Center for Embedded Computer Systems
University of California, Irvine

System-Level Modeling and Refinement of a Canny Edge Detector

Xu Han ,Yasaman Samei and Rainer Dömer

Technical Report CECS-12-13
November 7, 2012

Center for Embedded Computer Systems
University of California, Irvine
Irvine, CA 92697-3425, USA
(949) 824-8059

{hanx, ysameisy, doemer}@uci.edu
http://www.cecs.uci.edu/

System-Level Modeling and Refinement of a Canny Edge Detector

Xu Han ,Yasaman Samei and Rainer Dömer

Technical Report CECS-12-13
November 7, 2012

Center for Embedded Computer Systems
University of California, Irvine
Irvine, CA 92697-3425, USA
(949) 824-8059

{hanx, ysameisy, doemer}@uci.edu
http://www.cecs.uci.edu

Abstract

Electronic System Level design methodology and supporting tools simplifies the design of complex embedded systems. In this paper, a case study with an application example of canny edge detector is presented. We recoded a C reference code of canny to an initial specification model in SpecC. In order to yield best design from automated system level design tools, we refined the specification model by exploiting data-level parallelism, pipelining and converting floating-point computation to fixed-point in the initial model. With the optimized specification model, we used System-on-Chip Environment (SCE) to explore the design space, find allocation and mapping scheme which can achieve real-time computing, and refinement the specification model to Transaction level Model.

Contents

List of Figures

List of Tables

System-Level Modeling and Refinement of a Canny Edge Detector

Xu Han ,Yasaman Samei and Rainer Dömer

Center for Embedded Computer Systems
University of California, Irvine
Irvine, CA 92697-3425, USA

{hanx, ysameisy, doemer}@uci.edu
http://www.cecs.uci.edu

Abstract

Electronic System Level design methodology and supporting tools simplifies the design of complex embedded systems. In this paper, a case study with an application example of canny edge detector is presented. We recoded a C reference code of canny to an initial specification model in SpecC. In order to yield best design from automated system level design tools, we refined the specification model by exploiting data-level parallelism, pipelining and converting floating-point computation to fixed-point in the initial model. With the optimized specification model, we used System-on-Chip Environment (SCE) to explore the design space, find allocation and mapping scheme which can achieve real-time computing, and refinement the specification model to Transaction level Model.

1 Introduction

The growing market for embedded systems, typically portable electronic devices, demands products with better functionality and shorter time to market. The complexity of designing, debugging and verifying systems containing increasing number of HW/SW components presents great challenge to embedded design methodology. Electronic system level (ESL) design engages the problem using models of higher abstraction level. In ESL design, a system is firstly specified using a System Level Design Language (SLDL). The initial model is called a specification model. Then supporting tools can efficiently perform design space exploration, high-level synthesis, and software refinement to cycle-accurate level on the specification model.

Figure 1 [3] presents an ESL design methodology and tool sets SCE [2] using SpecC SLDL. Firstly, the product specification is captured using SpecC. Then, architecture refinement is performed, which involves allocation of processing elements (PE), mapping behaviors, channels and

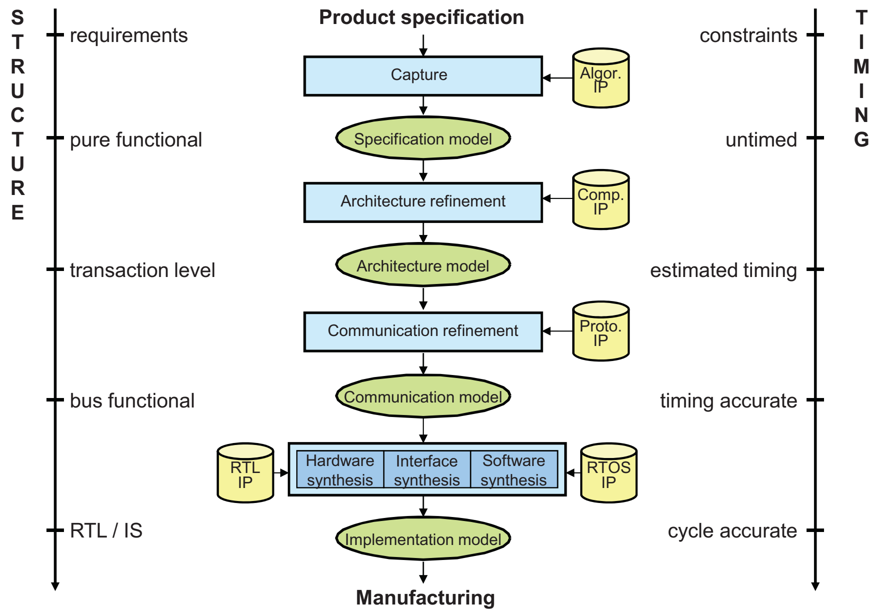


Figure 1: Top-Down System-Level Design Flow

variables to PEs. The result of architecture refinement is a system architecture model of concurrent PEs with abstract communication in channels. Next, scheduling refinement is performed to serialize the execution of behaviors on each PE according to either static or dynamic scheduling (RTOS support) chosen by designer. The result of this step is a system model with abstract RTOS inserted in each PE. Next, communication refinement implements the abstract communication channels between PEs. By allocating system busses and map the channels to them in SCE, we can then generate a bus-functional model of the system. Finally, the SW and HW components of bus-functional model is synthesized.

This report focuses on a case study of the ESL design methodology using an application example of canny edge detecter (canny). Canny is modeled in SpecC, optimized for design and refined to pin-accurate level using SCE.

2 Overview of Canny Edge Detector

The canny edge detector is developed by Prof. John F. Canny in 1986 and our work is based on a reference implementation [1]. The algorithm applies five functions in sequential to an input image and detects all the edges in it (Figure 2).

The five functions are:

- **gaussian_smooth** creates a gaussian kernel based on input parameter *SIGMA* (the standard deviation of the gaussian smoothing filter), and then used the kernel to filter or blur each pixel of the image to reduce the noise. The blurring occurs first horizontally and then vertically.

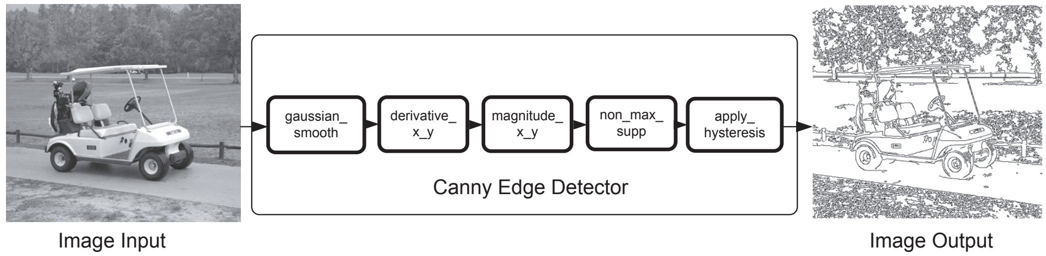


Figure 2: Canny Edge Detector Flowchart

- **derivative_x_y** computes the first derivative of the image in both the x any y directions.
- **magnitude_x_y** computes the magnitude of the gradient - the square root of the sum of the squared derivative values.
- **non_max_supp** applies non-maximal suppression to the magnitude of the gradient image. The pixels which are not part of local maxima are set as non-edges.
- **apply_hysteresis** finds edges that are above some high threshold or are connected to a high pixel by a path of pixels greater than a low threshold. Parameter *TLOW* and *THIGH* specifies these two thresholds.

3 System Level Modeling of Canny Edge Detector

3.1 Initial Specification Model

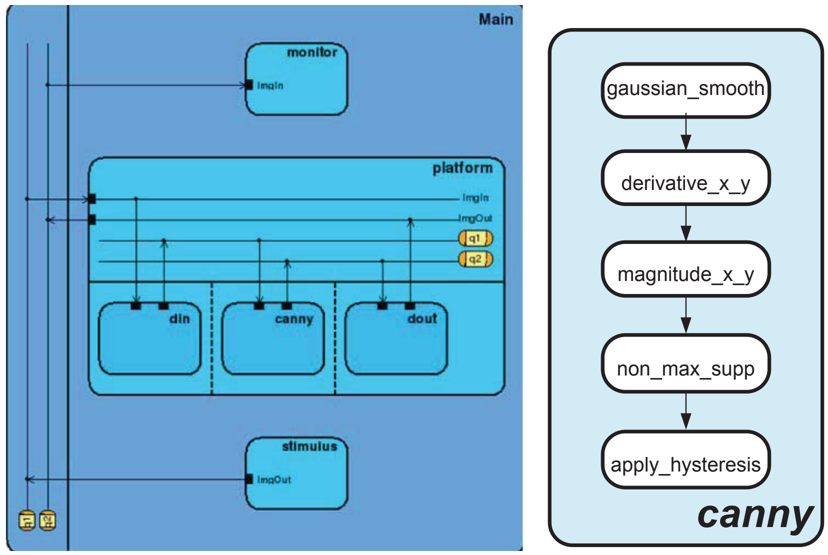
The first step of ESL design on canny is to create a specification model. We recoded the unstructured and sequential C reference code into SpecC model by works including encapsulating functions into behaviors, creating channels and hierarchy, and creating a testbench. The resulting specification model is described as Figure 3(a) where *stimulus* sends incoming images to *platform*, I/O units(*din* and *dout*) send input to DUT *canny* and send output to *monitor*. DUT *canny* consists of 5 behaviors (Figure 3(b)) performing the 5 functions of canny algorithm.

