focus on the "worst" elements on the design.

2.3 Applications

Regardless of the underlying algorithm used, any evaluative metric, embedded in the DesignEye environment, can deliver strong organizational/political benefits. In a multidisciplinary design team, different groups have different agendas and are often talking on "different wavelengths." Having an objective, quantitative measure of a desirable visual property can encourage discussion and help designers work together towards a commonly defined goal. In a way, Feature Congestion clutter can be an element of the "language" that previous research has found essential to a happy crossfunctional design team.

The Statistical Saliency Model (SSM) and Feature Congestion Measure (FCM) have important applications in real-world situations, and bringing the power of the models to bear on these problems was the main motivation for building the tool. A quantitative metric of interface "goodness" is a boon to design teams both from a technical and organizational angle. Technically, the transforms allow designers to verify that the elements they are trying to make salient are in fact so, and encourages them to work on de-cluttering regions before trying to place any salient features there. With an operator in the loop to make the judgment calls about tradeoffs between presenting more features vs. presenting a more cluttered interface, the tool can be a powerful way of iteratively improving designs.

Chapter 3

The DesignEye implementation

Starting with the saliency and clutter algorithms implemented in MATLAB, the goal of my project was to implement an interface to these algorithms that also met the design requirements specified in Chapter 1. This chapter describes the important features and the rationale for each.

