

Time-Machine Computing: A Time-centric Approach for the Information Environment

Jun Rekimoto

Interaction Laboratory

Sony Computer Science Laboratories, Inc.

3-14-13, Higashi-gotanda, Shinagawa-ku, Tokyo 141-0022 Japan

Phone: +81-3-5448-4380

E-Mail: rekimoto@acm.org, <http://www.csl.sony.co.jp/person/rekimoto.html>

ABSTRACT

This paper describes the concept of *Time-Machine Computing (TMC)*, a time-centric approach to organizing information on computers. A system based on Time-Machine Computing allows a user to visit the past and the future states of computers. When a user needs to refer to a document that he/she was working on at some other time, he/she can travel in the time dimension and the system restores the computer state at that time. Since the user's activities on the system are automatically archived, the user's daily workspace is seamlessly integrated into the information archive. The combination of spatial information management of the desktop metaphor and time traveling allows a user to organize and archive information without being bothered by folder hierarchies or the file classification problems that are common in today's desktop environments. TMC also provides a mechanism for linking multiple applications and external information sources by exchanging time information. This paper describes the key features of TMC, a time-machine desktop environment called "TimeScape," and several time-oriented application integration examples.

KEYWORDS: Time-machine computing, time traveling, desktop environment, document management, information visualization, inter-application communication.

'... But you are wrong to say that we cannot move about in Time. For instance, if I am recalling an incident very vividly I go back to the instant of its occurrence: I become absent-minded, as you say. I jump back for a moment.'
H.G. Wells, "The Time Machine"

INTRODUCTION

Organizing electronic information on computers is one of the central issues of concerning the design of user interfaces. We regularly have difficulty in finding files, and it is painful and often simply impossible to assign a suitable name to a newly

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

UIST '99, Asheville, NC

© 1999 ACM 1-58113-075-9/99/11... \$5.00



Figure 1: A snapshot of the author's computer desktop. As shown in this example, people often like to keep many icons directly on their desktops.

created document and classify the document correctly when putting it into a specific folder (many documents belong to more than one category). The management of contextual data (e.g., photographs or audio files) is even more problematic because there is no easy way to search for them. In addition, we use computers not only for document processing, but also for managing our daily schedules, browsing online information, and communicating with others by e-mail. These activities also create a large amount of data and, in a sense, represent our personal history. We certainly need computer support to managing them.

Malone et al. observed that people organize the documents on their real desktops spatially [13], and Mander et al. developed a 'Pile' metaphor for the casual organization of information represented as files piled on the computer desktop [14].

Many people prefer keeping document icons on the desktop instead of storing them in document folders (Figure 1). The popularity of this habit implies underlying difficulty in today's desktop environments. Directly placing objects looks more intuitive than folder hierarchies, in particular for novice users. Items are always there; people can access them immediately, and can arrange them spatially to form implicit structures [6]. As well as document icons, digital PostIt notes on the desktop are also used in managing daily schedules and as reminders.

The arrangement of these items often represents the context of a user's current activity. However, this simple method does not scale; desktop space is not large enough to contain all the items.

Freeman et al. argued that a time-ordered approach to the organizing and archiving of information is more suitable than a spatial approach [4]. Their time-ordered system, called the "Lifestreams," stores entire documents in *streams*, time-ordered 1D lists of information. We can also find similar time-ordered lists in some applications, such as an e-mail list or a web history list.

Since our personal activity is tightly coupled to the flow of time, there are a number of reasons why time-ordered systems work. However, if we store all the information in a 1D list, we give up many possible advantages of 2D graphical interfaces. In many cases, what we need is not only to find files but also to reconstruct the activity status (what we were doing at that time). For this reason, time-ordered lists are more suitable for archiving data than for providing workspaces. In addition, there can still be ambiguity for ordering. If the list is sorted according to file creation date, a thesis you have been working on for a year would be listed far from the current position of the time-ordered list.

Although there has been a debate on the relative merits of these two approaches [3], we think they are not exclusive but complementary. One of our ideas is to exploit the advantages of both by combining a spatial arrangement of information with a chronological navigation mechanism; in other words, to create a time machine for the information environment.

Imagine that your computer has a dial for time-traveling. When you create a document you can simply leave it on the desktop. You can also remove documents at any time. If you later need to refer to the information in that document, you can time-travel to the day when that document was on the desktop. You might then also see related items that were on the computer screen at the same time, and these items would help you to recall the activity context at that time. We call this time-centric approach "*Time-Machine Computing*," or TMC. The "everything-on-the-desktop" approach (Figure 1) now becomes a more attractive and practical because we can deal with only a single desktop surface. Operations are significantly simpler, and we do not have to be bothered with folder hierarchies.

This paper describes the concept of Time-Machine Computing and introduces two examples, a time-machine desktop system called "*TimeScape*" and a time-based inter application communication method called "*time casting*."

TIME-MACHINE USER INTERFACES

The Time-Machine Computing is a new concept of user interfaces, allowing a user to visit past and future information spaces. There are four key features that characterize this concept:

Lifelong archival of information history

In principle, all the user activities performed on computers are permanently archived. File creation and modification logs, web visiting histories, and e-mail sending/receiving histories are examples of user activities. Some other external information sources, such as web news, can also be integrated in the user's history.

Time-traveling: chronological navigation over archived information

Archived information can be used to reconstruct the state of the computer (the user's information space) at any specified time. In other words, the system allows computer "time-traveling." As a result, a user's computer desktop can also be an archive of information. A user creates a document and places it directly on a desktop. All information is immediately visible and the user can spatially arrange documents and other desktop items (e.g., PostIt notes).

