PhotoSpread: A Spreadsheet for Managing Photos

Sean Kandel, Eric Abelson, Hector Garcia-Molina, Andreas Paepcke, Martin Theobald
Stanford University

ABSTRACT
PhotoSpread is a spreadsheet system for organizing and analyzing photo collections. It extends the current spreadsheet paradigm in two ways: (a) PhotoSpread accommodates sets of objects (e.g., photos) annotated with tags (attribute-value pairs). Formulas can manipulate object sets and refer to tags. (b) Photos can be reorganized (tags and location changed) by drag-and-drop operations on the spreadsheet. The PhotoSpread design was driven by the needs of field biologists who have large collections of annotated photos. The paper describes the PhotoSpread functionality and the design choices made.

Author Keywords
Photo browsing and analysis, spreadsheet.

ACM Classification Keywords
H.5.1 Multimedia Information Systems; H.5.2 User Interfaces; H.2.3 Languages.

INTRODUCTION
With the proliferation of digital cameras and scanners, vast collections of digital images have become common, and effectively *analyzing* such collections and their associated metadata has become critical. For example, we are working with biologists who use outdoor camera traps (remotely triggered photographic equipment) to generate thousands of animal photographs in the Jasper Ridge Nature Preserve. Each photo is automatically tagged with the date, the temperature, the location of the trap, and other metadata. In addition, biologists manually add tags describing the animal species, the identity of the individual animal (if known), and other facts. Biologists analyze these photos to discover trends and anomalies. For instance, they need to select groups of photos by their characteristics; groups of photos must be compared side-by-side; the scientists need to compute various statistics for particular sets of photos (e.g., average temperature), and so on. In addition, the biologists need to continuously edit the metadata, to correct errors or to enter additional facts they discover related to given photos.

Consider the following actual analysis as an example for what our current target user needs to produce. A biologist is studying the interaction between deer (*Odocoileus hemionus*) in the nature preserve, and its main predator, the mountain lion (*Puma concolor*). The biologist notices that in some photos the deer are bolting away from the camera, and wants to know why. A student has previously tagged each photo with the species of the photographically captured animal.

The scientist now needs to assemble all the photos that show deer, and must add a "bolting" tag where appropriate. To test for a possible correlation between mountain lions and bolting deer, the biologist needs to determine the mean, median, mode and range of times when deer were skittish. Congruence of these times with mountain lions' dawn and evening hunting hours would demonstrate correlation. To see if temperature impacts behavior as well, the biologist additionally needs to examine temperatures at the time photos were captured.

An alternative hypothesis may be that deer are most skittish during the new moon, because the associated darkness makes predators hard to see. To verify, the biologist needs to divide the number of images that occur during a new (or 1st/3rd quarter) moon by the total number of images. After completing this skittish deer analysis, the biologist may need to replicate the process on other species, such as raccoons or bobcats.

Similar photo analysis needs arise in other domains. For example, a journalist needing photos for an article, may need to examine, filter or group photos of relevant events. An astronomer looking for patterns may need to examine large numbers of photos related to a particular area of the sky. Museums have digitized significant parts of their collections, both for preservation and for wider dissemination. These vast digital archives must be curated, which requires analysis and organization of materials. Last but not least, an amateur photographer may also want to organize and analyze his travel or family photos.

Figure 1. PhotoSpread Interface (image retouched for clarity).

Fortunately, as our needs for photo analysis grow, computer display sizes have grown as well, either as individual, or multi-displays. This gain in screen real-estate has introduced new opportunities for coordinated, side-by-side photo set viewing. However, photographers with large collections of data-rich photos, like our biologist colleagues, currently use three types of tools that are not well integrated with each other: Photo browsers display images, spreadsheets afford computation over metadata, and databases (like MS Access) store the metadata. We will argue in this paper that no tool by itself fully addresses the increasing analysis needs.

To address this lack of appropriate software, we designed and implemented PhotoSpread, a spreadsheet tool where photos and groups of photos are first class citizens. To illustrate, Figure 1 shows a typical PhotoSpread sheet. (The video that accompanies this paper shows this example in action, and illustrates other features.) Like any spreadsheet tool, PhotoSpread displays an array of cells, but in addition to atomic data values, like strings and integers, groups of photos can be displayed within any cell. For example, the `load` command populated cell `A1` with a set of photos from an external source. These photos are from camera traps at Jasper Ridge Nature Preserve. PhotoSpread decides how best to utilize the given real estate in the cell to display all or a subset of the photos. Users can change cell dimensions by dragging the grid lines. The size and number of the photos in the cells are automatically adjusted.

Anchored to the right of the spreadsheet grid is a *workarea*. This "big cell" shows the contents of the currently active cell—in this case `C3`—for better viewing. The metadata for a particular photo can be viewed via the blue rollover tooltip window, and can be edited in the workarea (not shown in this figure). If the current cell contains a formula, the formula is shown, and can be edited, in the formula window at the top of the workarea. Keep in mind that Figure 1 shows our interface in a small area. On a large display, the photos, especially those in the workarea, can be much better appreciated.

Say we want to organize these photos along two dimensions: the species and the age of each animal. To keep this example simple, we limit ourselves to two species, `bobcat` and `deer`, and two age groups, `adult` and `juvenile`. Figure 1 shows how we have arranged these values into a grid.