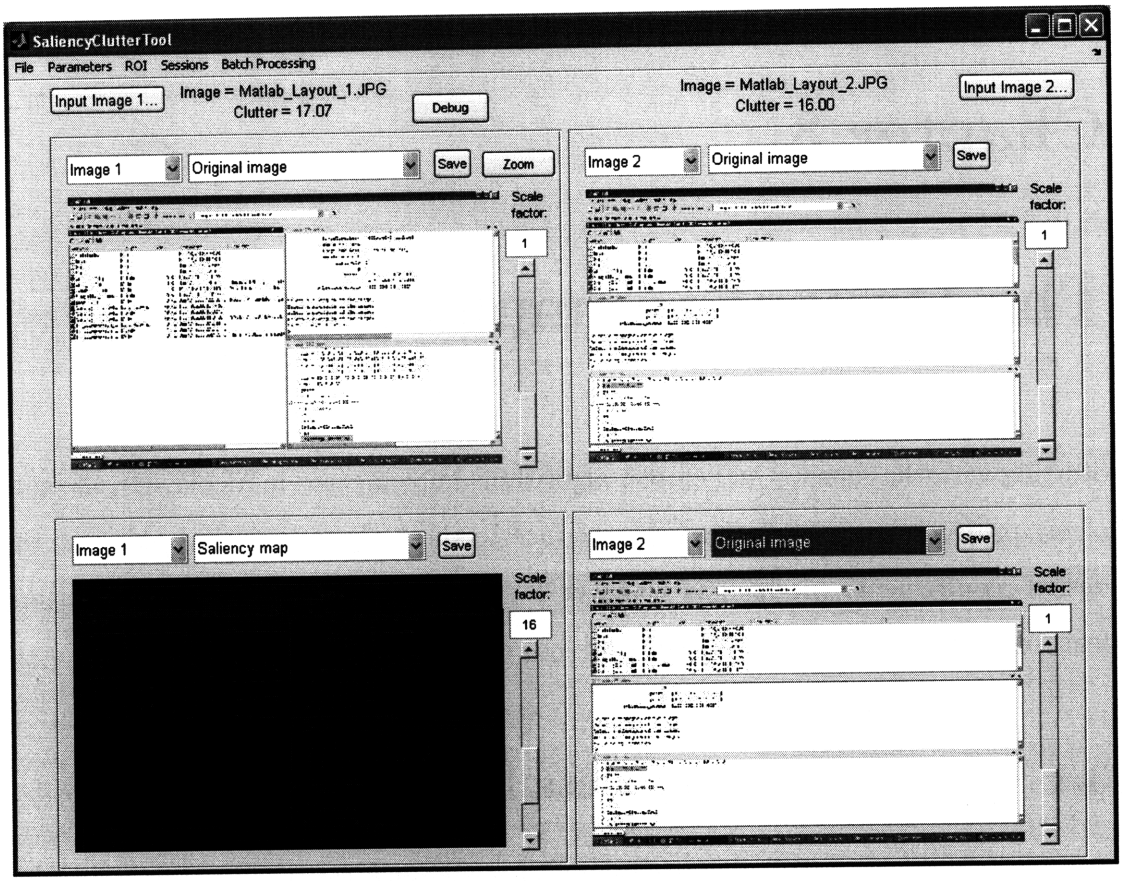
3.1 Implementation Details

To implement this graphical tool, I could have compiled the core algorithms to x86 and then built an interface around them with a cross-platform GUI builder like Java, but I chose to build the interface inside MATLAB. This was for two reasons: first, MATLAB's GUIDE GUI design tool, while leaving much to be desired, is an effective way of building interfaces to scientific algorithms. Second, MATLAB's cross-platform compatibility is easier to work with than Java's.

This did impose some limitations on the flexibility of the design. MATLAB is a single-threaded environment by design, so it is impossible for my tool to run computation in the background which could be a very useful feature extension in the future. GUIDE has very poor support for custom icons and graphical elements, so all actions are triggered by text-based buttons.

Because of these limitations, the interface of the tool itself is not optimal when evaluated through the lens of our algorithm. In fact, our tool would do rather poorly

Figure 3-1: The DesignEye tool upon starting and loading two images for comparison.



on a saliency and clutter evaluation itself; we accept this modicum of hypocrisy as the cost of cross-platform flexibility and the ability to quickly add new features to the tool.

3.2 Saliency and Clutter Maps

The most basic functionality the tool offers is the ability to compute and display saliency and clutter maps in an interactive design program. When a user loads an image into the software, the GUI pauses while saliency and clutter are computed, and stores these in a background array. The overall clutter value of the image is displayed on-screen, and the user can select one of five options from a dropdown menu: the

original image, the saliency map, the combined clutter map, and the three individual clutter maps: color, contrast, and orientation. Any of these maps can be saved to disk for future reference, and the brightness "scale" can be adjusted to adjust for systematically low or high clutter values.

Loading another image puts the new image at the front of the queue, and users can jump between images at any time. Even this basic functionality empowers users to turn a critical eye on the clutter of their designs. Using only a single image at a time, a designer can iteratively strive for lower clutter values and ensure that the correct elements are salient. If the designer has prepared a set of design variants ahead of time, he can load the entire run and quickly see which are the least cluttered.

