1. XIP: A simple image processing package

XIP is a basic image processing system that permits the user to process and display images. The supported functions are limited to those required by the homework. XIP is made available to you so that you can quickly and easily verify the correctness of your homework solutions. It consists of a graphical user interface (GUI) based on the X Window System that permits you to invoke any of the homework solutions on any image of your choice and display the output on the screen.

The purpose of these notes is to describe the XIP user interface. The role of the interface is to allow the user to interactively invoke the small library of image processing functions. A typical session consists of a series of repeated actions: select an input image, specify the position of the output image, select a command (image operation), and supply any needed parameters. It is not hard to see that the screen can quickly become cluttered if images are allowed to appear at any location. Furthermore, continually respecifying the input image and repositioning the output image is tedious and cumbersome. Therefore, one of the main goals of the XIP user interface is to enforce an orderly and uncluttered workspace, and relieve the user of the tedious task of managing images on the screen.

2. Screen Layout

The XIP interface divides the entire screen area into three major regions: the menu display, dialog box, and image display regions. Commands are issued to the system from the menu display region in the upper-left part of the screen. When additional information is necessary from the user, or when messages must be relayed to the user, the communication takes place in the dialog box in the lower-left part of the screen. This is essentially a pseudo-xterm window, in which ordinary text can be entered and printed. Both the menu display and dialog box share the same width. The remaining part of the screen is devoted to the image display region.

Commands can be invoked by using either the mouse or the keyboard. For instance, any command can be invoked by positioning the cursor over the desired operation and clicking once with the left mouse button. Alternatively, a command can be invoked by typing the appropriate keystroke, as identified by the single character at the right end of each command label. Since this does not require the need to reach over for the mouse and position it properly over the selection, users often find this mode more expedient. In fact, the term keyboard accelerator is given to this approach because many advanced users find it faster to use keyboard shortcuts instead of menus.

In order to enforce an orderly and uncluttered workspace, XIP partitions the image display region into non-overlapping rectangular tiles, or windows, each reserved for holding one image. The windows are identified by a numbering system that increases from left to right, and top to bottom. By default, the screen is split into 9 windows, although up to 64 windows may be specified by the user at program startup. The dimensions of the windows are automatically computed and printed in the main titlebar that spans the screen. Note that the number of windows cannot be changed once XIP begins.
3. Window Titlebar

Each window has a titlebar on top that specifies its status. The general structure of the titlebar is shown below.

```
INP wxh <num> x,y
```

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
</tr>
<tr>
<td>N</td>
</tr>
<tr>
<td>P</td>
</tr>
<tr>
<td>w</td>
</tr>
<tr>
<td>h</td>
</tr>
<tr>
<td>num</td>
</tr>
<tr>
<td>x,y</td>
</tr>
</tbody>
</table>

The first three characters in the titlebar are reserved for symbols I, N, and P, in left-to-right order. The I symbol denotes that window which contains the input image. We shall refer to it as window I. Any subsequent command requiring an input image will take it from window I. A dash appears in place of the I if the window does not contain an input image. There can only be one window I among all the windows. This implies that there can only be one input image defined at any one time. Operations that require more input images (i.e., arithmetic operations) will request the user to specify all the necessary input images at the time the command is invoked.

The N symbol specifies the next window into which an image will be displayed. We shall refer to it as window N. Note that the next displayed image can either refer to the next output or the next image read into the system. Any image already in window N will be overwritten. A dash appears in place of N if the window is not the target for the next displayed image. As with window I, there can only be one window N on the screen.

Images which do not fit in the window are automatically clipped to the window by the X Window System. In that case, a P appears to indicate that the displayed image is only partially exposed. Note that clipping occurs only at the display; there is no destructive effect on the image actually stored internally.

