VNC-BASED ACCESS TO REMOTE COMPUTERS FROM CELLULAR PHONES

Buntarou Shizuki, Masato Nakasu, and Jiro Tanaka
Institute of Information Sciences and Electronics, University of Tsukuba,
Tennoudai 1-1-1, Tsukuba, Ibaraki, JAPAN (zip:305-8573)
{shizuki,baru,jiro}@iplab.is.tsukuba.ac.jp

ABSTRACT

We propose a virtual network computing (VNC) based architecture for accessing the desktops of remote computers from a cellular phone. A viewer is provided on the cellular phone that enables the user to see and manipulate the desktop of various remote systems such as MS Windows, Macintosh, and UNIX. The system to be accessed must be running a VNC server and it must be attached to a network. A proxy is used to send the image of the desktop to the cellular phone, to convert different devices, to suppress network traffic, and to support recovery from an unscheduled disconnection. To reduce user effort and solve problems inherent to the cellular phone's small screen, several functions are provided on the cellular viewer. Frequently used screen areas can be assigned and restored quickly by using the *Shortcut* function. The *Guidance* function can be used to show the current key assignments. Two areas can be viewed at the same time by the *Twin view* function. A prototype of the proposed architecture has been implemented using Java and has been tested on a Java-enabled cellular phone emulator.

KEY WORDS
Virtual Network Computing, cell phone, small screen problem, iAppli, Java.

1 Introduction

Cellular phones have shown a dramatic improvement in their functionality to a point where it is now possible to have cellular phones execute Java programs. As a result, cellular users throughout Japan are now able to read and write e-mail, browse Web pages, and play Java games using their cellular phones. This trend has prompted us to propose the use of a cellular phone as a device for remotely controlling computers. For example, if a cellular user is able to remotely access computers (such as workstations in offices and personal computers (PCs) in homes) or other networked digital appliances, it would provide the user with the following capabilities:

- to see the contents of a file placed on the desktop of a remote computer.
- to reboot a remote server as an administrator.

While it is not very difficult to develop a specific system to satisfy each of the above operations separately, it lacks the generality needed for performing several such functions with one device.

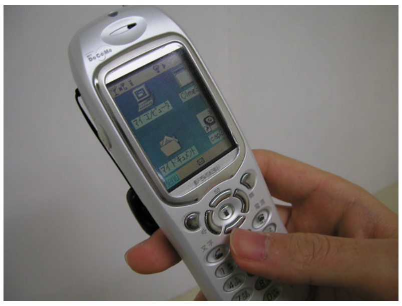


Figure 1. A cellular phone SVNC viewer accessing the desktop of a remote MS Windows system

This paper presents a virtual network computing (VNC)[1] based architecture for accessing the desktop of various remote systems (such as MS Windows, Macintosh, and UNIX systems) from a cellular phone. It is assumed that the remote computer system is running a VNC server and that it is attached to a network. The cellular user can see and manipulate the desktop on the cellular phone. Figure 1 and Figure 2 shows the viewer that we have implemented on a Java-enabled cellular phone.

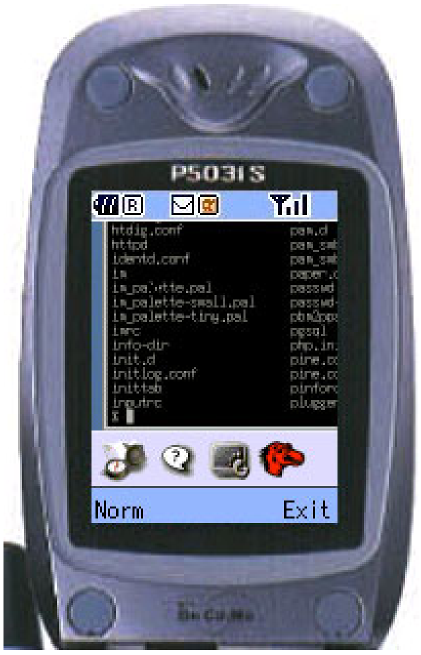
Figure 1 shows our viewer running on a cellular phone accessing the desktop of a remote MS Windows system. Figure 2 shows our viewer running on an emulator[1] accessing the desktop of a remote Linux system. This paper also describes several user interfaces, based on our architecture, which overcome some of the problems associated with the small screen[2] of cellular phones.