When the user feels that one of these documents is no longer immediately needed, he/she can cleanup the desktop by simply dragging that document to the trash-can. When he/she again needs that document later, he/she specifies a time when that document was on the desktop. The system adjusts time accordingly, and the document appears again on the desktop.

There are various ways to specify time. The simplest one is to directly say "go to January 1, 1999" but it is also possible to say "go to when this file was created," "go to when I received this e-mail," or "go to when I attached a PostIt note that contains the string *project ABC*." It might also possible to use external (physical) contexts, such as "go to when I was in the meeting room."

Visualizing time in various ways

In some cases, simply restoring the past state would not be enough. We might want to view, from different perspectives, what was happening during a week or a month or perhaps the user's whole lifetime. Thus TMC also provides multiple ways to visualize the information space. Timeline and calendar views are typical examples.

Time-casting: inter-application communication of time

It is also desirable to restore the application's context as well as the document states. For example, if a web-browser can restore its state at any specified time, we can easily access the previously visited web pages. In many cases, it is also useful if various applications can exchange time information and change their states according to the time. Imagine that an e-mail browser and a desktop environment are exchanging time information. When a user select an e-mail on the mail-browser, its arrival time is transmitted to the desktop and the desktop state at that time is reconstructed. As a result, a user would find important notes he created on arrival of that e-mail. In this way, exchanging time information among software applications and computers provides a new way of integrating computer services. We call this kind of inter-application communication "time-casting."

Although the ultimate version of TMC will record and recover the history of user's activities perfectly, there are many

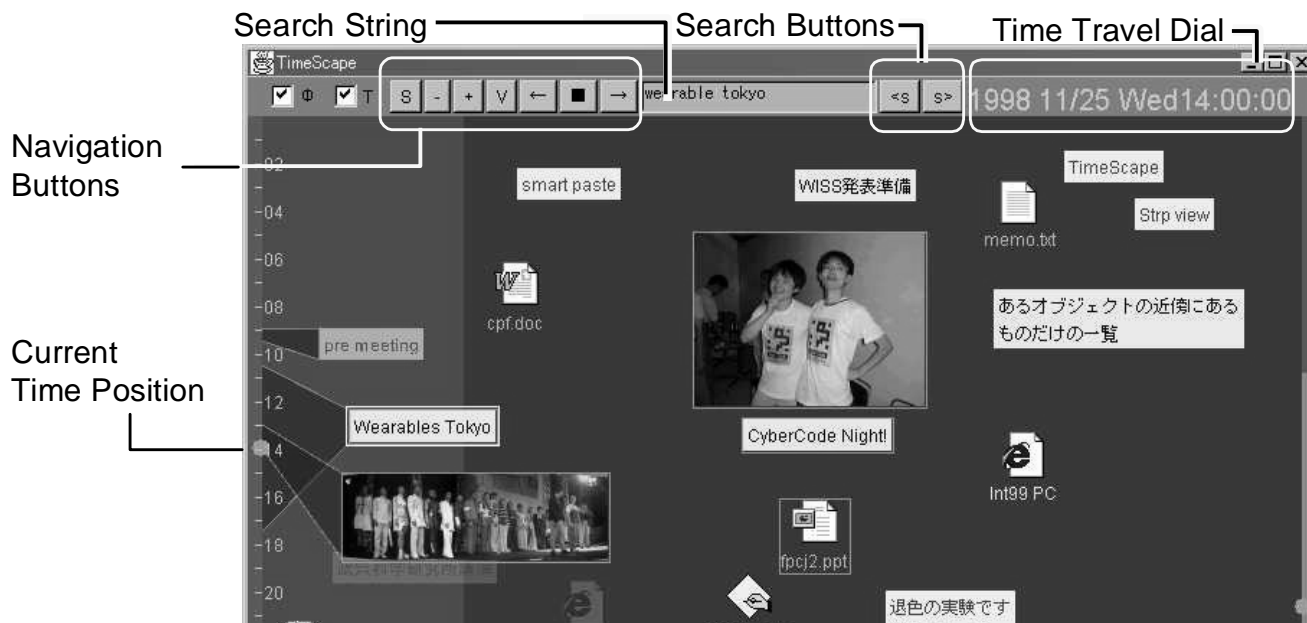


Figure 2: A screenshot of the TimeScope desktop.

intermediate levels of implementation. The following section introduces a system called “TimeScope,” a desktop environment based on TMC.¹

TIMESCAPE: A TIME-MACHINE DESKTOP ENVIRONMENT

We built a first instantiation of the proposed TMC concept, a desktop environment called “TimeScope.” It is a prototype software system written in Java and is designed to be used instead of native desktop environments (e.g., Windows GUI / Macintosh GUI). Figure 2 shows a typical screenshot of the TimeScope desktop. There are no folders in this environment. All the items (file icons, application icons, PostIt notes, voice note icons and digital images) are placed directly on the desktop. As with other desktop environment, a user can open documents and launch applications by double-clicking them. Desktop items can also be spatially arranged to organize information (e.g., related objects can be placed closer together to form a group).

Time Traveling

TimeScope provides several ways to change the time of the desktop. For example, when a user manipulates a time-travel dial (top-right part of the screen), the system restores the state of the desktop to that of the designated time. The system also provides “go back” and “go forward” buttons (arrow buttons on the toolbar) for event-based navigation. For example, pressing the “go back” button brings the user back to the time just before the most recent desktop change (item creation, deletion, etc.). The user can also select one of desktop objects and go to the time that item was created or

deleted. As described later, the user can also specify a time point using text search.

