Department of Electrical Engineering

State University of New York at Stony Brook

Stony Brook, New York 11794-2350

Stony Brook

Computer Vision Laboratory

Tech. Report 92.05.27 May 27, 1992

Image Defocus Simulator: A Software Tool

Ming-Chin Lu and Muralidhara Subbarao

Abstract

Image Defocus Simulator (IDS) is an interactive research software tool developed to simulate the image sensing/formation process in a typical CCD camera system. IDS takes as input the camera parameters and the scene parameters. It produces as output a digital image of the scene as sensed by the camera. IDS consists of a number of distinct modules each implementing one step in the computational model proposed in our previous work. The modular design of IDS also make it portable to almost any machine with a C compiler in it. IDS was created to give the researcher in the field of computer vision or image processing a flexible, interactive, and user-friendly environment in which he/she can analyze and/or synthesize images without putting too much effort in building the hardware equipments and/or setting up a laboratory. IDS will help in the verification of computer vision theories.

Image Defocus Simulator: A Software Tool [1]

User's Manual

Ming-Chin Lu and Murali Subbarao
Department of Electrical Engineering
State University of New York
Stony Brook, NY 11794

Abstract

Image Defocus Simulator (IDS) is an interactive research software tool developed to simulate the image sensing/formation process in a typical CCD camera system. IDS takes as input the camera parameters and the scene parameters. It produces as output a digital image of the scene as sensed by the camera. IDS consists of a number of distinct modules each implementing one step in the computational model proposed in our previous work. The modular design of IDS also make it portable to almost any machine with a C compiler in it. IDS was created to give the researcher in the field of computer vision or image processing a flexible, interactive, and user-friendly environment in which he/she can analyze and/or synthesize images without putting too much effort in building the hardware equipments and/or setting up a laboratory. IDS will help in the verification of computer vision theories.

[1]Copyright © 1992 by Research Foundation of SUNY

Acknowledgement: This work is supported in part by the National Science Foundation and the Olympus Optical Corporation.

1 Introduction

The Image Defocus Simulator (IDS) is an interactive/batch-mode research software tool to simulate the image formation process in a typical CCD camera system. IDS consists of a simulation engine and three user interfaces. The simulation engine is a machine-independent module to carry out all the computational steps involved in image sensing/formation process while the user interfaces are used to provide menu- or command-driven I/O interfaces.

IDS was created to give the researcher in the field of computer vision or image processing a flexible, interactive, and user-friendly environment in which he/she can analyze and/or synthesize images without putting too much effort in building the hardware equipments and/or setting up a laboratory. Researchers can use IDS to generate all the possible candidate images for a given object and a set of camera settings. After that, they can use the synthesized images to verify the computer vision theory under investigation and/or to debug the implementation of the theory.

Two graphics interfaces and a non-graphics interface are included in IDS. They are Sunview Graphics Interface (SGI), X window Graphics Interface (XGI), and Dummy TTY Interface (DTI). SGI and XGI provide menu-driven interfaces for users to interactively simulate the behavior of image defocus while DTI provides line mode interactive/batch simulation.

This report is organized as follows: Section 2 gives an overview of IDS; Section 3 describes the usage of IDS and currently supported platforms; Section 4 briefly describes the machine-independent simulation engine; Section 5 presents the graphics environments; Section 6 presents the line mode commands/environment; and finally, Section 7 concludes this report.

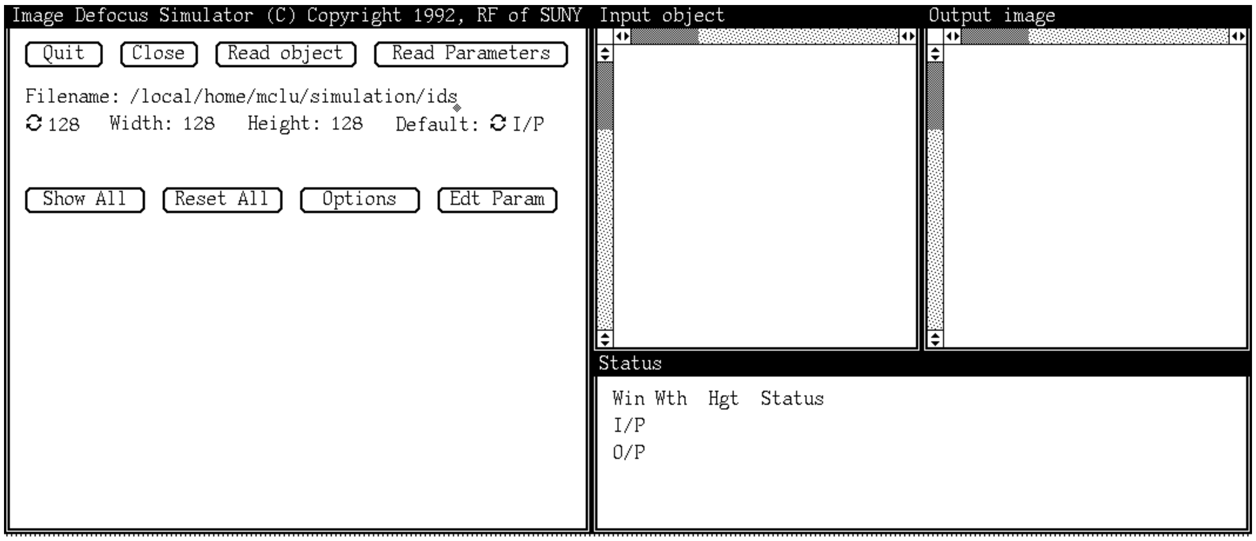
2 An Overview of IDS

IDS has a simulation engine and three user interfaces: SGI, XGI, and DTI. The only difference between SGI and XGI is that SGI runs under SunView environment while XGI runs under X window environment. The DTI interface is designed for users to run their simulations on a dummy terminal. It provides all the functions of IDS except the capability of displaying images and menus. The simulation engine and DTI are written in ANSI C; the SGI is written in SunView programming environment [7, 8]; while the XGI is written in Xlib and MIT X11R4 Athena Widget set [2, 3, 4, 5, 9].

In graphics environment (SGI or XGI), the input command is processed by the event handler; while in non-graphics environment (DTI), the input command is processed by the built-in LR(1) parser [1]. The syntax of the commands is listed in Appendix A. The processed event id or tokens are then interpreted and executed. The processed image can be inspected, printed using halftone algorithm, and saved for further processing.

