Appendix A

Function Details

Analog_pro

performs analog projection for a gray scale image.

function prototype

int **Analog_pro**(char *a,int *N1*,int *M1*,int *N2*,int *M2*)

where *a* is the input image buffer.

N1,M1 is the start coordinate in the image.

N2,M2 is the end coordinate in the image.

BB_thres

performs threshold value selection based on boundary characteristics.

function prototype

char ***BB_thres**(char **str*,char *type*)

where *str* is the input image buffer.

type is the type of objects.

0 for black objects on white background.

1 for white objects on black background.

DFT

performs discrete Fourier transform of the input image.

function prototype

char ***DFT**(char **str*,int *type*)

where *str* is the input image buffer.

type is the type of operation.

0 for shift center frequency.

1 for non-shift center frequency.

Digital_pro

performs projection for a binary image.

function prototype

int **Digital_pro**(char *a,int *N1*,int *M1*,int *N2*,int *M2*)

where *a* is the input image buffer.

N1,M1 is the start coordinate in the image.

N2,M2 is the end coordinate in the image.

Global_T

performs global thresholding.

function prototype

char ***Global_T**(char **str*)

where *str* is the input image buffer.

ViewGif

displays an image in GIF format on a screen.

function prototype

void **ViewGif**(char **str*)

where *str* is the buffer of an image in GIF format.

ViewImg

displays an image in IMG format on a screen.

function prototype

void **ViewImg**(char **str*)

where *str* is the buffer for input image in IMG format.

_13Vtextxy

displays text in graphic mode 13h. The color of the text is calculated automatically.

function prototype

void **_13Vtextxy**(unsigned *x*,unsigned *y*,char **msg*)

where x,y is the start coordinate.

$0 \leq x \leq 320$ and $0 \leq y \leq 200$

msg is the message to be displayed.

_13box

draws a square box.

function prototype

void **_13box**(int *x1*,int *y1*,int *x2*,int *y2*,int *color*,int *paint*)

where *x1,y1* is the start coordinate of the box.

$0 \leq x1 \leq 320$ and $0 \leq y1 \leq 200$

x2,y2 is the end coordinate of the box.

$0 \leq x2 \leq 320$ and $0 \leq y2 \leq 200$

color is the color of the box.

paint is 1 if the box is painted.

_13capture_screen (screen capture)

captures the screen and saves to the file in IMG format.

function prototype

void **_13capture_screen**(char **a*,WORD *X1*,WORD *Y1*,WORD *X2*,WORD Y2)

where *a* is the file name to be saved.

X1, Y1 is the start coordinate of the window.

$0 \leq X1 \leq 320$ and $0 \leq Y1 \leq 200$

X2, Y2 is the end coordinate of the window.

$0 \leq X2 \leq 320$ and $0 \leq Y2 \leq 200$

_13getimage

copy the data from display memory, in display mode 13h, to an image buffer.

function prototype

void **_13getimage**(*x1,y1, x2,y2,buff*)

where *x1,y1* is the start coordinate of the window.

$0 \leq x1 \leq 320$ and $0 \leq y1 \leq 200$

x2,y2 is the end coordinate of the window.

$0 \leq x2 \leq 320$ and $0 \leq y2 \leq 200$

buff is the image buffer.

_13imagesize

calculates the memory size required for stored the image.

function prototype

unsigned **_13imagesize**(*x1,y1,x2,y2*)

where *x1,y1* is the start coordinate of the window.

$0 \leq x1 \leq 320$ and $0 \leq y1 \leq 200$

x2,y2 is the end coordinate of the window.

$0 \leq x2 \leq 320$ and $0 \leq y2 \leq 200$

_13putimage

displays an image from the buffer saved by function _13getimag.

function prototype

void **_13putimage**(unsigned *x*,unsigned *y*,BYTE **buff*)

where *x,y* is the start coordinate of the image.

$0 \leq x \leq 320$ and $0 \leq y \leq 200$

buff is the image buffer.

_13rotage

rotates the displayed image on the screen.

function prototype

void **_13rotage**(unsigned int *x1*,unsigned int *y1*,int *x2*,int *y2*,float *ang*)

where *x1*,*y1* is the start coordinate of the window.

$0 \leq x1 \leq 320$ and $0 \leq y1 \leq 200$

x2,*y2* is the end coordinate of the window.

$0 \leq x2 \leq 320$ and $0 \leq y2 \leq 200$

ang is the angle of rotation in radian.

_13textxy

displays text on the screen.

function prototype

void **_13textxy**(unsigned int *x*,unsigned int *y*,char **msg*,unsigned char *color*)

where *x*,*y* is the start coordinate of the displayed text.