3.3 Regions of Interest

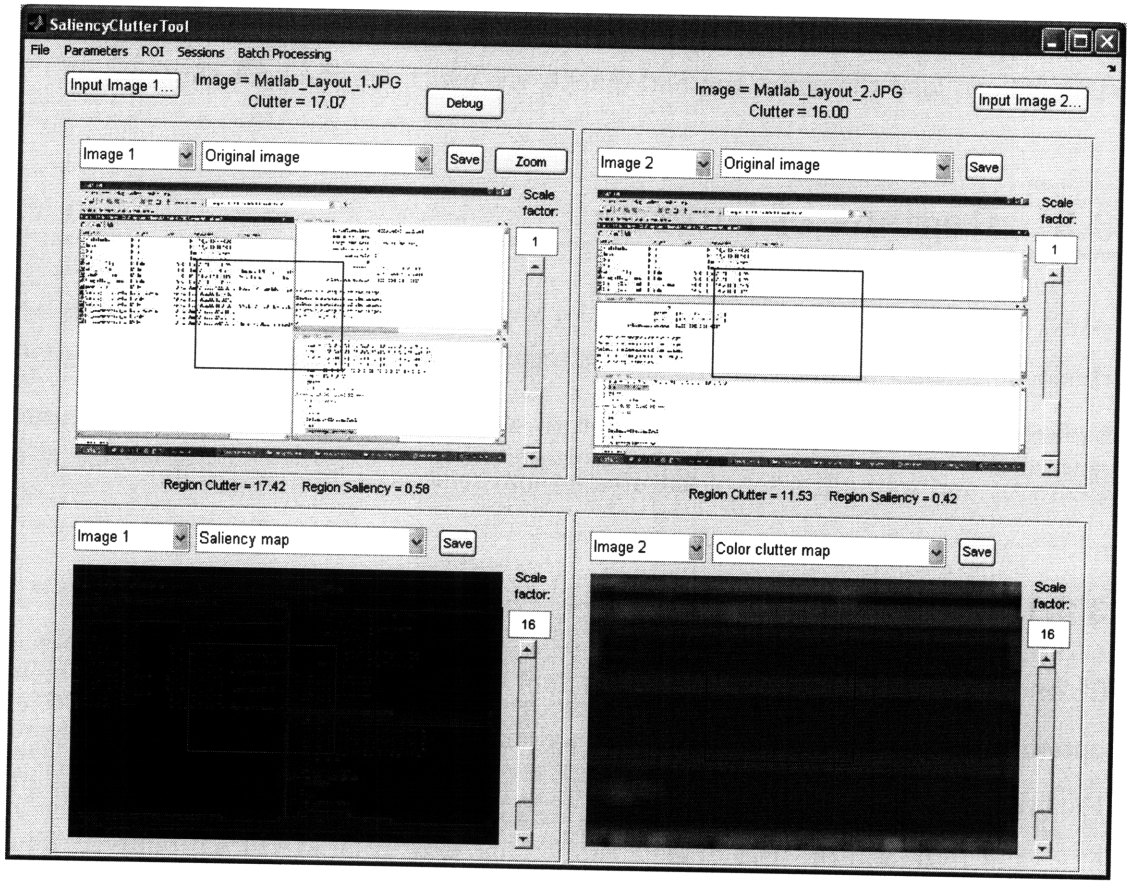
After implementing the core functionality, we found that users often want to assess the local saliency and clutter of a specific region within an image. The images might be largely identical, or the user may want to see the clutter contribution of individual elements, and we built a set of features around regions of interest (ROIs).

3.3.1 Selection

To select an ROI, the user first puts the tool in ROI selection mode through a menu option. On this event, the tool registers a global `OnMouseDown` listener which looks for the first click that a user makes. If it is on an image, the point becomes the beginning of a valid ROI, and the user drags a rubberband box using MATLAB built-in `rbbox` to define the ROI. We are now faced with three problems:

- It is yet unknown which corner of the box was the initial point
- We must continously redraw the rubberband box on top of the figure every time we update the plot
- MATLAB's `plot` command expects a series of points to plot in a continuous line,

Figure 3-2: Identical ROIs selected in two different interface designs.



which requires us to construct the rubberband box as a series of four distinct plot segments

We use the following MATLAB M-code to solve these three problems:

```
handles.point1 = point1(1,1:2);       % extract x and y
handles.point2 = point2(1,1:2);

p1 = min(handles.point1,handles.point2);      % calculate locations
offset = abs(handles.point1-handles.point2); % and dimensions


%% ACTUALLY DRAW RECTANGLE IN BLUE
handles.subregion_plotx = round([p1(1) ...
p1(1)+offset(1) p1(1)+offset(1) p1(1) p1(1)]);

handles.subregion_ploty = round([p1(2) ...
p1(2) p1(2)+offset(2) p1(2)+offset(2) p1(2)]);

handles.subregion_coords= round([ p1(2) ...
p1(2)+offset(2) p1(1) p1(1)+offset(1)]);

hold on

% redraw in dataspace units
plot(handles.subregion_plotx,handles.subregion_ploty);
```

For more advanced shapes, the user can choose a polygonal region, which allows a more arbitrarily defined shape as an ROI.

After the user has selected the ROI, the tool goes to the already-computed clutter and saliency maps and averages the values within the ROI, displaying them in text

below each image as "ROI clutter" and "ROI saliency." Because of the redrawing mentioned above, this ROI outline and associated text stays visible even as the user toggles between clutter maps, saliency maps, and the original image.