4. Mouse Buttons

The left-to-right ordering of the I, N, and P symbols has been made to conveniently correspond to the left, middle, and right mouse buttons, respectively. The I, N, and P status of any window can be changed by positioning the cursor on the window of interest and clicking the appropriate mouse button. For instance, any image can be chosen to be the next input image by moving the cursor to the desired window and clicking on the left mouse button. The change in window I will be reflected in the titlebars: the old window I will have its I flip to a dash, and the new window I will have its left dash flip to an I. Similarly, the next output image can be placed in any desired window (including window I) by clicking the middle button while the cursor is over the desired window. This will cause the middle symbol to flip from an N to a dash in the previous window N, and from a dash to an N in the new window N.
Images which are partially exposed (clipped) can be made fully exposed by clicking on the right button. This will cause the window to be resized to the image dimensions and allow the image to be fully displayed. Although this inevitably causes overlapping windows (and clutter), it permits the user to freely deal with images of arbitrary dimensions. If the user desires to reclip the image and have the window restored to its initial dimensions, the right mouse button must be clicked again. Therefore, repeated clicking on the right button toggles the P symbol for windows containing images larger than the window dimensions.

5. Command Selection

Most image processing commands are of the image-in/image-out variety. That is, it becomes necessary to specify window I and window N for virtually all operations. In order to limit the amount of tedious user interaction necessary to specify these windows, the following rules are adopted. Window I remains the same unless explicitly changed by the user with the left mouse button. Window N, on the other hand, cycles around the screen, allowing the output image to be displayed in the next available window. It, however, is not permitted to overwrite window I, thus leaving the input image intact. Therefore, if there are 9 windows in the image display region and the input image is currently in window 6, subsequent image operations will display its output in windows 7, 8, 0, 1, 2, ..., 4, 5, 7, 8, 0, 1, etc.

These rules imply that the same input image will undergo several different operations, or perhaps the same operation with different parameters. Meanwhile, the output will cycle around the windows so that side-by-side comparisons can be made. Once the user is satisfied with an operation, a different image may be explicitly chosen as the input image. This saves a lot of energy on the part of the user.

Windows I and N must be set properly before a command is selected. (There is no opportunity to set them later, even during parameter collection). A command is selected by positioning the cursor over the desired operation and clicking once with the left mouse button. The menu display is then redrawn with a lighter typeface to indicate that it is now disabled. If the command needs further input, it will interact with the user via the dialog box. Once the operation begins, a clock cursor appears to request you to wait while the input image is being processed. After the processing is completed, the output image is drawn in window N, N moves on to the next available window (passing over window I), and the menu is redrawn with the original solid typeface to indicate that it is now enabled and ready to accept the next command.

6. Image Scrolling

XIP supports image scrolling, allowing the user to position an image under a window in a physically intuitive way. This enables the user to pan across an image when only a portion of it is visible. An image can be partially visible if it is clipped, larger than the screen, or partially hidden by another window. The image can be scrolled in one of three modes: mouse, scroll key, and <shift> scroll key. All of these image scrolling modes are entered by pressing the <ctrl> key. The cursor icon changes shape to signal this event.

6.1. Using the mouse

The fastest way to scroll an image by an arbitrary amount is achieved by using the mouse. First, the cursor must be placed anywhere over the image. Next, the <ctrl> key must be pressed down. This changes the shape of the cursor and enables image scrolling. While holding the <ctrl> key down, the
image can be dragged around in the window by moving the mouse. The point under the cursor at the time the <ctrl> key was held down remains stationary relative to the cursor.

The coordinates of the image origin (upper-left corner) are listed in the window titlebar. These coordinates are relative to the \(x, y\) window coordinate system, where \(x\) increases from left-to-right, \(y\) increases from top-to-bottom, and the origin is at the upper left corner of the window. The titlebar always reflects the position of the image origin, even as the image is being dragged around. Note that once the cursor crosses a window boundary, image scrolling is disabled, as reflected in the cursor shape. This is necessary to avoid accidental displacements of neighboring images. In order to resume scrolling the original image, the cursor must be brought back into the window and the <ctrl> key must again be pressed down.

