PixelTone: A Multimodal Interface for Image Editing

Gierad Laput[1,2], Mira Dontcheva[2], Gregg Wilensky[2], Walter Chang[2], Aseem Agarwala[2], Jason Linder[2], and Eytan Adar[1]

[1]University of Michigan
School of Information
105 S. State Street
Ann Arbor, MI, United States
{glaput, eadar}@umich.edu

[2]Adobe Research
601 Townsend Street
San Francisco, CA, United States
{mirad, wilensky, wachang}@adobe.com
{asagarwa, linder}@adobe.com

Figure 1. With PIXELTONE, users *speak* to edit their images instead of hunting through menus. a) The user selects the person's shirt and says "This is a shirt." PIXELTONE associates the tag "shirt" with the selected region. b) The user tells PIXELTONE to "Change the color of the shirt," and c) PIXELTONE applies a hue adjustment to the image and offers a slider so that the user can explore different colors.

ABSTRACT

Photo editing can be a challenging task, and it becomes even more difficult on the small, portable screens of mobile devices that are now frequently used to capture and edit images. To address this problem we present PIXELTONE, a multimodal photo editing interface that combines speech and direct manipulation. We observe existing image editing practices and derive a set of principles that guide our design. In particular, we use natural language for expressing desired changes to an image, and sketching to localize these changes to specific regions. To support the language commonly used in photo-editing we develop a customized natural language interpreter that maps user phrases to specific image processing operations. Finally, we perform a user study that evaluates and demonstrates the effectiveness of our interface.

Author Keywords

multimodal interfaces; natural language; image editing

ACM Classification Keywords

H.5.2 User Interfaces: Natural language

CHI 2013, April 27–May 2, 2013, Paris, France.

INTRODUCTION

Photo editing can be a daunting task with a steep learning curve. Not only are interfaces for photo editing often complex, but they also expect the user to learn the language of image processing. Users must understand image properties such as hue, saturation, levels, and cropping, and learn how they are changed and combined to achieve a desired effect. To add complexity, effective image edits are often localized to a specific region, e.g., to brighten a face, recolor an eye, or make a sunset more vivid; this task usually requires sophisticated direct manipulation. Such manipulations are easier with the large displays available in most desktop environments. However, the reality is that image editing is now frequently performed on small, portable devices such as camera phones, which makes complex interactions even more challenging.

Speech interfaces can make complex tasks more accessible because they allow users to simply state goals without first learning an interface. Research on integrating speech interfaces into software applications starting in the 1980s [5, 26] gave rise to today's systems. Popular speech interfaces like Apple's Siri [3] allow users to efficiently perform complex operations (e.g., sending text messages and getting navigation information). However, image editing is hard to perform with speech alone, since people are not good at describing spatial locations; previous work has shown that visual tasks benefit from a combination of speech and direct manipulation interfaces [20, 11, 12]. If we look at how professionals communicate desired photo edits (Figure 2), we find a combi-

nation of symbols, shapes, and text; a type of communication that we believe would lend itself well to a *multimodal interface* that combines speech and direct manipulation. Notably, even professionals have very limited shorthand for describing the changes they want to make and frequently resort to more natural instructions (e.g., "even out skin tone," "clean up floor slightly," or "lighten midtones overall.").

In this paper, we explore the effectiveness of a multimodal interface that combines speech and direct manipulation for photo editing. To guide our exploration, we distill a set of design principles for multimodal photo editing applications by observing existing editing practices. We implement these principles in PIXELTONE, a multimodal photo editing application for a tablet device that allows end-users to express desired changes in natural language and localize those changes by sketching. For example, a user can point to a person in an image and say "this is Bob," and then say "make Bob brighter." PIXELTONE detects Bob's face and associates it with the tag "Bob." When the user refers to Bob, PIXELTONE applies the image editing operations to Bob's face. Users can easily refer to spatial and tonal locations in the image (e.g., "Blur the shadows on the left."), and they can even use subjective ambiguous terms (e.g., "Make this fun").

To make PIXELTONE effective, we developed a natural language interpreter for the photo-editing domain that maps user phrases to image processing operations. Our interpreter segments each user phrase into parts-of-speech components and then maps the parts-of-speech components to predefined phrase templates. PIXELTONE handles ambiguous terms by looking for synonyms or close matches for the ambiguous word that are part of its vocabulary. When PIXELTONE is not able to interpret the user's intent, it offers a gallery of options that may be appropriate. This graceful fallback allows the user to learn the vocabulary supported by the system while successfully editing an image. To evaluate the benefits of a multimodal interface for image editing, we collected feedback from fourteen people. Participants used the different modalities offered, preferring the speech interface when they knew what they wanted to do, but using a more traditional gallery interface to explore available commands.

In the process of building PIXELTONE, we observed that a two-tiered approach that first tries a domain-specific method before resorting to a general purpose one worked best. For example, we combined a very constrained local system with a cloud-based general purpose system for more accurate speech recognition. We also combined a more precise interpreter using parts of speech pattern matching with a bag-of-words approach for improving speech interpretation. Additionally, we also learned that adjusting our vocabulary model towards the photo editing domain made the synonym-matching component of PIXELTONE more robust, which we achieved by mining online photo editing tutorials.

This research makes the following contributions:

- a set of design guidelines for integrating a speech interface into image editing applications,
- a multimodal interface that combines speech, direct manipulation, and galleries for the purpose of photo editing,
- a set of specialized fallback techniques for improved speech recognition and interpretation,
- and an algorithm for turning natural language phrases with unconstrained vocabulary into image editing operations.