$0 \leq x \leq 320$ and $0 \leq y \leq 200$

msg is the message to be displayed.

color is the color of the message.

add_CG

returns the address of the character generator ROM.

function prototype

char ***add_CG**(char *mode*)

where *mode* is the number of character sets.

average_f

performs average filtering.

function prototype

char ***average_f**(char **str*,int *Nw*,int *Nd*)

where *str* is the name of image file in IMG format.

Nw and *Nd* specify template sizes, e.g. 3x3,4x4 etc..

capture_video

captures the image from the video camera and saves to the image file in IMG format.

function prototype

void **capture_video**(char *a,WORD *X1*,WORD *Y1*,WORD *X2*,WORD *Y2*)

where *a* is the name of the file to be saved.

X1,*Y1* is the start coordinate of the window.

$0 \leq X1 \leq 320$ and $0 \leq Y1 \leq 200$

X2,*Y2* is the end coordinate of the window.

$0 \leq X2 \leq 320$ and $0 \leq Y2 \leq 200$

chain_code

performs chain coding.

function prototype

int **chain_code**(char *a,char *b,int *TMAX*,int *N1*,int *M1*,int *N2*,int *M2*,int *TYPE*)

where *a* is the name of input image file in IMG format.

.*b* is the name of output image file in CHN format.

TMAX is the maximum points, default is 3000.

N1,*M1* is the start coordinate of the window.

N2,*M2* is the end coordinate of the window.

TYPE is the type of chain code used, 4 or 8.

clsg

clears screen in graphic mode.

function prototype

void clsg(void)

contract

performs image contraction.

function prototype

int contract(char *a,char *b,int t,int *ND1*,int *MD1*,int *N1*,int *M1*,int *N2*,int *M2*)

where *a* is the file name of the input image.

b is the file name of the output image.

int *t* is the contraction factor.

ND1,*MD2* is the start coordinate of the destination buffer.

N1,*M1*is the start coordinate of the window.

N2,*M2* is the end coordinate of the window.

curve_split

performs polygon approximation.

function prototype

int curve_split(char *a,float *T*,int *n*)

where *a* is the file name of the input image in IMG format.

T is the error specified, default is 0.1.

n is the maximum points, default is 10000.

cutimage

cuts the area in the image.

function prototype

void cutimage(char *a,char *b,int *Ns*,int *Ms*,int *Ne*,int *Me*)

erosion

perform morphological erosion.

function prototype

void **erosion**(char *a,char *b)

where *a* is the input file name in IMG format.

b is the output file name in IMG format.

erosion_buffer

performs morphological erosion and saves the result to the buffer.

function prototype

void **erosion_buff**(BYTE **buf1*,BYTE **buf2*,int *W*,int *H*)

where *buf1* is the input buffer.

buf2 is the output buffer.

W is the width of the image.

H is the height of the image.

expand

performs image expansion.

function prototype

int **expand**(char *a,char *b,int *t*,int *ND1*,int *MD1*,int *N1*,int *M1*,int *N2*,int *M2*)

where *a* is the input buffer.

b is the output buffer.

t is the expansion factor.

ND1,MD2 is the start coordinate of the destination buffer.

N1,M1 is the start coordinate of the image.

N2,M2 is the end coordinate of the image.

freez

freezes the live display, i.e. stops image conversion, in order to communicate with the Video Blaster card.

function prototype

void **freez**()

halftone

performs image halftoning.

function prototype

int **halftone**(char *a,char *b,int *cn*,int *N1*,int *M1*,int *N2*,int *M2*)

where *a* is the file name of the input image.

b is the buffer of an output image.

cn is the number of halftone.

N1,M1 is the start coordinate of the image.

N2,M2 is the end coordinate of the image.

huffman_decode

decodes Huffman-encoded image.

function prototype

void **huffman_decode**(char **file1*,char **file2*)

where *file1* is the input file name in Huffman-encoded format.

file2 is the output file name in IMG format.

huffman_encode

encodes an image file in an IMG format using Huffman coding.

function prototype

void **huffman_encode**(char **file1*,char **file2*)

where *file1* is the input file name in an IMG format.

file2 is the file name of an Huffman-compressed image.

initFont

initializes the Thai font of Thai word processor (CW).

function prototype

void **initFont**(char *a)

where *a* is the file name of the font file.

initGDT

initializes the Global Describtion Table (GDT) in order to use the memory that is above1MB.

function prototype

void **initGDT**(void)

initMODE

initializes the display card in high resolution graphics mode.

function prototype

void **initMODE**()

init_video

warm-starts the Video Blaster card and specifies the display windows to 320 x 200 in mode 13h.

function prototype

void *init_video*(WORD *WIDTH*,WORD *HIGH*)

where *WIDTH* is the width of the display image.