2 An Architecture for Cellular Phones

The use of existing remote display protocols such as the X Window System Protocol[3] and the Remote Desktop Protocol[4] of Microsoft's Windows 2000 Terminal Services[5] does not provide the required capability to realize the following goals:

Convert different devices. A cellular phone is physically limited. The typical size of the screen is 2.2 inches with 120x130 pixels. The phone usually has

[1]Hereafter, we use captured screen images of i-JADE released by Zentek Technology Japan, Inc. (http://www.zentek.com/) that emulates an i-mode cellular phone with a built-in KVM.



```
htdig.conf            pam.d
httpd                 pam_smb
identd.conf           pam_smb
im                    paper.c
im_pa:'ette.pal       passwd
im_palette-small.pal  passwd-
im_palette-tiny.pal   pbm2ppa
imrc                  pgsql
info-dir              php.in
init.d                pine.co
initlog.conf          pine.co
inittab               pinforc
inputrc               plugger
$ █
```

Figure 2. SVNC viewer on a emulator accessing the desktop of a remote Linux

about sixteen keys. In contrast, current desktops are manipulated with keyboards that have over 100 keys and pointing devices such as a mouse.

Suppress network traffic. The wireless transmission bandwidth available for a cellular phone is limited. Currently, it is 384k bps, even on IMT-2000 based services (only downstream at this transmission rate).

Recover from an unscheduled disconnection. Because of its wireless nature, stable network connectivity cannot be expected. For example, when the user goes into a tunnel or a building, established connections can be lost. In addition, in order to use the same cellular phone to talk to someone, the user must terminate the network connection.

Suppress computational resource use. CPU performance and memory size are limited on a cellular phone to achieve portability and to lower power consumption.

2.1 The VNC-based architecture

To achieve the above-mentioned goals while at the same time considering portability and generality, we propose a VNC[1] based architecture. VNC is an implementation of a remote display system based on a Remote Frame Buffer (RFB) protocol[6].

Structure

Figure 3 depicts the VNC architecture. It consists of VNC servers running on one or more remote computers, a Smart VNC (SVNC) proxy, and a SVNC viewer on a cellular phone. A VNC server sends a remote desktop display as bitmap images in RFB protocol.

A SVNC proxy converts (crops, shrinks and resamples) the display image and then transfers the converted image to a SVNC viewer in response to a user request that was received from that SVNC viewer. The transfer is performed in our own Compact RFB (CRFB), our simplified RFB protocol. Then, the SVNC viewer displays the transferred images. Key events received by the SVNC viewer are transmitted to a SVNC proxy that coverts them and sends them to the server.

When the user first tries to connect to a remote computer, he must specify his user name and password for authentication as well as the host name of the computer that is running a VNC server. If authentication succeeds, the SVNC proxy establishes a *session* with the VNC server and the SVNC viewer starts user services.

To suppress network traffic, encoding is changed depending on contexts. Usually, colored display images are transferred from the SVNC proxy to the SVNC viewer. However, while the user is manipulating the remote desktop, such as scrolling and moving the pointing device, the display images are gray-scaled to reduce the number of bytes required to encode the image.

Maintaining the states of a session

In order to recover quickly from an unscheduled disconnection, the SVNC proxy maintains its own database containing each session's unique information such as the user name, the password, the target host name, and other internal states.