3.2 Model Refinements

We profiled the initial model using SCE profiler. The results (Figure 4) show that when ARM7 CPU is allocated, Behavior takes more than 50% of total computation. In order to yield better design, we considered to optimize *guassian_smooth*.

3.2.1 Parallelization of *guassian_smooth*

Parallelization is one desired approach to optimize the heaviest function *guassian_smooth*. We observed that *guassian_smooth* firstly creates a Gaussian kernel used to blur the image (we call this



(a) Stimulus, Platform and Monitor (b) DUT: canny

Figure 3: Initial Specification Model

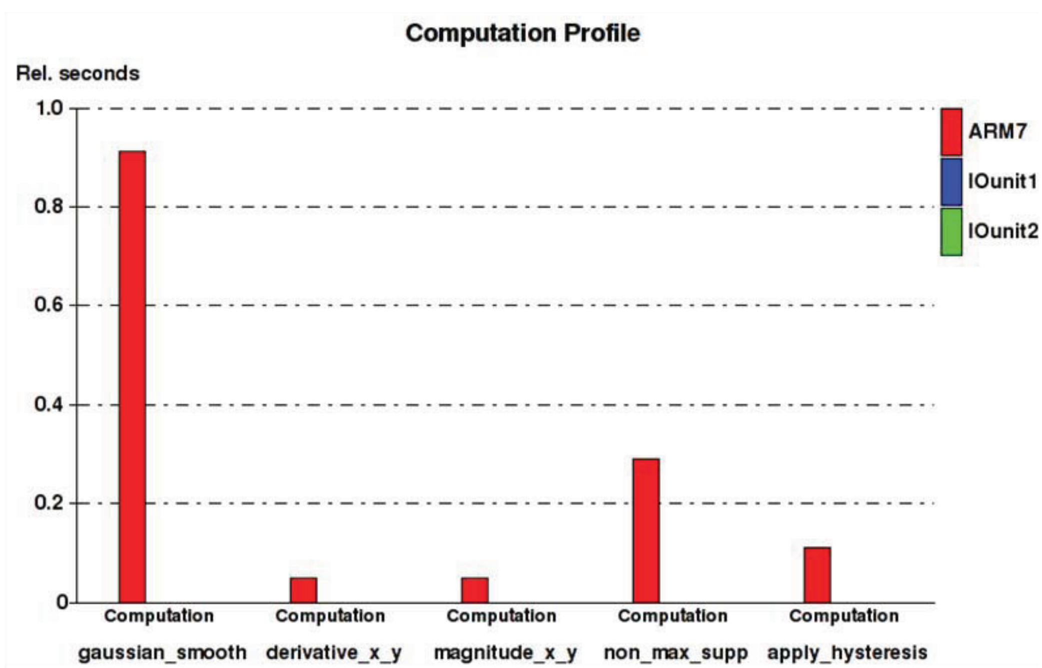


Figure 4: Canny Profile using SCE

task 'Prep'), then blurs the image in horizontally by filtering each pixel using its neighbors in X-direction (BlurX), and finally blurs the image in vertically by filtering each pixel using its neighbors in Y-direction (BlurY). One parallelization strategy is to run BlurX on horizontal slices of the image in parallel and BlurY on vertical slices. In this way, the parallelization can be dependency free.

To model parallel tasks, we partitioned *guassian_smooth* into 3 behaviors, namely *Prep*, *BlurX* and *BlurY*. Assuming 4 available processing elements, we created the new hierarchy for *guassian_smooth* as Figure 3.2.1.

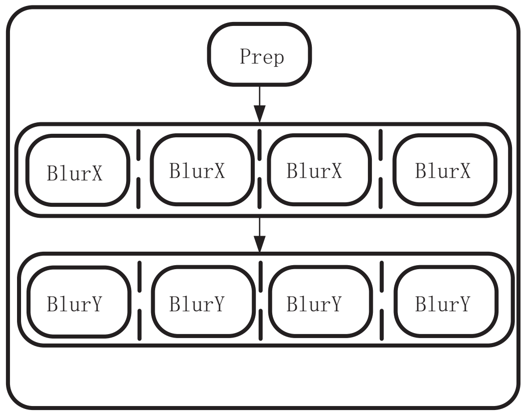


Figure 5: Parallel *guassian_smooth*

3.2.2 Pipelining

Though the reference C code only processes a single image, we adapted the specification model to process a sequence of images with the number of images specified by the user. We have pipelined the 5 functional blocks (Figure 6) so that the model is able to process multiple incoming images in parallel.

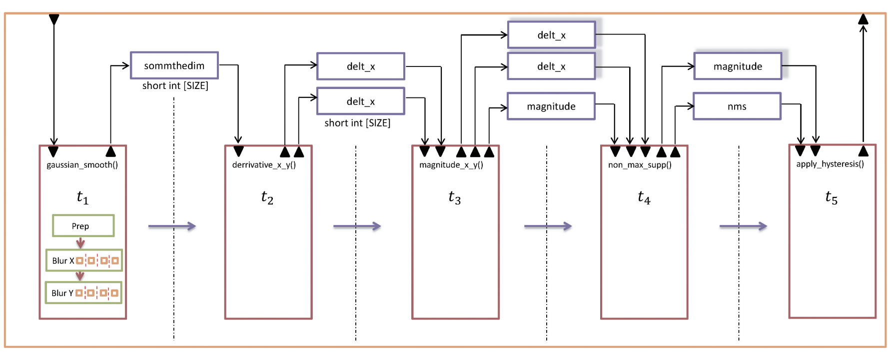


Figure 6: 5-stage Pipelined Canny Edge Detector [5]

3.2.3 Converting Floating-point to Fixed-point

Figure 7 shows updated profiling results when *guassian_smooth* is parallelized using 4 processing unit and tentatively mapped onto two hardware unit using SCE. Now *non_max_supp* contains the largest amount of computation in all behaviors and becomes the bottleneck in the pipeline.

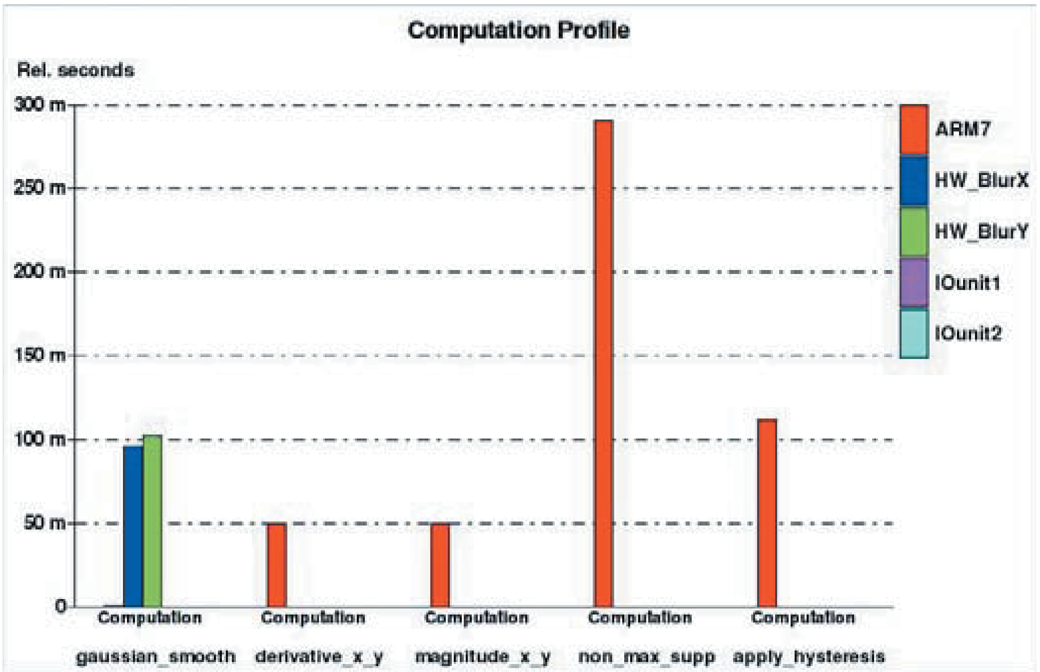
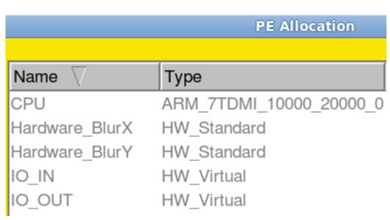


Figure 7: Canny Profile using SCE after *guassian_smooth* parallelized

From SCE profiling statistics, 95.6% of the operations in *non_max_supp* are floating-point. In order to balance the pipeline stages, we converted floating-point computation to fixed-point without loss of accuracy [4].

The source code of the refined specification model can be found in Appendix A.1.