Under SGI or XGI, users are provided with eight command buttons, two choices, and three editable text input areas at startup as shown in Figure 1 and Figure 2. All the I/O operations and the "Read Parameters" command take the string in the "Filename" text area as the target filename. The command "Read Parameters" reads a file which contains all the user-controllable parameters. These parameters are parsed through a built-in LR(1) parser to generate tokens and detect possible syntax errors. The parsing results are then interpreted by an interpreter to generate parameter values. These values can be modified by the "Edt Param" command which pops up a window as shown in Figure 3. Note that, the values of the parameters in Figure 3 are the default values at system startup.

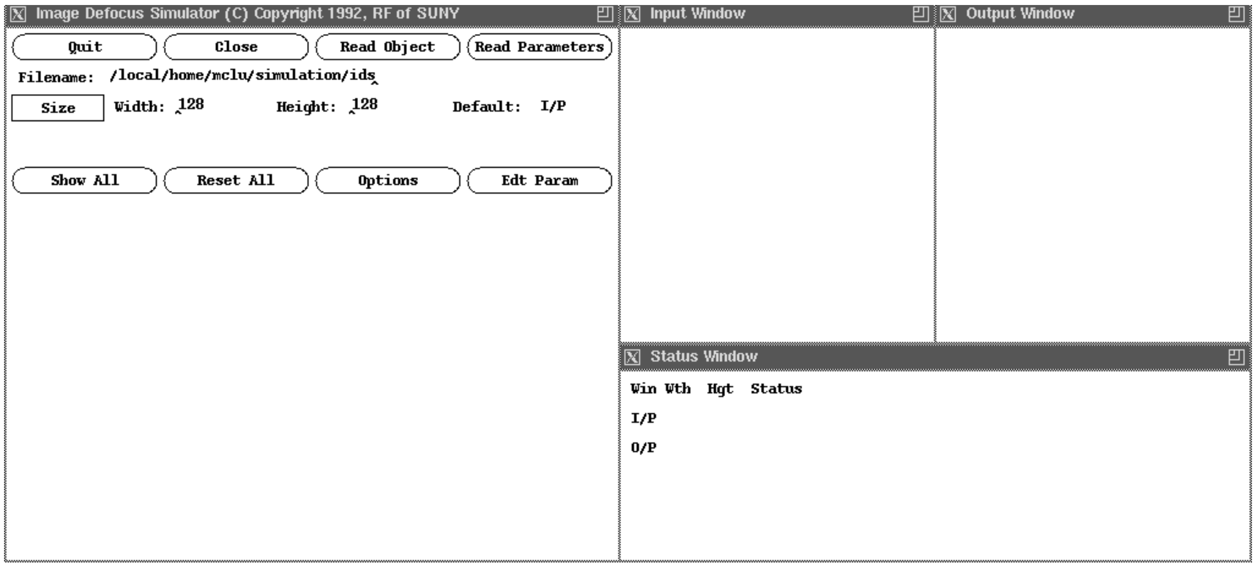
The "Options" command controls the system-wide options such as input/output image format, convolution method, and how to handle the image border. The input/output image format can be binary/ASCII integers, binary/ASCII floating numbers, or even a n-th order



Filename: /local/home/mclu/simulation/ids
128 Width: 128 Height: 128 Default: I/P

Win Wth Hgt Status
I/P
O/P

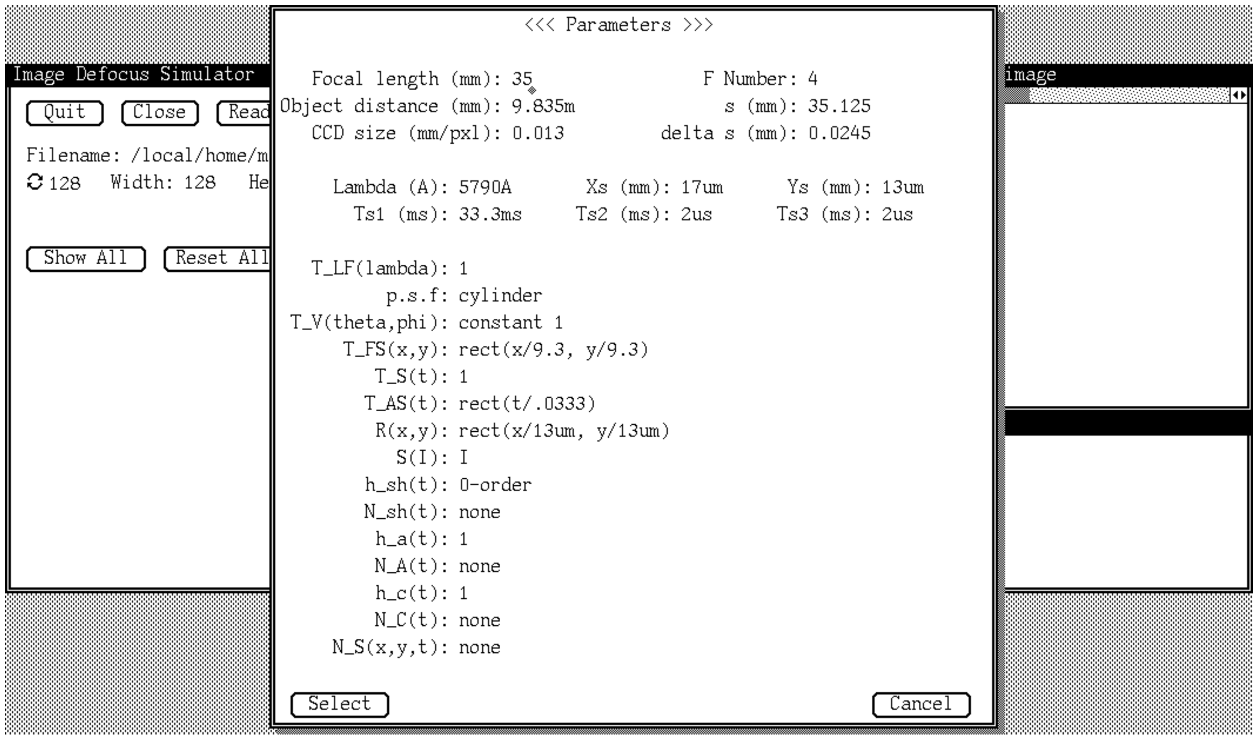
Figure 1: Startup screen of the simulation system under SGI