Cell `C3` corresponds to adult (`B3`) bobcats (`C2`), so we enter the formula `=A1[species = C2 & age = B3]`. (As in Excel, dollar signs `$` could be added to control how a formula is copied into other cells.) As soon as the formula is entered, the appropriate photos from `A1` are displayed in `C3`. When we copy the formula into the other three grid cells, we populate our display as desired. Our biologist collaborators tell us that this type of photo array (concurrently displaying the results of multiple queries) is critical for visually identifying patterns or anomalies, and is not available in any tools they use.

Cells `E3` and `E4` show the average temperature across the adult and juvenile rows. For instance, the average for adults (in `E3`) is computed as `=average(C3:D3.temp)`. Here, `C3:D3` is a range of cells containing the adult photos, and the `.temp` component extracts their temperature tag. As mentioned earlier, such metadata com-

putations are important for biologists as they analyze their photos.

In addition to using formulas to organize and view photos and other data, PhotoSpread introduces a novel way of entering or updating metadata. Say for example that we see in our grid a photo that is misclassified, e.g., the photo appears in the cell for adult deer, but it is really an adult bobcat. We can simply drag the photo from cell `D3` to the cell for adult bobcats `C3`, and this action *forces* the metadata to change, so that the photo complies with the formula of its new cell. Later in this paper we present other semantics for dragging, so that a new copy of the photo is made, or so that the metadata is not changed but instead the underlying formula of the receiving cell is automatically changed to cover the new photo. In addition, strings can be moved to cells with photos (or vice versa) to tag photos. All these simple and intuitive drag-and-drop actions give the user a number of options for adding or changing the metadata of photos, as the photos are being analyzed and studied. Again, our biologist collaborators tell us that this type of re-tagging is very common, and that tagging by moving photos is preferable to using traditional editors.

In summary, PhotoSpread introduces two important extensions to common spreadsheet systems:

- *Tagged sets are first class objects.* Sets of objects, and in particular, sets of photos, can reside and be displayed in any cell. The objects—photos and other data types—can be conveniently tagged. A powerful formula language can select and manipulate objects, referring to tags in expressions. The language builds on widespread knowledge of standard spreadsheet formulas.

- *Drag-and-Drop (Re)Organization.* Simple actions on photos and strings change metadata and formulas. Tagging photos through direct drag/drop manipulation has been found to be effective for tagging data [11], and PhotoSpread allows the user to configure such manipulations to his needs.

PhotoSpread has been implemented as both a standalone and Web application using Flex. In the current implementation, photos and other objects are stored in memory. However, we are developing a storage layer that uses a backend database for storage and query processing. All the PhotoSpread features described in this paper have been implemented.

While the extensions for tagged sets and drag-and-drop organization are very natural, we had to address a number of challenges to ensure a usable and effective realization:

- *Data Model and Formula Language.* Should the system support explicit photo "containers" (work areas that hold photos, like a directory does in a conventional system) or can we use the spreadsheet cells as containers? Can we extend our notion of "photo sets" to sets of strings, tags, numbers, etc.? Such an extension would enable us to extract all the tags in a particular photo set, and then assign these tags to another set. What formula language should we define, so that Excel users feel at ease, but can have access to the new set and tag features? We answer these questions in the Model section below, where we present our data model and associated formula language.

- *Drag-and-Drop (Re)Organization.* There are two fundamental types of sets of objects: materialized sets (containers) and virtual sets defined via formulas. The semantics of moving (or copying) a photo or other object from/to a set depend on the type of set. For example, moving a photo between two materialized sets is just like moving a photo from one physical pile to another. But what is the meaning of moving a photo to or from a virtual set (i.e. formula cell)? Does copying a photo create a new instance or just a reference to the original photo? In the section on Reorganization, we explore the options, and determine what choices give the user a powerful way to reorganize photos.

- *Interface Issues.* In the section on the Interface Design, we discuss some of the interface design challenges we faced. For example, what is the best way to use the large workspace area? Is this area a special type of cell (or container) or is it simply a window into an active spreadsheet cell? What is the best way to display photos in the limited-size cells? If all the photos do not fit in a cell display area, which ones do we show? How does one select multiple photos for a set operation?

After describing our design following the outline above, we conclude by comparing PhotoSpread to existing solutions and by discussing how it is currently being used by our biologist colleagues.

BASIC MODEL AND FORMULAS

In this section we describe the underlying PhotoSpread model for the spreadsheet, its objects, and their annotations. Our general philosophy is to use Excel as a starting point. We chose this approach not because Excel is necessarily the best available spreadsheet application, but because it is the most popular package, and well known in particular to our current target audience. We extend Excel to handle and display in a cell a *set of objects*, and in particular a set of photos annotated with tags. The full details of our model and formula language are presented in [12]; here we only summarize the key points, mainly using examples.

As in Excel, our spreadsheet is a two-dimensional array of *cells*, where each cell is referred to by its column and row identification. For example, cell `C2` is at column `C` and row `2`. Each cell has a *display*, i.e. an associated rectangular "screen canvas" where its "contents" can be displayed. Each cell `Xi` can be in one of two states:

- Cell Xi can be a *container* holding a set of materialized objects, i.e., objects that are stored in the PhotoSpread database. Excel, by comparison, can only store a single object in a cell. Objects can be moved in and out of containers, just as if containers were physical piles of objects. Cell Xi's canvas shows some or all of its contained contents.