HIGH is the height of the display image.

inverst

turns the image into the negative image.

function prototype

void **inverst**(char *a,char *b)

where *a* is the name of the input file in IMG format.

b is the name of the output file in IMG format.

Laplace

performs Laplacian edge detection.

function prototype

int **Laplace**(char *a,char **b*,int *N1*,int *M1*,int *N2*,int *M2*)

where *a* is the input image buffer.

b is the output image buffer.

N1,M1 is the start coordinate of the image.

N2,M2 is the end coordinate of the image.

linear_histogram

peforms linear histogram.

function prototype

void **linear_histogram**(char *a,char **b*)

where *a* is the file name of the input image in IMG format.

b is the name of the output image in IMG format.

local_en

performs local enhancement.

function prototype

char ***local_en**(char **str*,int *type*)

where *str* is the file name of the input image in IMG format.

type can be 0 or 1.

median_f

performs median filtering.

function prototype

char *median_f(char *str,int Nw,int Nd)

where *str* is the name of the input file inIMG format.

Nw is the width of the window.

Nd is the length of the window.

move2hi

moves the data from the memory below 640 KB to the memory above 1 MB

function prototype

void move2hi(unsigned long *S*,unsigned long *D*,unsigned *Length*)

where *S* is the source address (segment:offset).

D is the destination address (real address).

Length is the number of bytes to be moved.

move2lo

moves the memory above 1 MB to the memory below 640 KB.

function prototype

void move2lo(unsigned long *S*,unsigned long *D*,unsigned *Length*)

where *S* is the source address (real address).

D is the destination address (segment:offset).

Length is the number of bytes to be moved.

msd

performs morphological shape decomposition.

function prototype

int msd(char *a,char *b,char *c, char *d,int *ki*, int *N1*,int *M1*,int *N2*,int *M2*)

where *a* is the input image buffer .

b, c are the intermediate buffers.

d is the output image file.

ki is 1 for square structuring elements.

N1,M1 is the start coordinate.

N2,M2 is the end coordinate.

op_mask

performs mask processing.

function prototype

void **op_mask**(char *a,char *b,int *d)

where *a* is the name of input file in IMG format.

b is the name of the out file in IMG format.

d is the pointer to mask (3x3).

operate

performs point processing of two images.

function prototype

void **operate**(char *a,char *b,char *c,char *d)

where *a* is the name of the first file in IMG format.

b is the name of the second file in IMG format.

c is the name of the output file in IMG format.

d is '-' or '+', '-' for image subtraction, '+' for image addition.

outthaiXY

displays Thai text in graphics mode which is initialised using **initMODE()** or Turbo-C's **initgraph()**.

function prototype

void **outthaiXY**(int *x*,int *y*,int *c*,BYTE **msg*)

where *x,y* is the start coordinate of the text.

c is the specified color.

msg is the message to be displayed.

quadtree

performs quadtree encoding.

function prototype

void **quadtree**(char **img_in*,char **img_out*,int *Ns*,int *Ms*,int *level*)

where *img_in* is the buffer for input image in .IMG format.

img_out is the buffer for input image in .IMG format.

Ns,Ms is the start coordinate of interested window.

level is the size of the interested window.

quantiz_decode

performs quantization decoding

function prototype

void **quantiz_decode**(char **a*,char **b*)

where *a* is the buffer for input image in .IMG format.

b is the buffer for input image in .IMG format.

quantiz_encode

performs quantization encoding

function prototype

void **quantiz_encode**(char **a*,char **b*)

where *a* is the buffer for input image in IMG format.

b is the buffer for input image in IMG format.

range

performs range edge detecion using local maximum and local minimum values.

function prototype

int **range**(char *a,char *b,int *N1*,int *M1*,int *N2*,int *M2*)

where *a* is the buffer for input image.

b is the buffer for output image.

N1,M1 is the start coordinate of the image.

N2,M2 is the start coordinate of the image.

read2buff

reads image data from Video Blaster card and saves it to buffer.

function prototype

void **read2buff**(BYTE **buff*,WORD *X1*,WORD *Y1*,WORD *X2*,WORD *Y2*)

where *buff* is buffer for image data in VIDEO BLASTER card.

X1, Y1 is the start coordinate of the windows.

$0 \leq X1 \leq 320$ and $0 \leq Y1 \leq 200$

X2, Y2 is the end coordinate of the windows.