Filename: /local/home/mclu/simulation/ids
Width: 128 Height: 128 Default: I/P

Win Wth Hgt Status
I/P
O/P

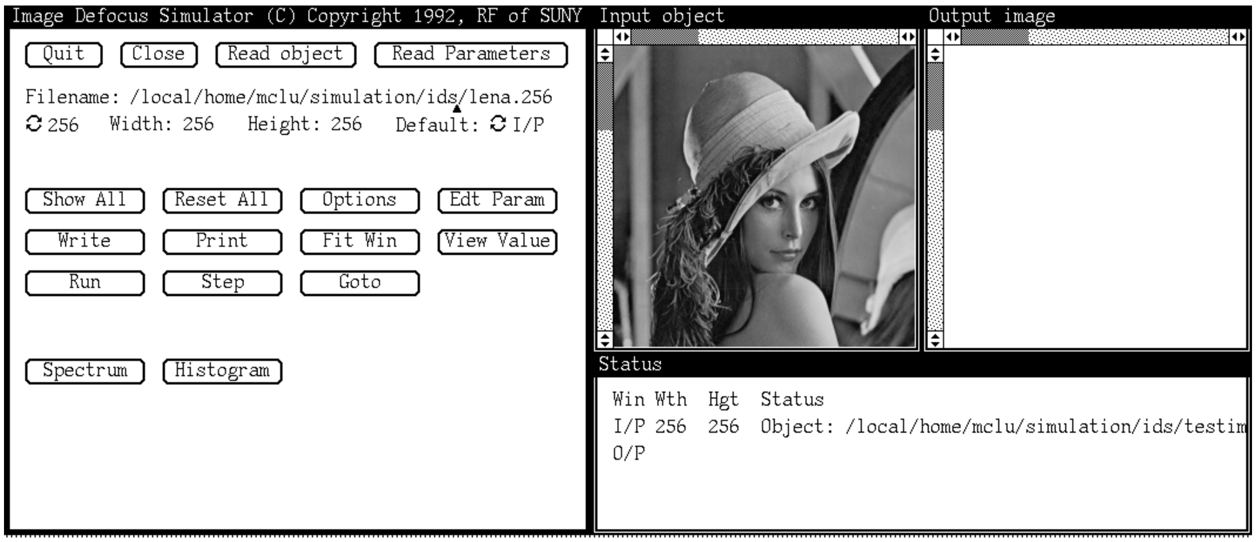
Figure 2: Startup screen of the simulation system under XGI



```
                    <<< Parameters >>>

  Focal length (mm): 35               F Number: 4
Object distance (mm): 9.835m            s (mm): 35.125
  CCD size (mm/pxl): 0.013         delta s (mm): 0.0245

     Lambda (A): 5790A      Xs (mm): 17um      Ys (mm): 13um
       Ts1 (ms): 33.3ms    Ts2 (ms): 2us      Ts3 (ms): 2us

   T_LF(lambda): 1
          p.s.f: cylinder
 T_V(theta,phi): constant 1
      T_FS(x,y): rect(x/9.3, y/9.3)
         T_S(t): 1
        T_AS(t): rect(t/.0333)
         R(x,y): rect(x/13um, y/13um)
           S(I): I
        h_sh(t): 0-order
        N_sh(t): none
         h_a(t): 1
         N_A(t): none
         h_c(t): 1
         N_C(t): none
     N_S(x,y,t): none
```

Figure 3: Popup Window invoked by "Edt Param" command button

polynomial (input only) specified by the order of the polynomial and the corresponding coefficients. The convolution method can be direct, FFT, or *smart* mode implementation. In direct (FFT) mode, no matter how big (small) the image/psf size is, direct (FFT) convolution is carried out; while in smart mode, the type of convolution used depends on the number of operations expected. This heuristic is used to speed up the computation of the convolutions. The image border can be treated as a zero-padded, mirrored, or periodic image during convolution. For "zero-padded" option, the image outside the field of view is simply treated as a dark area, *i.e.*, an area with zero values. For "periodic" option, the image is considered as $f(aM+m, bN+n) = f(m,n)$ for an $M \times N$ image where $a, b \in \{0, \pm 1, \pm 2, \cdots\}$. And for "mirrored" option, the image is first reflected along its right side border, then the resulting image is reflected along its top border; this gives an image which is four times larger than the original image. This four times larger image is then taken to be wrapped around at its



```
Filename: /local/home/mclu/simulation/ids/lena.256
256   Width: 256   Height: 256   Default: I/P
```

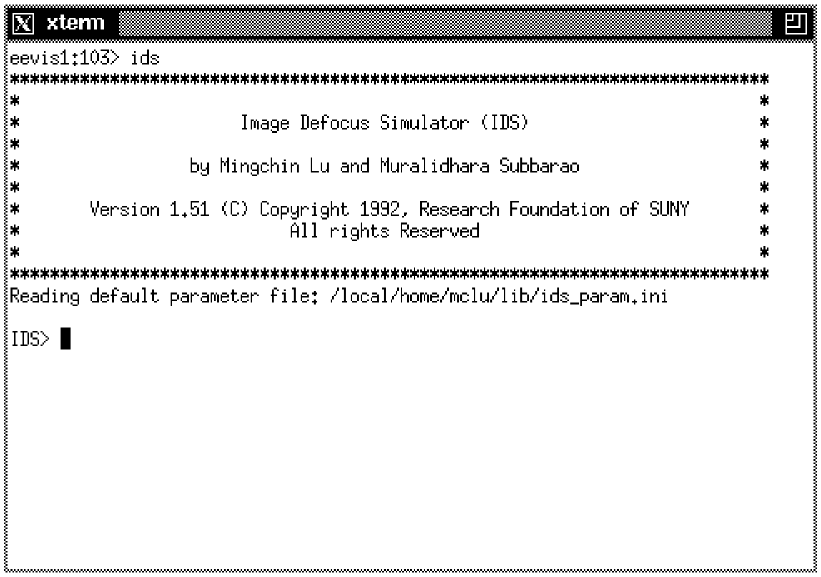
```
Win Wth  Hgt  Status
I/P 256  256  Object: /local/home/mclu/simulation/ids/testim
O/P
```

Figure 4: Full operation menu

borders. Note that, the "mirrored" option gives a periodic image whose period is twice that of the "periodic" option. After the input image and the parameters are loaded, additional operations are available as shown in Figure 4.

