

Synthesizing evocative imagery through design patterns

Daniel M. Russell, Andreas Dieberger
IBM Almaden Research Center
{Daniel2, AndreasD}@us.ibm.com

Abstract

Automatically creating images that give a sense of content can be useful in a number of settings. Such images can summarize a larger collection of media in novel ways, giving an evocation of the whole, rather than a precise linguistic summary. Paradoxically, such visual summaries are often found to be more useful as indicators of the gestalt of a media than more traditional language summaries. We demonstrate a design pattern based method that creates visual compositions by selecting imagery from the source document(s) and instantiating several design patterns. Each pattern has a set of slots and roles that are filled with source imagery and then constrained to fit within the pattern visual specifications. Examples are given of the process and its end result. A pilot study has shown that these visual summaries lead to faster retrieval of relevant documents than textual summaries.

1. Introduction

Summaries are immensely useful things, especially if well made, accurate and intrinsically interesting. In our information rich age, there are many situations in which it would be useful to be able to have a single, easily understandable summary of a document collection. A common element of many tasks is browsing through large collections of documents – for example, when trying to understand the contents of an infrequently visited directory on a hard drive or server. Such tasks are commonplace, yet extant tools scarcely support this overview function.

Collection Summarizer (CS) is a program that takes a collection of documents and synthesizes a single summary image, attempting to create a summary that is both informative and aesthetically pleasing. This work builds on our previous efforts in creating summaries of videos, extending it to include more sophisticated image synthesis. [7]

CS creates a summary image montage by matching a document collection to a “best fit” pattern description of a summary. The pattern is then filled-in with elements from the collection. Although this seems surprisingly simple, the larger surprise is how well the method seems to work.

A reasonably small number of design patterns serve to create a rich set of summaries, and ones that seem to test well in our pilot study. This property of apparently sophisticated output is more a reflection of the complexity of the world in the relatively simple patterns. It is, like Simon’s ant [8], a measure of the complexity of the world seen through a simple mechanism, rather than the result of intrinsically complex processing.

2. What should a content summary do?

Current summary genres such as film trailers, previews or television commercials are extraordinary examples of designed compositions that summarize, but are not simple reductions of the original.

CS is an tool to automatically create an overview of a collection in a way that is both simple to understand and quickly understandable.

One way to achieve that goal is to just create a “contact sheet” that shows a representation of each item in the collection. But our goal is to be better than that. A grid of the complete document collection *can* be a useful overview, especially with image collections as in Figure 1.

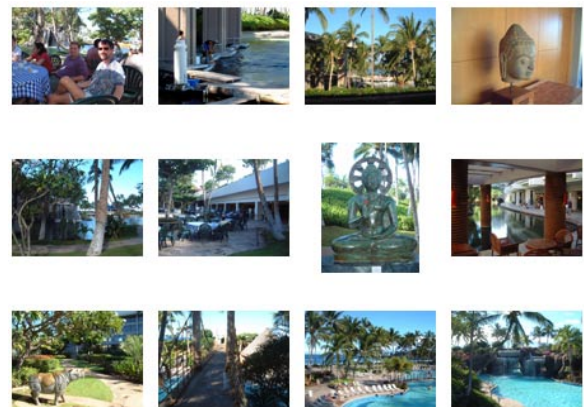


Figure 1. A collection summary needs to be better than simply displaying all of the elements of the collection. While this grid layout (in this case, a built-in Adobe PhotoShop tool made this) is effective for images, it works much less well for heterogenous collections, containing diverse items such as text documents, program code and spreadsheets.

But a collection summary can be a great deal more, if we adopt a few stylistic conventions, introduce the ability

to use more sophisticated graphical design elements (e.g. layering or montage) and have an ability to do minimal reasoning about the contents of the collection.

Figure 2 shows a snippet of a Win2K file browser that examines the same collection as shown in Figure 1. As is apparent, the overview function is not especially well served here. A user can individually examine each line item in the table, but that can be a tedious and time-consuming operation.