When it is first connected to a SVNC viewer, the SVNC proxy searches its own database using the user name and the target host name pair as a key. It determines whether or not there has been any previously established session. If such a session did exist, the proxy restores the stored states. A session's state in the proxy is discarded when the user explicitly terminates it on the viewer.

2.2 Benefits from the architecture

The VNC-based architecture described in the previous section has the following benefits.

The VNC protocol is an image-based protocol in which updates to a screen by applications are captured

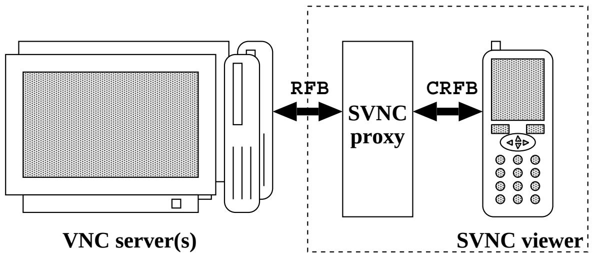


Figure 3. VNC-based architecture

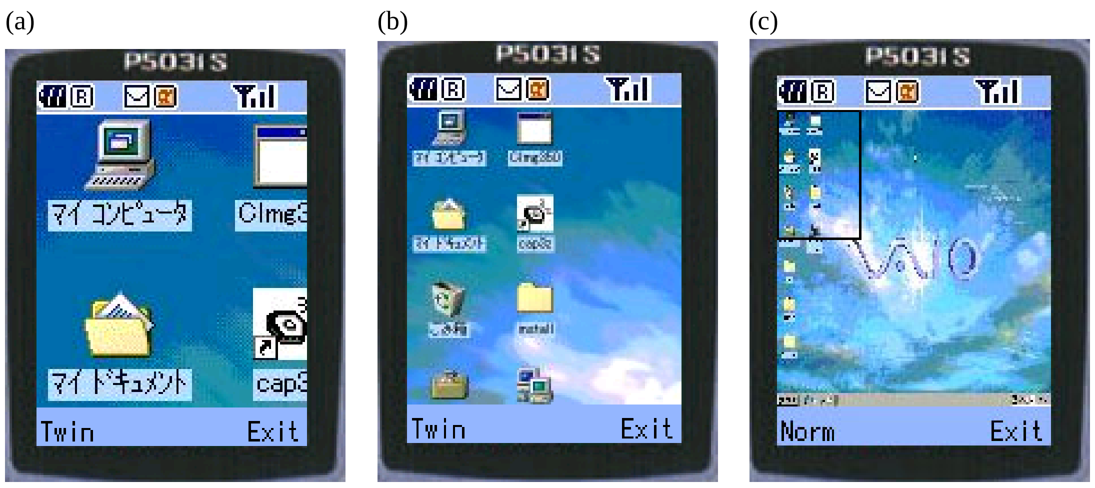


Figure 4. Snapshots of a SVNC viewer

and transferred in bitmap images to the viewer. Therefore, we can manipulate the applications running on the remote system by browsing the same image that we would be browsing if we were sitting at the remote computer.

We can utilize the general availability of VNC servers. VNC is becoming widely available as an infrastructure for controlling remote computers and for linking home appliances with PCs (such as μVNC[7] and [8]) due to the portability of the RFB protocol.

3 User Interfaces on a Cellular Phone

Building a remote accessing system for cellular phones based on an image-based protocol has additional problems that must be addressed beyond those that must be considered when building remote accessing systems for PDAs (such as the PalmVNC[9] and Windows CE clients for Windows 2000 Terminal Services). These are:

- the cellular phone usually has no pointing device, so pointing is difficult. In contrast, a user can easily point anywhere on the screen of PDAs using a stylus.

- the screen is smaller than that of PDAs, so it is impossible to directly show the entire display area of the remote desktop system.

The following gives a description of the basic operations provided by the SVNC viewer. Then, advanced user interfaces are introduced that enable the user to quickly jump and point to a desired area on the remote desktop display.