The "Run", "Step", and "Goto" commands control the execution of the simulation. Users can step or go to any particular step described in [6] to examine the output of a specific module. The value of the pixels of the image can be viewed via "View Value" command. We also provide Fourier spectrum and histogram analysis of a given image by using the "Spectrum" and the "Histogram" commands. During these two commands, a popup status window is displayed to keep track of the status of each spectrum or histogram popup window such as — when the operation was called and where the source image came from. The magnitude of the spectrum can also be inspected by simply pressing a mouse button. Finally, the synthesized image can be saved to a file and/or sent to printer for printing using the halftone algorithm.

The DTI runs on virtually any dummy terminal. Figure 5 is a typical run-time startup screen. Online help is available in this mode by typing "`help`" after the command prompt.



```
eevis1:103> ids
******************************************************************************
*                                                                            *
*                       Image Defocus Simulator (IDS)                        *
*                                                                            *
*                by Mingchin Lu and Muralidhara Subbarao                     *
*                                                                            *
*       Version 1.51 (C) Copyright 1992, Research Foundation of SUNY         *
*                            All rights Reserved                             *
*                                                                            *
******************************************************************************
Reading default parameter file: /local/home/mclu/lib/ids_param.ini

IDS> █
```

Figure 5: DTI user interface

3 Usage and System Requirements

Two command line options are provided in the current version: `-linemode` and `-openwin`. The first option enters line mode user interface. This is useful in batch mode simulation. Users can prepare an ASCII file which contains the line commands and then redirect that file to IDS to generate images in background. Appendix B gives such an example. The second option runs XGI on OpenWindow 2.0 or higher (default: MIT X11R4 with Athena Widget set). Note that, for XGI to work properly on OpenWindow 2.0 or higher, `-openwin` option must be explicitly specified.

The requirements to run the provided interfaces are listed in Table 1 where SUN 3, SUN 4, SUN SPARC, SunOS, OpenWindows, and SunView are trademarks of SUN Microsystems, Inc.; UNIX is a trademark of AT&T Bell Laboratories; VAX/VMS is a trademark of Digital Equipment Co.; X Window System is a trademark of the Massachusetts Institute of Technology; and VM/XA is a trademark of IBM, Inc. Table 2 lists the possible usage of IDS.

Interface	Hardware	Software
SGI	SUN 3, SUN 4 or SPARC	SunOS 4.0 or higher and SunView
XGI	terminal capable of running X window system	X Window System or OpenWindows 2.0 or higher
DTI	any terminal	UNIX or VAX/VMS or VM/XA

Table 1: System requirements to run different user interfaces

Environment		*Command*
Graphics	SunView	`ids`
	MIT X Window System	`ids`
	OpenWindows	`ids -openwin`
Non-Graphics		`ids`
Batch mode		`ids -linemode < filename &`

Table 2: Using IDS

4 Simulation Engine

Sixteen steps are included in the simulation engine to carry out the image sensing/formation process proposed in [6]. The input and output functions are listed here for reference. Note that, the input function of the current step is the output function of the previous step unless otherwise specified.

(Step.1) *Light filtering.* The input is the power of light of wavelength λ incident on the entrance pupil from the direction (θ, ϕ) at time t denoted as $f(\theta, \phi, \lambda, t)$. The I/O relationship is:

$$f_1(\theta, \phi, \lambda, t) = f(\theta, \phi, \lambda, t) \cdot T_{LF}(\lambda) \tag{1}$$

(Step.2) *Vignetting.*

$$f_2(\theta, \phi, \lambda, t) = f_1(\theta, \phi, \lambda, t) \cdot T_V(\theta, \phi) \tag{2}$$

(Step.3) *Optical system.* Using standard transformation from spherical to Cartesian coordinates, $f_2(\theta, \phi, \lambda, t)$ and the point spread function $h(\theta, \phi, \theta', \phi', r(\theta, \phi), \vec{e})$ can be equivalently represented as $f_2'(x, y, \lambda, t)$ and $h'(x, y, x', y', r(x, y), \vec{e})$, respectively. Hence, the output is:

$$f_3(x', y', \lambda, t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h'(x' - x, y' - y, r(x, y), \vec{e})\, f_2'(x, y, \lambda, t)\, dx\, dy \tag{3}$$

(Step.4) *Field stop.*

$$f_4(x, y, \lambda, t) = f_3(x, y, \lambda, t) \cdot T_{FS}(x, y) \tag{4}$$

(Step.5) *CCD spectral sensitivity.*

$$f_5(x, y, \lambda, t) = f_4(x, y, \lambda, t)\, T_s(\lambda) \tag{5}$$

(Step.6) *CCD transduction.*

$$f_6(x, y, t) = \int_{-\infty}^{\infty} f_5(x, y, \lambda, t)\, d\lambda \tag{6}$$

(Step.7) *Exposure.*

$$f_7(x,y,t) = \int_{-\infty}^{\infty} f_6(x,y,\tau)\, T_{AS}(\tau - t)\, d\tau \tag{7}$$

(Step.8) *CCD output.*

$$f_8(x,y,t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_7(\alpha,\beta,t)\, R(\alpha - x, \beta - y)\, d\alpha\, d\beta \tag{8}$$

(Step.9) *CCD sensor noise.*

$$f_9(x,y,t) = +f_8(x,y,t)\, n_s(x,y,t) \tag{9}$$

(Step.10) *Sampling.*

$$f_{10}[i,j,k] = f_9(i \cdot x_s, j \cdot y_s, k \cdot \tau_{s1}) \tag{10}$$

(Step.11) *Sensor response.*

$$f_{11}[i,j,k] = S(f_{10}[i,j,k]) \tag{11}$$

(Step.12) *Interpolation filter.*

$$\begin{aligned} f_{12a}[i + j \cdot M + k \cdot M \cdot N] &= f_{11}[i,j,k] \\ f_{12}(t) &= h_{sh}(t) \star \left[\sum_{l=0}^{K \cdot M \cdot N - 1} f_{12a}[l]\delta(t - l \cdot \tau_{s2}) \right] \end{aligned} \tag{12}$$