RELATED WORK

While natural language interfaces have been studied extensively in a number of domains, including databases [1], automating web tasks [14], command lines [17], mobile phones [3], car interfaces [7], and home media systems [8], natural language interfaces for image editing have received little attention. There are techniques that use natural language to correct colors in images [28, 30]; however, they focused on global operations and limited the user interaction to natural language only. Our language parser supports a much wider variety of commands, including tonal corrections, localized edits, and object tagging. Furthermore, previous findings comparing speech-only and multimodal interface show that a multimodal interface is more effective for visual tasks [20].

Researchers have also studied how to integrate natural language interfaces into sketching tasks. For example, Speak'n'Sketch [27] lets artists issue commands, such as "group", "rotate", or "thicker brush"' as they are sketching. This frees up the artist to continue sketching on the canvas rather than switching between the menu system and the sketching canvas. Pausch and Leatherby [23] showed that adding voice to a drawing application reduced time to completion by up to 56%, with results showing an average reduction of more than 21%.

Researchers have combined image capture and the hardware capabilities of mobile devices (e.g., sensors, GPS) to create new applications for mobile visual computing. For example, Gelfand et. al. [9] described techniques for real-time preview and capture of high dynamic-range (HDR) scenes entirely on a mobile camera. In addition, Xiong and Pulli [29] presented an image blending algorithm that can be applied for panoramas and image compositing on mobile devices.

Consumer mobile applications such as Instagram[1], Camera+[2], Photoshop Express[3], Photosynth[4], SnapSeed[5], and Vapp[6] have made it easier for users to edit and share images. However, these applications mostly support global image edits (e.g., preset filters and simple adjustments), and none of them include a speech interface as an interaction component.

NATURAL LANGUAGE AND PHOTO EDITING

To understand how natural language fits into the editing process, we studied how professional photographers communicate photo edits, and carried out two studies: one small lab study, largely to test prototype features, and one larger online study on Amazon Mechanical Turk, to help us create an initial dictionary of phrases (i.e., lexicon).

[1]To learn more about these mobile applications, please visit: instagram.com, [2]campl.us, [3]photoshop.com/products/mobile/express, [4]photosynth.net, [5]snapseed.com, and [6]vappapp.com.

Figure 2. Samples of professionally annotated photographs. Provided by Chris Buck (top image) and Widen Enterprises (bottom two).

Professional Annotation

Historically, professional photographers and art directors annotated their proofs with instructions on how the images should be modified or printed (see Figure 2). Because edits were to be performed by another person, most often a retoucher or color/printing professional, the description of changes needed to be specific and unambiguous. Annotations typically include a circled region and description of changes. Despite this high-level norm, the subjective nature of images and the language to describe them has made a formal notation system difficult to achieve.

What developed instead was a mix of shorthand for common tasks (e.g., $-M\ vv\ slt$ means a very very slight decrease in magenta [21]) with a significant use of "long form," unstructured, natural language, descriptions of desired changes ("soften the eyes," or "lighten the boots" or "open slightly, too tan"). Instructions can range from the very specific (e.g., $+D.5$, to indicate a one half stop density increase) to extremely subjective and open (e.g., "too green, fix"). Additionally, we note the use of highly metaphorical language: "too hot" (too red), "too cool" (too blue), "flat" (not enough texture), or "can be opened" (brightened) instead of direct references to image features (e.g., color levels, hues, saturation, etc.). Finally, the example images we studied demonstrate the tactics of localization either by circling or referencing an object in the text (e.g., shoe, face, or teeth).

Conventional wisdom for photographer-retoucher workflow is that ambiguity is to be avoided. For example, one would prefer to say, "remove wrinkles" instead of, "lighten wrinkles," which has two meanings (change exposure or eliminate) [25]. This tension between expressive descriptions, range of editorial control (from "do it exactly as I indicate" to "make it look good"), and demand for low ambiguity have led to reduced use of shorthand and increased use of unstructured natural language.

Novice Editing Behavior

In addition to studying professionals, we wanted to see how casual photographers would interact with a speech interface.

Pilot Lab Experiment

We recruited five participants for a pilot experiment (three novice, two advanced). We asked the participants to tell us how they would improve a given image (four images). Next, we showed them two images (image A and image B) and asked them to tell us what they would say if they had to direct another person to transform image A into image B (six pairs of images). Finally, we asked the participants to edit images using an early prototype of the system that mapped keywords to commands (five images). Each participant performed all 15 image tasks.

Crowdsourcing

To gather additional natural language phrases, we used Amazon Mechanical Turk. Our experiment was very similar to the one performed in the lab. The Turkers were asked to do two things: 1) describe how they might improve an image, and 2) create instructions for transforming one image into another. Although our experiment on Amazon Mechanical Turk did not use a speech interface, the images themselves were the same as the lab study, and the Turkers entered descriptions as unstructured text. The Turkers were paid $0.05 for each image or image pair. We showed 10 individual images and 14 image pairs and collected 10 responses for each task (240 responses). Users saw each image and image pair once. After filtering out bad data (e.g., instances where Turkers gave an unrelated one-word description of what is in the image), we had 211 valid responses from 35 unique workers. We analyzed and coded the data using an affinity diagram.

Study Findings

One of the key findings from these experiments was that while users had a common language for describing changes, discoverability of the lexicon terms was difficult without guidance. After trying out our early prototype, users in the lab study noted that it was hard to remember the commands.

Participants used their prior editing experience to guide them on what to say. For example, more experienced users performed more adjustments, and used more advanced terminology (e.g., "adjust the mids," "bring the black levels up," "add a soft vignette,"). Consistent with previous findings [20], users mentioned that specific, localized and elaborate changes were challenging to describe, while global operations were easier. Users also naturally referred to objects within an image, but they varied in their degree of precision (e.g., "the front," "the person on the left"), and they also pointed to specific regions of the image (e.g., "this," "here").