All modifications of a desktop are archived automatically. When the user feels some documents are not immediately necessary, the user can simply remove them from the desktop by dragging them to the trash-can. This action looks like the file deletion on ordinary desktop systems, but the file is still maintained in the TimeScope archive.² The user can always retrieve removed data by traveling to the “past” of the desktop. This capability supports the effective usage of the desktop real estate; the user can keep the number of desktop objects organized by removing unnecessary items. This feature is particularly suitable for maintaining “To-do” items on the desktop. It is also possible to determine the duration of attached objects: at a specified time, these objects automatically disappear from the desktop.

There is a notable difference between our 2D + time approach and other 1D time-ordered systems. We can keep many items on the TimeScope desktop simultaneously, while in Lifestreams everything is stored in a one-dimensional stream. The spatial arrangement of items gives a user a more concrete work context and helps the user recall his/her activity when he/she revisits a state by time traveling. When a user is working on a document, he/she can place supporting material (related documents, URLs, diagrams, PostIt notes) around the document, as we do on real-world desktops. The user can also spatially arrange and make groups of such items on the desktop in order to represent multiple activities. These objects might be put apart on a time-ordered list because their

¹A brief introduction of TimeScope can also be found in [17].

²TimeScope also provides, mainly for maintenance purposes, a command for permanent deletion of objects.

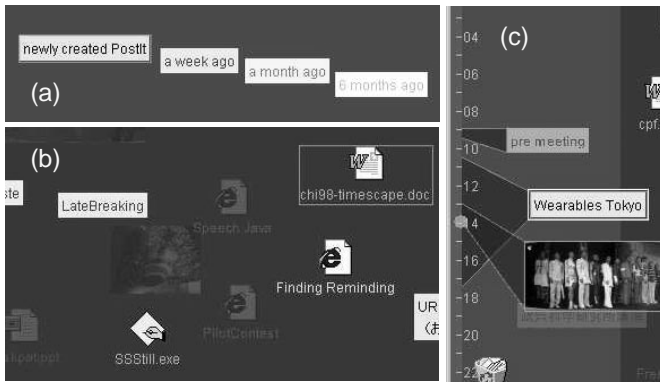


Figure 3: Visualization techniques used on the TimeScope desktop: (a) PostIt notes gradually fade over time, (b) recently removed objects are still slightly visible from the current desktop, and (c) objects close to the left edge show their durations.

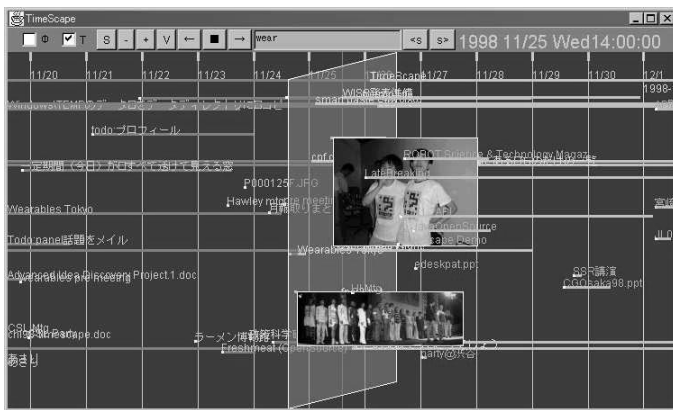


Figure 4: The timeline view.

creation dates are often quite different.

Like Lifestreams, TimeScope also supports time-travel to the future part of the desktop. When you travel to the future and create a PostIt note, it becomes a reminder. This scheduled object appears on the desktop automatically at the arrival of the appointed time. The system also provides reminding objects, such as “chime” or “blink.” When a specified time arrives, these objects are activated to make a sound or to flash the screen.

Visualization techniques

To support browsing along the time dimension, TimeScope provides several views of the information space. The current implementation of the system provides *desktop*, *timeline*, and *calendar* views. The desktop view is basically the same as that in ordinary desktop environments but has some enhancements for visualizing time information (Figure 3). For example, color of PostIt notes attached to the desktop gradually changes over time to subtly indicate its passing of time (Figure 3-a). And the background color of desktop surface changes to indicate whether the system is in the current, past, or future mode. The transparency level of the background is also controllable, so that past (or future) information can be



Figure 5: Animated transition shows connection between desktop and timeline views.

dimly seen from the current time (Figure 3-b). The left and right edges of the desktop respectively indicate the current time and date. When a desktop item is placed near these edges, its duration appears on a time-band (Figure 3-c). Using this feature, a user can use PostIt notes to represent daily schedules.

In the timeline view (Figure 4), desktop items are visualized as horizontal lines on a time-line diagram. The left and right endpoints of these lines respectively represent object creation and deletion dates. The current desktop is visualized as a semitransparent slanted rectangle in the middle of the screen, and the left and right parts visualize the past and future of

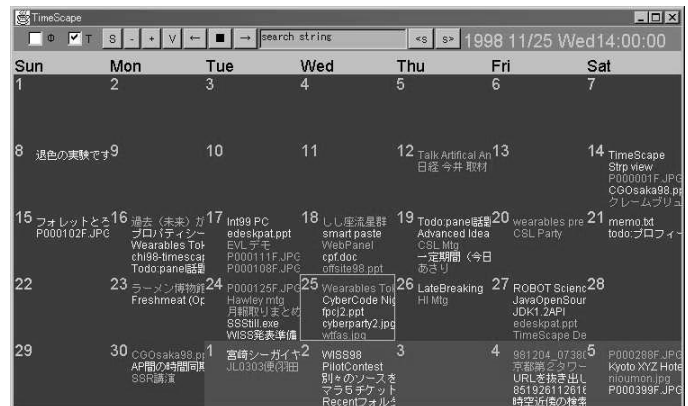


Figure 6: The calendar view provides a familiar perspective on the information space.