Figure 2. Most people viewing collections use a tool such as this file browser. While convenient for walking hierarchies, it does not solve the problem of understanding the contents of a large collection. As seen here, the file list runs off the bottom of the window, and the many arbitrarily named JPGs don't provide much information.

In Figure 3, CS has created a single montage that attempts to summarize the contents of the same collection seen in Figure 2. Here, the contents of the collection are not completely rendered – much information is dropped from the display (you can't, for example, determine file size or creation dates), but the sense of the whole collection is vastly improved.

To create Figure 3, the document collection was handed to CS, a design pattern was selected from the library and instantiated. The value of such a display lies not in its overall information density, but rather in its ability to quickly evoke a sense of “what's in the collection” without requiring extensive training in how to read the idiosyncratic details of tools like file browsers.

In this instance, the design pattern specifies the layout of the 4 images (a “most important” image, the largest, at the top of the frame), a text overlay (that is the first “important abstraction” from a text document), and a short summary of the contents of the collection (the icons with number counts and file name when the document count is 1). These 7 elements are all specified in a simple design pattern, and then instantiated with items from the collection.

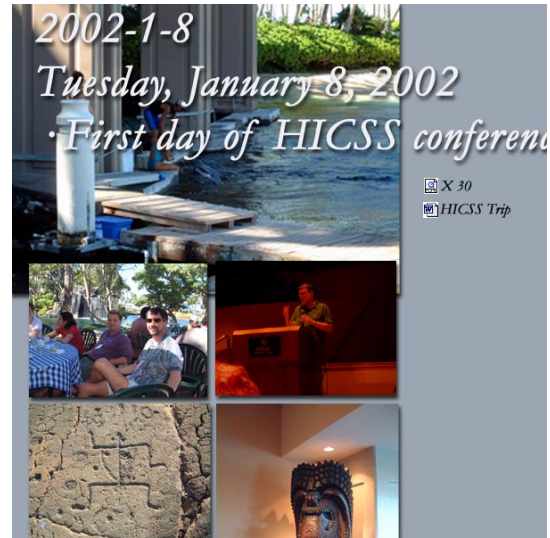


Figure 3. A sample output of CS pulls together a number of image and textual resources, attempting to create a layout that summarizes the contents of the collection. This is same document collection as seen in Figures 1 and 2.

3. Source content: Documents and metadata

The summaries are created from a collection of individual documents. Ideally, a summary is the kind of montage that you might create for a colleague, with the intent of giving a single glance sense of the collection.

Content of a variety of forms can be processed for use in the summary – with images (JPG, GIF, PNG) and text (DOC, PDF, ...) being the most common. But additionally, small, strongly typed documents can also be used: calendar entries, notes from a journal, emails, and so on.

CS analyzes a document collection by running a number of filters over the collection as a whole and creating the “Content Description” (CD) data structure. Each filter is applied to the collection, which then populates object/property/value triples in the CD.

In the current version, filters identify common terms (by analyzing file names), creation and last-modification dates, document type (e.g., genre identification), and common content terms (by extracting text from text documents and doing concordance analysis).

Each document in the collection creates at least one entry in the CD (<filename> <property> <value>), and frequently subsequent filters will identify additional information about the document. Properties that are true of the collection are placed under the \$Collection symbol e.g., <TRIPLE \$Collection \$PrimaryType \$Text> - indicating that the “primary type” of this collection is “text documents,” meaning that the majority of the contents are .DOC, .TXT, etc.

In essence, the analysis phase searches for information about what is salient in the collection about both individual elements and the collection as a whole. The analysis phase represents that in the CD for handoff to the design pattern instantiator.

4. Patterns for evocative summaries

Summary Design Patterns (or SDPs) are similar to those template descriptions used in [7]. Like the SDPs used for video summarization, the SDPs used here in CS specify layout in 2D *plus* information about how to composite the visual object within the whole.