3.3.2 Mapping between images

One very typical use case for ROIs is when a user wants to compare the same region in two parallel images; for example, comparing the menu bar in two different design variants for a new release of software. In this case, the ROIs must be identical in both images for appropriate comparison.

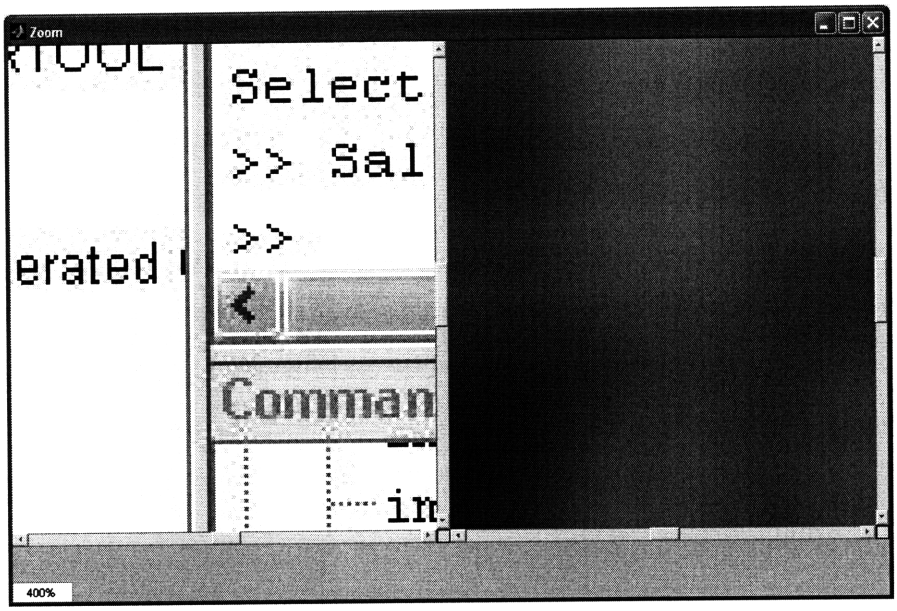
To assist this, DesignEye provides ROI copy-and-paste; when a user has selected an appropriate ROI, he can copy it, placing it in a temporary location in GUIDE's `handles` object, and pasting it into the other images he intends to compare it with. A user-settable parameter allows ROIs to be automatically copy-and-pasted to any new image of exactly the same dimensions; this saves time when a user is working exclusively with a set of screenshots of hypothetical design variants.

3.3.3 Modifying and Saving

One common design modification is moving an element from one place to another, and observing how that affects clutter and saliency. With static ROIs, direct comparisons become difficult since the ROI must be drawn to exactly the same shape and size on two separate images. To address this issue, DesignEye implements drag-and-drop repositioning of ROIs, independently within an image. Thus, a user can copy-and-paste a ROI from one image to another, and then select the "Move ROI" menu item. This attaches a drag-and-drop listener to the new ROI, upon which the user can move it to coincide with the target element's new location.

In addition, the last 5 ROIs for a given image are retained in the background, so they can be recalled at a later time, even from a restored session. This allows precisely defined ROIs to be reused repeatedly.

Figure 3-3: Zooming in on a particular section for side-by-side clutter map comparison.



3.4 Side-by-side comparison

A novel feature of DesignEye is the facility for side-by-side image comparison which allows designers to correctly compare images for clutter and salient features. The dual-dropdowns for selecting the image for comparison and the type of output allow the user to either view multiple maps for one image, the same type of map for multiple images, or a combination of the two. This can create consistency problems with the *scale* of each image; the level by which brightness is adjusted. *Locked scaling* is our solution to the problem.

3.4.1 Locked scaling

The key enabler of side-by-side comparison is *locked autoscaling*, a methodology which prevents the tool from showing misleading results. Normally, autoscaling for a single image raises the overall brightness of an image until its brightest pixel is pure white (255,255,255). Thus, regardless of the absolute clutter, the clutter map is scaled for

maximum information content.