- Cell Xi can instead be a *formula* defining a virtual set of objects. Cell Xi's display can show either the formula (when Xi is active) or some of the objects that are selected by the formula.

Initially, all cells are empty containers. A cell can be transformed into a formula cell by typing a formula into the formula editor. As in Excel, cell contents are lost when a new formula is entered, although the actual objects are retained in the PhotoSpread database.

In the current system objects can be integers, decimal numbers, dates, strings, or they can be photos. We represent each object by an unordered set of *tags*, where each tag is an attribute–value pair. For example, a particular photo may have tags `date: 1/1/08`, `location: Paris`, `VALUE <bits>`, `ID: 12345`. All objects have a `VALUE` tag that holds the physical object representation, and an internal `ID` tag that uniquely identifies the object in the PhotoSpread database. In the future we are planning to add a `PROVENANCE` tag that indicates the origins of a photo.

Note that non-photo objects are also represented by a set of tags. For example, the number "123" may be represented by `size: small`, `VALUE: 123`, `ID: 23456`. Here `size` is a user-defined tag. We will see that handling all objects in this flexible and uniform fashion gives us significant power in selecting, grouping and organizing not just photos but any values in PhotoSpread.

Our model allows multiple tags with the same attribute name, as well as multiple tag values. For example, a photo with tags `species: squirrel` and `species: fox; cat` shows a squirrel and either a fox or a cat.

One interesting question that arose in our design is how to handle object copies. In particular, say object `O` is in container cell `C1` and we make a copy into cell `D2` (see Reorganization section for the respective interaction details). Is the object in `D2` a different object than `O`, with a different `ID` and the same `VALUE`, or is it the same `O`? That is, should copying be executed by-value or by-reference? We decided to go with the former. Allowing an object to be in more than one container would save storage but would complicate our system, as containers would have to hold indirect references to objects. Indirect references also complicate formula processing, as discussed below. Thus, PhotoSpread handles object as unique entities, even if they were generated from a copy operation.

A formula defines a set of objects based on the contents of other cells. Within a formula we refer to other cells by their names, e.g., `C2` or `D5`. We can also refer to ranges of cells, e.g., `B2:C3` refers to the union of the contents of cells `B2`, `B3`, `C2`, `C3`. As in Excel, PhotoSpread allows dollar signs `$` in cell references to control how equations are copied. For example, if a formula with reference `A$2` is copied to a cell that is three horizontal units over and two vertical units down, the reference becomes `D$2`. The vertical coordinate 2 is not shifted because of the dollar sign.

PhotoSpread provides three set operators in formulas, which we illustrate with examples. `union(C1, D3:D5)` represents the objects that are either in `C1` or in `D3` or in `D4` or in `D5`. `intersect(C1, D3, F2)` represents the objects that appear in all three cells `C1`, `D3`, and `F2`. `minus(C1, D3)` represents the objects that are in `C1` but not in `D3`.

Filter expressions let us select particular objects from a set. For example, `C1[species=fox & date < 1/1/08]` selects the objects in `C1` with a `species:fox` tag and a date tag valued earlier than 1/1/08. Note that quotes around string values are not required, because the underlying computational engine can determine this data type by context. To select `C1` photos that show both a fox and a deer, we can write either `intersect(C1[species=fox], C1[species=deer])` or `C1[species=fox & species=deer]`.

Note that filter expressions may include sets. For example, say `C1` contains two string objects, one with `VALUE: fox`, and another with `VALUE: deer`. The formula `D1[species=C1]` is equivalent to `D1[species={fox, deer}`, and selects photos with either deer or foxes. As illustrated in the Introduction, the power to refer to cells with strings and other data types makes it possible to display on the spreadsheet the values that are used (via indirection) for filtering.

A value selection expression lets us extract values from the tags on an object. For example, `C1.location` returns the set of values associated with the attribute `location` of any object in `C1`. Similarly, `C1[species=fox].location` returns locations of any fox objects in `C1`.

Finally, PhotoSpread allows aggregation operators like `maximum`, `minimum`, `sum`, `average` and `count`. For instance, `count(C1)` returns the number of objects in `C1`. For instance, `average(C1[species=fox].age)` returns the average age of foxes in `C1`.

Readers familiar with database query languages like SQL will see that PhotoSpread provides a lot of the power of such languages (we have left out some functionality such as GROUP BY and JOINS), using a formula language that is relatively simple, and most important, similar to that used by Excel. Furthermore, several of