Through both the lab and crowdsourcing experiments we found consistent use of both imperative and declarative language to describe desired changes. *Imperative* phrases contain verbs and act on the entire image or portions of it (e.g., "make this bright," "increase the saturation on the image a little bit"). *Declarative* phrases indicate a problem in the image needing correction without specifying the operations that corrects it (e.g., "this is too dark," "this is way too blurry").

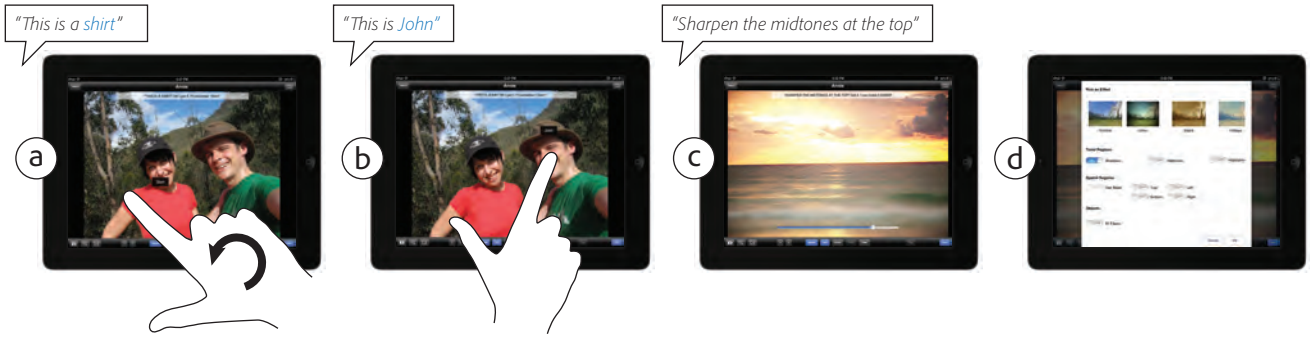

Figure 3. a) The user can select an object in the image and tag it by saying the name of the object, in this case a shirt. b) The user can also point to an object. PIXELTONE makes use of face detection to associate the tag "John" with the face on the right. c) PIXELTONE supports more advanced photo editing language as well. Here the midtones in the sky are sharpened. d) Through PIXELTONE's gallery interface, the user can explore available commands and learn the system vocabulary.

Design Guidelines

In addition to design directions that were consistent with previous studies (e.g., incorporate multiple input modalities [20], fail gracefully [14]), our analysis of professional annotations and study findings helped us distill key design guidelines for incorporating natural language into image editing interfaces.

Incorporate object references. Both professional and novice photographers often refer to objects in an image when using natural language (e.g., "the background," "this person," " the shoes"). Systems should leverage object recognition and offer tagging as a way to reference objects in an image.

Support declarative and imperative sentences. When users know how to improve an image, they tend to use imperative phrases that describe what should be done. But users, especially novices, don't always know how to improve an image. They simply know what's wrong with it. In such cases, they are more likely to use declarative phrases that state what they don't like about an image.

Guide users with available commands. Due to the diverse and large set of possible utterances in the image editing domain, extra care is necessary to properly address discoverability and learnability.

Allow localized image edits. Users may want to edit only part of an image, so a system should facilitate selecting a variety of regions including spatial regions ("the bottom left," "the top"), tonal regions ("shadows," "midtones," "highlights"), and colors ("the reds," "greens"). Different modalities may be used to support different types of referents.

USER EXPERIENCE

Following the design guidelines listed in the previous section, we created PIXELTONE, a multimodal photo editing application that lets users describe their goals with natural language and direct manipulation (see Figure 3). Let's follow Eve as she uses PIXELTONE to improve her photos.

Image Adjustments

First, Eve opens PIXELTONE and loads a photo that she took from a recent vacation. She notices that the image looks a little dark and needs adjustment. Eve says "lighten the image" and PIXELTONE immediately enhances the brightness of the photo. Eve is presented with a slider, and she uses it to fine-tune the degree of brightness. Once done, Eve taps the "Save" button and saves changes to her newly enhanced photo. Next, Eve notices that specific areas in the photo need some enhancements. She says "sharpen the bottom" and PIXELTONE applies a "sharpness" filter to the bottom region of the image. Next, Eve wants to improve the dark areas in the image. Eve says "make the shadows more contrasty" and PIXELTONE enhances the contrast on the low-intensity regions of the image. Finally, Eve draws a circle around one of the grassy areas of the photo and says "make this greener," and PIXELTONE applies a "green tint" to the region Eve selected.

Using Arbitrary Words

Users may not always know how to improve an image. For instance, when Eve wants to enhance one of her skyline photos but she is not quite sure what to do, she can say "make this better", and PIXELTONE finds an effect that approximates her goal. In this case, PIXELTONE applies the "auto-color" operation to Eve's photo. When PIXELTONE does not understand what Eve wants and cannot find a suitable approximation, it offers a fallback interface which offers a list of image operations (Figure 3d).

Implicit References

When Eve wants to enhance a recent photo of her friends, she loads the image in PIXELTONE, points at a person in the photo, and says "This is Roslyn." PIXELTONE identifies "Roslyn" as one of the names in Eve's contact list, uses face recognition to find a person in the region where Eve pointed, and *tags* it as "Roslyn" in the photo. Then, Eve says "add soft-focus on Roslyn," and PIXELTONE applies a "soft focus" filter to the region in the image with Roslyn's face.