(Step.13) *Amplification.*

$$f_{13}(t) = f_{12}(t) \star h_a(t) + n_a(t) \tag{13}$$

(Step.14) *Cable noise.*

$$f_{14}(t) = f_{13}(t) \star h_c(t) + n_c(t) \tag{14}$$

(Step.15) *Sample and hold circuit.*

$$\begin{aligned} f_{15a}(t) &= \frac{1}{\tau_{s3}} \mathrm{comb}\left(\frac{t}{\tau_{s3}}\right) \cdot f_{14}(t) \\ f_{15b}(t) &= f_{15a}(t) \star h_{sh}(t) + n_{sh}(t) \\ f_{15}(t) &= \frac{1}{\tau_{s3}} \mathrm{comb}\left(\frac{t - \tau_{s3}/2}{\tau_{s3}}\right) \cdot f_{15b}(t) \end{aligned} \tag{15}$$

(Step.16) *Quantization.*

$$f_{16}[i] = Q\left(\int_{(i+\frac{1}{2})\tau_{s3}^{-}}^{(i+\frac{1}{2})\tau_{s3}^{+}} f_{15}(t)\,dt\right) \tag{16}$$

5 Graphics User Environment

5.1 Startup Screen

Under SGI and XGI, the startup screen is divided into three sections as shown in Figure 1. The control panel is located at left-hand portion of the screen and consists of input parameters, choice buttons, and command buttons. The status window is located next to the right of the control panel and contains output information concerning the input object and output image status. The upper right-hand portion of the screen contains two image display windows: input object window and output image window to display the object and the processed images, respectively. The appearance of the XGI is user configurable. Appendix C provides such an example.

5.2 Text Input Area

There are three text input areas located at the top of the control panel with their associated default values. To alter the value of a text input area, the user must first place the cursor over the current value and click the left button of the mouse, and then use the delete key to erase the previous value and type in the desired new value. The definitions of the text input areas are as follows:

1) **Filename** (Default value: current working directory name)

   Specifies the name of the object/image file to be read/written or the name of the parameter file to be read. Tilde character is recognized. Filename completion can also be done.

2) **Width** (Default value: 128)

   The width of the headerless image to be read/written. The image size can also be specified by using the choice button located to the left of the "Width" text input area.

3) **Height** (Default value: 128)

   The height of the headerless image to be read/written. The image size can also be specified by using the choice button located to the left of the "Width" text input area.

5.3 Choice Buttons

There are two choice button inputs. In SGI, the value of the choice button inputs can be altered by placing the cursor over the desired choice button and clicking the left button of the mouse. This will set the choice button to the next possible value. The choice button inputs can also be altered by placing the cursor over the desired choice button and pressing the right button of the mouse. This will give the user a pop-up menu from which he can choose one of the possible values of the choice button. In XGI, the value of the choice button inputs can be altered by placing the cursor over the desired choice button, pressing and holding the left button. This will give the user a pop-up menu from which he can choose one of the possible values for that choice button.

1) **Image Size** (Default value: 128×128)

   Chooses the image size of 32×32, 64×64, 128×128, 256×256, 512×512, and 1024×1024.

2) **Default** (Default value: Input object window)

   Chooses the default window that the command buttons apply to. This can also be changed by moving the cursor to the corresponding display window and clicking the left button of the mouse.

5.4 Command Buttons

At initialization, the available command buttons are “Quit”, “Close”, “Read object”, “Read Parameters”, “Show All”, “Reset All”, “Options” and “Edt Param”. Additional command buttons will be available when the object file is loaded. They are “Write”, “Print”, “Fit Win”, “View Value”, “Spectrum”, “Histogram”, “Run”, “Step”, and “Goto”. A command is selected by placing the cursor over the desired command button and clicking the left button of the mouse. The definitions of the command buttons are as follows:

1) **Quit**

   Exits from the IDS environment.

2) **Close**

   All windows become transparent to the user. The IDS Control Panel is iconized.

3) **Read object**

   Reads an image from the file given by the “Filename” text input area into the Input object window. The “Width” and “Height” input areas specify the image size.

4) **Read Parameters**

   Reads user defined parameter values from the file given by the “Filename” input area. The first four characters, serve as a magic number, in a parameter file must be “%!CS”. Each parameter takes one line. The format for each parameter is: “param_id= param_value”. Note that, there is NO space between param_id and the equal sign. The available param_id and param_value are described in the following Line Mode Commands section. Any line begins with the character % is treated as comment and ignored. During ids startup, the default parameter file “ids_param.ini” is read.

5) **Show All**

All display windows and status window become visible to the user.

6) **Reset All**

All windows are reset to their origin size and location.

7) **Options**

Sets the values for the spectrum display threshold, convolution border, convolution method, and I/O format. When this command button is pressed, a window is popped up which contains one text input area and three choices at the current version. The spectrum display threshold input area controls the display of the spectrum. If the value of the spectrum at a particular frequency is greater than the spectrum display threshold it is displayed in the popped up spectrum window as a foreground pixel. Otherwise, it is displayed as a background pixel. The convolution border can be mirroring (default), periodical, or zero-padded. The convolution method can be smart (default), FFT, or direct convolution as explained in the previous section. The I/O format specifies the data format when I/O operation is performed. The available I/O formats will be described later in Line Mode Commands section.

8) **Edt Param**

Sets the values for the parameters. When this command button is pressed, a window is popped up which contains all the parameters and two buttons: “Select” and “Cancel”. The possible values for each parameter will be described in Line Mode Commands section.

9) **Write**

Writes the image in the window specified by the “Default” choice to the file given by the “Filename” text input area. If the file already exists, an alert window will be popped up to ask for confirmation. However, no confirmation will be asked in DTI interface.

10) **Print**

Sends the image in the display window specified by the "Default" choice to the printer using halftone algorithm. When this command button is pressed, a window is popped up which contains a "Printer type" choice, a "Printer name" text input area, a "Reset" command button, and a "Cancel" command button. The available printer types are HP LaserJet III or compatibles and PostScript printer. The printer name parameter specifies the name of the printer as indicated in the /etc/printcap file. The printer name parameter must be terminated with a RETURN key to call up other input area, choice, and "Select" button to print the images. These additionally popped items specify the resolution of the printer, the size of the output image, and other information for printing the images.