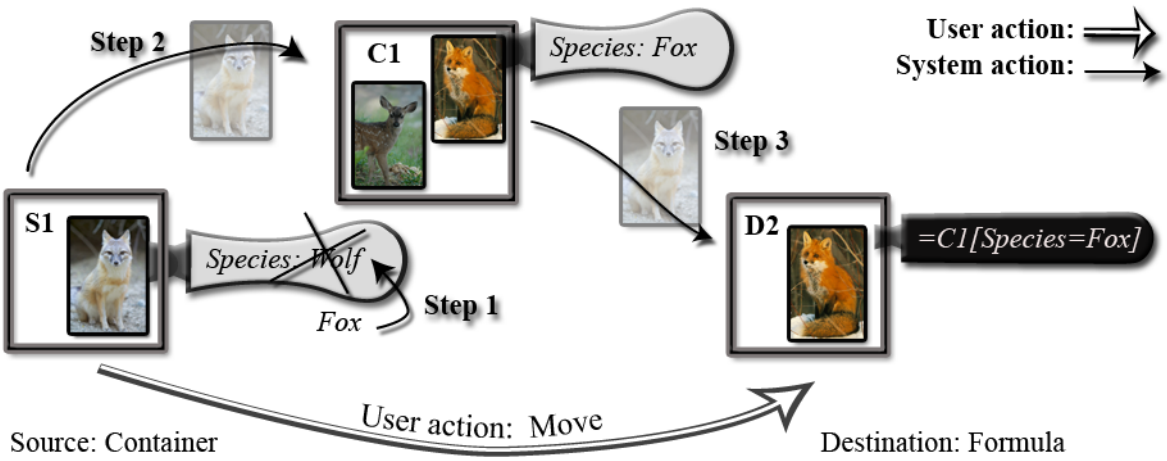

Figure 2. Example of container to formula move.

our design decisions allow the underlying computation engine to be of manageable complexity.

For instance, as we saw in an earlier example, when a formula refers to a cell that is also a formula we get a nested formula, which may then be simplified. For example, assume that `C1` is defined by `union(D1, D2)`. Then the formula `union(C1, D2)` is effectively `union(D1, D2, D2)`, which can be simplified to `union(D1, D2)`.

We can further simplify formulas when they refer to container cells, due to our 'no duplicate object' assumption. For example, if `C1` and `C2` are containers, then `intersection(C1, C2)` is empty! However, if `C2` is a formula, then we cannot tell if the result is empty. For instance, if `C2` is defined by `union(C1, D3)` and `D3` is a container, then `intersection(C1, C2)` is equivalent to container `C1`.

This example illustrates that it is convenient to expand a nested formula until all cell references are to containers, because at that point intersection and minus operations can be simplified. We call formulas that only refer to containers *base formulas*. When discussing reorganizations in the next section we will also see that it is useful to convert a formula to its equivalent base formulation.

DIRECT CONTENT (RE)ORGANIZATION

Like any spreadsheet, PhotoSpread formulas can group and filter displayed photos (and other objects). However, with PhotoSpread one can go further and (*i*) organize or reorganize photos, and (*ii*) tag photos by dragging and dropping. That is, drag/drop operations can be used to conveniently change photo metadata.

How this direct manipulation reorganization works depends on three factors:

- Whether the source/destination cell is a container or a formula;
- Whether the action is a copy or a move;
- Whether the intended semantics are force (metadata change) or not.

Due to space limitations we cannot discuss all the combinations of factors in detail. Instead we discuss a few key scenarios from which the reader can infer the other cases. For now let us assume we are moving/copying a single photo. Also note that we defer interface issues (e.g., how does one tell PhotoSpread that a drag is a copy with force semantics) to the Interface Design section that follows.

To start, let us assume that photo `X` is dragged from source cell `S1` to destination cell `D2`, and that both `S1` and `D2` are containers. In this case, copy and move work as expected, as if one were moving a physical photo from one pile of photos to another. That is, moving a photo removes it from `S1` and adds it to `D2`. Photo `X`'s metadata is not changed. Copying a photo creates a new copy at `D2`, with the same metadata (except for the `ID` tag; a new `ID` is generated). Since metadata does not change, force semantics are not applicable here.

Next, assume that source `S1` is a container, and destination `D2` is a formula. The key issue here is how cell `D2` "absorbs" `X`, driven by whether force semantics are on:

- *Force Semantics (default in PhotoSpread).* Consider a *move* action with force semantics. In this case, X's metadata is changed to satisfy the `D2` formula. For example consider the scenario of Figure 2: Cell `D2` contains the formula `C1[species=fox]`, which refers to container `C1`. Since one of the `C1` photos is a fox, this fox photo is currently displayed in `D2`. Source cell `S1` contains a misclassified photo `X`: the image is not of a wolf (as currently annotated) but of a fox. To correct the error, the user drags `X` into `D2`. What needs to change so that `X` is displayed in `D2`? Three things happen because of this move action (with force semantics): Step 1, `X`'s metadata is modified, so the

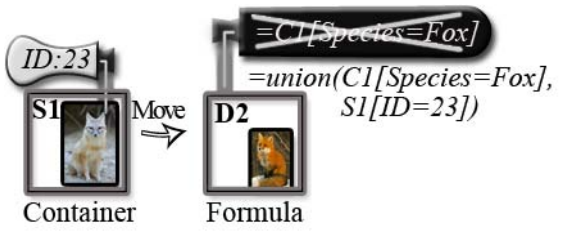

Figure 3. Example of non-force move.

`species: wolf` tag becomes `species: fox` (if no `species` tag existed, it would be added). Step 2, photo `X` is moved from `S1` to container `C1`. Note the photo is not moved to `D2`. Cell `D2` is simply a filtered view of `C1`, so if `X` is to be seen in `D2`, it belongs in `C1`. Step 3, photo `X` is now automatically displayed in `D2` (showing two foxes) since it now satisfies the formula.

In general, to determine the necessary changes we first transform a formula like `D2` to a base formula that refers only to containers (see Model section). Then we determine the changes necessary to make `X` be one of the results of `D2`. In some cases the changes are not unique. For example, say `D2 = union(C1, C2)`. To satisfy this formula, `X` can either be placed in `C1` or in `C2`. There are several options for handling non-unique transformations: (1) ask the user for clarification, (2) pick any choice, or (3) disallow such transformations. For now, in PhotoSpread we have taken option (3), but are likely to switch to option (1) because our biologist collaborators prefer that option.