SYSTEM DESCRIPTION

We implemented PIXELTONE on a tablet device (iPad), and used a client-server architecture to distribute computationally intensive processing functions on remote servers. PIXELTONE consists of three main components: 1) a *speech recognition* component that converts users' voice into text, 2) an *interpreter* that parses an input phrase into parts-of-speech, matches them with a set of *phrase patterns*, and maps them into different components of an image processing request,

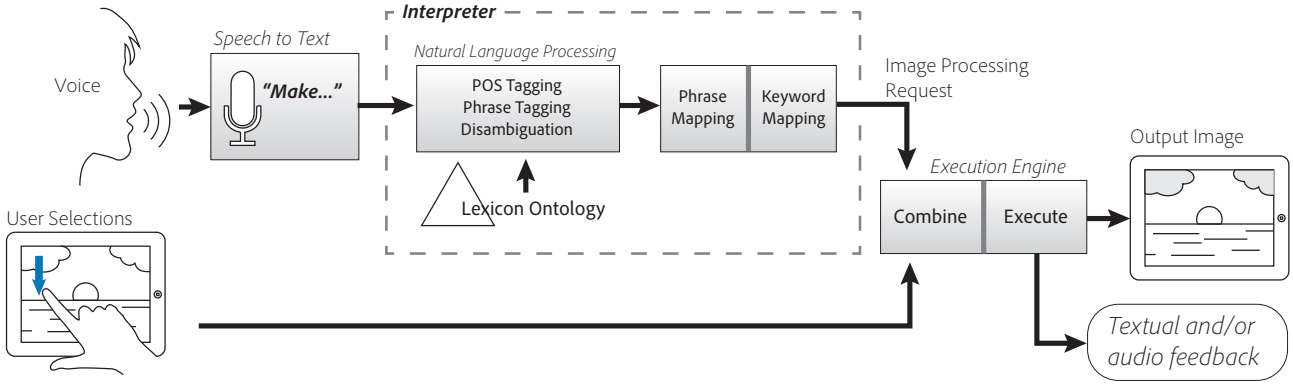

Figure 4. PIXELTONE Architecture. A user interacts with PIXELTONE through a combination of voice and direct manipulation. A speech recognition component converts users' voice into text; an interpreter combines natural language processing and keyword matching techniques to map a user's input phrase into an image processing request; and an execution engine combines the interpreted command with user's direct-manipulation input to execute an image processing operation.

and 3) an *execution engine* that combines the interpreted command with users' direct-manipulation input to execute an image processing operation (see Figure 4).

Speech recognition

We use existing speech recognition technology to convert voice into character strings that PIXELTONE can process and understand. We explored several speech-to-text implementations and identified two classes of systems: 1) local speech recognition capable of understanding a finite set of words, and 2) cloud-based speech recognition geared towards a more generalized vocabulary. Neither implementation was satisfactory in isolation, but we found through iterative testing that a multi-pass approach worked well for our purposes.

Local speech recognition

In the first pass, we employ a local speech recognition engine. We use data gathered from our user study and crowdsourcing experiments as the corpus for the local speech recognition component. For utterances that fall within the corpus, this approach yields high accuracy and a relatively fast response. PIXELTONE uses OpenEars [24] for local speech recognition.

Remote speech recognition

If the local speech recognition component encounters an "out-of-vocabulary" utterance (i.e., instances where the confidence level is lower than 10%), PIXELTONE sends the recorded voice data to a remote speech recognition server. Although this approach is slower compared to local speech recognition, it provides PIXELTONE with access to a more generalized vocabulary when needed. We use the iSpeech HTTP API [13] for remote speech recognition.

Speech interpretation

Once speech is converted to text, the *interpreter* (Figure 4) analyzes and maps the input command into an action request. Our approach uses a variant of the keyword-command technique proposed by Little and Miller [15]. Their approach breaks an input command into tokens and recursively builds a function tree by scanning for keywords that match functions and data types. Instead of a recursive approach, we simplify the process by taking advantage of the fact that the functions we support in the photo-editing domain follow a more generalized syntax (i.e., most functions only require the name of an effect, along with optional parameters on where to localize the effect). We extend their approach by segmenting an input command into parts-of-speech tags, matching tags against a repository of *phrase templates*, and then remapping keywords into functions and parameters. This technique allows us to scan for functions and parameters to specific words that play a "grammatical" role within an input phrase. For example, if we know that a verb in a phrase pattern maps to an image operation (i.e., "brighten the image"), we can match for operations that fit with that verb and not to other parts of the sentence.

If the interpreter cannot find a matching phrase template, our system ignores the parts-of-speech tags and scans for keywords by treating the input phrase as a *bag-of-words*. This approach works well when word order is jumbled or when a sentence is awkwardly phrased (e.g., "left shadow brighten"). The bag-of-words approach also allows users to not follow strict sentence structures.

In the following sections, we describe the algorithm in detail.

Parsing phrases

First, we tokenize an input command into constituent words. For each word, we determine its part-of-speech tag using a customized Brill transformation-based tagger [6]. In our current implementation, we use a two-level tag hierarchy for each input command. The lower portion of the tag hierarchy consists of Penn Treebank tags [16], and the upper level consists of verb, noun, or adjective expressions formed by groups of tags for each phrase element.

Phrase Level At the phrase-level, an input phrase is tagged into broader "phrase expressions." For example, the sentence "make the shadows on the left slightly brighter" is tagged for Verb, Noun, and Adjective Expressions (VX, NX, and AX respectively) as:

$$\textit{make} \rightarrow \text{VX} \mid \textit{the shadows on the left} \rightarrow \text{NX} \mid \textit{slightly brighter} \rightarrow \text{AX}$$

Word Level At the word-level, each word within a sentence expression is further tagged into more granular parts-of-speech components. Using the same example sentence, its adjective expression "slightly brighter" is tagged as:

slightly → Adverb (VB) | *brighter* → Adjective (JJ)

To summarize, our phrase-level tags identify potentially complex subject, predicate, and objects, while word-level parts of speech tags are needed to differentiate the types of words that make up a phrase element. Our phrase templates work at the phrase-level level.