$0 \leq X2 \leq 320$ and $0 \leq Y2 \leq 200$

requantize

requantizes the image in the form that is ready for image compression.

function prototype

void **requantize**(char *a,char *b,float *percent_err*)

where *a* is the buffer for input image in IMG format.

b is the buffer for input image in IMG format.

percent_err is the acceptable error.

round

converts floating point value to integer value.

function prototype

int **round**(float *a*)

where *a* is the value to be converted.

run_length

performs run length coding.

function prototype

void **run_length**(char **file1*,char **file2*,char *option*)

where *file1* is the buffer for the input image in IMG format.

file2 is the buffer for the output image in IMG format.

option is '1' for encoding, '2' for decoding.

set_BW

converts a colour image to a gray-scale image.

function prototype

void **set_BW**(void)

shannon_decode

decompresses an image using Shannon-Fano algorithm.

function prototype

void **shannon_decode**(char *a,char *b)

where *a* is the buffer for the input image in IMG format.

b is the buffer for the output image in IMG format.

shannon_encode

compresses an image using Shannon-Fano algorithm.

function prototype

void **shannon_encode**(char *a,char *b)

where *a* is the buffer for input image in IMG format.

b is the buffer for output image in IMG format.

sharpen

edge-enhances an image.

function prototype

char ***sharpen**(char *str,int type)

where *str* is the buffer for the input image in IMG format.

type is the value between 0 to 4 which specifies the type of masks.

showCHN

displays a chain coded image.

function prototype

int showCHN(char *a)

where *a* is the buffer for input image in CHN format (chain-coded image).

showIMG

displays an IMG image.

function prototype

int showIMG(char *a)

where *a* is the buffer for input image in IMG format.

show_SGN

shows a signature image

function prototype

void show_SGN(char *a,int *Type*)

where *a* is the buffer for the input image in SGN format (signatured image).

Type is 0 for start at minimum, 1 for start at maximum.

show_histogram

shows histogram of the image.

function prototype

void show_histogram(char *a,BYTE *color*)

where *a* is the buffer for the input image in IMG format.

color is the colour of the histogram.

signature

finds the charateristics of image using its signature.

function prototype

void signature(char *a,char *b,int *Type*)

where *a* is the buffer for the input image in CHN format (chain coded image).

b is the buffer for the output image in SGN format.

Type is 0 for start at minimum, 1for start at maximum.

skeleton

finds the skeleton of an input image using mathematical morphology.

function prototype

int skeleton(char *a,char *b,char *d,int *ki*,int *N1*,int *M1*,int *N2*,int *M2*)

where *a* is the buffer for the input image.

b,c are intermediate buffers.

d is the buffer for the output image.

ki is the type of structuring elements, 1 for square structuring element.

N1,M1 are starting points in X and Y directions.

N2,M2 are ending points in X and Y directions.

sobel

performs edge detection using Sobel masks.

function prototype

int sobel(char *a,char *b,int *N1*,int *M1*,int *N2*,int *M2*)

where *a* is the pointer to the input image in IMG format.

b is the pointer to the output image in IMG format.

N1,M1 are starting points in X and Y directions.

N2,M2 are ending points in X and Y directions.

space

calculates the percentage of area of objects to background.

function prototype

float space(char *a)

where *a* is the file name of the input image in IMG format.

terminate_video

terminates Video Blaster driver.

function prototype

void terminate_video()

threshold

converts a gray scale image to a corresponding binary image using a specified threshold value.

function prototype

void threshold(char *a,char *b,int *th*)

where *a* is the file name of the input image in IMG format.

b is the file name of the output image in IMG format.

th is the threshold.

threshold_exp

converts a gray scale image to a corresponding binary image using the threshold value which is calculated from the minimum value of pixels in the image.

function prototype

void **threshold_exp**(char *a,char *b,int *exp*)

where *a* is the file name of the input image in IMG format.

b is the file name of the output image in IMG format.

exp is the expansion value.

i.e. threshold = MinValue + exp

threshold_red

converts a gray scale image to a corresponding binary image using the threshold value which is calculated from the maximum value of pixels in the image.

function prototype

void **threshold_exp**(char *a,char *b,int *red*)

where *a* is the file name of the input image in IMG format.

b is the file name of the output image in IMG format.

red is the reduction value.

i.e. threshold = MaxValue - red

unfreez

recapture image data for Video Blaster card.

function prototype

void **unfreez**()

video_display_mode

sets the modes for Video Blaster card.

function prototype

void **video_display_mode**(WORD MODE)

where *MODE* is one of Video Blaster modes.