4 Model Refinements and Implementation using SCE


PE Allocation

Name	Type
CPU	ARM_7TDMI_10000_20000_0
Hardware_BlurX	HW_Standard
Hardware_BlurY	HW_Standard
IO_IN	HW_Virtual
IO_OUT	HW_Virtual

Figure 8: PE Allocation

Figure 9: Behavior Mapping

In this section different steps of design space exploration shown in Figure 1, are performed over Canny example. The SCE tool is used to perform a step by step implementation. The refinement steps start with architecture refinement. In this step processing elements(PE) are added to design and canny's behaviors are allocated to them. The added PEs and their corresponding behaviors are shown in the Figure 8 and Figure 9.

The canny algorithms are running on a ARM7TDMI target processor and as it is shown in Figure 9 two sub behavior of the *guassian_smooth* function are allocated to separate hardware units. Since the *guassian_smooth* function is computationally expensive, extra hardware units are added to improve the performance of the design.

In the next step of system level design process different behaviors of the design are scheduled. In the canny example no dynamic scheduling algorithm is applied and behaviors execution order is static and same as the original specification of the design. After scheduling the behaviors, decisions on communication between system components are made. Two system BUSes are selected in canny example; AMBA BUS belong to allocated ARM CPU and a Hardware bus. The allocated BUSes

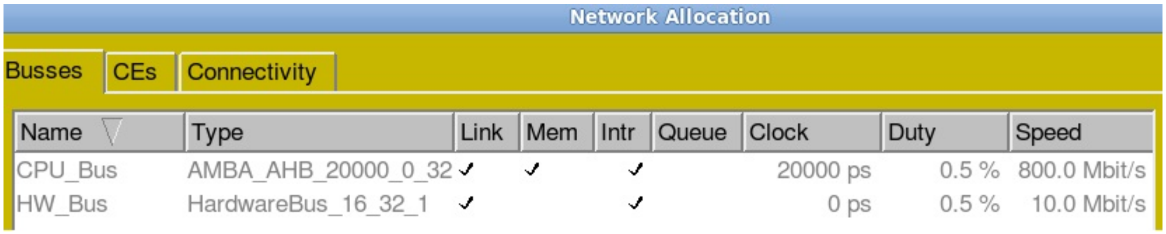


Figure 10: Network Allocation

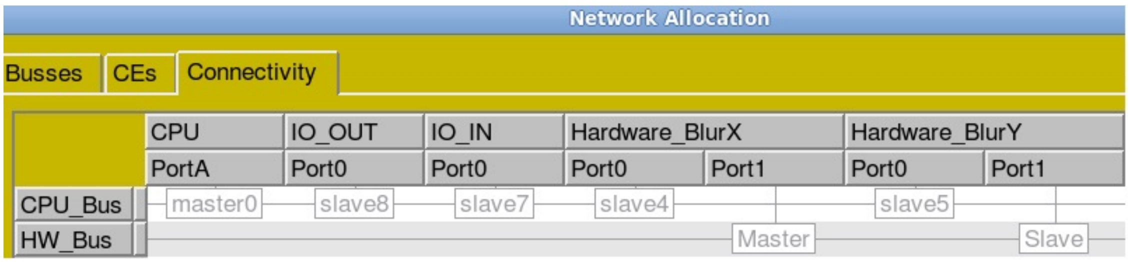


Figure 11: Network Connectivity

are illustrated in Figure 10. Following the BUS allocation, components connectivity, Masters and Slaves are defined. In Figure 11 component's connections over the BUS are clarified.

Finally the implementation is brought down to Transaction level Model(TLM). Table 1 reflects the simulation results in each step. The detailed configuration of the TLM generation is described in the Makefile in Appendix .

Table 1: Timing of various Models

Model	Simulation Time(micro second)
Specification Model	0
Architecture Model	1982972
Scheduled Architecture Model	1982972
Network Model	1982972
TLM Model	2012235

5 Conclusion

In this report, we have created a system-level model of a canny edge detector based on its C reference code. We refined the model by exploiting parallelism, pipelining and converting floating-point to fixed-point for the bottleneck pipelining stage. We then use SCE to find an allocation and mapping scheme and refine the model automatically to Transaction level Model. The model is verified by simulation.

References

[1] Canny Edge Detector. ftp://figment.csee.usf.edu/pub/Edge_Comparison/source_code/canny.src.

[2] Rainer Dömer, Andreas Gerstlauer, Junyu Peng, Dongwan Shin, Lukai Cai, Haobo Yu, Samar Abdi, and Daniel Gajski. System-on-Chip Environment: A SpecC-based Framework for Heterogeneous MPSoC Design. *EURASIP Journal on Embedded Systems*, 2008(647953):13 pages, 2008.

[3] Andreas Gerstlauer, Rainer Dömer, Junyu Peng, and Daniel D. Gajski. *System Design: A Practical Guide with SpecC*. Kluwer, 2001.

[4] Jiang Wan. Modeling of a canny edge detector system-on-chip for a digital camera. EECS222A 2012 Spring Course Project Report.

[5] Ching-Yao Wang. Modeling of a canny edge detector system-on-chip for a digital camera. EECS222A 2012 Spring Course Project Report.

A Appendix

A.1 Source Code of Canny Edge Detector in SpecC

```
/* Canny Edge Detector for image stream
   X.Han, Rainer Doemer, Oct.2012 */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

#include <sim.sh>
#include <c_typed_queue.sh>     /* make the templates available */
#include <c_typed_double_handshake.sh>

#define VERBOSE 0
#define TIME_BASE        (1 MICRO_SEC)   // print time in units of micro-seconds

#define NOEDGE 255
#define POSSIBLE_EDGE 128
#define EDGE 0
#define BOOSTBLURFACTOR 90.0

#define COLS 640
#define ROWS 360
#define SIZE COLS*ROWS

//do not include picture number and ".pgm" of the filename
#define FILENAME "image"

#define SIGMA 0.6
#define TLOW  0.3
#define THIGH 0.8

#define IMG_NUM 4  //it can be any number
#define AVAIL_IMG 3
#define SHIFT_BIT 16

/* an upper bound for removing dynamic calloc
 * SIGMA must be less than 4
 * check for 'windowsize' below
 */
#define WINSIZE 21

// patch for the non_max_supp function
#define SMP_PATCH

// fixed point computing for non_max_supp() function
#define FIXED_POINT

typedef unsigned char img[SIZE];        /* define our communication data types */
```

```
typedef short imgs[SIZE];

DEFINE_I_TYPED_SENDER(img, img)          // creates interface i_img_sender
DEFINE_I_TYPED_RECEIVER(img, img)        // creates interface i_img_receiver
DEFINE_I_TYPED_TRANCEIVER(img, img)      // creates interface i_img_tranceiver

DEFINE_C_TYPED_QUEUE(img, img)           // creates channel c_img_queue


behavior Monitor(i_img_receiver ImgIn, in sim_time StartTime[IMG_NUM])
{
  unsigned char edge[SIZE];

  /****************************************************************************
  * Function: write_pgm_image
  * Purpose: This function writes an image in PGM format. The file is either
  * written to the file specified by outfilename or to standard output if
  * outfilename = NULL. A comment can be written to the header if coment != NULL.
  ****************************************************************************/
  int write_pgm_image(char *outfilename, unsigned char *image, int rows,
      int cols, const char *comment, int maxval)
  {
     FILE *fp;

     /*************************************************************************
     * Open the output image file for writing if a filename was given. If no
     * filename was provided, set fp to write to standard output.
     *************************************************************************/
     if(outfilename == NULL) fp = stdout;
     else{
        if((fp = fopen(outfilename, "w")) == NULL){
           fprintf(stderr, "Error writing the file %s in write_pgm_image().\n",
              outfilename);
           return(0);
        }
     }

     /*************************************************************************
     * Write the header information to the PGM file.
     *************************************************************************/
     fprintf(fp, "P5\n%d %d\n", cols, rows);
     if(comment != NULL)
        if(strlen(comment) <= 70) fprintf(fp, "# %s\n", comment);
     fprintf(fp, "%d\n", maxval);

     /*************************************************************************
     * Write the image data to the file.
     *************************************************************************/
     if((unsigned)rows != fwrite(image, cols, rows, fp)){
        fprintf(stderr, "Error writing the image data in write_pgm_image().\n");
        if(fp != stdout) fclose(fp);
        return(0);
```

```
    }

    if(fp != stdout) fclose(fp);
    return(1);
  }

  void main()
  {
          char outfilename[128];    /* Name of the output "edge" image */
          sim_time t, t2;
          sim_time_string buf, buf2;
          int i,n;

          for(i=0; i<IMG_NUM; i++)
          {
                 ImgIn.receive(&edge);

                 t = now();
                 printf("%8s: Monitor received image%d.\n",time2str(buf, t/TIME_BASE),i);
                 t2 = t - StartTime[i];
                 printf("%8s: Image processing took %8s micro seconds.\n",
                             time2str(buf, t/TIME_BASE), time2str(buf2, t2/TIME_BASE));


          /*************************************************************************
          * Write out the edge image to a file.
          *************************************************************************/

       n=i%AVAIL_IMG;

       sprintf(outfilename, "%s_s_%3.2f_l_%3.2f_h_%3.2f_%d.pgm", FILENAME, SIGMA, TLOW, THIGH, i);
            if(VERBOSE) printf("Writing the edge iname in the file %s.\n", outfilename);
            if(write_pgm_image(outfilename, edge, ROWS, COLS,"", 255) == 0){
             fprintf(stderr, "Error writing the edge image, %s.\n", outfilename);
             exit(1);
            }
          }//for


          exit(0);      // done testing, quit the simulation
   }
};

behavior Stimulus(i_img_sender ImgOut, out sim_time StartTime[IMG_NUM])
{
    unsigned char image[SIZE];

    /*************************************************************************
    * Function: read_pgm_image
    * Purpose: This function reads in an image in PGM format. The image can be
    * read in from either a file or from standard input. The image is only read
    * from standard input when infilename = NULL. Because the PGM format includes
```

```
* the number of columns and the number of rows in the image, these are read
* from the file. Memory to store the image is allocated OUTSIDE this function.
* The found image size is checked against the expected rows and cols.
* All comments in the header are discarded in the process of reading the
* image. Upon failure, this function returns 0, upon sucess it returns 1.
****************************************************************************/
int read_pgm_image(const char *infilename, unsigned char *image0, int rows, int cols)
{
   FILE *fp;
   char buf[71];
   int r, c;

   /***************************************************************************
   * Open the input image file for reading if a filename was given. If no
   * filename was provided, set fp to read from standard input.
   ***************************************************************************/
   if(infilename == NULL) fp = stdin;
   else{
      if((fp = fopen(infilename, "r")) == NULL){
         fprintf(stderr, "Error reading the file %s in read_pgm_image().\n",
            infilename);
         return(0);
      }
   }

   /***************************************************************************
   * Verify that the image is in PGM format, read in the number of columns
   * and rows in the image and scan past all of the header information.
   ***************************************************************************/
   fgets(buf, 70, fp);
   if(strncmp(buf,"P5",2) != 0){
      fprintf(stderr, "The file %s is not in PGM format in ", infilename);
      fprintf(stderr, "read_pgm_image().\n");
      if(fp != stdin) fclose(fp);
      return(0);
   }
   do{ fgets(buf, 70, fp); }while(buf[0] == '#');  /* skip all comment lines */
   sscanf(buf, "%d %d", &c, &r);
   if(c != cols || r != rows){
      fprintf(stderr, "The file %s is not a %d by %d image in ", infilename, cols, rows);
      fprintf(stderr, "read_pgm_image().\n");
      if(fp != stdin) fclose(fp);
      return(0);
   }
   do{ fgets(buf, 70, fp); }while(buf[0] == '#');  /* skip all comment lines */

   if((unsigned)rows != fread(image0, cols, rows, fp)){
      fprintf(stderr, "Error reading the image data in read_pgm_image().\n");
      if(fp != stdin) fclose(fp);
      return(0);
   }