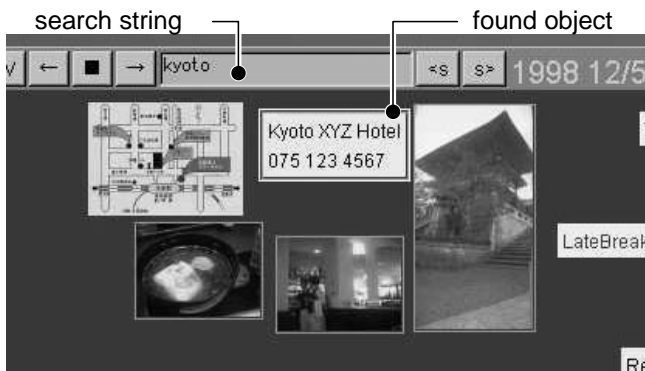


Figure 7: The result of a search for “kyoto.” When the system stops at a PostIt containing “Kyoto,” the user also sees a picture of Kyoto temple.

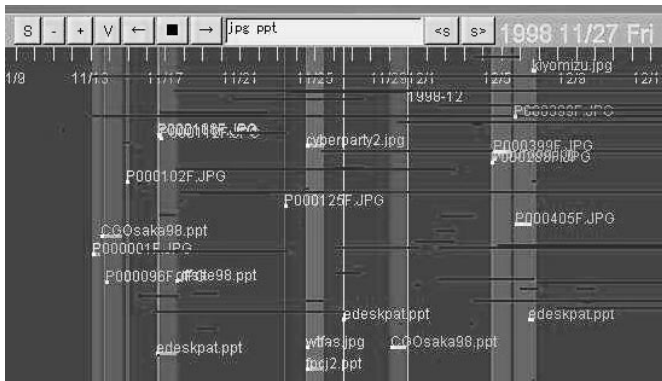


Figure 8: Using dynamic filtering to incrementally visualize objects containing search keywords. Note that semitransparent gray bands appear on the timeline indicating the density of the number of found objects.

the desktop. The user can use “zoom-in” and “zoom-out” commands to change the time-line scale and thus browse activities from a day, a week, a month, or an entire lifetime. To maintain a constant level of complexity on the screen, the labels of objects that have shorter durations fade away when the screen is zoomed out.

The calendar view displays items in a calendar format (Figure 6). Each cell contains items created on the date of the cell. This view is suitable for browsing a schedule and appointments in a way familiar to many schedule-software users.

These views are switched with a smooth animation to show the relationships between them (Figure 5), and the combination of these views helps a user recall the contexts of past activities. For example, a user travels with a desktop view to the specific time, then switches to a timeline or calendar view in order to see what the user was doing around that time.

Information search over time and space

Combining the above features, TimeScape can be seen as a space-time continuum of the user’s information space. The system also provides a text search on this continuum. For example, when a user enters keywords in the keyword entry box (on the navigation bar) and presses a “search forward”

or a “search backward” button, the system travels to the past or to the future until an object is found. Then the user looks at objects nearby (in terms of space and time) by switching view types (e.g., desktop and timeline).

Figure 7 shows the result of a search “kyoto.” Thanks to the attached PostIt note containing the word “kyoto,” the user can also find a picture of the temple even though the system does not understand image contents. As demonstrated in this example, items appearing on the desktop simultaneously can be used as an index to the activity history when a user attaches/detaches PostIt notes according to the user’s activities. This information is not only useful for representing daily *to-do* items, but also acts as bookmarks for later information retrieval.

In timeline and calendar views, the system can also enable a “dynamic filtering” mode, a variation of dynamic query techniques [1]. In this mode, the system dynamically filters out items that do not contain the specified keywords (Figure 8).

TIMECASTING: INTER-APPLICATION EXCHANGE OF TIME INFORMATION

Many software applications other than TimeScape also have a notion of “time.” For example, an e-mail browser could have a “current-time” state if we define “current-time” as the arrival time of a currently selected e-mail message. In this case, an e-mail browser that has a time-ordered list of mails would accept a time-travel command and select the e-mail message whose arrival time is closest to the specified time. Similarly, a photograph browser would display a photograph whose creation data is closest to the specified time. In this way, many software applications could change their states (e.g., displayed information, appearances according to the specified time information. Even though these actions would not strictly be “time-traveling,” they would still be quite helpful by making it easier for a user to recall the activity status at that moment.

Furthermore, if these applications were used together, combined information would be even more helpful. For example, suppose that a user selects a portrait of someone in a photograph browser. The user remembers that this picture was taken at a meeting but cannot recall the name of the person in the picture. If other applications, such as the e-mail browser or the desktop, could also change their states according to the time that picture was taken, the user could find the name of the person on the desktop schedule. Similarly, the user could more easily to recover the activity context if the photograph browser automatically selected a photograph according to the arrival date of a selected e-mail message. We call such inter-application communication of time “time-casting.”

The essence of time-casting is summarized by the following three features:

- Each software system (application) has the notion of “current-time.”
- When the “current-time” of one application is changed according to user interactions or for some other reasons, the application notifies the other applications of this changed time.