3.1 Basic Operations

Figure 4 shows snapshots of the SVNC viewer as the user is accessing the desktop of a remote MS Windows system. In Figure 4a, the currently displayed area of the desktop, called the *viewport*, is the upper-left corner. The following functions are available on the cellular viewer as basic operations to manipulate the viewport and to send events.

Panning and zooming: The user can move the viewport horizontally and vertically. The viewport can be widened (zoom out) to browse its contents and narrowed (zoom in) to see the display in greater detail. Figure 4b shows the viewport after the user has zoomed out from Figure 4a. The viewport of Figure 4b shows an area twice as large as the viewport of Figure 4a, both in width and in height.

Overviewing: In order to browse the entire area of the desktop display and to choose a specific area within it, the overviewing mode is provided. When the user turns this mode on, the aspect ratio is changed so that the whole area is rendered to fit the screen of the cellular phone. For example, Figure 4c is an example of this mode applied to Figure 4b. Note that, in this view, the viewport to be restored when the overview mode is turned off is indicated by a black rectangle as shown in Figure 4c. The user can move the black rectangle horizontally or vertically and shrink or enlarge it by pressing keys to change the viewport while examining the desktop display. This helps the user adjust the viewport to the desired area of the desktop display.

Pointing and clicking: The user can move the pointer on the remote desktop display vertically and horizontally by pressing keys. Dragging can be executed by pressing a key to specify the start of the dragging operation, then moving the pointer, and finally pressing the same key to indicate the end of the dragging operation. When the pointer approaches the edge of the viewport, the viewport is automatically panned to follow the pointer. Clicking mouse buttons can be performed by pressing the corresponding keys on the cellular phone. Double-clicking can be executed by pressing a specific key as a prefix.

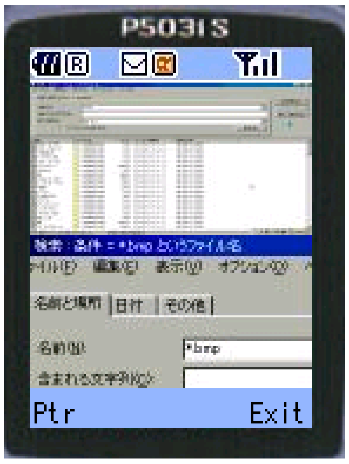


Figure 5. Twin view

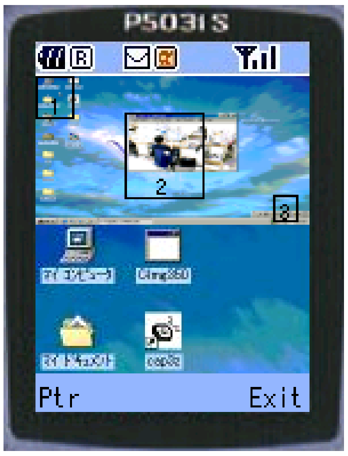


Figure 6. Guidance (upper half of the screen)

Inputting text: Text is entered and edited locally on the cellular phone using the built-in text input capability of the cellular phone. After editing on the cellular phone, the text is transmitted to the VNC server via the SVNC proxy. The same mechanism can be used to send control characters such as backspace, delete, carriage return, and line feed, by entering escape sequences (the same approach as is used with Rajicon[10]). For example, the user can list the files on a remote UNIX system by entering the ls command with a carriage return as four characters "`ls\n`" on a terminal emulator.

3.2 Shortcut Assignment

Common GUI operations, such as pressing GUI buttons and opening pull-down menus, become very tiresome when only basic operations are provided. For example, to push a GUI button that is not currently displayed on the viewer, the user has to zoom out, pan several times, and may then have to zoom in to show the button.