6.2. Using the scroll keys

While using the mouse may seem natural, it is not suitable for instances in which the user may want to align the edges or corners of the image with those of the window. This can be achieved more expediently by using the <ctrl> key together with one of the nine designated scroll keys. In order to keep things simple, the \(3 \times 3\) grid of keys on the left side of the keyboard have been selected to be the scroll keys:

\[
\begin{array}{ccc}
q & w & e \\
a & s & d \\
z & x & c \\
\end{array}
\]

The nine positions of these keys specify the nine directions in which an image can be moved: northwest (q), north (w), northeast (e), west (a), center (s), east (d), southwest (z), south (x), and southeast (c). Pressing <ctrl> with one of these keys while the cursor is in the window will align the specified corner/edge of the image to that of the window. For instance, pressing <ctrl>q aligns the upper-left corner of the image to the upper-left corner of the window, <ctrl>a aligns the left edge of the image with that of the window, <ctrl>s will center the image in the window, and <ctrl>c aligns the lower-right corner of the image with that of the window. In general ‘q’, ‘e’, ‘z’, and ‘c’ refer to the four image and window corners, while ‘a’, ‘w’, ‘d’, and ‘x’ refer to the four edges.

6.3. Using the <ctrl> <shift> scroll keys

Finer control is achieved by pressing the <ctrl> and <shift> keys together with one of the keys listed above. This moves the image one pixel at a time in the specified direction, as reflected in the titlebar. Note that ‘s’ has no function in this context. Although the mouse can be used for this purpose, it is generally difficult to achieve single pixel displacements that way. Therefore, the mouse is used for fast arbitrary displacements, the scroll keys are used for corner/edge alignment, and <ctrl> <shift> scroll keys are used for single-pixel displacements.

7. Parameter Collection

Now we discuss the method by which parameters are passed along to the system. At various times during the execution of a command, the user may be requested to supply parameters in the dialog box. The message prompts come in three formats:

Request for character string: <default>
In the first instance, a character string prompt is specified by two fields: the request and the default response. Defaults are always surrounded by `<' and `>', and are used whenever a carriage return is entered in response to a prompt.

In the second instance, a request is made for a yes or no response. This is evident from the value of the default field: y or n. These two characters are the only valid responses.

In the third instance, a request is made for an integer, real, or hex number. Notice that this prompt is specified by three fields: the request, a range for the response, and the default value. The min and max values are always surrounded by `[` and `]`. This is not to be confused with the possible brackets that may be contained in the request field. As long as the response does not lie within the range given (inclusive), the request is continually re-issued.

The distinction between an integer, real, or hex request is made as follows. If the number has any alphabetic characters between A and F (in upper or lowercase), it is in hex and a hex response is valid (i.e. 3F). If a decimal point appears in the number, then a real number is a valid response (an integer suffices as well). Any other number is an integer. If a real is given in response to an integer request, the fractional part is lost.

As the message at the bottom of the dialog box states, the escape (ESC) key can be pressed any time during parameter collection to quit the command. This gracefully returns the user to the state that existed before the command was selected. Note that the escape key cannot quit the command once the parameters have been supplied and the image operation is in progress.

8. A Digression: X Windows and Images

A little digression is in order here to explain how images are displayed on the screen. Although the user never really needs to know this, it does explain the differences between the actual images in memory and what you see on the display monitor. The user will learn that XIP attempts to enforce the ‘‘what you see is what you get’’ (WYSIWYG) paradigm. However, there are certain instances when that is not possible. This section explains why.

The earliest implementation of XIP did not run under any window system. It ran on an HP 9000 UNIX workstation that supported 24-bit true color images. Although the system was robust, it was not portable, at least in terms of image display. Special calls to the HP Starbase graphics library had to be made to display images and draw lines. Porting XIP to other systems required the device-dependent parts of the program to be modified considerably.