11) **Fit Window**

The size of display window, specified by the "Default" choice, is set to the same size of the image that is located within the display window.

12) **View Value**

View the value and location of pixels within the display window specified by the "Default" choice. The coordinates and value of the pixel pointed by the cursor will be displayed below the "Width" and "Height" areas. To exit, click the right button of the mouse.

13) **Run**

Run the simulation.

14) **Step**

Run the simulation step by step.

15) **Goto**

Goto a specific step. When this command button is pressed, a text input area and two command buttons will pop up at the bottom of the control panel. The input area specifies the destination step while the buttons ask for "Execute" or "Cancel".

16) **Spectrum**

Computes the spectrum for the image in the display window specified by the "Default" choice. When the computation is done, a display window is popped up to display the spectrum. The DC component is in the center of the display window. A popup status window is also displayed at the bottom of the control panel to keep track of the status of each spectrum or histogram (described next) popup window such as when the operation was called and where the source image came from. This popup window will be automatically closed if all the popups are closed.

The magnitude of each frequency component can be viewed by moving the cursor to the popup window and clicking the left mouse button. The value and the frequency components will be displayed under the "Width" area in the control panel. To stop viewing value, move the cursor to the display window and click the right mouse button. To destroy the display window, move the cursor to the title bar, click and hold the right mouse button to choose quit operation (in SGI interface) or move the cursor to the display window and click the right mouse button (in XGI interface).

17) **Histogram**

Computes the histogram for the image in the display window specified by the "Default" choice. When the computation is done, a display window is popped up to display the histogram. A popup status window is also displayed at the bottom of the control panel to keep track of the status of each histogram or spectrum (described above) popup window such as when the operation was called and where the source image came from. This popup window will be automatically closed if all the popups are closed.

To destroy the display window, move the cursor to the title bar, click and hold the right mouse button to choose quit operation (in SGI interface) or move the cursor to the display window and click the right mouse button (in XGI interface).

5.5 Status Window

In the Status Window there is an output table that keeps track of the status of each display window. The execution status is also displayed at the bottom of this window.

1) **Wth**

   Indicates the width of the image in the display window.

2) **Hgt**

   Indicates the height of the image in the display window.

3) **Status**

   Indicates the status of the display window.

6 Line Mode Commands

Line mode provides users an environment to run their simulations on any TTY terminal. Users can also put their job to the background using this mode. All the commands and option keywords in this mode can be abbreviated to 1, 2 or 3 characters long if there is no conflict between any two commands. The available commands are (formal syntactical definition can be found in Appendix A):

1) **help**

   Displays the on-line help messages.

2) **exit**
   quit

   Exits from the IDS environment. In line mode, EOF is equivalent to quit.

3) **shell**
   ![shell_command]

   Invokes a C-shell session ("shell" or "!") or issues a shell command.

4) **echo string**

   Echoes a string "string" to stdout.

5) **read object filename width height**
   read parameters filename

   Reads an object or the parameters from the file named "filename".

6) **write filename [from x y [size width height]]**

   Writes the processed image to the file named "filename" from the position (x,y) with file size width by height. The default (x,y) is (0,0), i.e., left-upper corner. The default width and height are (image_width - x) and (image_height - y), respectively.

7) **view { object | image } x y**
   view spectrum x y [normalized]

   Displays the pixel value or the value of the spectrum, both magnitude and complex value, at (x,y) to stdout. When normalized is requested, the DC component is normalized to one. The frequency range is (-image_width/2) to (image_width/2 - 1) and (-image_height/2) to (image_height/2 - 1).

8) **show { status | options | parameters }**

Displays the current status information, values for the options, and values for the parameters to stdout.

9) **set parameter param_id param_value**
set option option_key

Sets the value for the parameter or option. The param_id and param_value are:

a) **f number**

Specifies focal length. The default unit is mm.

b) **u number [unit]**

Specifies object distance. The available units are: mm (default), cm, m, km, in[ch], ft, and mi[le].

c) **f/number number**

Specifies f/number.

d) **s { number | { step integer } }**

Specifies the second principle position. The default unit for "number" is mm. If the second format is used, the value in parameter "delta_s" is used to compute the s in mm. The formula used is: $s = f + (integer) * delta\_s$.

e) **delta_s number**

Specifies the delta s used for each camera step. The default unit is mm.

f) **Lambda number [unit]**

Specifies the wavelength. The available units are: um, mm, cm, and Å(default). Note that, Å= 10^{-8}cm.

g) **ccd_size number**

Specifies the CCD pixel size in mm/pixel.

h) **psf cylinder**

Specifies cylindrical point spread function. The blur circle radius is computed using geometric optics.

i) **psf delta(x,y)**

Specifies delta point spread function at (x,y).

j) **psf file filename width height**

Reads the point spread function from the file named filename. The size of the psf is width by height. The format is determined by I/O option.

k) **psf mtf file filename width height**

Reads the MTF from the file named filename. The size of the mtf is width by height. The format is determined by I/O option.

l) **psf gaussian(sigma_x, sigma_y)**

Specifies the gaussian point spread function with sigma_x and sigma_y.

m) **psf wave_optics**

Computes point spread function using wave behavior.

n) **Xs number [unit]**

Specifies the x sampling in spatial domain. The available units are: um, mm (default), cm, and in.

o) **Ys number [unit]**

Specifies the y sampling in spatial domain. The available units are: um, mm (default), cm, and in.

p) **Ts1 number [unit]**
Ts2 number [unit]
Ts3 number [unit]

Specifies the sampling information in time domain.

q) **T_LF number**

Specifies the transfer function of the light filtering.

r) **T_V constant number**
T_V gaussian(sigma_x, sigma_y)
T_V file filename width height

Specifies the transfer function of vignetting to be a constant, a gaussian mask with (sigma_x,sigma_y), or a file read from filename of size width by height. The file I/O type is determined by I/O option.

s) **T_FS rect(x/number, y/number)**

Specifies the transfer function of field stop.

t) **T_S number**

Specifies the spectral sensitivity of CCD sensor.

u) **T_AS rect(t/number)**

Specifies the transfer function of aperture stop.

v) **R rect(x/number, y/number)**

Specifies the area for integration over space, i.e., photo-sensitive area.

w) **S I**
S aI + b
S I^(number)

Specifies the sensor response (a, b are numbers; 'I' must stay as is).

x) **h_sh 0-order**
h_sh number
h_c number
h_a number

Specifies the transfer function for sample-and-hold circuit, amplifier, and cable.

y) **N_sh**
N_A
N_C
N_S

Specifies the sample-and-hold circuit (sh), amplifier (A), cable (C), and

CCD sensor additive (S) noise.