To shorten the time necessary to access frequently used display areas, the SVNC viewer offers a mechanism called a *shortcut*. This mechanism enables the user to register the current viewport to a specific numerical key by pressing a numerical key from Key-1 to Key-9 after pressing Key-Asterisk twice. For subsequent operations, the user can easily restore the viewport by pressing the designated numerical key after pressing Key-Asterisk once. For example, when accessing a remote MS Windows system, the user may assign shortcuts to the current working area, to the upper-left corner of the current application window (i.e., the area around "File" menu), and to the lower-left corner (i.e. the area around "Start Menu").

3.3 Twin View

Sometimes, it is convenient to display two areas of the desktop simultaneously. Suppose that we are going to search for a file that matches a given condition on a remote computer. When we are sitting at the computer, we can easily see the area for entering a search condition and the area that shows the result. We can enter test conditions and observe the results simply by moving our line-of-sight slightly. Thus, we can quickly acquire a condition that produces the desired result. However, on a SVNC viewer, even though we can use shortcuts (Section 3.2), we still have to press keys many times to enter the test conditions and view the results repeatedly. To solve this problem, the SVNC viewer provides a view called *Twin view*. This view divides the display area of the cellular phone into two parts. The user can control the upper half and the bottom half independently. Therefore, he can assign both halves independently to facilitate his desired task.

In Figure 5, the user has selected the window of the search tool. On the upper half, the entire window of the tool is shown. On the bottom half, the control area of the tool is displayed for giving search conditions. Now, the user can see whether the given condition produces the desired results by watching the upper half of the display immediately after he starts a search. If the desired result is not obtained, he can launch another search by entering different conditions using the bottom half of the cellular display.

3.4 Guidance

A user will often want to know which keys have already been assigned as shortcuts and which keys are free, especially when he wants to assign a new key to a different area. The *Guidance* function is provided to show this information. The Guidance function depicts each display area assigned to a shortcut as a rectangle with the key number.

The upper half of Figure 6 is a snapshot of the Guidance function. In this figure, the Guidance function shows that Key-1, Key-2, and Key-3 are assigned to the upper-left corner area, to the area around the center, and to the area around the bottom-right corner respectively. Therefore, the user can easily determine which keys can be assigned. In addition, this view can be utilized to remind the user of the key assignments that have been made. For example, it can be seen that just pressing Key-Asterisk and Key-1 will change the view to Figure 4a to access "My Computer" on the remote desktop.

4 Implementation

We implemented the proposed architecture including the SVNC proxy and the SVNC viewer using Java.

4.1 Platform

The SVNC proxy was implemented by modifying the Java version of the VNC viewer released by AT&T Laboratories, Cambridge. The proxy runs as a servlet on an HTTP server with the servlet API. We are currently using Apache Tomcat 4.0[2] as the HTTP server. We have installed the proxy on a PC with a Windows 2000 workstation operating system.

The SVNC viewer has been implemented using the J2ME Wireless SDK released by NTT DoCoMo. The code of the SVNC viewer has been placed on the HTTP server of the SVNC proxy and is downloaded in response to requests from the cellular phone.

We have tested our viewer using a real cellular phone and have successfully accessed remote computers.

4.2 Handshakes in CRFB Protocol

The viewer periodically requests the SVNC proxy to send the desktop display of the remote computer as a frame. This polling action is required to ensure that the Java running on a NTT DoCoMo device follows the definition of Doja[3] This definition requires that applications on a cellular phone must explicitly send a request to the proxy to start communication. Moreover, the proxy can only return one message in response to one request from the viewer. For each frame, the viewer sends the position and size of the desired viewport with its zoom level. It should be noted that the proxy can generate a frame by shrinking the original image with anti-aliasing depending upon the zoom level,.

Usually, bitmaps transferred from the proxy to the viewer are encoded in 120x130x8 bits with compression. However, during scrolling and dragging, bitmaps are gray-scaled into 120x130x3 bits with compression.