```

```
        if(fp != stdin) fclose(fp);
        return(1);
     }

   void main()
   {
           sim_time t;
           sim_time_string buf;
           int i=0,n=0;
           char infilename[40];




           for(i=0;i < IMG_NUM; i++)
           {
             n=i%AVAIL_IMG;

             sprintf(infilename, "%s%d.pgm",FILENAME,n);

             if(VERBOSE) printf("Reading the image %s.\n", infilename);
             if(read_pgm_image(infilename, image, ROWS, COLS) == 0){
              fprintf(stderr, "Error reading the input image, %s.\n", infilename);
              exit(1);
              }
                   t = now() / TIME_BASE;
                   printf("%8s: Stimulus sends image%d.\n", time2str(buf, t), i);
                   StartTime[i] = t;

                   ImgOut.send(image);

           }


   }
};


behavior Prep(i_img_receiver ImgIn, inout int center,
                   inout float kernel[WINSIZE], inout img image)
{
 int    windowsize;        /* Dimension of the gaussian kernel. */

 void make_gaussian_kernel(float sigma)
 {
    int i;
    float x, fx, sum=0.0;

    windowsize = 1 + 2 * ceil(2.5 * sigma);
    center = windowsize / 2;

    if(VERBOSE) printf("      The kernel has %d elements.\n", windowsize);
```

```
    for(i=0;i<windowsize;i++){
        x = (float)(i - center);
        fx = pow(2.71828, -0.5*x*x/(sigma*sigma)) / (sigma * sqrt(6.2831853));
        kernel[i] = fx;
        sum += fx;
    }

    for(i=0;i<windowsize;i++) kernel[i] /= sum;

    if(VERBOSE){
        printf("The filter coefficients are:\n");
        for(i=0;i<windowsize;i++)
            printf("kernel[%d] = %f\n", i, kernel[i]);
    }
 }

  void main()
  {
    ImgIn.receive(&image);

   /****************************************************************************
    * Create a 1-dimensional gaussian smoothing kernel.
    ****************************************************************************/
    if(VERBOSE) printf("   Computing the gaussian smoothing kernel.\n");
    make_gaussian_kernel(SIGMA);
  }
};


behavior BlurX(in img image, in int center, in float kernel[WINSIZE],
               inout float tempim[SIZE], in int rowStart, in int rowEnd)
{

void main()
{
   int r, c, cc;      /* Counter variables. */
   float dot,         /* Dot product summing variable. */
         sum;         /* Sum of the kernel weights variable. */

   /****************************************************************************
   * Blur in the x - direction.
   ****************************************************************************/
   if(VERBOSE) printf("   Bluring the image in the X-direction.\n");
   for(r=rowStart;r<rowEnd;r++){
      for(c=0;c<COLS;c++){
         dot = 0.0;
         sum = 0.0;
         for(cc=(-center);cc<=center;cc++){
            if(((c+cc) >= 0) && ((c+cc) < COLS)){
               dot += (float)image[r*COLS+(c+cc)] * kernel[center+cc];
               sum += kernel[center+cc];
```

```
            }
        }
        tempim[r*COLS+c] = dot/sum;
    }
  }
}
};


behavior BlurY(out imgs smoothedim, in int center, in float kernel[WINSIZE],
               in float tempim[SIZE], in int colStart, in int colEnd)
{

void main()
{
  int r, c, rr;          /* Counter variables. */
  float dot,             /* Dot product summing variable. */
        sum;             /* Sum of the kernel weights variable. */

  /*************************************************************************
  * Blur in the y - direction.
  *************************************************************************/
  if(VERBOSE) printf("   Bluring the image in the Y-direction.\n");
  for(c=colStart;c<colEnd;c++){
    for(r=0;r<ROWS;r++){
      sum = 0.0;
      dot = 0.0;
      for(rr=(-center);rr<=center;rr++){
        if(((r+rr) >= 0) && ((r+rr) < ROWS)){
          dot += tempim[(r+rr)*COLS+c] * kernel[center+rr];
          sum += kernel[center+rr];
        }
      }
      smoothedim[r*COLS+c] = (short int)(dot*BOOSTBLURFACTOR/sum + 0.5);
    }
  }
}
};


behavior BlurX_par(in img image, in int center, in float kernel[WINSIZE],
                   inout float tempim[SIZE])
{
   BlurX blurX1(image, center, kernel, tempim, ((ROWS/4)*0), ((ROWS/4)*1));
   BlurX blurX2(image, center, kernel, tempim, ((ROWS/4)*1), ((ROWS/4)*2));
   BlurX blurX3(image, center, kernel, tempim, ((ROWS/4)*2), ((ROWS/4)*3));
   BlurX blurX4(image, center, kernel, tempim, ((ROWS/4)*3), ((ROWS/4)*4));

    void main()
    {
        par{
            blurX1.main();
```

```
            blurX2.main();
            blurX3.main();
            blurX4.main();
          }
    }
};


behavior BlurY_par(out imgs smoothedim, in int center, in float kernel[WINSIZE],
                   in float tempim[SIZE])
{
    BlurY blurY1(smoothedim, center, kernel, tempim, ((COLS/4)*0), ((COLS/4)*1));
    BlurY blurY2(smoothedim, center, kernel, tempim, ((COLS/4)*1), ((COLS/4)*2));
    BlurY blurY3(smoothedim, center, kernel, tempim, ((COLS/4)*2), ((COLS/4)*3));
    BlurY blurY4(smoothedim, center, kernel, tempim, ((COLS/4)*3), ((COLS/4)*4));

     void main()
     {
         par{
            blurY1.main();
            blurY2.main();
            blurY3.main();
            blurY4.main();
          }
     }
};

behavior Blur_done( in imgs smoothedim, out imgs smoothedimg )
{
        void main()
    {
                smoothedimg = smoothedim;
    }
};

behavior Gaussian_Smooth (i_img_receiver ImgIn, out imgs smoothedimg)
{
     img image;
     int center;
     float kernel[WINSIZE];
     float tempim[SIZE];
     imgs smoothedim;

     Prep prep(ImgIn, center, kernel, image);
         BlurX_par blurX_par(image, center, kernel, tempim);
         BlurY_par blurY_par(smoothedim, center, kernel, tempim);
     Blur_done blur_done(smoothedim, smoothedimg);

     void main()
     {
           prep.main();
           blurX_par.main();
```

```
        blurY_par.main();
        blur_done.main();
    }
};


behavior Derivative_X_Y(in imgs smoothedim, out imgs delta_x, out imgs delta_y)
{
        /*******************************************************************************
        * PROCEDURE: derrivative_x_y
        * PURPOSE: Compute the first derivative of the image in both the x any y
        * directions. The differential filters that are used are:
        *
        *                                          -1
        *         dx =  -1 0 +1       and     dy =  0
        *                                          +1
        *
        * NAME: Mike Heath
        * DATE: 2/15/96
        *******************************************************************************/
        void derivative_x_y(short int *smoothedimg, int rows, int cols,
             short int *deltax, short int *deltay)
        {
           int r, c, pos;

           /****************************************************************************
           * Compute the x-derivative. Adjust the derivative at the borders to avoid
           * losing pixels.
           ****************************************************************************/
           if(VERBOSE) printf("   Computing the X-direction derivative.\n");
           for(r=0;r<rows;r++){
              pos = r * cols;
              deltax[pos] = smoothedimg[pos+1] - smoothedimg[pos];
              pos++;
              for(c=1;c<(cols-1);c++,pos++){
                 deltax[pos] = smoothedimg[pos+1] - smoothedimg[pos-1];
              }
              deltax[pos] = smoothedimg[pos] - smoothedimg[pos-1];
           }

           /****************************************************************************
           * Compute the y-derivative. Adjust the derivative at the borders to avoid
           * losing pixels.
           ****************************************************************************/
           if(VERBOSE) printf("   Computing the Y-direction derivative.\n");
           for(c=0;c<cols;c++){
              pos = c;
              deltay[pos] = smoothedimg[pos+cols] - smoothedimg[pos];
              pos += cols;
              for(r=1;r<(rows-1);r++,pos+=cols){
                 deltay[pos] = smoothedimg[pos+cols] - smoothedimg[pos-cols];
              }
```

```
                deltay[pos] = smoothedimg[pos] - smoothedimg[pos-cols];
            }
        }

    void main()
    {
                imgs deltax, deltay;

                derivative_x_y(smoothedim, ROWS, COLS, deltax, deltay);

                delta_y=deltay;
                delta_x=deltax;
    }
};

behavior Magnitude_X_Y(in imgs delta_x, in imgs delta_y, out imgs magnitude,
                                     out imgs delta_x_p1, out imgs delta_y_p1)
{
        /*************************************************************************
        * PROCEDURE: magnitude_x_y
        * PURPOSE: Compute the magnitude of the gradient. This is the square root of
        * the sum of the squared derivative values.
        * NAME: Mike Heath
        * DATE: 2/15/96
        *************************************************************************/
        void magnitude_x_y(short int *deltax, short int *deltay, int rows, int cols,
                short int *mag)
        {
            int r, c, pos, sq1, sq2;

            for(r=0,pos=0;r<rows;r++){
                for(c=0;c<cols;c++,pos++){
                    sq1 = (int)deltax[pos] * (int)deltax[pos];
                    sq2 = (int)deltay[pos] * (int)deltay[pos];
                    mag[pos] = (short)(0.5 + sqrt((float)sq1 + (float)sq2));
                }
            }
        }

    void main()
    {
         imgs mag;

         magnitude_x_y(delta_x, delta_y, ROWS, COLS, mag);

         magnitude = mag;
         delta_x_p1 = delta_x;
         delta_y_p1 = delta_y;
    }
};

behavior Mag_Delta(in imgs smoothedim, out imgs delta_x, out imgs delta_y,
```

```
                    out imgs magnitude)
{
        imgs delta_x_p1;
        imgs delta_y_p1;

        Derivative_X_Y  derivative_x_y(smoothedim, delta_x_p1, delta_y_p1);
        Magnitude_X_Y   magnitude_x_y(delta_x_p1, delta_y_p1, magnitude,
                                                        delta_x, delta_y);

        void main()
        {
                derivative_x_y.main();
            magnitude_x_y.main();
        }
};

behavior Non_Max_Supp(in imgs delta_x, in imgs delta_y, in imgs magnitude,
                        out img nms, out imgs magnitude_p1)
{

           /*******************************************************************************
           * PROCEDURE: non_max_supp
           * PURPOSE: This routine applies non-maximal suppression to the magnitude of
           * the gradient image.
           * NAME: Mike Heath
           * DATE: 2/15/96
           *******************************************************************************/
           non_max_supp(short *mag, short *gradx, short *grady, int nrows, int ncols,
               unsigned char *result)
           {
               int rowcount, colcount,count;
               short *magrowptr,*magptr;
               short *gxrowptr,*gxptr;
               short *gyrowptr,*gyptr,z1,z2;
               int gx,gy;
               short m00;
#ifdef FIXED_POINT
                 int mag1,mag2,xperp,yperp;
#else
                 float mag1,mag2,xperp,yperp;
#endif
               unsigned char *resultrowptr, *resultptr;

              /*****************************************************************************
              * Zero the edges of the result image.
              *****************************************************************************/
               for(count=0,resultrowptr=result,resultptr=result+ncols*(nrows-1);
                   count<ncols; resultptr++,resultrowptr++,count++){
                   *resultrowptr = *resultptr = (unsigned char) 0;
               }

               for(count=0,resultptr=result,resultrowptr=result+ncols-1;
```

```
                count<nrows; count++,resultptr+=ncols,resultrowptr+=ncols){
                *resultptr = *resultrowptr = (unsigned char) 0;
            }

            /*******************************************************************
            * Suppress non-maximum points.
            *******************************************************************/
            for(rowcount=1,magrowptr=mag+ncols+1,gxrowptr=gradx+ncols+1,
               gyrowptr=grady+ncols+1,resultrowptr=result+ncols+1;

#ifdef SMP_PATCH
               rowcount<=nrows-2;
#else
               rowcount<nrows-2;
#endif

               rowcount++,magrowptr+=ncols,gyrowptr+=ncols,gxrowptr+=ncols,
               resultrowptr+=ncols){
               for(colcount=1,magptr=magrowptr,gxptr=gxrowptr,gyptr=gyrowptr,
#ifdef SMP_PATCH
                  resultptr=resultrowptr;colcount<=ncols-2;
#else
                  resultptr=resultrowptr;colcount<ncols-2;
#endif
                  colcount++,magptr++,gxptr++,gyptr++,resultptr++){
                  m00 = *magptr;
                  if(m00 == 0){
                     *resultptr = (unsigned char) NOEDGE;
                  }
                  else{
#ifdef FIXED_POINT
                     gx = *gxptr;
                     gy = *gyptr;;
                     xperp = -(gx<<SHIFT_BIT)/m00;
                     yperp = (gy<<SHIFT_BIT)/m00;
#else
                     xperp = -(gx = *gxptr)/((float)m00);
                     yperp = (gy = *gyptr)/((float)m00);
#endif
                  }

                  if(gx >= 0){
                     if(gy >= 0){
                           if (gx >= gy)
                           {
                              /* 111 */
                              /* Left point */
                              z1 = *(magptr - 1);
                              z2 = *(magptr - ncols - 1);

                              mag1 = (m00 - z1)*xperp + (z2 - z1)*yperp;