SDP 12:

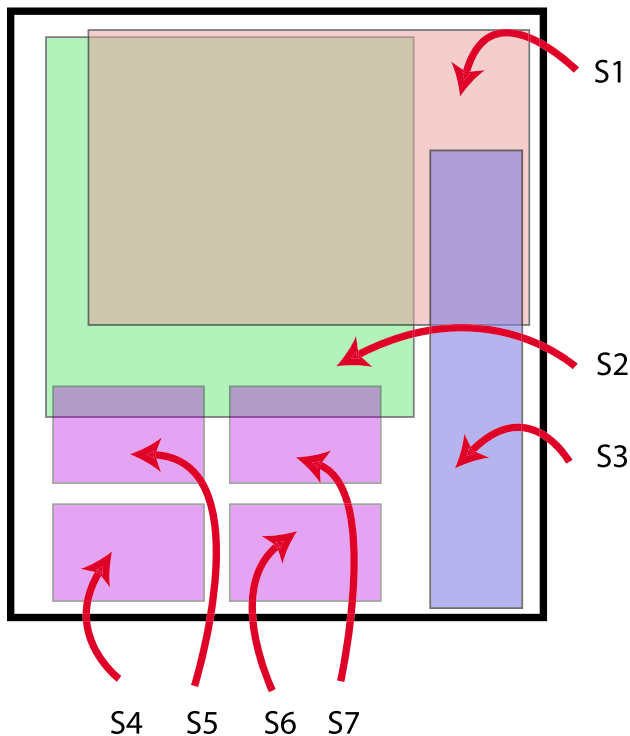


Figure 4. A Summary Design Pattern (SDP) specifies the layout of graphical elements in an evocative summary. SDPs are laid-out by hand, but instantiated at application time with items from the document collection. S1 is a text object (the most salient text found, < 100 chars), S2 is the most salient image found, S3 is the summary of documents in the collection, S4, S5, S6, S7 are the highest rated images from the collection.

Each SDP item has a layout, a possible set of constraints on its layout, and a procedure for adding it into the overall summary being created. In Figure 4, item S1 is a text block that is in the outermost layer, Gill Sans, italic, and drop shadowed.

An SDP is just an XML descriptor of layout elements (as in Figure 4) -- each elements has (1) relative X, Y,

Width, Height measures; (2) a composition procedure -- essentially a script that specifies layer sequence, selection type and properties, translucency, etc.; (3) a trigger pattern that specifies what kind of object can satisfy the element (e.g., \$Text, \$Image, \$List, \$Date...)

The SDPs are created by a visual designer and stored in an SDP library. Ideally, an SDP can generate many different kinds of layouts based on the details of what happens to be in the source collection.

SDPs form, naturally enough, the real source of design knowledge in the system: the better the SDPs, the better the resulting overview montages.

5. Building a summary by template filling

Collection Summarizer creates a summary by the following process:

1. Analyze the source collection by creating the collection description. How many items are present? What kind of items are they? (Text vs. image vs. short document kinds.) The Collection Description data structure binds individual documents to symbols (e.g., S1, S2, etc.) and properties of each (e.g., image-type, size, creation-date, etc.)
2. Search the library of SDPs for a best-fit between the feature set, user specified parameters and the SDPs element slots. (Match on presence / absence of elements for the slots.)
3. Instantiate the SDP by matching elements to slots.
4. Render the SDP to JPG, filling in text elements and visual transitions. Each item is rendered by running a "render script" that specifies how to layer and merge the item into the overall montage.

The heart of this process is step 3, the place where the template is filled in with content. To instantiate an SDP, a matching process selects the appropriate media elements from the collection.

Working from this relatively rich resource, the SDP matcher sorts the slots into priority order: required slots first, followed by increasingly less important slots to fill. Each slot has a priority rating that is a function of its overall contribution to the composition.

In general, the instantiation process proceeds by searching over the space of possible slot fillers, satisfying constraints between the slots and constraints placed on the template instantiation as a whole. These additional constraints, such as number of items in the visual display, may be specified by the viewer before the template is filled out.