Pointing and clicking mouse buttons are achieved by the translation of these events on the proxy side. When the proxy accepts a request from the viewer, it generates corresponding event sequences and sends the sequences to the VNC server. Inputting text is also realized in the same way. For example, when the user inputs the text "`ls\n`", the proxy generates an event sequence consisting of six events; *KeyPress("l"), KeyRelease("l"), KeyPress("s"), KeyRelease("s"), KeyPress(Return), and eyRelease(Return).*

[2]Apache Tomcat 4.0 is released by the Apache Software Foundation (`http://www.apache.org/`).

[3]Doja is a profile of Connected Limited Device Configuration (CLDC)[11], that is defined by NTT DoCoMo.

The bindings of shortcuts (Section 3.2) are stored in the proxy. When the user assigns an area to a key, the current size and position of the viewport and the zoom level of the viewport are saved. When the user uses a shortcut, the key is sent to the proxy and the proxy sends the viewport information with its image back to the viewer.

5 Related Work

There are several remote accessing systems available that provide a cellular phone with access to the desktop of a remote computer. [12] uses a GUI-CUI transformation enabling the user to access the GUI applications of a remote PC. The mechanism analyzes the structure of the GUI components of an application and sends the result in Compact HTML, which is displayed with the Web browser on the cellular phone. The user's manipulation of the Web browser is converted into application events by the CGI interface. CAFEMOON@HOME[13] is another system that uses this approach to access networked appliances. However, the developer must prepare a software module to produce the semantics of the target application and/or an algorithm to analyze the semantics. These software modules are not required in our proposed system.

Desktop On-Call[14] Version 5 and Rajicon[10] are based on an image-based approach. In this sense, these systems use the same approach as ours does. Desktop On-Call enables the user to see the image of a remote desktop from the cellular phone, but the user cannot manipulate the desktop. Rajicon[10] enables the user to see and manipulate the desktop of a remote PC. However, it is limited to accessing MS Windows systems, since the system also utilizes information captured from the MS Windows operating system, such as the positions of windows, to support minimizing and maximizing of the windows.

6 Discussions

The size of a frame is approximately 12.2k bytes in 120x130x8 bits. This results in 3.9 frames per second (fps) for a system with a transmission capability of 386k bps. However, since images consisting of a desktop are relatively simple (a small number of colors used and they have a great number of rectangular areas of the same color), this is almost the worst case and the speed is usually at least doubled. When the user moves the pointer, the frame rate is approximately 13.6 fps, since the size of a gray-scaled frame of 120x130x3 bits is approximately 3.51k bytes.

By incorporating Shortcuts, our advanced interface has successfully reduced the user effort required to move the viewport and to point. This mechanism can be easily extended to include text inputting. Since many operations of modern GUI applications can be executed using keyboard shortcuts, the time required to perform a specific task can be significantly reduced by registering key sequences to execute the necessary operations and viewing operations (panning and zooming) as one shortcut. Furthermore, pattern-matching techniques can be employed to improve the user interface by utilizing the properties of GUI applications. Typically, GUI applications on the same desktop share the same bitmaps. Examples include GUI buttons (such as “Close Window” and “Minimize Window”) and pull-down menus (such as “File” and “Edit”). Therefore, it will become easier to remotely press GUI buttons and open pull-down menus by first registering the bitmaps of those GUI components and later searching those components using pattern matching techniques. Consequently, the viewer can provide the user with intelligent mapping. This enables the user to quickly move the viewport to the GUI components that the user wants to manipulate. This pattern matching can be combined with the shortcut mechanism. The result can be considered as an extension of [15].