```

```
                /* Right point */
                z1 = *(magptr + 1);
                z2 = *(magptr + ncols + 1);

                mag2 = (m00 - z1)*xperp + (z2 - z1)*yperp;
            }
            else
            {
                /* 110 */
                /* Left point */
                z1 = *(magptr - ncols);
                z2 = *(magptr - ncols - 1);

                mag1 = (z1 - z2)*xperp + (z1 - m00)*yperp;

                /* Right point */
                z1 = *(magptr + ncols);
                z2 = *(magptr + ncols + 1);

                mag2 = (z1 - z2)*xperp + (z1 - m00)*yperp;
            }
        }
        else
        {
            if (gx >= -gy)
            {
                /* 101 */
                /* Left point */
                z1 = *(magptr - 1);
                z2 = *(magptr + ncols - 1);

                mag1 = (m00 - z1)*xperp + (z1 - z2)*yperp;

                /* Right point */
                z1 = *(magptr + 1);
                z2 = *(magptr - ncols + 1);

                mag2 = (m00 - z1)*xperp + (z1 - z2)*yperp;
            }
            else
            {
                /* 100 */
                /* Left point */
                z1 = *(magptr + ncols);
                z2 = *(magptr + ncols - 1);

                mag1 = (z1 - z2)*xperp + (m00 - z1)*yperp;

                /* Right point */
                z1 = *(magptr - ncols);
                z2 = *(magptr - ncols + 1);

