Royal Institute of Technology
Stockholm, Sweden

University of Stockholm
Sweden

DEPARTMENT OF NUMERICAL ANALYSIS
AND COMPUTING SCIENCE

CanApp — the Candela Application Library

by
Gårding, Lindeberg

CVAP–TN02

Technical Note from Computer Vision and
Associative Pattern Processing Laboratory

CanApp—the Candela Application Library

Jonas Gårding and Tony Lindeberg
Computer Vision and Associative Pattern Processing Laboratory (CVAP)
Royal Institute of Technology, S-100 44 Stockholm, Sweden

November 28, 1989

Abstract

This paper describes *CanApp*, the Candela Application Library. CanApp is a software package for image processing and image analysis. Most of the functions in CanApp are available both as stand-alone programs and as C subroutines.

CanApp currently comprises some 50 programs and 75 subroutines, and these numbers are expected to grow continuously as a result of the joint efforts of the members of the CVAP group at the Royal Institute of Technology in Stockholm.

CanApp is currently installed and running under UNIX on Sun workstations.

1 Introduction

This paper describes *CanApp*, the Candela Application Library. CanApp is a software package for image processing and image analysis. Most of the functions in CanApp are available both as stand-alone programs and as C subroutines.

CanApp currently comprises some 50 programs and 75 subroutines, and these numbers are expected to grow continuously as a result of the joint efforts of the members of the CVAP group at the Royal Institute of Technology in Stockholm.

CanApp is based on the *Candela* [1] subroutine package for low-level image handling. It is currently implemented and running under the UNIX operating system.

1.1 Background

The main purpose of CanApp is to facilitate the sharing of image processing software among the individual members of the CVAP research group. CanApp achieves this by providing some standardization guidelines to be used by each program developer and a minimal "central administration" for maintaining the library and the documentation.

CanApp was created to change a situation where each user had developed his own programming standards, making the sharing of software very difficult or even impossible.

All the software in CanApp has been developed for some specific purpose, in most cases by the person who needed it—no program or subroutine has been written for the sole purpose of putting it in the library.

Although CanApp was initially meant for "internal use" in the CVAP group, it now has many other users as well, including undergraduate students.

2 Organization of the library

The CanApp library has three parts:

- the program library
- the C subroutine library
- the online documentation library

2.1 The program library

The program library consists of stand-alone programs, all beginning with the three letters "can". The programs are either compiled C programs or executable shell scripts. Most programs are simple filters that take an image as input and produces another image as output.

These programs can be connected conveniently by means of the pipe mechanism in UNIX. Assume for example that we want to histogram equalize an image, then apply the Canny-Deriche edge detector, and finally convert the image to PostScript and print it. This can all be done with a single command:

```
canhisteq x.can | canderiche | can2ps | lpr
```

All programs accept switches in the standard syntax. Assume that we want to supply the parameter $\alpha = 2.0$ to "canderiche", and that we want to draw a frame around the printed image and add the title "Example 1". This is done with the following command:

```
canhisteq x.can | canderiche -a 2.0 | can2ps -f -T "Example 1" | lpr
```

The available switches are listed in the documentation for each program. All programs accept the switch "–d" (meaning "debug"), which makes the program print out tracing information during the execution.

2.2 The subroutine library

With a few exceptions it is always required that a program that is added to the library is also added as a subroutine to the subroutine library. In many cases there will actually be more than one subroutine for each program, since the program may perform several more or less independent steps.

There is no standard naming scheme for the subroutines, but care is always taken to avoid name collisions with other libraries.

The name of the CanApp subroutine library is simply "canapp", and this library depends on the libraries "candela" (low level image handling) and "m" (standard mathematical functions). To compile and link a program that calls CanApp routines, the command is thus

```
cc foo.c -o foo -lcanapp -lcandela -lm
```

2.3 The online documentation library

No program or subroutine is put in the library unless it is properly documented, which in this context means that there is a "man page" in standard UNIX format that can be read with the "man" command and printed with "troff".

The name of the man page is always the same as the program or the subroutine. In the cases where several subroutines are documented in the same paper, the name of the man page is the major (or first) subroutine.

There are also a few pages of documentation that apply to CanApp as a whole:

man 7 canapp contains general information about where to find things.

man 1 canapp contains a list of the programs in CanApp, with a one-line description of each program.

man 3 canapp contains a list of the subroutines in CanApp, with a one-line description for each subroutine (or set of subroutines).

man 7 canappstandards describes the programming standards for CanApp.

man 7 canappmaint contains information for the system administrator about the organization of the CanApp library.