This autoscaling runs into difficulties when two different images are being displayed; should each image be scaled up to full white independently? If so, one image could be at a much higher scale than another, and thus appear to be more cluttered, when it is in fact far less cluttered. However, if one forces a fixed scale onto all of the images, they will necessarily stay dark as a single bright image could cause a "race to the bottom" of the scale factor.

Locked scaling dynamically looks at the type of map being shown for each image and scales accordingly. If three saliency maps are on screen at the same time, the highest value of the three is scaled to full white and the other images receive the same scaling factor for direct comparability.

This is, to our knowledge, a unique way of addressing the common problem of autoscaling granularity.

3.4.2 Detail analysis

Another key aspect of side-by-side comparison is the inset window we provide for detailed inspection of images. When the user clicks "Zoom," a new window offers a zoomed-in set of tracking windows which allow a user to pan around the original image and see the corresponding pixels of the saliency and clutter maps. This allows very fine-grained analysis of saliency and allows designers to strive for "pixel perfection" rather than just high-level de-cluttering.

3.5 Session State

A novel feature for tools of this broad category is the ability to save an entire working session and reload it, even on a different computer. When a user decides to "Save Session," the entire catalog of open images, the active ROIs, the active clutter maps, and the computed clutter and saliency maps are rolled up from MATLAB's `handles` object and saved into a single file (which can get quite large). This file can then be transferred to colleagues, who can see not only the program's output but the user's

rationale in tuning parameters / selecting ROIs / etc. We believe that this feature is essential for true collaborative design because it encourages team members to share far more of their thought process than a single numerical output value.

3.6 Batch Input/Output

The clutter and saliency maps are computationally expensive to produce, and many users with dozens or hundreds of images to process will not want to load them manually. DesignEye offers a batch processing mode where users can input an arbitrary list of images and have them process directly to files. In addition, DesignEye can create an HTML file that organizes these output files for easy browsing.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4

Applications

I have described the goals of DesignEye, its specific design requirements, and its implementation. This chapter discusses the possible applications of the tool. These are not hypothetical applications; we have been conducting user studies with design teams from multiple industries, and have found interest in our tool from almost everyone we talk to. First, I will discuss the specific applications of clutter and saliency metrics, and then extend the discussion to the broader applications of the DesignEye system.

4.1 Clutter and Saliency Applications

As computing power increases and electronic devices become cheaper to build, there has been a natural tendency towards device proliferation and interface complexity in everything from toasters to automobiles. BMW's much-maligned iDrive system was accused of being too difficult to see while driving, and the term "information overload" has been coined in response to what many people perceive as too much information with too little structure in its presentation. Making an appropriate tradeoff between clutter and information, and ensuring that the most important elements are indeed the most visually salient, is essential for a well-received interface. This can have serious implications; an insufficiently salient fuel gauge on a car could cause someone to run out of gasoline and get stranded. Even in other domains, the difference between

a clean, de-cluttered interface and a fouled interface can be the difference between commercial success and failure.

How can large amounts of information be presented to users in ways that allow them to quickly find what they are looking for? I look at three major examples.

4.1.1 Software GUIs

Software is a product whose complexity is limited by only the imagination of the developer; as such, cluttered and intractable interfaces are almost the norm in the business. The Windows desktop with 100 icons or the software interface with 50 buttons is nearly unusable, and software vendors such as Apple and Firefox have built competitive advantages in clean, functional interfaces. How can all software developers strive for this level of visual streamlining?

Software is perhaps the most natural first outlet for DesignEye as the teams that design software will be the most comfortable using a new and untested piece of software themselves. Because of the ease of prototyping, much interface research has been in the human-computer interaction (HCI) domain, and DesignEye can provide an easier way to incorporate cognitive research results into the software design process.

By using DesignEye, software designers can keep their software up to date with the theoretical results from perception research, even if they are not familiar or not focused on these results themselves. For example, most interface designers are familiar with Fitts' Law, a model which predicts the time a user needs to place mouse focus on a target area. Fitts' Law implies that commonly used screen elements should be placed at the edges to minimize this time, but Windows 2000 placed a 2 pixel gap between the "Start" button and the corner of the monitor, effectively "snatching defeat from the jaws of victory." (TOG) Had a Fitts' Law transform been embedded in a DesignEye tool and been an integral part of the design process, this mishap could have been averted.