The value of **option_key** is:

a) **border { 0 | 1 | 2 }**

Sets the convolution border. 0 means mirroring (default); 1 means periodical; and 2 means zero-padded.

b) **method { 0 | 1 | 2 }**

Sets the convolution method to be used. 0 means smart (default); 1 means FFT; and 2 means direct convolution.

c) **io { 0 | 1 | 2 | 3 | 4 | 5 }**

Sets the I/O format to be used. 0 means unsigned char (default); 1 means ASCII integer; 2 means binary float with MSB first (SUN-4); 3 means binary float with LSB first (PC); 4 means ASCII float; and 5 means binary double with MSB first (SUN-4).

10) **set object x y value**

Sets the value at pixel (x,y) to "value".

11) **run**

Run the simulation.

12) **run spectrum { object | image }**

Computes the spectrum for the input object or the output image.

13) **step**

Run the simulation step by step.

14) **goto step**

Goto a specific step.

7 Conclusion

The advantages of modular design and the decoupling of simulation engine and user interfaces make IDS an extensible and powerful tool in simulating the image sensing/formation process in computer vision applications. IDS is machine independent and can be very easily ported to many platforms. IDS can be extended to simulate the images sensed in stereo-vision applications. This topic is currently being investigated.

Appendix A Syntax

A.1 Command Syntax

The syntax of the commands in IDS can be expressed as the following context-free grammar where non-terminals are enclosed by left- and right-angle pairs.

```
⟨help⟩        ::= help
⟨exit⟩        ::= exit | quit
⟨shell⟩       ::= shell | { ![ ⟨shell_command⟩ ] }
⟨echo⟩        ::= echo ⟨string⟩
⟨read⟩        ::= read { { object ⟨filename⟩ ⟨width⟩ ⟨height⟩ } |
                         { parameters ⟨filename⟩ } }
⟨write⟩       ::= write ⟨Filename⟩ [ from ⟨x⟩ ⟨y⟩ [ size ⟨width⟩ ⟨height⟩ ] ]
⟨view⟩        ::= view { { { object | image } ⟨x⟩ ⟨y⟩ } |
                         { spectrum ⟨xf⟩ ⟨yf⟩ [ normalized ] } }
⟨show⟩        ::= show { status | options | parameters }
⟨set⟩         ::= set { { parameter ⟨param_id⟩ ⟨param_value⟩ } |
                        { option ⟨option_key⟩ } | { object ⟨x⟩ ⟨y⟩ ⟨gray_value⟩ } }
⟨run⟩         ::= run [ spectrum { object | image } ]
⟨step⟩        ::= step
⟨goto⟩        ::= goto ⟨step_num⟩
⟨step_num⟩    ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16
⟨x⟩           ::= 0 | ⟨positive_integer⟩
⟨y⟩           ::= 0 | ⟨positive_integer⟩
⟨xf⟩          ::= ⟨x⟩ | -⟨x⟩
⟨yf⟩          ::= ⟨y⟩ | -⟨y⟩
```

⟨gray_value⟩	::= 0..255
⟨param_id⟩	::= ⟨optical_info⟩ | ⟨obj_info⟩ | ⟨samp_info⟩ | ⟨CCD_info⟩ | ⟨misc_info⟩
⟨optical_info⟩	::= `f` | `s` | `f/number` | `delta_s` | `psf` | `Lambda`
⟨obj_info⟩	::= `u`
⟨samp_info⟩	::= `Ts2` | `Ts3`
⟨CCD_info⟩	::= `T_S` | `T_AS` | `Xs` | `Ys` | `Ts1` | `R` | `S` | `ccd_size`
⟨misc_info⟩	::= `T_LF` | `T_V` | `T_FS` | ⟨noise⟩ | ⟨transfer⟩
⟨noise⟩	::= `N_sh` | `N_A` | `N_C` | `N_S`
⟨transfer⟩	::= `h_sh` | `h_a` | `h_c`
⟨option_key⟩	::= ⟨border⟩ | ⟨method⟩ | ⟨io⟩
⟨border⟩	::= `border` { ⟨mirroring⟩ | ⟨periodical⟩ | ⟨zero_pad⟩ }
⟨method⟩	::= `method` { ⟨smart⟩ | ⟨fft⟩ | ⟨direct⟩ }
⟨io⟩	::= `io` { ⟨ByteChar⟩ | ⟨ASCInt⟩ | ⟨BinFltMSB⟩ | ⟨BinFltLSB⟩ | ⟨ASCFloat⟩ | ⟨BinDblMSB⟩ }
⟨mirroring⟩	::= 0
⟨periodical⟩	::= 1
⟨zero_pad⟩	::= 2
⟨smart⟩	::= 0
⟨fft⟩	::= 1
⟨direct⟩	::= 2
⟨ByteChar⟩	::= 0
⟨ASCInt⟩	::= 1
⟨BinFltMSB⟩	::= 2
⟨BinFltLSB⟩	::= 3
⟨ASCFloat⟩	::= 4
⟨BinDblMSB⟩	::= 5
⟨width⟩	::= 0..1024
⟨height⟩	::= 0..1024

A.2 Parameter Syntax

The syntax of the user adjustable parameters in IDS can be expressed as the following rules.