3 The program library

In this section the programs in the CanApp program library are listed. Specific documentation for each program is obtained with "man *name*", where *name* is the name of the program.

An up-to-date version of this list is obtained with "man 1 canapp".

3.1 Basic operations

cantocan Convert an image to Candela format.

canfromcan Convert an image from Candela format (e.g. to raw binary).

canabout Display header information from Candela pictures.

cancutout Cut a subwindow from a picture.

canunpack Read selected layers from a multi-layer picture.

3.2 Point operations

canpointop General point operations.

canlin Linear point transformations.

canfindposneg Detection of positive and negative values.

3.3 Local operations

canlinfilt General linear filtering.

canmedfilt Median filtering.

3.4 Scale-space operations

cangaussconv Convolution with the Gaussian kernel.

candiscgaussconv Convolution with the discrete analog of the Gaussian kernel.

3.5 Image size modification

canhalfsize Reduce the image size with a factor of 2 in each coordinate direction.

canenlarge Enlarge an image by pixel replication.

canextend Extend an image in various ways.

3.6 Histogram operations

canhisprint Compute and print histogram.

canhisview Compute and display histogram on a Sun workstation running SunView.

canhisteq Adaptive histogram equalization.

candiffhisprint Compute and print absolute difference histogram.

3.7 Edge detection etc.

canfocus Edge focusing.

canderiche Canny-Deriche edge detection.

candiffgrad Gradient calculations with various masks (Sobel etc.)

canlaplace Filter with the discrete Laplacian.

cansobel Filter with Sobel derivative masks.

cancentraldiff Filter with central difference masks.

cangradthresh Gradient thresholding with hysteresis.

3.8 Edge tracking

canedgetrack Edge tracking (segments edge images)

3.9 Interest points

canmoravec Determines Moravec interest points.

canputpad Puts pads at coordinate positions in an image.

3.10 Fourier transforms

canfft Forward FFT of an image.

canifft Inverse FFT of an image.

3.11 Digital topology

canconncomp Connected component labelling algorithm.

canfindextrema Extrema detection in the gray-level landscape.

canmarkbound Marks region boundaries.

canmarkregions Marks regions.

3.12 Distance transformations

candistancetransform Approximate and euclidean distance transformations.

3.13 Multilayer images

canmap Apply a csh command to all layers in a multilayer image.

3.14 Image sequences

canseqmap Apply a sh command to each file in one/many sequence(s).

3.15 Display and hardcopy

can2ps Convert an image to PostScript.

can2pegas Display an image on the Pegasus display unit.

can2sun Display an image on a Sun workstation running SunView.

cancrossplot Plot gray-level values along a cross-section parallel to one of the coordinate axes.

3.16 Frame grabbing

cangrab Grab a picture from a frame grabber.

3.17 Display of multi-layer images

canstack Stack a set of images into separate bitplanes.

canstackview Display a sequence of stacked images on a Sun workstation.

3.18 Image synthesis

canplot Execute UNIX plot instructions in a Candela image.

canregionmean Create a mean value image from a segmented image.

3.19 Miscellaneous

caned Interactive picture editor.

4 The subroutine library

In this section the subroutines in the CanApp subroutine library are listed. Specific documentation for each subroutine is obtained with "man *name*", where *name* is the name of the subroutine.

Some of the names in the list below is actually the name of a *group* of subroutines. In those cases the name of the group is usually the same as the name of the "major" subroutine in the group.

An up-to-date version of this list is obtained with "man 3 canapp".

4.1 Basic operations

minmax Find the minimum and maximum value in an image.

4.2 Point operations

pointop General point operations.

lintrans Linear point transformations.

4.3 Local operations

linearfilter General linear filtering.

filters Linear filtering with some common filters.

medianfilter Median filtering.

4.4 Scale-space operations

gaussconv Convolution with the Gaussian kernel, its discrete analog and three-kernels.

4.5 Image size modification

halfsize Reduces the image size with a factor of 2 in each coordinate direction.

picenlarge Enlarge an image by pixel replication.

picextension Extend the size of an image in various ways.

4.6 Histogram operations

hiscomp Compute and print histogram.

hisview Compute and display histogram on a Sun workstation running SunView.

histeq Adaptive histogram equalization.

diffhiscomp Compute and print histogram over the absolute values of the differences.

4.7 Edge detection etc.

diffgrad Gradient calculations with various masks (Sobel etc.)

derihyst Canny-Deriche edge detection. Thresholding with hysteresis.