```

```
            mag2 = (z1 - z2)*xperp  + (m00 - z1)*yperp;
        }
    }
}
else
{
    if ((gy = *gyptr) >= 0)
    {
        if (-gx >= gy)
        {
            /* 011 */
            /* Left point */
            z1 = *(magptr + 1);
            z2 = *(magptr - ncols + 1);

            mag1 = (z1 - m00)*xperp + (z2 - z1)*yperp;

            /* Right point */
            z1 = *(magptr - 1);
            z2 = *(magptr + ncols - 1);

            mag2 = (z1 - m00)*xperp + (z2 - z1)*yperp;
        }
        else
        {
            /* 010 */
            /* Left point */
            z1 = *(magptr - ncols);
            z2 = *(magptr - ncols + 1);

            mag1 = (z2 - z1)*xperp + (z1 - m00)*yperp;

            /* Right point */
            z1 = *(magptr + ncols);
            z2 = *(magptr + ncols - 1);

            mag2 = (z2 - z1)*xperp + (z1 - m00)*yperp;
        }
    }
    else
    {
        if (-gx > -gy)
        {
            /* 001 */
            /* Left point */
            z1 = *(magptr + 1);
            z2 = *(magptr + ncols + 1);

            mag1 = (z1 - m00)*xperp + (z1 - z2)*yperp;

            /* Right point */
            z1 = *(magptr - 1);
```

```
                z2 = *(magptr - ncols - 1);

                mag2 = (z1 - m00)*xperp + (z1 - z2)*yperp;
            }
            else
            {
                /* 000 */
                /* Left point */
                z1 = *(magptr + ncols);
                z2 = *(magptr + ncols + 1);

                mag1 = (z2 - z1)*xperp + (m00 - z1)*yperp;

                /* Right point */
                z1 = *(magptr - ncols);
                z2 = *(magptr - ncols - 1);

                mag2 = (z2 - z1)*xperp + (m00 - z1)*yperp;
            }
        }
    }

    /* Now determine if the current point is a maximum point */

    if ((mag1 > 0) || (mag2 > 0))
    {
        *resultptr = (unsigned char) NOEDGE;
    }
    else
    {
        if (mag2 == 0)
            *resultptr = (unsigned char) NOEDGE;
        else
            *resultptr = (unsigned char) POSSIBLE_EDGE;
    }
    }
  }
  return 0;
}

void main()
{

    img result;
    int i;
    //initialise nms to all zero by jiangwan
    for( i=0; i<SIZE; i++ )
    {
        result[i] = 0;
    }

    non_max_supp(magnitude, delta_x, delta_y, ROWS, COLS, result);
```

```
                    magnitude_p1 = magnitude;
                    nms = result;

            }
};


behavior Apply_Hysteresis(in imgs magnitude, in img nms, i_img_sender ImgOut)
{
    unsigned char edge[SIZE];

            /*******************************************************************************
            * PROCEDURE: follow_edges
            * PURPOSE: This procedure edges is a recursive routine that traces edgs along
            * all paths whose magnitude values remain above some specifyable lower
            * threshhold.
            * NAME: Mike Heath
            * DATE: 2/15/96
            *******************************************************************************/
            void follow_edges(unsigned char *edgemapptr, short *edgemagptr, short lowval,
               int cols)
            {
               note _SCC_ANALYSIS_IgnoreParThAnalyzeRecursiveFunction=true;
               short *tempmagptr;
               unsigned char *tempmapptr;
               int i;

               int x[8] = {1,1,0,-1,-1,-1,0,1},
                   y[8] = {0,1,1,1,0,-1,-1,-1};

               for(i=0;i<8;i++){
                  tempmapptr = edgemapptr - y[i]*cols + x[i];
                  tempmagptr = edgemagptr - y[i]*cols + x[i];

                  if((*tempmapptr == POSSIBLE_EDGE) && (*tempmagptr > lowval)){
                     *tempmapptr = (unsigned char) EDGE;
                     follow_edges(tempmapptr,tempmagptr, lowval, cols);
                  }
               }
            }

            /*******************************************************************************
            * PROCEDURE: apply_hysteresis
            * PURPOSE: This routine finds edges that are above some high threshhold or
            * are connected to a high pixel by a path of pixels greater than a low
            * threshold.
            * NAME: Mike Heath
            * DATE: 2/15/96
            *******************************************************************************/
            void apply_hysteresis(short int *mag, unsigned char *nmsimg, int rows, int cols,
                 float tlow, float thigh, unsigned char *edge0)
```

```
{
   int r, c, pos, numedges, highcount, lowthreshold, highthreshold,
        hist[32768];
   short int maximum_mag;

   /****************************************************************************
   * Initialize the edge map to possible edges everywhere the non-maximal
   * suppression suggested there could be an edge except for the border. At
   * the border we say there can not be an edge because it makes the
   * follow_edges algorithm more efficient to not worry about tracking an
   * edge off the side of the image.
   ****************************************************************************/
   for(r=0,pos=0;r<rows;r++){
      for(c=0;c<cols;c++,pos++){
      if(nmsimg[pos] == POSSIBLE_EDGE) edge0[pos] = POSSIBLE_EDGE;
      else edge0[pos] = NOEDGE;
      }
   }

   for(r=0,pos=0;r<rows;r++,pos+=cols){
      edge0[pos] = NOEDGE;
      edge0[pos+cols-1] = NOEDGE;
   }
   pos = (rows-1) * cols;
   for(c=0;c<cols;c++,pos++){
      edge0[c] = NOEDGE;
      edge0[pos] = NOEDGE;
   }

   /****************************************************************************
   * Compute the histogram of the magnitude image. Then use the histogram to
   * compute hysteresis thresholds.
   ****************************************************************************/
   for(r=0;r<32768;r++) hist[r] = 0;
   for(r=0,pos=0;r<rows;r++){
      for(c=0;c<cols;c++,pos++){
      if(edge0[pos] == POSSIBLE_EDGE) hist[mag[pos]]++;
      }
   }

   /****************************************************************************
   * Compute the number of pixels that passed the nonmaximal suppression.
   ****************************************************************************/
   for(r=1,numedges=0;r<32768;r++){
      if(hist[r] != 0) maximum_mag = r;
      numedges += hist[r];
   }

   highcount = (int)(numedges * thigh + 0.5);

   /****************************************************************************
   * Compute the high threshold value as the (100 * thigh) percentage point
```

```
        * in the magnitude of the gradient histogram of all the pixels that passes
        * non-maximal suppression. Then calculate the low threshold as a fraction
        * of the computed high threshold value. John Canny said in his paper
        * "A Computational Approach to Edge Detection" that "The ratio of the
        * high to low threshold in the implementation is in the range two or three
        * to one." That means that in terms of this implementation, we should
        * choose tlow ~= 0.5 or 0.33333.
        ****************************************************************************/
        r = 1;
        numedges = hist[1];
        while((r<(maximum_mag-1)) && (numedges < highcount)){
           r++;
           numedges += hist[r];
        }
        highthreshold = r;
        lowthreshold = (int)(highthreshold * tlow + 0.5);

        if(VERBOSE){
           printf("The input low and high fractions of %f and %f computed to\n",
           tlow, thigh);
           printf("magnitude of the gradient threshold values of: %d %d\n",
           lowthreshold, highthreshold);
        }

        /****************************************************************************
        * This loop looks for pixels above the highthreshold to locate edges and
        * then calls follow_edges to continue the edge.
        ****************************************************************************/
        for(r=0,pos=0;r<rows;r++){
           for(c=0;c<cols;c++,pos++){
           if((edge0[pos] == POSSIBLE_EDGE) && (mag[pos] >= highthreshold)){
                 edge0[pos] = EDGE;
                 follow_edges((edge0+pos), (mag+pos), lowthreshold, cols);
           }
           }
        }

        /****************************************************************************
        * Set all the remaining possible edges to non-edges.
        ****************************************************************************/
        for(r=0,pos=0;r<rows;r++){
           for(c=0;c<cols;c++,pos++) if(edge0[pos] != EDGE) edge0[pos] = NOEDGE;
        }
     }

   void main()
   {
      apply_hysteresis(magnitude, nms,  ROWS, COLS, TLOW, THIGH, edge);
      ImgOut.send(edge);
   }
};
```

```
/*****************************************************************************
* DUT: To perform canny edge detection.
*****************************************************************************/

behavior DUT(i_img_receiver ImgIn, i_img_sender ImgOut)
{
//changed by Jiang Wan for pipeline
   piped imgs smoothedim;
   piped imgs delta_x;
   piped imgs delta_y;
   piped imgs magnitude;
   piped imgs magnitude_p1;
   piped img nms;
   int i;

   Gaussian_Smooth      gaussian_smooth(ImgIn, smoothedim);
   Mag_Delta            mag_delta(smoothedim, delta_x, delta_y, magnitude);
   Non_Max_Supp         non_max_supp(delta_x, delta_y, magnitude, nms, magnitude_p1);
   Apply_Hysteresis     apply_hysteresis(magnitude_p1, nms, ImgOut);

    void main()
    {
     pipe(i=0; i<IMG_NUM; i++)
     {
        gaussian_smooth.main();
        mag_delta.main();
        non_max_supp.main();
        apply_hysteresis.main();
     }
    }
};


behavior DataIn(i_img_receiver ImgIn, i_img_sender ImgOut)
{
    unsigned char image[SIZE];

    void main()
    {
          while(1)
          {
         ImgIn.receive(&image);
         ImgOut.send(image);
          }
    }
};


behavior DataOut(i_img_receiver ImgIn, i_img_sender ImgOut)
{
    unsigned char image[SIZE];
```

```

    void main()
    {
            while(1)
            {
              ImgIn.receive(&image);
              ImgOut.send(image);
            }
    }
};


behavior Platform(i_img_receiver ImgIn, i_img_sender ImgOut)
{
  c_img_queue    q1(2ul),
                 q2(2ul);
  DataIn         din(ImgIn, q1);
  DUT            canny(q1, q2);
  DataOut        dout(q2, ImgOut);

  void main()
  {
    par{
       din.main();
       canny.main();
       dout.main();
       }
  }
};


behavior Main()
{
   sim_time      t[IMG_NUM];
   c_img_queue   q1(2ul),
                 q2(2ul);

   Stimulus      stimulus(q1, t);
   Platform      platform(q1, q2);
   Monitor       monitor(q2, t);

   int main()
   {
     par{
         stimulus.main();
         platform.main();
         monitor.main();
        }
     return 0;  // never reached
   }
};
```