4.1.2 Vehicle Instrumentation

As automotive instrumentation becomes more digitized, more complex, and more integrated with electronics such as navigation systems and Bluetooth mobile phones, designers face the continuous challenge of keeping the instrumentation uncluttered and ensuring that the major safety readings (speed, engine temperature, fuel level, etc.) are highly salient. Automotive designers can use DesignEye to ensure that stray elements such as song title readouts and navigation notices do not overwhelm these important elements. Furthermore, the multidisciplinary teams that DesignEye is optimized for are very common in automotive, with engineers, marketers, and designers all having to agree on each element of the design.

The automotive applications are what prompted me to include the functionality for non-rectangular ROIs. In car dashboards, many elements are non-rectangular, and arbitrary polygonal ROIs are essential for honing in on one particular functional element of the design. Because this feature introduces substantially more complexity into the DesignEye interface, it must be separately enabled.

With computerized "brains" controlling more of the car's functions, multimodal interactions will become more common; perhaps the car will interrupt your music to tell you to refuel. This is an exciting area for future work but not an avenue addressed by DesignEye.

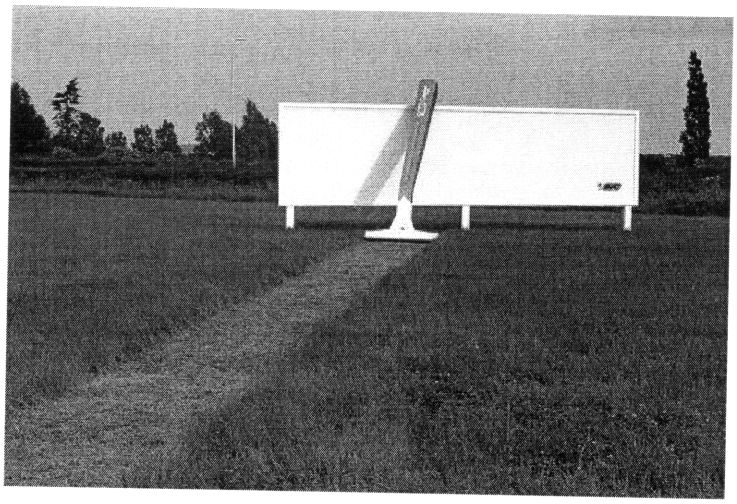
4.1.3 Billboard advertising

Advertisers often have the problem of conveying a commercial message to people with very low attention devoted to the ad. In billboard advertising in particular, advertisers need to capture a person's attention in a sea of visual stimuli. How can advertisers make their messages more salient?

While ad agencies rely on rules of thumb like using no more than seven words on a billboard, these rules are applied only to the advertisement itself, and not to the billboard in the context of its surroundings.

DesignEye allows users to create a billboard that jumps out from its surroundings;

Figure 4-1: DesignEye will never create a billboard concept this creative. It can, however, tell the user what color to make the billboard.



by creating mockups of the proposed billboards in the physical context, advertisers can better predict true saliency. For example, a dark red billboard will not be very salient in a cityscape with predominantly brick buildings; DesignEye can take this reasoning several steps farther to achieve optimum saliency.

4.2 Extensibility: DesignEye as a platform

As mentioned in Chapter 1, DesignEye is not just a tool for measuring saliency and clutter; this is only its first incarnation. DesignEye is a platform for building a design tool around any vision model which can give the user feedback about his design, and the future applications are limited only by the ability of cognitive scientists to create effective models for other aspects of human vision.

As a fanciful example, someone might develop a transform that quantifies the level of "color coordination" from an aesthetic point of view, and a mapping transform that highlights parts of an image that are chromatically discordant. By running DesignEye over this transform, interior designers or web designers could improve color coordination in the same way that they use the current version of DesignEye to

reduce clutter and improve saliency.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 5

Contributions

Does the design community need another tool? Does DesignEye do anything that SILK/LADDER/Denim do not? The answer is a resounding yes, and here I summarize the four major contributions of the tool.