The example in the Introduction showed how move with force semantics can be useful for either adding new tags to photos or re-tagging photos that were incorrectly tagged. While PhotoSpread does have a tag editor for individual photos (in the workarea), forced moves allow a much more intuitive and rapid organization of photographs.

Copying `X` from a container to a formula works as with a move, except that a new photo is created with the same tags, which are then forced as appropriate.

- *Non-Force Semantics.* When the user moves (or copies) photo `X` from container `S1` to formula destination `D2` with non-force semantics, she does not wish to change `X`'s tags. So to make `X` appear in `D2` the underlying engine changes the formula at `D2`. In particular, the engine can change the formula to `union(prev, S1[ID=Xid])`, where `Xid` is the unique identifier of photo `X`, and `prev` is the original `D2` formula. Notice that `X` remains in cell `S1` (for both move and copy actions): since the user is moving to a formula and does not want to change `X`, the engine's only alternative is to leave `X` where it is.

  Figure 3 illustrates a non-force move. Photo `X` with `ID=23` in container `S1` is moved to cell `D2` with formula `= C1[species=fox]`. The non-force move changes the `D2` formula to `=union(C1[species=fox], S1[ID=23])`, causing both fox photos to be displayed in `D2`. Photo `X` remains in `S1`.

  Non-force semantics are relevant when users want quickly to construct a complex formula without the formula editor. For example, by dragging and dropping, a user can rapidly create a virtual set of particular photos without having to write out a formula such as `union(C1[Id=123], C1[ID=234], C2[ID=345])`.

  Non-force semantics can also be helpful when a formula has exceptions. For instance, a biologist may know that all "squirrels" have a particular disease, but she also knows of two individual "gophers" that are afflicted. To view all the sick animals, she can define a formula to display photos with `species = squirrel` and then manually add photos of the two sick gophers.

The options when a user moves a photo `X` out of a *formula* source `S1` are analogous: the compute engine changes `X`'s tags (force) or the formula (non-force) so that `X` does not appear in `S1`. In this case, force semantics are probably not very useful since there are many ways to change tags such that `X` moves out of `S1`. In PhotoSpread, the default semantics when `S1` is a formula is therefore non-force.

Finally, there are two ways to generalize the functionality we have described. First, users can re-organize groups of photos by control/option selecting the target photos in the source cell and then dragging the selection as a group.

Second, we can generalize to cells that have *strings* as opposed to photos. We have analyzed all combinations of factors when strings are involved in a re-organization, and we have identified two main scenarios that are useful, and are implemented in PhotoSpread:

- String Case 1: Source container `S1` has one or more strings, which are dragged to photo destination `D2` (container or formula) with *force* semantics. For example, say the string `species: fox` is dragged to `D2`. The result is to add the tag `species: fox` to the `D2` photos if they do not have a `species` tag, or to change their `species` tag to `species: fox` if a `species` tag is present. If the copied string is simply `fox`, PhotoSpread prompts the user for the attribute to use. Note that non-force semantics are not as useful here, because they result in a cell containing a mixture of photos and tags.
- String Case 2: Photos from source `S1` are dragged onto destination `D2` containing one or more strings (with *force* semantics). The effect is analogous: the metadata of the dragged photos is changed to include the `D2` tags. The photos do not actually move, unless their new metadata causes them to move.

In summary, PhotoSpread offers a variety of methods for changing the location of photos and for tagging and retagging photos. In talking to scientists that handle large number of photos, we have found that each has their own style of working, manually or with existing

tagging tools. Some like the paradigm of moving tags onto photos, others prefer to move photos to "places" that represent their characteristics. Yet others prefer to manually enter tags using a tag editor. PhotoSpread offers all the choices within the same framework: A user that prefers to move tags, can lay out an array of tags at the bottom or top of his spreadsheet, and then move these tags onto photos or groups of photos as necessary. If a user prefers moving photos, she can set up an array of photo "piles" (defined with formulas), as we did in the example in the Introduction. At any time, the user can inspect and modify tags of individual photos by using the object editor we provide as part of the workspace.

INTERFACE ISSUES

Next, we discuss design issues for the user interface. Many of the biologists we worked with were most comfortable using Windows operating systems and Microsoft office software, especially Excel. Therefore much of our design is based on these products' interfaces.

Screen Layout

We considered many options for the main application layout. In many spreadsheet applications, by default, the entire window is used to present the spreadsheet. Our design deviates from this layout by including the separate workspace, roughly one-third the size of the window. The workspace reduces the overall size of the spreadsheet and the average size of a cell in the sheet, but allows users to enlarge the contents of a given cell. This allows biologists to focus on photos of interests while maintaining the context of working within the larger spreadsheet.

Workspace

The dedicated workspace allows users to enlarge the contents of a selected cell, such as `C1`. User actions in the workspace have the same effect as if they had been executed in the respective cell. For instance, a user can delete a `C1` image by choosing "Delete Image" from a context menu in the workspace while it is displaying the contents of `C1`. This operation has the same effect as deleting the image by interacting with cell `C1` directly. In this way, the workspace contents are always intimately tied to the spreadsheet cell that is in focus.