A.2 Makefile for TLM generation in SCE

SCE tool has a graphical user interface for design space exploration. However, in order to ease the regeneration of the TLM model, a Makefile has been created. The Makefile contains the commands to SCE tool for different design settings and refinements.

```
#clock period for ARM CPU (in nanoseconds)
CPUCLKP         = 10000

#bus
BUSCLKP         = 20000

SPECC           = /opt/sce
SCE_PATH        = $(SPECC)
SCEDB_PATH      = $(SCE_PATH)/share/sce/db
PROCDB_PATH     = $(SCEDB_PATH)/processors
COMMDB_PATH     = $(SCEDB_PATH)/communication
BUSDB_PATH      = $(SCEDB_PATH)/busses

SYSC_PATH       = /opt/pkg/systemc-2.1.v1

SCENV           =SPECC=$(SPECC) SCERC_PATH=$$PWD/.sce \
                PATH=$(SPECC)/bin:$${PATH}      \
                LD_LIBRARY_PATH=$(SPECC)/lib$${LD_LIBRARY_PATH:+:}$${LD_LIBRARY_PATH}


PROC_CACHE      = .sce/processors
COMM_CACHE      = .sce/communication
BUS_CACHE       = .sce/busses
RTL_CACHE       = .sce/rtl
SCE_LOCK        = .sce/.scerc.lock

FINALPKG        = final.tar.gz

MAINSIRFILES    = cannySpec.sir cannyArch.sir cannySched.sir  \
                cannyNet.sir cannyTlm.sir cannyComm.sir       \
                cannyRTLC.sir cannyISS.sir

SYSC            = g++ -m32
SYSCOPT         = -Di386
SYSCINC         = -I$(SYSC_PATH)/include
SYSCLIB         = -L$(SYSC_PATH)/lib-linux -lsystemc

TIME            = /usr/bin/time

SCC             = scc
SCCOPT          = -ww -vv -d  -xlx
SCCINC          =-Isrc
SCCIMP          =-Psrc
SIR_RENAME      = sir_rename
SIR_RENAMEOPT   = -v


SIR_STATS       = sir_stats
SIR_STATSOPT    =

SIR_GEN         = sir_gen
SIR_GENOPT      = -vv

SIR_IMPORT      = sir_import
SIR_IMPORTOPT   = -v

SCSH            = scsh
SCSHOPT         =

SCE_RETYPE      = sce_retype
SCE_RETYPEOPT   = -v

SCE_IMPORT      = sce_import
SCE_IMPORTOPT   = -v
```

```
SCE_ALLOCATE    = sce_allocate
SCE_ALLOCATEOPT = -v

SCE_TOP         = sce_top
SCE_TOPOPT      = -v

SCE_MAP         = sce_map
SCE_MAPOPT      = -v

SCE_SCHEDULE    = sce_schedule
SCE_SCHEDULEOPT = -v

SCE_CONNECT     = sce_connect
SCE_CONNECTOPT  = -v

SCE_PROJECT     = sce_project
SCE_PROJECTOPT  = -v

SCPROF          = $(TIME) scprof
SCPROFOPT       = -v

SCAR            = $(TIME) scar
SCAROPT         =

SCOS            = $(TIME) scos
SCOSOPT         = -v

SCNR            = $(TIME) scnr
SCNROPT         = -v -O -falign -fmerge

SCCR            = $(TIME) sccr
SCCROPT         = -v -O

ECHO            = echo
MAKE            = make
# ----- TARGET RULES ---------------------------------------------------------------


all:
	$(SCENV) $(MAKE) cannyTlm.SIM_OK

test:
	$(SCENV) $(MAKE) cannySpec.SIM_OK

final:	$(FINALPKG)

clean:
	$(RM) cannySpec cannyArch cannySched cannyNet          \
	      cannyTlm cannyComm cannyRTLC cannyISS
	$(RM) $(FINALPKG)
	$(RM) *.ins *.sysc
	$(RM) *.si *.sir *.cc *.h *.o *.cpp *.hpp
	$(RM) *.dpr *.prf
	$(RM) *.SIM_OK
	$(RM) core
	$(RM) image_s_0.60_l_0.30_h_0.80_*
	$(RM) -r $(PROC_CACHE)
	$(RM) -r $(COMM_CACHE)
	$(RM) -r $(BUS_CACHE)
	$(RM) -r $(RTL_CACHE)
	$(RM) $(SCE_LOCK)

stats:	$(MAINSIRFILES)
	$(SIR_STATS) $(SIR_STATSOPT) $(MAINSIRFILES)

loc:	$(MAINSIRFILES)
	$(SIR_STATS) $(SIR_STATSOPT) -BCI $(MAINSIRFILES)


# ----- GENERAL RULES ------------------------# -o $@ $*------------------------------
```

```
.SUFFIXES:
.SUFFIXES:      .ins.sir .sir .ins .ana.sir
.SUFFIXES:      .pgm .SIM_OK

.sir.ins.sir:
	@$(ECHO) "***"
	@$(ECHO) "*** Instrumenting $* for profiling ..."
	@$(ECHO) "***"
	$(SCPROF) -v -m  -i $< -o $@ $*

.sir:
	@$(ECHO) "***"
	@$(ECHO) "*** Compiling $* for execution ..."
	@$(ECHO) "***"
	$(SCC) $* -sir2out $(SCCOPT) $(SCCINC) $(SCCIMP)

.ins.sir.ins:
	@$(ECHO) "***"
	@$(ECHO) "*** Compiling $* for execution with profiling ..."
	@$(ECHO) "***"
	$(SCC) $* -sir2out $(SCCOPT) $(SCCINC)$(SCCIMP) -i $< -o $@ -lscprof

.pgm.SIM_OK:
	@$(ECHO) "***"
	@$(ECHO) "*** Simulating $*"
	@$(ECHO) "***"
	$(TIME) ./$*.ins  | tee log_execution_$*
	@$(ECHO) "***"
	@$(ECHO) "*** Simulation successful!"
	@$(ECHO) "***"

.sir.ana.sir:
	@$(ECHO) "***"
	@$(ECHO) "*** Back-annotating profiling data and running estimation ..."
	@$(ECHO) "***"
	$(SCPROF) -E $(SCPROFOPT) -i $< -o $@ $*

# --- SPECIFIC RULES ----------------------------------------------------------

# --- compile the sources
cannySpecpre.sir: canny.sc
	@$(ECHO) "***"
	@$(ECHO) "*** Compiling the sources ..."
	@$(ECHO) "***"
	$(SCC) canny -sc2sir $(SCCOPT)  $(SCCINC) $(SCCIMP)\
		-i canny.sc  -o cannySpecpre.sir

cannySpec.sir:  cannySpecpre.sir
	@$(ECHO) "***"
	@$(ECHO) "*** Setting top level of design under test ..."
	@$(ECHO) "***"
	$(SCE_TOP) $(SCE_TOPOPT) -s Platform \
	-i cannySpecpre.sir -o cannySpec.sir cannySpec

cannySpec.ins.sir: cannySpec.sir
cannySpec:      cannySpec.sir
cannySpec.ins:  cannySpec.ins.sir
cannySpec.pgm:  cannySpec.ins
cannySpec.SIM_OK:       cannySpec.pgm
cannySpec.ana.sir: cannySpec.sir cannySpec.SIM_OK

# --- back-annotate raw profiling data
cannySpec.prof.sir: cannySpec.sir cannySpec.SIM_OK
	@$(ECHO) "***"
	@$(ECHO) "*** Back-annotating raw profiling data ..."
	@$(ECHO) "***"
	$(SCPROF) -p $(SCPROFOPT) -i cannySpec.sir -o cannySpec.prof.sir cannySpec

cannySpec.3.sir:  cannySpec.prof.sir
	@$(ECHO) "***"
	@$(ECHO) "*** Allocating the component ..."
```

```
	@$(ECHO) "***"
	$(SCE_ALLOCATE) $(SCE_ALLOCATEOPT)\
		-g PORTA_HCLK_PERIOD=$(BUSCLKP) -g CLOCK_PERIOD=$(CPUCLKP) -p CPU=ARM_7TDMI\
		-g CLOCK_PERIOD= -p IO_OUT=HW_Virtual  \
		-g CLOCK_PERIOD= -p IO_IN=HW_Virtual  \
		-g CLOCK_PERIOD= -p Hardware_BlurX=HW_Standard\
		-g CLOCK_PERIOD= -p Hardware_BlurY=HW_Standard \
		-i cannySpec.prof.sir -o cannySpec.3.sir cannySpec

cannySpec.4.sir:	cannySpec.3.sir
	@$(ECHO) "***"
	@$(ECHO) "*** Map the components.."
	@$(ECHO) "***"