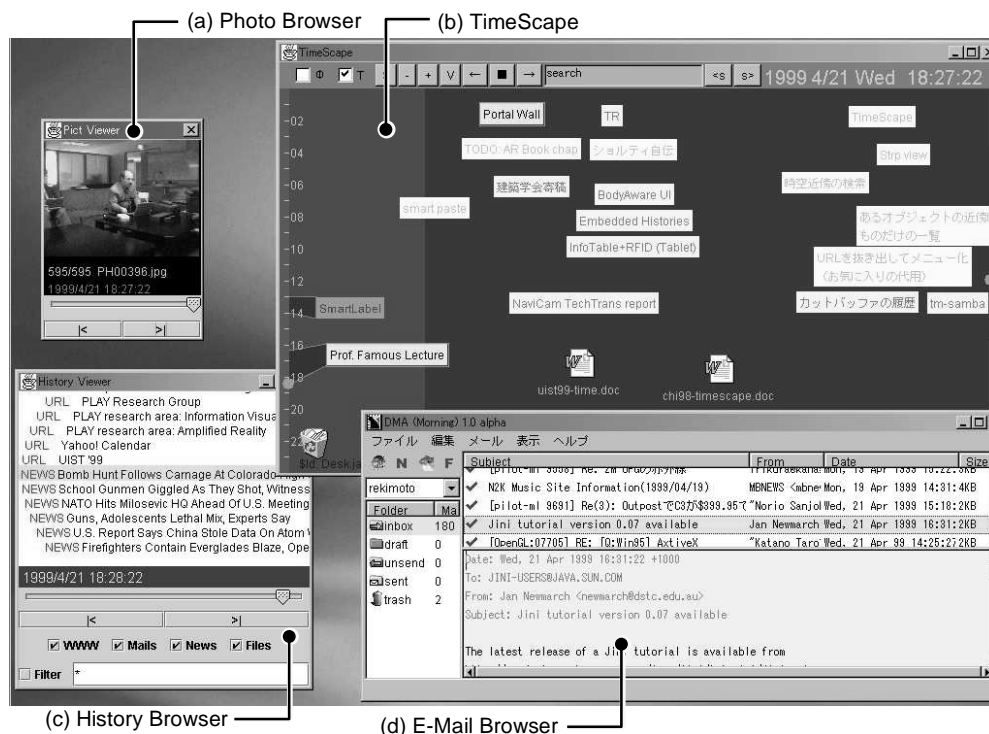


Figure 9: Time-casting: linking multiple applications by exchanging time information. A photo browser (a), the TimeScope desktop (b), the web history browser (c), and an e-mail browser (d) are exchanging time information. Any application can be used and others change their “current-time” according to the transmitted time. This example shows a snapshot when the user selected a photograph showing a person in the photo browser. The user may find the name of the person, on the TimeScope. The history browser and the e-mail browser also help to user recall his/her activities at that time.

- The other applications change their “current-time” states according to this information.

Figure 9 shows several usages of time-casting. In this example, a photo browser (b) and the TimeScope desktop (a) are exchanging time information. When the user travels time on the TimeScope environment, the photo browser automatically changes a picture displayed to correspond to the desktop’s “current time.” Conversely, the user can also flip through the photo browser, and the TimeScope desktop changes according to the date the displayed photograph was taken. Although PostIt notes on the desktop are created independent of the actual photography capturing, they often act as captions for the photograph (e.g., portrait picture at a meeting and a meeting agenda on the desktop).

We also have developed a text-editor compatible with time-casting. This text editor remembers the creation time of each character: when the user enters text, characters of the text are assigned a time-stamp. Like a font attribute or text-color attribute, this time attribute is saved with the text. When the user opens the text and puts the cursor on any character in it, the text-editor propagates the creation time of the character. As a result, the user can see what the user was doing when he/she created this text (more precisely, when creating the particular sentence containing the document).

Connecting to other information sources

Other possible combination includes automatic recording of user’s activities and other external events. Figure 9 also shows then interaction of a web history browser (c) and TimeScope (a). This web history browser automatically records the user’s web-visiting history, and displays a list of URLs that were visited around the specified time.³ For example, when a user needs to re-open a web site that was first visited at a meeting room, he/she first searches for the time of meeting by time-traveling on TimeScope and then finds that the URL link needed is available from the history browser. The web history browser also periodically retrieves information from other online sources (e.g., Web newspapers or weather reports). Such additional information can serve as a context for reconstructing the user’s activity state and can also be used to search for information on other applications.

Inter-computer time-casting

The area of time-casting is not limited to a single computer. Figure 10 shows the interaction of a desktop computer and a schedule browser on a hand-held computer (Palm Pilot). In this case, when the user selects a particular appointment on the PalmPilot, it transmits the appointment time to other computers.⁴ Upon receiving this information, TimeScope on the desktop automatically changes to that time to show the

³This information is obtained by modifying a web proxy server.

⁴We are currently using a serial cable between Pilot and computer for time-casting, but this could be done by using infrared communication.

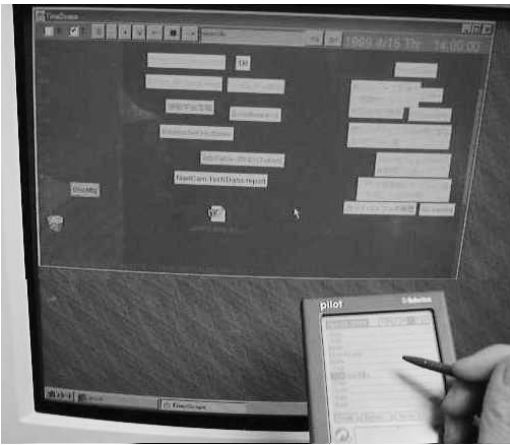


Figure 10: Time-casting between desktop and palmtop computers.

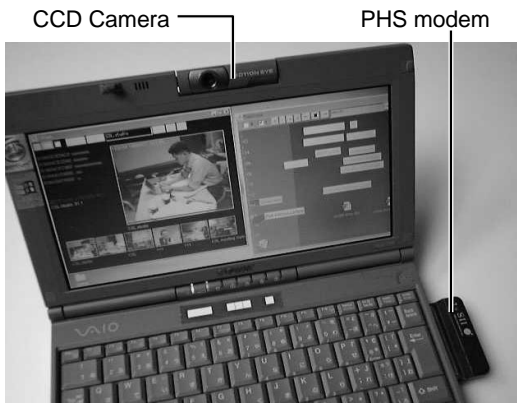


Figure 11: A notebook PC with a camera and a PHS wireless modem. The PHS modem can also be used to identify the current physical location.

user's activity context.