An alternative would have been to have the workspace function as an external library, containing objects that may or may not exists in the spreadsheet. We decided against this approach to stay as close to the spreadsheet paradigm as possible. In most spreadsheet applications, values and formulas only exist within the spreadsheet and there is no notion of an external library of values.

In addition to enlarging the contents of a cell, the workspace provides functionality to view and edit metadata, edit cell formulas, load images into cells and facilitate tagging. By rolling over an object in the workspace, users can view the object's metadata, and by right clicking the object, users can edit its metadata using an in-place editor.

Located at the top of the workspace are a text field, a drop-down menu (which in Figure 1 is set to the choice "Edit Formula"), and a "Submit" button. Using the drop-down menu, a user can choose to edit a cell's formula, load images into a cell, or tag objects that are located in a cell. The text field allows users to enter information relevant to the various actions, such as the text of a formula. Clicking the submit button executes the requested action.

To facilitate formula editing, the text field provides context sensitive auto-suggest hints, such as attribute types and function names. Any attribute type ever defined in the spreadsheet will appear as an option in the auto-suggest list.

To load images into a cell the user enters either a list of image URLs, Flickr image IDs, or Flickr set IDs into the text field. The images are retrieved from the local disk or over the Web as appropriate.

Finally, the user can enter an attribute `A` into the text field to accelerate tagging. While `A` is present in the text field, any string dropped onto selected photos in the workspace is taken to be a value assigned to the photos' `A` attribute.

For example, if a biologist wanted to add tags with attribute `Gender` and value `Male` or `Female` to photos in the workspace, she might type `Gender` into the text field, `Male` into cell `A2`, and `Female` into cell `B2`. She could then proceed screen by screen, each time selecting all males and dragging string `Male` from `A2` onto the selected photos; females would be tagged analogously.

Controlling Motion

As described in the Reorganization section, users can drag and drop objects within the spreadsheet, and the result of this action depends on the intended semantics. We considered both moded and non-moded alternatives as affordances for communicating force, or non-force intent during drag/drop interactions.

The biologists we met with often work on specific, extended tasks. They dedicate some time to tagging photos, and then later spend time querying and organizing the images. For these specific tasks, a moded interface has advantages over a modeless interface, as biologists do not have to repeatedly indicate their intentions, which remain consistent throughout a given task. When scientists switch tasks, they simply switch modes as well. A disadvantage of the moded choice would, as always, be the potential for confusion as system behavior changes in response to an action.

We opted for the non-moded choice, loosely basing our interaction design on conventions for drag/drop behav-

iors in desktop GUIs. However, in order to make our prototype accessible to both Macintosh and Windows users among our participating biologists, we avoided requirements for a two-button mouse.

If the user (left)-drag/drops a group of objects, default semantics are applied. As per the Reorganization section above, defaults are Forced Move. Control-drag/drop results in a Forced Copy operation. Finally, shift-drag/drop raises a context menu with all the available Move/Copy and Force/Non-Force options. This latter facility is similar to the right-drag/drop on Windows. During our initial, informal trials, our biologists were comfortable with these choices. In a fully developed system, separate Macintosh and Windows implementations would presumably be preferable.

We based the choice of Force for default semantics on a workflow analysis of our initial target users. Our biologists explained that much of their time focuses on tagging and editing photo metadata. Once their photos are tagged, they use queries to organize and filter their collections. They do not (currently) tend to spend time manually laying out their photos. Force semantics allows the biologists to quickly tag and edit photos, while Non-Force only functions to rearrange the photo layout within the spreadsheet. Force semantics are therefore more appropriate for the majority of the biologists' tasks. Note that extended availability of PhotoSpread for layout activities might change this behavior, and we may have to revisit our choice.

Photo layout

Cells in PhotoSpread may contain an arbitrary number of objects, and in particular, an arbitrary number of images. Although with a large display each cell can be relatively large, it is still critical for PhotoSpread to maximize the size and number of images that appear in the cells. We simplify the problem by resizing each image to be square. We also constrain resizing to retain a minimum size of m. Users can thus generally perceive some details in the images. For collections of realistic size, some images are, of course, not displayed in their cell. The display engine must thus determine a representative set of images to present. Many algorithms, including last-in-first-out (LIFO), FIFO, and more sophisticated summarization approaches are candidates for use as display group constructors. We chose LIFO so that the effect of drag and drop operations on a cell containing many images is immediately visible.

Within each cell we must compute the largest feasible size for displaying each photo. We perform this computation as follows. Let x be the horizontal dimension of the cell, y the vertical dimension, and N the number of photos to display. For now, assume that all N images fit in the cell without violating the minimum size m constraint introduced above. Let $r = x/y$ be the ratio of the width of the cell to its height. We want to compute

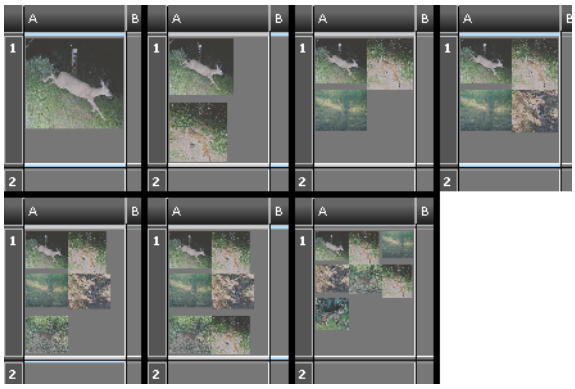

Figure 4. Example of cell photo layout.