Modeling an image processing request

Next, we define a model for how an input phrase is mapped into a valid image processing request. Based on data gathered from our initial user study and crowdsourcing experiments, we segment the user's input phrase into three main components: Image Operations, Masks, and Parameters.

An *Image Operation* is the effect or filter that a user wants to apply to the image. (i.e., "brighten", "blur", "warm"). A *Mask* defines the target of the effect. This could either be global (i.e., the entire image), within a selected region (i.e., through pointing or scribbling), within a geometric area (i.e., top, bottom, upper-right), within a specified tonal region (i.e., shadows, midtones, highlights), within a particular range of colors (i.e., red, blue, "the orange parts"), or through object references (i.e., the background, the sky, or a person such as Sara). Finally, the user specifies *Parameters* which indicates the degree to which the user wants the effect applied (i.e., more, less, slightly, decrease).

Matching with phrase templates

Once an input phrase is categorized into parts-of-speech components, we use its phrase-level parts-of-speech tags to find any matching *phrase templates*. Phrase templates are simple "rulesets" that map words and phrases into the different components of an image processing request. Here is the phrase template mapping for the example above:

Production Rule	*Example*
Match for pattern → "VX NX AX"	see example above
Adjective in AX → *image operation*	"brighter" → BRIGHTEN
Nouns in NX → *masks*	"shadows" & "left" → SHADOW & LEFT
Adverbs → *parameters*	"slightly" → SLIGHT
Require → *presence of JJ*	"brighter" → *true*

In our current implementation of PIXELTONE, we use 12 phrase templates, which were manually defined in code.

Mapping unknown words to known operations

A general problem is handling speech input containing words outside our core vocabulary of operations, nouns, and adjectives. We therefore map unknown words to the set of known operations that PIXELTONE supports, which allows the system to provide reasonable responses in the presence of unknown words and allows us to extend the vocabulary.

We use lexicon ontologies such as Wordnet for term disambiguation similar to SenseRelate [22], based on earlier work by Banerjee et. al. [4]. We developed a modified shortest path distance function to compute the semantic similarity between two or more terms with ties resolved by closeness of the nearest common hypernym. Using noun, verb, adjective, and adverb lexicon ontologies, we compute the shortest path distance within the ontology between any two terms. Terms that are synonyms in the same synset ring (semantically equivalent terms) have path distance zero; non-synonym terms that have the same hypernym parent have distance 2. Terms further away within the ontology will have higher path distances.

Adding interaction

PIXELTONE uses a canvas layer overlaid on top of the displayed image to allow users to make arbitrary selections for localizing effects. For example, a user can draw arbitrary scribbles in the image, and PIXELTONE applies an effect only to that particular selection. In most cases, users want to adjust the "degree" of a particular effect. PIXELTONE provides sliders to fine-tune an effect. Sliders prevent the user from repeating particular words like "more, more" or "less, less" to tweak an effect.

Implicit tagging

PIXELTONE uses object detection to find an object within an image. For example, a user can point to a person in an image and say "This is Kurt." In our current prototype, we use face detection (using the iOS face detection API) to identify faces within an image, and we combine this information with the point location to resolve the face whose bounding box falls within the neighborhood of the pointed region. The bounding boxes of the objects are used as masks, and the captured tag (e.g., Kurt) is used as the identifier for the reference.

PIXELTONE also makes it possible to store arbitrary selections within the image. For example, a user can draw a selection on a specific part of the image and store that selection by saying "This is the background."

When PIXELTONE detects a previously named reference from an input phrase, it retrieves the stored selection and applies the effect to that region. For example, a user can say "brighten Kurt" or "blur the background" and PIXELTONE applies the particular effects within the region specified by those tags.

Image operation execution

Finally, the *execution engine* component processes the interpreted command and combines it with users' direct-manipulation input to execute an image processing operation. If a mask is specified, the execution engine localizes the operation only to that region. In addition, the execution engine blends multiple masks in cases where more than one mask is specified (e.g., "darken the midtones on the upper right," or "make Sara and John brighter").

Implementation

We developed PIXELTONE as an iPad application running on iOS 6. PIXELTONE supports 21 image processing operations. Fourteen of the operations were provided by the iOS core image library, and seven were newly implemented by us.

USER EVALUATION

We designed our evaluation to answer two questions: What is the qualitative experience of using PIXELTONE? And how does our proposed multimodal interface compare with a traditional image editing interface?

Methodology

We compared two interfaces: PIXELTONE, and PIXELTONE without the speech interface.

Participants. We recruited 14 users (8 female) from an open e-mail list at a large public university. The participants had diverse experiences with photo-editing, and six users self-reported having novice experience with photo-editing software. In addition, the age of the participants ranged from 19 to 47, with eight participants between the age of 22 to 34. Four users were non-native English speakers.

Training. At the beginning of the study, we trained our participants on how to use our prototype. We read a written script, and walked them through relevant features of the system, including the speech interface and the gallery mode. Afterwards, we allowed participants to interact with the prototype. During that period, we also gave them a hand-out that showed examples they could try.

Tasks. We asked participants to complete 16 tasks, which were segmented into two parts (8 tasks per part). In each part, we randomly assigned either PIXELTONE or PIXELTONE without voice. Similar to our pilot lab study, we gave participants two types of tasks: 1) a before-after image transformation task, and 2) an open-ended image improvement task. To decrease learning effects, we gradually increased the tasks' level of difficulty. At the beginning of each task, we also asked users for a high-level description of their intended actions. The images were counterbalanced across interfaces and tasks.