Focal length (mm)	:	`f=`⟨number⟩
F/number	:	`f/number=`⟨number⟩
s(mm)	:	`s=`⟨number⟩
		`s=step` ⟨integer⟩

Delta s (mm)	: `delta_s=`⟨number⟩
Wavelength Lambda (A)	: `Lambda=`⟨number⟩[⟨unit⟩]
CCD pixel size (mm/pixel)	: `ccd_size=`⟨number⟩
Object distance (mm)	: `u=`⟨number⟩[⟨unit⟩]
psf	: `psf=cylinder`
	`psf=delta(`⟨i⟩,⟨j⟩`)`
	`psf=file` ⟨filename⟩ ⟨width⟩ ⟨height⟩
	`psf=gaussian(`⟨sigma_x⟩,⟨sigma_y⟩`)`
	`psf=wave_optics`
Light Filtering	: `T_LF(lambda)=`⟨number⟩
Vignetting	: `T_V(theta,phi)=constant` ⟨number⟩
	`T_V(theta,phi)=gaussian(`⟨sigma_x⟩,⟨sigma_y⟩`)`
	`T_V(theta,phi)=file` ⟨filename⟩ ⟨width⟩ ⟨height⟩
Field Stop	: `T_FS(x,y)=rect(x/`⟨number⟩, `y/`⟨number⟩`)`
Sensor spectral sensitivity	: `T_S(lambda)=`⟨number⟩
Aperture stop	: `T_AS(t)=rect(t/`⟨number⟩`)`
Integration over space	: `R(x,y)=rect(x/`⟨number⟩, `y/`⟨number⟩`)`
	`R(x,y)=file` ⟨filename⟩ ⟨width⟩ ⟨height⟩
Sampling in spatial domain	: `Xs=`⟨number⟩[⟨unit⟩]
	`Ys=`⟨number⟩[⟨unit⟩]
Sampling in time domain	: `Ts1=`⟨number⟩[⟨unit⟩]
	`Ts2=`⟨number⟩[⟨unit⟩]
	`Ts3=`⟨number⟩[⟨unit⟩]
Sensor response	: `S(I)=I`
	`S(I)=I^`⟨number⟩
Sample-and-hold	: `h_sh(t)=0-order`
	`h_sh(t)=`⟨number⟩
Amplifier	: `h_a(t)=`⟨number⟩
Cable	: `h_c(t)=`⟨number⟩

Appendix B An Example Batch Mode Session

```
read object lena.128 128 128
read parameters ids_param.ini
set parameter delta_s 0.024500
set option border 1
% Object at step 10
set parameter u 5035.0000mm
set parameter s step 10
```

```
run
write r10s10.img
set parameter s step 40 run
write r10s40.img
set parameter s step 70 run
write r10s70.img
% Object at step 30
set parameter u 1701.6666mm
set parameter s step 10
run
write r30s10.img
set parameter s step 40
run
write r30s40.img
set parameter s step 70
run
write r30s70.img
% Object at step 50
set parameter u 1035.0000mm
set parameter s step 10
run
write r50s10.img
set parameter s step 40
run
write r50s40.img
set parameter s step 70
run
write r50s70.img
% Object at step 70
set parameter u 749.2857mm
set parameter s step 10
run
write r70s10.img
set parameter s step 40
run
write r70s40.img
set parameter s step 70
run
write r70s70.img
% Object at step 90
set parameter u 590.5555mm
set parameter s step 10
run
write r90s10.img
set parameter s step 40
run
write r90s40.img
set parameter s step 70
run
write r90s70.img
exit
```

Appendix C An Example X Resource File for IDS

```
!  default settings for ids
!
!  Put the following line in $HOME/.cshrc
!  setenv XAPPLRESDIR $HOME/lib
!
*Background:  lightblue
*StatusWindow*background:  lightblue
*StatusWindow*foreground:  black
*highlightThickness:  3
*Quit*background:  aquamarine
*Close*background:  aquamarine
*ReadObject*background:  aquamarine
*ReadParameters*background:  aquamarine
*ShowAll*background:  aquamarine
*ResetAll*background:  aquamarine
*Options*background:  aquamarine
*EdtParam*background:  aquamarine
*Write*background:  khaki
*Print*background:  khaki
*FitWin*background:  khaki
*ViewValue*background:  khaki
*Run*background:  plum
*Step*background:  plum
*Goto*background:  plum
*Histogram*background:  plum
*Spectrum*background:  plum
```

```
*X*foreground:  red
*Y*foreground:  red
*Value*foreground:  red
*yes*background:  red
*no*background:  red
*ParamSelect*background:  red
*ParamCancel*background:  red
*PrintSelect*background:  red
*PrintReset*background:  red
*PrintCancel*background:  red
*OptionsSelect*background:  red
*OptionsCancel*background:  red
!
!  shape settings (Not Settable)
!
!  font settings
!
*dialog*icon.foreground:  red
*dialog*label*font:  *charter-bold-i-normal--17-120*
*dialog*value*font:  *new century schoolbook-bold-i-normal--14-100*
!
!  cursor settings (Not Settable)
!
!  Functional Resources
!
!  Main Menu translations
!
*menubox.Command.translations:\
   <EnterWindow>:  highlight() \n\
   <LeaveWindow>:  reset() \n\
   <BtnUp>:  set() notify() unset()
*menubox.menupane5.translations:\
   <LeaveWindow>:  checkRightAndPopupSubmenu()
!
!  Sub Menu translations
!
*subbox.translations:\
   <LeaveWindow>:  checkLeftAndPopdownSubmenu(subbox)
*subbox.Command.translations:\
   <EnterWindow>:  highlight() \n\
   <LeaveWindow>:  reset() \n\
⟨BtnUp⟩:  set() notify() unset()
```

References

[1] A. V. Aho, R. Sethi, and J. D. Ullman, *Compilers – Principles, Techniques, and Tools,* Addison-Wesley, Massachusetts, 1986.

[2] A. Nye, *Xlib Programming Manual,* Vol. 1, O'Reilly & Associates, Inc. 1989.

[3] A. Nye, (editor) *Xlib Reference Manual,* Vol. 2, O'Reilly & Associates, Inc. 1989.

[4] A. Nye and T. O'Reilly, 196z198z*X Toolkit Intrinsics Programming Manual,* Vol. 4, O'Reilly & Associates, Inc. 1989.

[5] T. O'Reilly, (editor) *X Toolkit Intrinsics Reference Manual,* Vol. 5, O'Reilly & Associates, Inc. 1989.

[6] M. Subbarao and M.-C. Lu, "Computer Modeling and Simulation of Camera Defocus," Technical Report No. TR.92.01.16, Computer Vision Laboratory, Department of Electrical Engineering, State University of New York, Stony Brook, 1992.

[7] " SunView Programmer's Guide," SUN Microsystems, Inc., Mountain View, CA, March 1990.

[8] " SunView System Programmer's Guide: Pixrect Reference," SUN Microsystems, Inc., Mountain View, CA, March 1990.

[9] V. Quercia and T. O'Reilly, *X Window System User's Guide,* Vol. 3, O'Reilly & Associates, Inc. 1989.