S, the maximum dimension of the square container for each photo, such that all photos can fit within the cell.

It can be shown that the largest feasible S value is

$$S = \mathrm{M}ax[\mathrm{M}in(x/a, y/b), \mathrm{M}in(x/c, y/d)],$$

where $a = \lceil\sqrt{rN}\rceil$, $b = \lceil N/a\rceil$, $c = \mathrm{Max}(\lfloor\sqrt{rN}\rfloor, 1)$, and $d = \lceil N/c\rceil$. If $S < m$ then we cannot display all N photos in the cell, so we set S to m. Once we have the desired single photo size S, we simply place photos side by side in the cell such that each photo is contained within a square of size S.

Figure 4 shows this equation in action. The Figure displays the same cell as more photos are added. Initially, the cell contains one image that is expanded to fill the available space. As photos are added, they are made as large as possible. If the minimum size constraint is hit, the photos will not shrink further. For instance, if the photo size with 7 photos were too small, the display would stay with 6 photos even as more were added.

DISCUSSION/EVALUATION

In this section we present a brief qualitative comparison of PhotoSpread to existing systems, looking at specific use scenarios. In particular, consider two tasks: Photo query and analysis, and photo annotation.

When users analyze a large collection of annotated photos, they need to identify groups of photos based on their metadata particulars. The users need to layout the resulting photo groups in a way that is meaningful for the domain and task at hand. Users also need to combine groups of photos, extract their metadata, and compute statistics over that extract. Existing photo browsers and spreadsheet applications (such as Excel) provide limited support for such querying and analysis.

Photo browsers typically allow users to filter collections using keyword search, or through metadata based facilities. PhotoSpread offers a more expressive query language [12] than most photo browsers. PhotoSpread formulas can be composed of cell references, cell ranges, set operators, filter predicates, value selections and object-type-specific aggregations. Also, queries are implicitly saved within the cells in which they are defined, en-

abling users to easily chain queries or simultaneously view the results of multiple queries. Finally, PhotoSpread allows users to generate sets of related queries using simple copy and paste operations.

Excel and other spreadsheet applications provide rich metadata analysis and visualization tools. However, they lack support for cells containing sets of data. PhotoSpread cells can contain sets of objects, including images, and formulas that operate on these sets. Furthermore, objects have properties which users can access and manipulate within the spreadsheet. Also, PhotoSpread allows for drag and drop operations, providing an intuitive interface for manipulating objects.

Database systems like MS Access also support storage, querying and visualization of data. However, their interfaces are tailored for business data, and do not provide the spreadsheet and reorganization functionality that PhotoSpread does.

The second task we consider is photo annotation. For this task, a user starts with a large collection of photos with missing tags. Some tags, such as time or geographic coordinates, may be automatically generated by the camera. But many tags, like an animal's species, or the name of a person in a photo require human input. Thus, the user must associate individual tags or groups of tags with individual or groups of photos. After the initial tagging, additional tags may be entered during the analysis phase, as conclusions are reached or as errors are discovered.

Many existing systems have introduced methods to facilitate photo annotation. Some approaches include automated content analysis or tag suggestions, while others concentrate on improving manual tagging interfaces.

Like TeamTag [11], PhotoSpread allows multiple methods for manually and semi-automatically tagging photos. Users can tag groups of photos with a single drag and drop operation. In addition, users can assign multiple attribute-value pairs to groups of photos with a single action. PhotoSpread also provides aids that suggest attributes based on previously used attributes, or on the contents of the text edit area. Similar methods allow users to edit existing photo metadata. Many photo browsers and Web applications such as Flickr also provide tagging interfaces. This support is usually limited to tagging one photo at a time, or a group of photos with one tag at a time.

RELATED WORK

There has been a great deal of research on extending the spreadsheet paradigm. Spreadsheets have been extended to include support for image analysis [8], end-user programming [7], and data visualization. Most work has concentrated on data visualization. Hasler [4], Chi [5], Varshney [17], and Ma [9] extended spreadsheets to support complex objects in cells, such as charts and graphs. The commercial product Tableau [16] provides an intuitive drag and drop interface allowing users to create charts and graphs from existing data sources such as spreadsheets.

Many systems have been created to facilitate annotation of photos. These systems generally do not use the spreadsheet paradigm. Rodden [13] investigated the methods used to tag personal photo collections, including voice annotations. ZoneTag [1], meaning [10] and context watcher [19] ease the process of tagging personal photograph collections by providing simple tagging interfaces, automatic content generation including tag suggestions, and integrating with third party photo sharing sites such as Flickr. TeamTag [11] allows multiple users to collaboratively tag images. Much of the research for photo tagging concentrates on personal photo collections and not on scientific collections, which typically are much larger and have more tags per photo.

Finally we discuss systems built for browsing and retrieving images. Shneiderman [14], introduced tree maps as an effective photo layout algorithm. PhotoMesa [2] provides a zoomable interface to enable users to quickly navigate through large photo collections. Snavely [15] created 3D viewing environments out of 2D image collections using image based rendering techniques. Other systems have organized photos using timelines [3,6] or hierarchical faceted metadata and dynamically generated query previews [20]. Many applications, including those above, provide mechanisms for image retrieval, with the majority of systems [18] employing content based image retrieval.