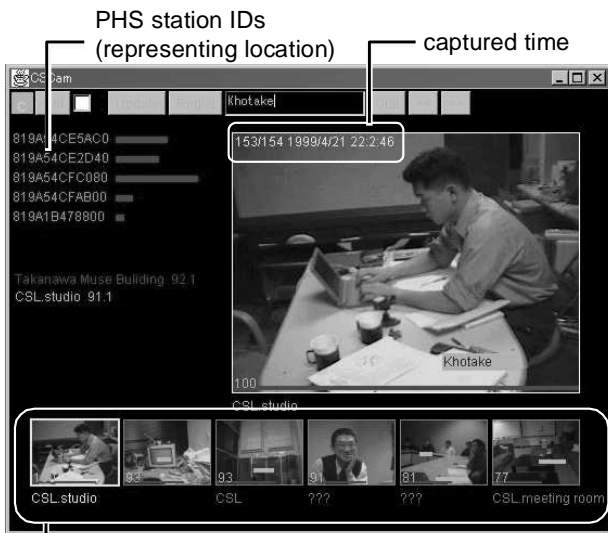


Figure 12: Linking time and location with a position sensitive photo-browser.

Another use of inter-computer time-casting is the linking of portable computers and installed computers (such as a digital whiteboard). A user in a meeting room, for example, could time-travel on his/her portable computer to the previous-meeting, and this information would be time-casted to the digital whiteboard. Then diagrams created during the previous meeting would appear on the whiteboard.

Linking physical contexts and time

Time-casting also makes it possible to incorporate physical contexts, such as locations, into a time-casting environment. Suppose you have a small position sensor that automatically records your physical location and the time. Then all other applications would be able to use this information by linking time and locations (instead of specifying an exact time, you can say "go to when I was visiting Kyoto").

Figure 11 shows a portable-PC equipped with a miniature camera and positioning system. A user can take a photograph at any time by pressing a shutter-button on a computer, and the picture is assigned the current location information as well as the captured time. We use a personal handy-phone system (PHS) modem as the location-sensing device. Since the PHS uses small radio cells for wireless communication, it is possible to identify the user's current location with reasonable precision by using the station-ID of the cell the user is in. When a user later selects a photograph on the browser (Figure 12), pictures that were taken near the time the selected photograph was taken are also listed in the thumbnail area.

Since the photograph browser is also "time-casting aware," we can link physical locations and time. For example, suppose that a user took a photograph in a meeting room. Later when the user attends the same meeting (room). He first starts the photograph browser, and the system lists photographs that were taken in the same room. Then he selects a photograph of previous meeting and he would also see the information related to the previous meeting on the TimeScape desktop, by traveling to the time of previous meeting.

SYSTEM ARCHITECTURE

All the above examples described in the earlier sections of this paper, except for the PalmPilot one described in the previous section, are written in Java and should run on any Java-enabled platform. We also implemented a drag-and-drop function that makes easy to exchange files and PostIt notes between TimeScape and native desktop environments.

Each object on the TimeScape desktop maintains duration (e.g., creation and deletion times), history of location on the desktop, references to external resources (e.g., path-name, URL), and some other attributes (e.g., labels for PostIt notes). This information is recorded automatically when a user changes the desktop state. Internally, these objects are linked from two time-ordered lists (one sorted by creation time and the other by deletion time), for rapidly reconstructing desktop states at a specified time.

TmSamba: a time-aware file system

The TimeScape desktop works on top of either an ordinary file system or a special time-aware file server. With a normal

file system, TimeScape simply maintains pathnames of files and users always see the latest contents of files (even when they are traveling to the past). When it is used with a time-aware file system, a user can access either the latest content or the version of a file at any specified time.

To restore the past contents of files, we developed a file server based on “Samba” [20], an open-source file server system for PCs that runs on Unix platforms. This modified Samba, called “TmSamba,” records all the modification logs to an internal database. TmSamba reconstructs the contents of a file when it is accessed by a pathname containing a special “time prefix” such as:

```
T:\@924247906550\My Documents\uist99.doc  
T:\@1999.4.20\My Documents\uist99.doc
```

In the first example, digits before the real pathname represent a time position (in seconds since 1 January 1970). The second example means a version of file “uist99.doc” on April, 20, 1999. This convention is also useful for non-GUI users, especially software developers. Without explicitly using a file versioning system, one can easily compare versions of file such as:

```
diff T:\@1999.1.1.12:30\App.java T:\App.java
```

Time-casting using UDP multicast

When an application changes its current-time, it sends time information to a specific UDP multicast group on the network. Multicasting allows information to be exchanged without explicitly using any central servers. Each time-casting-enabled software simply joins the multicast group and receives updated time information from other applications. For applications written in Java, we also provide the `TimeListener` interface following the Java event delegation and listener model [7]. Using this interface, application programmers can treat time-casting as a kind of Java input event:

```
public void timeHasChanged(TimeEvent event) {  
    long time = event.getTime();  
    // ... application specific actions ...  
}
```

DISCUSSIONS

User Experience

TimeScape and other time-aware applications have been used experimentally by the members of the authors’ group. Although a formal evaluation is still in preparation, informal observations revealed several issues.

PostIt notes based on time are turned out to be quite effective. People could quickly select text from web browsers or text editors and drag it to the TimeScape desktop to make a PostIt. The time history of PostIt creation/deletion represented a user’s activity patterns, and many users simply enjoyed playing them back to recall the past events. And the multiple

views connected by animation were effective in helping users understand the internal semantic model of TimeScape.