	$(SCE_MAP) $(SCE_MAPOPT) -p Platform.canny=CPU \
				 -p Platform.din=IO_IN \
				 -p Platform.dout=IO_OUT \
				 -p Platform.canny.gaussian_smooth.blurY_par=Hardware_BlurY\
				 -p Platform.canny.gaussian_smooth.blurX_par=Hardware_BlurX\
				 -i cannySpec.3.sir -o cannySpec.4.sir cannySpec

# --- analyze the profile given the behavior mapping
cannySpec.ana1.sir:	cannySpec.4.sir
	@$(ECHO) "***"
	@$(ECHO) "*** Analyzing the behavior mapping..."
	@$(ECHO) "***"
	$(SCPROF) -a $(SCPROFOPT) -i cannySpec.4.sir \
		-o cannySpec.ana1.sir canny

# --- estimate the profile given the behavior mapping
cannySpec.ana2.sir:	cannySpec.4.sir
	@$(ECHO) "***"
	@$(ECHO) "*** Estimating the behavior mapping..."
	@$(ECHO) "***"
	$(SCPROF) -e $(SCPROFOPT) -i cannySpec.4.sir \
		-o cannySpec.ana2.sir canny

cannySpec.arch.in.sir:	cannySpec.ana2.sir cannySpec.ana1.sir
	@$(ECHO) "***"
	@$(ECHO) "*** Importing the PEs into the design..."
	@$(ECHO) "***"
	$(SCE_IMPORT) $(SCE_IMPORTOPT) -a \
		-i cannySpec.ana2.sir -o cannySpec.arch.in.sir \
		cannySpec

cannyArch.sir:	cannySpec.arch.in.sir
	@$(ECHO) "***"
	@$(ECHO) "*** Performing architecture refinement..."
	@$(ECHO) "***"
	$(SCAR) cannySpec -b -m -c -w $(SCAROPT) \
		-i cannySpec.arch.in.sir -o cannySpec.arch.sir
	$(SIR_RENAME) $(SIR_RENAMEOPT) -i cannySpec.arch.sir \
		cannySpec cannyArch
	$(SIR_STATS) $(SIR_STATSOPT) $@

cannyArch.ins.sir: cannyArch.sir
cannyArch.ins:	cannyArch.ins.sir
cannyArch.pgm:	cannyArch.ins
cannyArch.SIM_OK:	cannyArch.pgm
cannyArch.ana.sir:	cannyArch.sir cannyArch.SIM_OK

cannyArch.2.sir:	cannyArch.ana.sir
	@$(ECHO) "***"
	@$(ECHO) "*** Deciding scheduling..."
	@$(ECHO) "***"
	$(SCE_SCHEDULE) $(SCE_SCHEDULEOPT) -r -t Platform\
		-k CPU=ARM_7TDMI_OSNONE_$(CPUCLKP)_$(BUSCLKP)_0 \
		-i cannyArch.ana.sir -o cannyArch.2.sir cannyArch

# --- import components for scheduling refinement
cannyArch.sched.in.sir: cannyArch.2.sir
	@$(ECHO) "***"
```

```
	@$(ECHO) "*** Importing components needed for scheduling ..."
	@$(ECHO) "***"
	$(SCE_IMPORT) $(SCE_IMPORTOPT) -s \
		-i cannyArch.2.sir -o cannyArch.sched.in.sir \
		cannyArch

# --- perform static scheduling refinement
cannyArch.sched.tmp.sir:	cannyArch.sched.in.sir
	@$(ECHO) "***"
	@$(ECHO) "*** Performing static scheduling refinement ..."
	@$(ECHO) "***"
	$(SCAR) cannyArch $(SCAROPT) -s \
		-i cannyArch.sched.in.sir -o cannyArch.sched.tmp.sir

# --- perform dynamic scheduling refinement
cannySched.sir: cannyArch.sched.tmp.sir
	@$(ECHO) "***"
	@$(ECHO) "*** Performing dynamic scheduling refinement ..."
	@$(ECHO) "***"
	$(SCOS) cannyArch $(SCOSOPT) \
		-i cannyArch.sched.tmp.sir -o cannyArch.sched.sir
	$(SIR_RENAME) $(SIR_RENAMEOPT) -i cannyArch.sched.sir \
		cannyArch cannySched
	$(SIR_STATS) $(SIR_STATSOPT) $@

cannySched.ins.sir: cannySched.sir
cannySched.ins: cannySched.ins.sir
cannySched.pgm:  cannySched.ins
cannySched.SIM_OK:	cannySched.pgm
cannySched.ana.sir: cannySched.sir cannySched.SIM_OK


cannySched.2.sir:	cannySched.ana.sir
	@$(ECHO) "***"
	@$(ECHO) "*** Allocating the CPU bus and the Hardware Bus ..."
	@$(ECHO) "***"
	$(SCE_ALLOCATE) $(SCE_ALLOCATEOPT) \
		-b CPU_Bus=AMBA_AHB \
		-b HW_Bus=HardwareBus\
		-i cannySched.ana.sir -o cannySched.2.sir cannySched

# --- connect the PE and the busses
cannySched.3.sir:	cannySched.2.sir
	@$(ECHO) "***"
	@$(ECHO) "*** Connecting PE and the busses ..."
	@$(ECHO) "***"
	$(SCE_CONNECT) $(SCE_CONNECTOPT) -c CPU,PortA=CPU_Bus,master0\
		-c Hardware_BlurX,Port0=CPU_Bus,slave4 -c Hardware_BlurX,Port1=HW_Bus,Master\
		-c Hardware_BlurY,Port0=CPU_Bus,slave5 -c Hardware_BlurY,Port1=HW_Bus,Slave\
		-c IO_OUT,Port0=CPU_Bus,slave8 \
		-c IO_IN,Port0=CPU_Bus,slave7 \
		-i cannySched.2.sir -o cannySched.3.sir cannySched

# --- import components for network refinement
cannySched.net.in.sir:  cannySched.3.sir
	@$(ECHO) "***"
	@$(ECHO) "*** Importing components needed for network refinement ..."
	@$(ECHO) "***"
	$(SCE_IMPORT) $(SCE_IMPORTOPT) -e \
		-i cannySched.3.sir -o cannySched.net.in.sir \
		cannySched

# --- perform network refinement
cannyNet.sir:   cannySched.net.in.sir
	@$(ECHO) "***"
	@$(ECHO) "*** Performing network refinement ..."
	@$(ECHO) "***"
	$(SCNR) cannySched $(SCNROPT) \
		-i cannySched.net.in.sir -o cannySched.net.sir
	$(SIR_RENAME) $(SIR_RENAMEOPT) -i cannySched.net.sir \
		cannySched cannyNet
```

```
	$(SIR_STATS) $(SIR_STATSOPT) $@

cannyNet.ins.sir: cannyNet.sir
cannyNet.ins:	cannyNet.ins.sir
cannyNet.pgm:	cannyNet.ins
cannyNet.SIM_OK:	cannyNet.pgm
cannyNet.ana.sir: cannyNet.SIM_OK cannyNet.sir

# --- define the link parameters
cannyNet.2.sir: cannyNet.ana.sir
	@$(ECHO) "***"
	@$(ECHO) "*** Defining the link parameters for the busses..."
	@$(ECHO) "***"
	$(SCE_MAP) $(SCE_MAPOPT) -l c_link_IO_IN__CPU=0x70000000,0 \
		-l c_link_CPU__IO_OUT=0x80000000,0\
		-l c_link_CPU__Hardware_BlurX=0x40000000,0\
		-l c_link_CPU__Hardware_BlurY=0x50000000,0\
		-l c_link_Hardware_BlurX__Hardware_BlurY=0x0000,MasterSync0\
		-i cannyNet.ana.sir -o cannyNet.2.sir cannyNet

# --- import the busses into the design
cannyNet.comm.in.sir:	cannyNet.2.sir
	@$(ECHO) "***"
	@$(ECHO) "*** Importing the busses into the design..."
	@$(ECHO) "***"
	$(SCE_IMPORT) $(SCE_IMPORTOPT) -c \
		-i cannyNet.2.sir -o cannyNet.comm.in.sir \
		cannyNet

# --- perform communication refinement to TLM
cannyTlm.sir:	cannyNet.comm.in.sir
	@$(ECHO) "***"
	@$(ECHO) "*** Performing communication refinement to TLM..."
	@$(ECHO) "***"
	$(SCCR) cannyNet $(SCCROPT) -t \
		-i cannyNet.comm.in.sir -o cannyNet.comm.sir
	$(SIR_RENAME) $(SIR_RENAMEOPT) -i cannyNet.comm.sir \
		cannyNet cannyTlm
	$(SIR_STATS) $(SIR_STATSOPT) $@

cannyTlm.ins.sir: cannyTlm.sir
cannyTlm.ins:	cannyTlm.ins.sir
cannyTlm.pgm:	cannyTlm.ins
cannyTlm.SIM_OK:	cannyTlm.pgm
cannyTlm.ana.sir:	cannyTlm.SIM_OK cannyTlm.sir
```