7 Conclusions with Future Work

We have proposed a system to remotely access a computer desktop using only a cellular phone, despite the physical and bandwidth limitations of cellular phones. The system has a VNC-based architecture for accessing a remote desktop from the cellular phone. A proxy is placed to convert different devices, to suppress network traffic, and to support recovery from an unscheduled disconnection. To increase user-friendliness and to solve the problem of the small screen, several functions are provided on the cellular viewer. Frequently used screen areas of the desktop can be registered and quickly restored quickly by using a Shortcut assignment. The Guidance function can be used to show the Shortcut assignments. Two areas of the desktop display can be viewed simultaneously using the Twin view function. We have implemented a prototype of this system using Java, and checked the operation on a Java-enabled cellular phone emulator.

Currently, we are extending our implementation to support incremental updating of the SVNC viewer image, to speed up the frame rate and to incorporate more intelligent navigation. We are also trying to provide integrated panning and zooming of the viewport to simplify these basic operations, by applying speed-dependent automatic zooming[16].

Acknowledgments

We are greatly indebted to Dr. Motoki Miura for his cooperation and numerous discussions on the subject of this work.

References

[1] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper. Virtual Network Computing. *IEEE Internet Computing*, 2(1), 1998, 33–38.

[2] A. Marcus, J. V. Ferrante, T. Kinnunen, K. Kuutti, and E. Sparre. Baby Faces: User-Interface Design for Small Displays. In *Proc. the conference on CHI 98 summary: human factors in computing systems*, ACM Press, April 1998, 96–97.

[3] R. W. Scheifler. *X Window System Protocol X Consortium Standard X Version 11, Release 6.4*. X Consortium, Inc.

[4] Microsoft Corporation. *Windows 2000 Server Remote Desktop Protocol (RDP) Features and Performance White Paper*, June 2000.

[5] Microsoft Corporation. *Windows 2000 Terminal Services: An Integrated, Server-Based Computing Solution*, September 1999.

[6] T. Richardson and K. R. Wood. The RFB Protocol Version 3.3. AT&T Laboratories Cambridge, January 1998.

[7] T. Haraikawa, T. Sakamoto, T. Hase, T. Mizuno, and A. Togashi. μVNC: A Proposal for Internet Connectivity and Interconnectivity of Home Appliances based on Remote Display Framework. *IEEE Transactions on Consumer Electronics*, 47(3), August 2001, 512–519.

[8] A. Hasegawa and T. Nakajima. A User Interface System for Home Appliances with Virtual Netowrk Computing. In *Proc. IEEE International Workshop on Smart Appliances and Wearable Computing (IWSAWC2001)*, April 2001.

[9] Harakan Software. PalmVNC: Virtual Network Computing Client for Palm Platform. http://www.harakan.btinternet.co.uk/PalmVNC/.

[10] N. Su, M. Tsukamoto, and S. Nishio. Rajicon: A System for Remote PC Access through a Cellular Phone. In *Proc. MultiMedia, Distributed, Cooperative and Mobile Symposium (DICOMO 2001)*, vol. 2001 of *IPSJ Symposium Series*. Information Processing Society of Japan, June 2001, 349–354.

[11] Sun Microsystems. J2ME(TM) Connected Limited Device Configuration Specification 1.0a, May 2000.

[12] H. Okada, K. Kato, T. Ikegami, Y. Tatsumi, and T. Asahi. Proposal of a PC Remote Control System by Mobile Devices. In *IPSJ SIG Notes*, volume 93 of *Human Interface*, Information Processing Society of Japan, May 2001, 1–6. (in Japanese).

[13] INPROBE Networks Inc. CAFEMOON@HOME. http://www.cafemoon.org/.

[14] IMB Corporation. *Desktop On-Call Version 5 Users' Guide*, 1.0 edition, November 2001. (in Japanese).

[15] K. Yamamoto. A Programming Method of Using GUI as API. *IPSJ Transactions on Programming*, 39(SIG1(PRO1)), December 1998, 26–33. (in Japanese).

[16] T. Igarashi and K. Hinckley. Speed-Dependent Automatic Zooming for Browsing Large Documents. In *Proc. the ACM Symposium on User Interface Software and Technology*, ACM Press, November 2000, 139–148.