Time-casting between a photo browser and a desktop also worked quite well. Since most of the trial users had a notebook computer with an integrated camera (Figure 11), they often take pictures as memos or as parts of a visual diary. Many of them thus had a large number of chronologically sorted photographs reflecting their daily activities. Scheduling or appointment PostIts managed on a TimeScape desktop served as “captions” for these photographs.

We initially intended that application icons could, as in conventional desktop environments, also be placed and managed on a TimeScape desktop. We soon realized, however, that applications themselves are not closely related to the user’s personal history. Only a few application icons were managed on the desktop and people launched applications in various other ways (e.g., by opening a document, or from a “start” menu),

Initially, we made the past state read-only; the user was not allowed to alter the past. Many users, however, often wished to create or attach data to the past, not to alter the past version of document but to add annotations describing the past contexts. The current system therefore provides a way to “unlock” the past state. The past versions of documents, though, are still immutable. If a user wishes to modify the past version, he/she first copies that document to the “current” desktop and then works on the copy.

Some users found it difficult to decide to remove objects from the desktop. Unless there is a clear boundary event (e.g., a conference is over), people tend to keep objects until there is no space for new ones. We are currently considering the automatic removing or shrinking of long-unaccessed objects.

Application Domains

We think our TMC approach is particularly suitable for personal information management because information processing and personal history are tightly coupled in that domain. In many cases, PostIt notes can be used both for daily schedule management and as bookmarks for information search. For application domains in which there are a large number of well-named files, on the other hand, such as software development, hierarchical file management might be more suitable.

We also expect TMC to work quite well on small mobile computers. The small screen of these computers make dealing with multiple folders cumbersome, and TimeScape’s single-desktop approach should thus be particularly advantageous. It would also be possible to time-travel with a small jog-dial, an input device popular in recent mobile devices (e.g., cellular phones, palm-sized computers). TMC should also be quite useful on home computers. TimeScape displays on the wall in the living room or on the refrigerator door could be used to keep records of daily housekeeping information as well as family’s history.

As described in the previous section, time-casting between desktop and mobile devices allows a user to integrate different

types of computers. For example, one can create a voice memo when reading an e-mail on a mail browser; later he/she can link these two different items based on time.

Archival Overhead

Some readers may be concerned about the storage cost of life-long archival. We think that as long as the system deals only with texts, this is no longer a problem, thanks to the recent progress of storage technologies. There are several approaches to deal with “heavy” multimedia information, and the easiest would be to keep only the latest version. Another approach would be to regard a local storage a “cache” for the entire archive; only documents created (accessed, modified) recently would be stored on a local storage, and the complete data would be maintained on a networked server. If we can assume temporal locality of information access (i.e., a document created a week ago is more likely to be accessed than one created years ago), the time-based organization of information should be more advantageous than the conventional methods using file hierarchies.

There is also a tradeoff between the granularity of history information and the cost of information retrieval. Even though it is possible to record literally all the keystrokes or mouse operations, finding useful information in such a huge amount of data is a daunting prospect. A text search of the user’s entire key-input history might result in an information overflow.

RELATED WORK

As describe in the introductory section, Lifestreams is one of the first time-ordered information management systems [4]. It archives documents in a time-ordered list called “streams.” It should also be noted that before Lifestreams appeared, Noguchi developed a similar time-based method – called the “hyper organization method” (*cho-seiri hou*) – for organizing real-world documents [16]. This well known method (a guidebook on this method became a best seller in Japan) uses labeled envelopes as document folders and sorts them chronologically in a bookshelf. Noguchi also suggested that electronic documents be organized by storing them on dated folders. The hyper organization method and Lifestreams share many concepts for organizing information; document archiving without categorization, a single store for all documents, and the use of time as the basis for ordering information. Although these systems stimulate our work, we are more concerned with restoring the workspace than with sorting documents in a one-dimensional list.

Forget-me-not [10] is a mobile information assistance system that continuously captures the user’s physical context (e.g., locations), as well as document transmission histories (e.g., beaming a document to the printer). Temporally threaded workspaces [5] allow a user to make a snapshot consisting of anchors to documents. Later, the user can revisit one of these snapshots.

Chimera [9] records a history of graphical drawing operations, and Timewarp [2] allows users to browse and manipulate document modification histories as a part of collaboration support. Flatland [15], which is a digital whiteboard system, also provides a method of “snapping” hand-drawn diagrams

to an interesting point in time. While these systems focus on the history of a single document, our work additionally deals with the history of work contexts consisting of multiple documents.

The author’s previous research on wearable user interfaces, called “Augment-able Reality,” also uses the time-machine concept [18]. This system allows a user to create digital information (e.g., voice memos and photographs) and attach to physical contexts (e.g., locations, things). It also supports time traveling in the sense that the user can view information that was attached a week ago as well as currently attached data. Experience with this system led us to generalize an idea of “computer as a time-machine” which became a background of the work described in this paper.

Using time to link multiple data is a common idea in video-editing and video-logging systems. Professional video recording systems often use “timecodes” to synchronize multiple videotapes and audio tapes. Some note-taking systems, such as [21, 19] also use time to link voice data and other kinds of logging information. Our contribution to this area is to provide a mechanism for more generalized inter-application domain. As a result, not only closed systems but also many independently developed applications can be integrated by exchanging times. Another interesting idea is to combine the time-oriented method with other inter-application methods, such as those used in CyberDesk [22]. For example, when a user select a text segment ‘4/1/1999’ on a word processor and the system automatically recognizes it as a date and causes time-casting.