Now that the system has been rewritten to run under X Windows, that onerous task is avoided. The device-dependent parts of XIP have been rewritten to use Xlib and Xt Toolkit library functions. Although a flashier user interface could have been written had Motif been used, the decision to use such low-level libraries as Xlib and Xt was based on one overriding issue: portability. These libraries were more widely distributed than Motif (at least at the time of code development).

In X nomenclature, the user program (e.g., XIP) is called the client, and the program that controls the display is known as the server. The server always runs on your local machine, i.e., the machine you are sitting at. Since X Windows is a network-oriented windowing system, the client may run on a remote
machine. This permits XIP to sit on one (fast) machine, and become available to a large set of users on
the network.

While using X makes XIP portable, it also adds a level of complexity to the simple process of
displaying an image. This is true because images are displayed in windows, and windows may become
hidden and exposed by other windows. Furthermore, XIP must consider that the system may run on
workstations that support anything between 8-bit pseudocolor and 24-bit true color images. The discus-
sion in this section is intended to explain how this is achieved. It has a direct bearing on what you see on
the screen.

All XIP images are stored internally in what is called image memory. Each image has its own
memory allotment in the image memory space. Before an image can be displayed it must be copied from
image memory to a pixmap, X Window’s copy of that image. The requirement for every image window
to have its own pixmap is due to the possibility that any particular window may have to be redrawn
depending on the activities of other windows. For instance, when the user presses the right button to fully
expose an image which had been clipped, a new window is created that fits the image dimensions, the
image is re-displayed in that large window, and neighboring image windows become obscured. At some
later point if the user chooses to clip that large image again, the obscured windows must be redrawn.
Since it is too late to have XIP recompute the output images that lied there, copies must be maintained in
pixmaps. It is XIP’s responsibility to keep track of which images are exposed and pass those pixmaps (or
parts of them) to the frame buffer, a region of memory that is dedicated to the display. Everything you
see on the screen comes directly from the frame buffer.

8.1. Colormap Flicker

Users of pseudocolor displays may find the screen flicker annoyingly as the cursor is moved across
window boundaries. This effect, which we call colormap flicker, is not a bug. It is just a limitation of the
hardware to simultaneously accommodate the colormaps of all the windows. In order to understand this,
it is necessary to review the relationship between pixmaps, colormaps, and the frame buffer.

Recall that each window has its own pixmap and colormap. Taken as a pair, the output will appear
fine: the displayed image window is the result of the pixmap passing through its corresponding colormap.
The frame buffer, on the other hand, is a collection of all visible pixmap regions. It too has a single
colormap: the one which is associated with the window under the cursor. Since that colormap is
appropriate only for its own window, only that part of the frame buffer will appear properly colored on
the screen. All other windows will appear falsely colored. Therefore, when inspecting the contents of a
window, make sure to position the cursor above it so that the proper colormap may be used.

9. Command Line Arguments

XIP is invoked from the command line by typing xip. The following options are supported:

xip [ -help ] [ -verbose] [ -images n] [ -windows n] [ -visual name] [ -maxgray n]

- help displays this help message.
- verbose turns off verbose mode.
- images n partitions the image display region into n windows.
-windows $n$ partitions the image display region into $n$ windows.

-visual $\text{name}$ uses the named visual.

-maxgray $n$ sets the maximum gray level to be $n$ (for clipping).

If any of the above options are not specified, default values are used. By default, verbose mode is set, the screen is partitioned into nine windows, a pseudocolor visual is implied, and the maximum gray value is 255.

Once the session starts, the main menu will appear containing only three commands: “Read Image,” “User Ops,” and “Exit.” All of the supported homework functions are found in the “User Ops” submenu. You will typically want to begin by reading in an input image and then selecting commands in the “User Ops” submenu to process it. Make sure to visually compare your output with that of xip to verify the correctness of your homework solution.