Computerized Feedback. Current design tools strive to capture the *intent* of the user, and offer no critical feedback on whether that intent is well-advised. DesignEye explicitly quantifies the clutter of each image and shows users exactly where the trouble spots are, so teams can use the software for direct technical guidance.

Designed for Collaboration. While many design tools claim to be collaborative, they are often too open-ended to be used in a multidisciplinary environment beyond the earliest prototyping phases. DesignEye puts users on a guided path rather than an open-ended field by giving designers access to the power of the Perceptual Science Group's vision models. The features of the tool are explicitly designed to simplify communications in terms of clutter values, saliency maps, and clutter maps.

Side-by-Side comparisons. DesignEye is built around not just building a design, but comparing different versions of a design. As described in Chapter 3, several novel features have been implemented to minimize the risk of any confusion while comparing saliency and clutter maps of different design candidates.

Arbitrarily Extensible Platform. Clutter and saliency represent only the first use case of DesignEye. As other vision scientists create models to quantify human responses to visual stimuli, designers will be able to use the familiar DesignEye

environment on top of these other algorithms. DesignEye thus becomes a design platform rather than a specialized tool.

DesignEye is a new take on the collaborative design tool – it is based on the idea that the computer can actually help the user make a better design, and makes the tool an active participant in the design process. We trust that it will find widespread adoption in all types of design teams.

Bibliography

[1] Piotr D. Adamczyk and Michael B. Twidale. Supporting multidisciplinary collaboration: requirements from novel hci education. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1073–1076, New York, NY, USA, 2007. ACM.

[2] A. Adler and R. Davis. Speech and sketching: An empirical study of multimodal interaction. EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling, 2007.

[3] Susanne Bodker, Christina Nielsen, and Marianne Graves Petersen. Creativity, cooperation and interactive design. In *DIS '00: Proceedings of the 3rd conference on Designing interactive systems*, pages 252–261, New York, NY, USA, 2000. ACM.

[4] Jan O. Borchers. A pattern approach to interaction design. In *DIS '00: Proceedings of the 3rd conference on Designing interactive systems*, pages 369–378, New York, NY, USA, 2000. ACM.

[5] Catherine M. Burns and Kim J. Vicente. A framework for describing and understanding interdisciplinary interactions in design. In *DIS '95: Proceedings of the 1st conference on Designing interactive systems*, pages 97–103, New York, NY, USA, 1995. ACM.

[6] Antonio Carlos, Pereira Maia, Ana Cristina, and Bicharra Garcia. A method for analyzing team design activity. In *DIS '95: Proceedings of the 1st conference on Designing interactive systems*, pages 149–156, New York, NY, USA, 1995. ACM.

[7] Tracy Hammond and Randall Davis. Ladder, a sketching language for user interface developers. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, page 35, New York, NY, USA, 2007. ACM.

[8] David G. Hendry. Communication functions and the adaptation of design representations in interdisciplinary teams. In *DIS '04: Proceedings of the 5th conference on Designing interactive systems*, pages 123–132, New York, NY, USA, 2004. ACM.

[9] J. Landay and B. Myers. Sketching interfaces: Toward more human interface design. *Computer*, 34(3):56–64, March 2001.

[10] J. Hong M. Newman, J. Lin and J. Landay. Denim: An informal web site design tool inspired by observations of practice. *Human-Computer Interaction*, 18:259–324, 2003.

[11] Ruth Rosenholtz. A simple saliency model predicts a number of motion popout phenomena. *Vision Research*, 39(19):3157–3163, 1999.

[12] Ruth Rosenholtz. Measuring visual clutter. *Journal of Vision*, 7(2):1–22, 2008.

[13] Anoop K. Sinha and James A. Landay. Visually prototyping perceptual user interfaces through multimodal storyboarding. In *PUI '01: Proceedings of the 2001 workshop on Perceptive user interfaces*, pages 1–4, New York, NY, USA, 2001. ACM.

[14] D. H. Sonnenwald. Communication roles that support collaboration during the design process. *Design Studies*, 17:277–301, 1996.