Success Scoring. We defined task success using a scale of 1 to 5 (5 as highest). We designed a method to assign a success score for each task. For example, a user who applied "hue" for a task that involved improving an underexposed image got a score of 3, while a user who applied "brightness" for the same task got a score of 5. To normalize expertise effects, we gave users one minute to do whatever they wished to accomplish for each task. After one minute, everyone received a "guide sheet" that showed them the operations for completing the task. For example, the guide sheet for a task could include the operations "brightness", "sharpen", and "vibrance."

Debrief. We debriefed the participants after they completed their tasks. We asked them for their overall impressions, the differences they encountered between the two types of interactions, and their overall feedback. In addition, we also asked the participants to complete a short questionnaire.

Results

Quantitative Results

Success rate for both interfaces were identical. We did not observe a significant difference in success rates between the two interfaces. The average success rate for multi-modal interface was 4.37 (SD=0.31) vs. 4.32 (SD=0.45) for the no-speech interface. The success rate of novice users with the multi-modal interface was slightly higher (4.38, SD=0.18) in comparison to the no-speech interface (4.08, SD=0.55), but this difference was not statistically significant. The success rate of experienced users was similar with both interfaces: 4.43 (multimodal), SD=0.20 vs. 4.48 (no speech), SD=0.16. Since the baseline interface already allowed a high success rate, the additional speech modality was not able to provide a significant improvement in rates of success.

Users preferred the multimodal interface. Consistent with previous findings [20], users rated the multi-modal interface as the one that they liked more (4.36, SD=0.50) compared to the interface without speech (3.64, SD=0.63).

Number and complexity of utterances varied from user to user. In one extreme, we observed a couple of users who successfully completed tasks using the speech interface alone, whether or not the gallery mode was available to them. In the other extreme, we also observed some users who rarely used the speech interface. However, we observed that the percentage of image operations invoked using the speech interface was higher among novices (76% of operations triggered by the speech interface) than with users having advanced image-editing experiences (47%). Similarly, we also found that native English speakers used the speech interface more often (67% vs. 45%).

We approximate the complexity of utterances using the mean length of utterances (MLU) measure [20] (Figure 5b). From a total of 386 utterances, the number of words used ranged from 1 to 6, and the average MLU across 14 users was 1.89 words per utterance (SD=1.10). Novices averaged 1.83 words per utterance (SD=1.14), compared to 1.94 words per utterance among experts (SD=1.07). Similarly, native English speakers average 2.04 words per utterance (SD=1.17) compared to 1.61 words per utterance for non-native speakers (SD=0.89). This finding among native english speakers vs. non-native speakers was statistically significant ($p<0.001$), which we measured using a standard t-test.

Speech engine implementation obtained high accuracy. We transcribed all user utterances and compared them with PIXELTONE's speech-to-text output (Figure 5a). The accuracy was calculated by comparing the Levenshtein distance (expressed as a percentage) between PIXELTONE's speech-to-text output and the actual transcribed utterance. From a total of 386 utterances across 14 users, our two-tiered speech recognition implementation obtained an average accuracy of 84% (SD=29%). For native English speakers, we obtained an accuracy of 90% (SD=20%) compared to 70% for non-native speakers (SD=38%) This finding was statistically significant ($p<0.001$) using a t-test. In addition, PIXELTONE was 84% accurate for more experienced users (SD=27%) compared to 82% for novices (SD=31%).

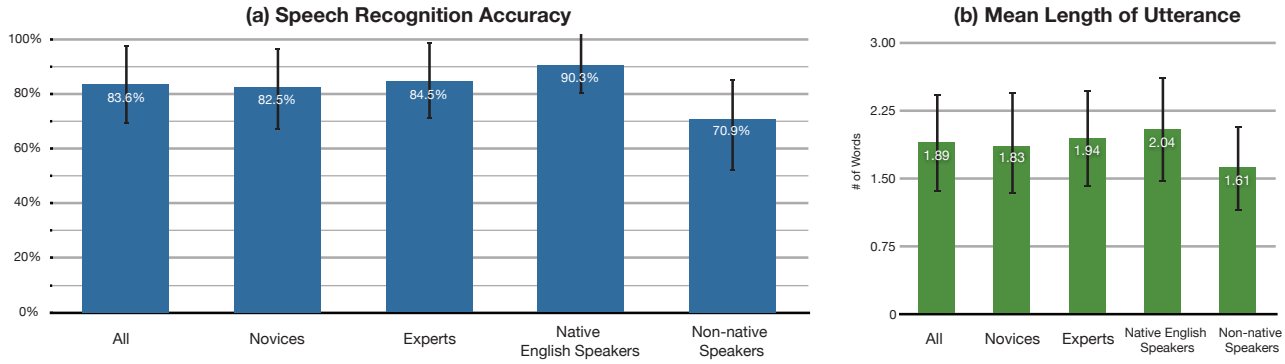

Figure 5. Speech recognition accuracy (a) and mean word length of utterance (b) from a total of 386 utterances across 14 users.

Qualitative Results

Our participants were overall very positive. Almost all users (13/14) said they would recommend PIXELTONE to their friends. Participants found the combination of speech, gallery-mode, sliders, and sketched-based selections easy to use. One user (U9) said *"It's possible that my mom and grandma would be able to use this."* Another user (U6) was impressed about the performance of the speech interface: *"Initially I had doubts about the speech interface because my experience tells me that speech doesn't really work in other apps. But this one, it worked well. I think it's better than Siri."* 13 out of 14 participants said they preferred the multi-modal interface because it allowed them to accomplish tasks in a way that met their needs. One user (U4) mentioned that the multi-modal interface allowed choosing the best approach for a particular task: *"For the tasks that were more complicated, it was easier to pick out using the gallery menu. For simple tasks like brighten, voice makes it easy."*

Users use the speech interface when they have a good idea of what they want to do. The participants mentioned that they used the speech interface to do simple things very quickly. In fact, the majority of the users (13/14) mentioned that when they knew exactly what they wanted to do, they used speech as their first choice because it saved them time.