There have been a number of systems visualizing “time” on computers. Many of them use 3D graphics, animation, and other visual effects such as semitransparent rendering. The Perspective Wall [11] is one of the first using 3D + animation. Spiral Calendar [12] combines several levels of calendars in one spiral view, and Dynamic Timelines [8] extensively uses 3D, zooming, and semitransparency in visualizing the history of photographers. Our TimeScape design also provides three different types of views (desktop, timeline, and a calendar) that are connected by smooth animations. Unlike the Spiral Calendar, we use only one view at a time, considering more efficient screen space usage. We also use semitransparency to make past and future objects visible from the current desktop, and use zooming to change the time-scale.

CONCLUSIONS AND FUTURE DIRECTIONS

This paper described the concept of Time-Machine computing (TMC), which let people easily navigate the past and future states of computers. We think this metaphor is interesting alternative to traditional folder-oriented desktops. We are particularly quite interested in applying this technique to the personal and home computing domains.

There are several ways to enhance TMC. One is to improve the visualization of time. We are currently use animation, zooming, and semitransparency, but there are many other techniques such as 3D and nonlinear (i.e., fisheye) zooming that can be used to visualize the space-time continuum.

Another direction is to understand the implicit spatial-temporal structure of information space. For example, if a user were to put several PostIt notes close to each other for some period of time, the system might regard them a group.

Another, and perhaps more interesting, direction is to integrate time-machine computing with real-world information, such as real PostIt notes on the office bulletin boards, books and notebooks on a real desktop, or user operations on several devices (e.g., operation history of VCR decks). If we could also use recognition technologies such as OCR and voice recognition, the information they provide would serve as bookmarks for our activities, even though their recognition accuracy is not perfect.

ACKNOWLEDGMENTS

We thank Toshiyuki Masui, Yoshifumi Ueno, and Tota Hasegara for early exploration of time-machine computing ideas. Shigeki Nakamura contributed to the implementation of the system. We are also indebted to Mario Tokoro and Toshi Doi for their continuing support of our research.

REFERENCES

1. Christopher Ahlberg, Christopher Williamson, and Ben Shneiderman. Dynamic queries for information exploration: an implementation and evaluation. In *CHI'92*, pages 619–626, 1992.
2. W. Keith Edwards and Elizabeth D. Mynatt. Timewarp: Techniques for autonomous collaboration. In *CHI'97 Proceedings*, pages 218–225, 1997.
3. Scott Fertig, Eric Freeman, and David Gelernter. "finding and reminding" reconsidered. *ACM SIGCHI Bulletin*, 28, January 1996.
4. E. Freeman and D. Gelernter. Lifestreams: A storage model for personal data. *ACM SIGMOD Bulletin*, March 1996.
5. K. Hayashi, T. Nomura, T. Hazama, M. Takeoka, S. Hashimoto, and S. Gudmundson. Temporally threaded workspace: A model for providing activity-based perspectives on document spaces. In *Hypertext'98*, 1998.
6. Frank M. Shipman III, Catherine C. Marshall, and Thomas P. Moran. Finding and using implicit structure in human-organized spatial layouts of information. In *CHI'95 Conference*, pages 346–353, 1995.
7. Sun Microsystems Inc. Java foundation classes. <http://www.javasoft.com/products/jfc/>.
8. Robin L. Kullberg. Dynamic Timelines: visualizing the history of photography. In *CHI'96 Conference Companion*, pages 386–387, 1996.
9. David Kurlander and Steven Feiner. A history-based macro by example system. In *Proceedings of UIST'92*, pages 99–106, 1992.
10. Mik Lamming and Mike Flynn. Forget-me-not: Intimate computing in support of human memory. In *FRIEND21 '94 International Symposium on Next Generation Human Interfaces*, 1994.
11. Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. The perspective wall: detail and context smoothly integrated. In *Proceedings of ACM CHI '91*, pages 173–179, 1991.
12. Jock D. Mackinlay, George G. Robertson, and Robert DeLine. Developing calendar visualizers for the information visualizer. In *Proceedings of UIST'94, ACM Symposium on User Interface Software and Technology*, pages 109–118, November 1994.
13. Thomas W. Malone. How do people organize their desks? implications for the design of office information systems. *ACM Trans. On Office Systems*, 1(1):99–112, 1983.
14. Richard Mander, Gitta Salomon, and Yin Yin Wong. A 'pile' metaphor for supporting casual organization of information. In *CHI'92*, pages 627–634, 1992.
15. Elizabeth D. Mynatt, Takeo Igarashi, W. Keith Edwards, and Antony LaMarca. Flatland: New dimensions in office whiteboards. In *CHI'99 Proceedings*, pages 346–353, 1999.
16. Yukio Noguchi. *"Cho-Seiri Hou" (the hyper organization system)*. Chuo-kouron publishing, 1993.
17. Jun Rekimoto. TimeScope: A time-machine for the desktop environment. In *Proceedings of ACM CHI'99 Extended Abstracts*, pages 180–181, 1999.
18. Jun Rekimoto, Yuji Ayatsuka, and Kazuteru Hayashi. Augment-able Reality: situated communication through physical and digital spaces. In *Proc. of the Second international symposium on wearable computers (IEEE ISWC'98)*, pages 68–75, 1998.
19. Lisa J. Stifelman. Augmenting real-world objects: A paper-based audio notebook. In *CHI'96 companion*, pages 199–200, 1996.
20. The Samba team. SAMBA: opening windows to a wider world. <http://www.samba.org>.
21. Karon Weber and Alex Poon. Marquee: a tool for real-time video logging. In *Proceedings of CHI'94*, pages 85–86, 1994.
22. Andy Wood, Anind K. Dey, and Gregory D. Abowd. Cyberdesk: Automated integration of desktop and network services. In *CHI'97 Technical Note*, pages 552–3, 1997.