7. Other systems

There have been a number of tools that create summaries and overviews of collections. The most interesting work has been in the area of video summarization [1, 2, 4, 7]. Most of these summarization systems have tried to identify the most salient, high-value information in a video stream, and then present only those portions. In our earlier work [7], we introduced the idea of design patterns as the foundational forms on which instances of video summarizations could be made. This work extends *that* model by adding simple reasoning about the ways in which template items could portray information about the collection. That is, in [7], slots in the templates could only be filled in by video content or transitions. Here, we include the ability to create entirely new representations of the content (for instance, a number indicating the quantity of images in a collection). This still follows the design template, but far extends the notion of simple instance replacement.

Other especially relevant work are the ComicChat [3], VideoManga [2] and ComicDiary [5] systems, each of which also has at their core a kind of design pattern that is instantiated. Again, the extension here has been to break out of the “comics school” and into a new kind of visual language that uses a large variety of elements *from* the source collection. Systems that create aesthetic displays [6] are roughly equivalent, but are targeted towards displays of a predetermined kind of information (e.g., stock variations over time, or working group awareness of individuals in a workspace). In addition, these systems do not attempt to solve a task in support of information browsing, but simply re-present an image collection, rather than a collection of arbitrary file types.

8. Future work

In the near term, we hope to test several new ideas to extend the range of utility of CS. In particular, we hope to be able to synthesize animations and incorporate video snippets, thereby transforming our earlier work [7] into a real, value-add tool that can supplement users work on their file systems.

One obvious direction to take CS is to output HTML versions of collection summaries (rather than JPGs) with a richer set of content parts and with active links for further drill-down. Such a summary of a web site (or portion thereof) could prove immensely valuable.

And to help with the analytic basis of CS, we are planning on incorporating taxonomy analysis tools that will create structures of similar components, reanalyzing the contents of a collection and giving a new kind of organizing information that the user can perceive.

9. Summary

The key idea of *Collection Summarizer* is that useful, evocative and pleasing gestalts of a collection can be created by instantiating a visual design pattern based on properties and contents in the collection.

The chief benefit of the design pattern summarization technique described here is that the resulting summary tends to have a more graceful presentation than standard “information visualizations” – and that such displays can be immensely useful in certain kinds of browsing or skimming tasks.

Future testing and development will test the idea that a synthetic summary can actually reveal contents and structure that is otherwise difficult to perceive.

10. References

- [1] J. Foote, J. Boreczky, A. Girgensohn, and L. Wilcox “An Intelligent Media Browser using Automatic Multimodal Analysis” *MULTIMEDIA '98*, ACM Press, 1998, pp. 375-380
- [2] S. Uchihashi, J. Foote, A. Girgensohn, J. Boreczky. “Video manga: Generating semantically meaningful video summaries.” In *Proceedings of MultiMedia '99*, p 383-392. ACM, 1999.
- [3] D. Kurlander, T. Skelly, D. Salesin. “Comic chat.” In *Proceedings of SIGGRAPH '96*, p 225-236. ACM, 1996.
- [4] Christel, M. G., Smith, M. A., Taylor, C. R., Winkler, D. B. “Evolving video skims into useful multimedia abstractions” *CHI '98*, ACM Press, April 1998, pp. 171-178.
- [5] Y. Sumi, R. Sakamoto, K. Nakao, K. Mase. “ComicDiary: Representing individual experiences in a comic style.” *Proceedings of UBIComp 2002*, p 16 – 32. Göteborg, Sweden. Springer Verlag.
- [6] J. Fogart, J. Forlizzi, S. E. Hudson “Aesthetic information collages: Generating decorative displays that contain information” *Proceedings of UIST-2001*, p 141-150, Orlando, FL, ACM Press.
- [7] Russell, D. M., “A Design Pattern-based Video Summarization Technique: Moving from Low-level Signals to High-level Structure” *HICSS 2000*, Maui, Hawai'i.
- [8] H. Simon, “*Sciences of the Artificial*” MIT Press, 1985.
- [9] Adobe. *Guide to Adobe Photoshop Scripting*. <http://www.adobe.com/support/techdocs/2bf56.htm?code=TA>