Users use the gallery mode when they want to explore options and compare different effects. When users were not exactly sure what to do, they often used the gallery mode as an exploration tool. Users mentioned that unlike the speech interaction, the gallery mode allowed them to have a visual representation of the effects, which they said was particularly useful when comparing options to accomplish a task.

Users use direct manipulation to fine-tune and explore. Thirteen out of fourteen users found the sketch-based selection particularly useful. They frequently used it for fine-tuning the edits they made on their images. One user (U2) said *"even without the speech, it was fun just to do the scribbles."* However, users also mentioned that sometimes they unknowingly "tap" the screen and make small scribbles, unintentionally localizing effects to undetected areas. To mitigate this issue, the system should provide more feedback on whether a local mask was applied to the image. Users also mentioned that they wanted to have control on the precision of selections. In addition, we observed that users utilized the sliders to tweak effects. The sliders also served as a feedback mechanism for exploring the capabilities of the system (e.g., moving the slider swiftly back-and-forth between extremes).

Non-native English speakers with accents used speech interaction much less. Non-native English participants often had a preconceived belief that their accent would impede their performance with the speech interface. Thus, they tended to use the speech interface less frequently. Moreover, they also reported being more self-conscious about their pronunciations, and the presence of a native English speaker (the user-study facilitator in this case) could have limited their inclination to use the speech interface.

DISCUSSION

While we are glad that users found PIXELTONE effective and useful, there are many ways we can improve it in future work.

Speech Recognition

To have a more robust and reliable speech interface, the speech recognition system must account for individual differences, including nuances in diction and pronunciation. It should also consider the diversity of non-native English speakers. Our studies suggest that for most users the speech interface must work accurately in the first three or four tries or it will impact their perception of its reliability and interfere with future use. Supporting non-native English accents was out of the scope of this project, and we leave the exploration of these techniques to future research in the speech recognition domain. However, it may be worthwhile to explore how commercial speech recognition systems such as Dragon [19] and Siri [3] account for the diversity of its users. In addition, future work should explore how to create a feedback mechanism to automatically train PIXELTONE to learn from the vocabulary and nuanced speaking styles of users.

Interpretation

In its current form our prototype has a limited vocabulary (both for commands and nouns such as the names of individuals for tagging). We mitigate this in part by using Wordnet for synonyms and disambiguation as well as mining the user's contact list to find names. Nonetheless, PIXELTONE does make mistakes and in future work we would like to create better mechanisms for improving the vocabulary (on the

back end) and better interactions (on the front end) to communicate uncertainty and corrections.

The current prototype performs well with simple commands (e.g., "make this image brighter"), and a limited set of high-level commands that require several image editing operations in a row (e.g.,"make this retro," which makes a mask, applies a vignette, and adjusts the hues). These high level commands are manually created and PIXELTONE may be better served by learning from examples [10] or mining online tutorials.

We have only begun to explore the different ways in which people use natural language in the context of photo editing. From our findings, we identified two general types of sentences that users' invoke: 1) imperative forms, and 2) declarative forms. Most of the phrase templates we designed were in the imperative form since they tend to be more common. However, the image-editing domain affords the use of declarative sentences (e.g., "the image is too bright"), and users may learn to use this form as they become more comfortable. Unlike imperative sentences, some ambiguity is introduced when interpreting declarative sentences. For example, the sentence "the image needs to be bright" might denote brightening the image (positive), while "the image is too bright" might denote darkening the image (negative).

Gestures

Our focus thus far has been mostly on direct manipulation, but we believe that adding gestures could bring further richness to the interaction. We have started experimenting with some gestures for cropping and specifying gradients. To let the user easily crop an image, PIXELTONE can use the stored touch points to determine the "bounding box" of an arbitrary selection. And to specify gradients, the user can draw a line. It remains future work to explore which types of gestures make sense in this context and whether users would be interested in learning them.

Additionally, PIXELTONE asks the user to use the different interface modalities one at a time (selection first, speech second) and does not integrate them. This means that the user can't sketch and talk at the same time. While this seems sufficient for now, a future goal would be to support integration (e.g., [18]) to offer a more flexible interface.

Scalability

Our current implementation supports 12 phrase templates that were manually written in code. A full implementation might require defining more phrase templates to allow users to fully express their intentions. Since these templates are based on grammatical structures of sentences, it is possible to semi-automate the process of defining templates by analyzing grammatical patterns on large corpora of users' utterances. Alternatively, users can train the system directly by verbalizing what they are doing as they are doing [10].

Currently, PIXELTONE supports a limited number of image processing operations. We picked them based on the most commonly used operations, similar to those supported by consumer-type photo editing applications such as iPhoto [2]. Extending the number of supported operations would require some effort on the computational side (e.g., writing code), and minor effort on the linguistic side (e.g., defining a set of keywords that represents an operation). Two future enhancements are possible in this area: supervised learning of additional image editing operation vocabulary from tutorials, and assigning correct word sense to the learned vocabulary words to improve disambiguation of unknown words.

Finally, we chose the tablet as our initial platform for PIXELTONE for practical reasons (e.g., availability of speech libraries, easier to transport a mobile device for immediate testing). It would be worthwhile to explore how our findings can translate to other platforms such as desktop computers.

CONCLUSIONS

In this paper we introduce PIXELTONE, a multimodal system to support image editing tasks through speech and direct manipulation. The system is motivated by a number of formative studies on how both professional and novice users make use of natural language and annotation to indicate the changes they would like to make. We found that the multimodal interface more naturally captures both existing work practice and desired functionality.