CONCLUSION

PhotoSpread combines photo computational notions with organizational tasks. In initial, informal testing, our biology collaborators report that it provides the facilities they need for their analysis of large annotated photo collections. This result is not surprising, since we developed PhotoSpread precisely to address their needs.

Before developing PhotoSpread, we observed the biologists as they applied their existing tools— often spreadsheets and databases—to gather photos and metadata, add tags, and study their collections.

Their spreadsheets contained only the metadata and links to photo files. They could group and analyze the metadata, but without seeing the respective photos as they proceeded. It was therefore difficult for these users to discover patterns that were hidden in the visuals of the images.

Once these users identified photos of interest in this purely textual context, they then needed to switch to a photo browser to see the images. During this step, connections to the results that had been computed over the metadata were lost. Updating the metadata was a painful process, both because it involved editing photos

individually, using a cumbersome editor, and because the metadata was stored in multiple places.

Because PhotoSpread supports sets of photos within cells, biologists can easily see, side by side, the groups of photos they are interested in. As PhotoSpread supports tags for photos and other objects, its formulas can naturally refer to object metadata. And since PhotoSpread supports drag-and-drop reorganization, adding or modifying tags is simplified, and can proceed in concert with the data being analyzed. PhotoSpread offers several options for reorganization within the same spreadsheet paradigm. Biologists can therefore select the method that best suits their style, such as moving photos to formulas or moving strings to photos.

Of course, our biologist collaborators are requesting additional features. The highest priority for them are tools for managing photos across multiple repositories. Typically, photos are taken by multiple users or sets of camera traps. In each instance, the photos "live" on some computer, and are copied into an analysis tool like PhotoSpread. After the metadata is edited, it is important to push back the changes to the original source. Sometimes the photos themselves are additionally changed in an image editor. What is needed, in essence, is a distributed photo repository that provides PhotoSpread's computational facilities, plus object versions, recovery, locking and other data management services. These services are beyond the current PhotoSpread, but we will be working to develop such tools.

Although in this paper we have focused on field biologists as our main PhotoSpread target users, we strongly believe that analogous photo analysis requirements exist elsewhere. As discussed in the Introduction, museum curators, journalists, astronomers, and others have large image collections that in addition to being admired, need to be understood and dissected. We believe that PhotoSpread can offer useful services to those communities as well.

REFERENCES

1. S. Ahern, M. Davis, D. Eckles, S. King, M. Naaman, R. Nair, M. Spasojevic, J. Hui-I Yang. Zonetag: Designing context-aware mobile media capture to increase participation. *PICS '06*, p. 3, 2006.

2. B. Bederson. Photomesa: a zoomable image browser using quantum treemaps and bubblemaps. *Proc. ACM symposium on User interface software and technology*, pp. 71–80, 2001.

3. S. Harada, M. Naaman, Y. Song, Q. Wang, A. Paepcke. Lost in memories: Interacting with photo collections on PDAs. *Proc. Joint Conference on Digital Libraries*, 2004.

4. A. F. Hasler, K. Palaniappan, M. Manyin, J. Dodge. A high performance interactive image spreadsheet (iiss). *Comput. Phys.*, 8(3):325–342, 1994.

5. E. Chi, J. Riedl, P. Barry, J. Konstan. Principles for information visualization spreadsheets. *IEEE Comput. Graph. Appl.*, 18(4):30–38, 1998.

6. D. Huynh, S. Drucker, P. Baudisch, C. Wong. Time quilt: scaling up zoomable photo browsers for large, unstructured photo collections. *CHI '05*, pp. 1937–1940, 2005.

7. E. Kandogan, E. Haber, R. Barrett, A. Cypher, P. Maglio, H. Zhao. A1: end-user programming for web-based system administration. *UIST '05*, pp, 211–220, 2005.

8. M. Levoy. Spreadsheets for images. *Computer Graphics*, 28:139–146, 1994.

9. K. Ma. Image graphs – a novel approach to visual data exploration. *VIS '99*, pp. 81–88, 1999.

10. Merkitys-meaning. Available from http://meaning.3xi.org/, 2007.

11. M. Morris, A. Paepcke, T. Winograd. Teamtag: Exploring centralized versus replicated controls for co-located tabletop groupware. *CHI'06*, 2006.

12. Authors omitted for anonymity. The Photospread query language. Technical report, 2007.

13. K. Rodden, K. Wood. How do people manage their digital photographs? *Proc. Human factors in computing systems*, pp. 409–416, 2003.

14. B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.*, 11(1):92–99, 1992.

15. N. Snavely, S. Seitz, R. Szeliski. Photo tourism: exploring photo collections in 3d. *ACM Trans. Graph.*, 25(3):835–846, 2006.

16. Tableau Software. Available from http://www.tableausoftware.com/, 2007.

17. A. Varshney, A. Kaufman. Finesse: a financial information spreadsheet. *INFOVIS '96*, p. 70, 1996.

18. R. Veltkamp, M. Tanase. Content-based image retrieval systems: A survey. TR UU-CS-2000-34, Dept. of Computing Science, Utrecht University, October 2002.

19. Context watcher. Available from http://contextwatcher.lab.telin.nl/contextwatcherportal, 2007.

20. K. Yee, K. Swearingen, K. Li, M. Hearst. Faceted metadata for image search and browsing. *Proc. Human factors in computing systems*, pp. 401–408, 2003.