4.8 Interest points etc.

moravec Detection of Moravec interest points.

putpad Puts pads in Candela images.

4.9 Digital topology etc.

conncomplabel Detection of connected components.

findextrema Detects local extrema in the gray-level landscape.

markregions Mark regions.

markboundaries Mark boundaries.

4.10 Distance transformations

distancetransform Approximate and euclidean distance transformations.

- A standard "Makefile" to be used for compiling the program and checking it with lint

See *man canappstandards* for information about where to find these files.

5.3 Programs

The following rules apply to main programs:

- The main program should in most cases only parse the command-line switches, open and close files, and call a subroutine that does the actual work. It should support the switches "–d" (debug mode) and "–v" (verbose mode). *Debug mode* means setting *candebug = 1*, and causes the low-level Candela routines to print entry and exit messages. As a rule the programs should be quiet when everything runs normally, but in *verbose mode* the programs and subroutines are allowed to print out informational messages. Both error and informational messages are always printed on stderr.
- The name of the program should begin with "can".
- The program should pass lint.
- Unless there are very strong reasons for not doing so, the program should be created by copying and modifying the "standard main" mentioned above.

5.4 Subroutines

The following rules apply to subroutines:

- Mutually connected subroutines (though not too many) should be grouped into one C file. Internal subroutines not intended to be exported should be declared *static* in order to be invisible outside the current file. If global variables are used, they must also be declared static in order not to interfere with variables having the same name in other files.
- The consistency of the input parameters should be checked as far as possible.
- If serious errors are detected, *cerror* should be called to print an error message and exit. The routine should *not* return any strange values or "error flags". Warnings may be printed to stderr.
- Input arguments (in particular Candela images) should not be affected by procedure calls. If a procedure really must change the contents of some of its input parameters then this effect must be explicitly stressed, both in the documentation and the C code.
- The history information in the Candela image free format label should be updated.
- The routine should be silent, i.e., if nothing serious happens then no output should be produced at the screen. If you do want log messages you may add the parameter *(int) verbose*. If its value is zero then the routine should be silent.

  Note that log messages should be printed to stderr, *not* stdout. Another possibility is to use a file pointer parameter *(FILE *) logfile*. If non-NULL then the log information should be printed to that file, otherwise the routine should be silent.

- If a routine makes use of specific constants (e.g. error thresholds, default parameter values etc), they should *not* be put in the middle of the code. Instead, define a symbol for that value and put the definition in the beginning of your C file.
- Procedures generating images as output should in general also allocate the corresponding memory space.
- If temporary working space is allocated dynamically, it should be released as soon as it is no longer needed.
- As many input image formats as possible should be supported. Algorithms operating on gray-level images should at least allow INT8BIT and FLOAT32BIT images as input. Similarly, algorithms for integer images should at least be able to work on INT8BIT, INT16BIT and INT32BIT images.
- The code should be commented, in particular the input and output formats and assumptions about the parameters.
- All compilations should be done with a Makefile (copy the standard Makefile described in *man canappstandards*).
- The code should be checked with lint.

5.5 Header files

There should always be a header file containing declarations for the routines which are to be exported. Name this file in accordance with your C file. (If the C file is named "myroutine.c" then the name of the header file should be "myroutine.h").

If you use special symbols which are necessary for an external user then their definitions should also be put in this (and only this) file. Use names which are descriptive enough such that name conflicts won't occur. Some care has to be taken since at installation this header file will be included in <candela/canapp.h>.

Note that only definitions which are really necessary (and don't occur anywhere else) may be put in this header file—internal definitions should normally be put in the corresponding C file.

However, if you for some special reason need to put your internal definitions in a header file, you should put them in a *separate* internal header file. Give this file a name in accordance with your other files, i.e. "myroutine.internals.h". At installation the external header file will be put at the common CanApp include directory, while the internal header file will be put together with the C-files.

5.6 Compilation and file organization

When a CanApp program is developed, it should be put in a separate directory. The standard Makefile should be copied and modified as needed (in most cases only the name of the program has to be changed).

This Makefile supports the following commands:

make Compile the program and install it in /bin/*arch*, where *arch* is the name of the machine architecture.

make debug Create an executable file suitable for dbx or gdb.

make lint Check the code with lint.

make clean Remove all object files, debug files, and core dumps.

More specific and up-to-date information is found in *man canappstandards*.

References

[1] M. Rehndal, *Candela Reference Manual*, Tech. Report, Royal Institute of Technology, Stockholm (in preparation).