Image editing is an incredibly difficult task, and the shift to mobile (i.e., small-screen) devices makes this task harder. Additionally, the language of image manipulation is varied, ambiguous, and subjective. By interpreting speech through a combination of local and remote speech recognition and customized natural language parsing, PIXELTONE provides users with a powerful mechanism to obtain desired results. While we identified future improvements, our user study found that users preferred the multimodal interface overall and were able to use it effectively for a realistic workload.

ACKNOWLEDGMENTS

We would like to thank our reviewers and study participants for their time and helpful feedback.

REFERENCES

1. Androutsopoulos, L. Natural language interfaces to databases - an introduction. *Journal of Natural Language Engineering 1* (1995), 29–81.
2. Apple. iPhoto. `http://www.apple.com/ilife/iphoto/`. Online; accessed Sept 18, 2012.
3. Apple. Siri. `http://www.apple.com/ios/siri/`. Online; accessed Sept 18, 2012.
4. Banerjee, S., and Pedersen, T. An adapted lesk algorithm for word sense disambiguation using wordnet. In *Proc. of the 3rd International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing '02, Springer-Verlag (London, UK, 2002), 136–145.
5. Bolt, R. A. Put-that-there: Voice and gesture at the graphics interface. *SIGGRAPH Computer Graphics 14*, 3 (July 1980), 262–270.
6. Brill, E. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics 21*, 4 (June 1995), 543–565.

7. Chang, J. C., Lien, A., Lathrop, B., and Hees, H. Usability evaluation of a volkswagen group in-vehicle speech system. In *Proc. of the 1st International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, AutomotiveUI '09 (2009), 137–144.
8. Coelho, J., Duarte, C., Biswas, P., and Langdon, P. Developing accessible tv applications. In *The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility*, ASSETS '11 (2011), 131–138.
9. Gelfand, N., Adams, A., Park, S. H., and Pulli, K. Multi-exposure imaging on mobile devices. In *Proceedings of the international conference on Multimedia*, MM '10 (2010), 823–826.
10. Gorniak, P., and Roy, D. Augmenting user interfaces with adaptive speech commands. In *Proc. of the 5th international conference on Multimodal interfaces*, ICMI '03 (2003), 176–179.
11. Hauptmann, A. G. Speech and gestures for graphic image manipulation. In *Proc. of the SIGCHI conference on Human factors in computing systems*, CHI '89 (1989), 241–245.
12. Healey, P. G. T., McCabe, R., and Katagiri, Y. A comparison of graphics and speech in a task-oriented interaction. In *Proc. of the 1st International Conference on Theory and Application of Diagrams*, Diagrams '00, Springer-Verlag (London, UK, 2000), 245–256.
13. iSpeech. iSpeech. `http://www.ispeech.org/api`. Online; accessed Sept 18, 2012.
14. Lau, T., Cerruti, J., Manzato, G., Bengualid, M., Bigham, J. P., and Nichols, J. A conversational interface to web automation. In *Proc. of the 23rd annual ACM symposium on User interface software and technology*, UIST '10 (2010), 229–238.
15. Little, G., and Miller, R. C. Translating keyword commands into executable code. In *Proc. of the 19th annual ACM symposium on User interface software and technology*, UIST '06 (2006), 135–144.
16. Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. Building a large annotated corpus of english: The penn treebank. vol. 19 (June 1993), 313–330.
17. Miller, R. C., Chou, V. H., Bernstein, M., Little, G., Van Kleek, M., Karger, D., and schraefel, m. Inky: a sloppy command line for the web with rich visual feedback. In *Proc. of the 21st annual ACM symposium on User interface software and technology*, UIST '08 (2008), 131–140.
18. Milota, A. D. Modality fusion for graphic design applications. In *Proc. of the 6th international conference on Multimodal interfaces*, ICMI '04 (2004), 167–174.
19. Nuance Communications. Dragon. `http://www.nuance.com/dragon/index.htm`. Online; accessed Sept 18, 2012.
20. Oviatt, S. Multimodal interactive maps: designing for human performance. *Human Computer Interaction 12*, 1 (Mar. 1997), 93–129.
21. Pajari, M. Color mark up terminology. Tech. rep., Widen Enterprises Inc., 2009.
22. Patwardhan, S., Banerjee, S., and Pedersen, T. Using measures of semantic relatedness for word sense disambiguation. In *Proc. of the 4th international conference on Computational linguistics and intelligent text processing*, CICLing'03, Springer-Verlag (2003), 241–257.
23. Pausch, R., and Leatherby, J. H. An empirical study: Adding voice input to a graphical editor. *J. of the American Voice Input/Output Society 9* (1991), 2–55.
24. Politepix. OpenEars: speech recognition and speech synthesis for the iPhone. `http://www.politepix.com/openears/`. Online; accessed Sept 18, 2012.
25. Riley, E. How to talk to a retoucher, 2012.
26. Samad, T., and Director, S. W. Towards a natural language interface for cad. In *Proc. of the 22nd ACM/IEEE Design Automation Conference*, ACM/IEEE DAC '85 (1985), 2–8.
27. Sedivy, J., and Johnson, H. Supporting creative work tasks: the potential of multimodal tools to support sketching. In *Proc. of the 3rd conference on Creativity & cognition*, C&C '99 (1999), 42–49.
28. Woolfe, G. Making color adjustment accessible to non-experts through the use of language. In *Proc. of IS&T 15th Color Imaging Conference* (2007).
29. Xiong, Y., and Pulli, K. Gradient domain image blending and implementation on mobile devices. In *Mobile Computing, Applications, and Services*. Springer Berlin Heidelberg, 2010, 293–306.
30. Zhao, Y., Bala, R., Braun, K. M., Langford, Z., Rolleston, R. J., and Stevens, M. T. Language-based color editing for mobile device. In *Proc. of the SPIE, Volume 7879* (2011).