#ifndef BLOB\_LOCATOR\_FPS //this checks if the frames per second is defined.If not, it is set to zero

#define BLOB\_LOCATOR\_FPS 0 ///< Default FPS (zero means run at camera fps)

#endif

PRINT\_CONFIG\_VAR(BLOB\_LOCATOR\_FPS)

#include "modules/computer\_vision/cv\_blob\_locator.h"

#include "modules/computer\_vision/cv.h"

#include "modules/computer\_vision/blob/blob\_finder.h"

#include "modules/computer\_vision/blob/imavmarker.h"

#include "modules/computer\_vision/detect\_window.h"

uint8\_t color\_lum\_min; //declaration of lower upper middle filter minimum range value

uint8\_t color\_lum\_max; //declaration of lower upper middle filter maximum range value

uint8\_t color\_cb\_min; //declaration of minimum range value blue chromiance

uint8\_t color\_cb\_max; //declaration of maximum range value blue chromiance

uint8\_t color\_cr\_min; //declaration of minimum range value red chromiance

uint8\_t color\_cr\_max; //declaration of maximum range value red chromiance

uint8\_t cv\_blob\_locator\_reset;

uint8\_t cv\_blob\_locator\_type;

int geofilter\_length = 5; //initializes the length of time the geofilter will be active

int marker\_size = 18; //sets the size of the marker

int record\_video = 0;

volatile uint32\_t blob\_locator = 0;

volatile bool blob\_enabled = false; //blob locator is disabled

volatile bool marker\_enabled = false; //marker is disabled

volatile bool window\_enabled = false;

// Computer vision thread

struct image\_t \*cv\_marker\_func(struct image\_t \*img);

struct image\_t \*cv\_marker\_func(struct image\_t \*img)

{

if (!marker\_enabled) { //checks if the marker is disabled

return NULL;

}

// sends the image and marker size and gets the deviation of the marker location with respect to the center

struct marker\_deviation\_t m = marker(img, marker\_size);

uint32\_t temp = m.x; //Assigns marker deviation of x axis

temp = temp << 16;

temp += m.y; //Assigns marker deviation of y axis

blob\_locator = temp;

return NULL;

}

#define Img(X,Y)(((uint8\_t\*)img->buf)[(Y)\*img->w\*2+(X)\*2])

// Computer vision thread

struct image\_t \*cv\_window\_func(struct image\_t \*img);

struct image\_t \*cv\_window\_func(struct image\_t \*img)

{

if (!window\_enabled) {

return NULL;

}

uint16\_t coordinate[2] = {0, 0}; //initializes the image coordinates to zero

uint16\_t response = 0; //initializes the response obtained to zero

uint32\_t integral\_image[img->w \* img->h];

struct image\_t gray;

//calls the function sending width, height and type of the image and initializes the image structure gray

image\_create(&gray, img->w, img->h, IMAGE\_GRAYSCALE);

//converts the image to grayscale mode

image\_to\_grayscale(img, &gray);

//detects the window size when image buffer, width, height, coordinates of the image and mode are passed

response = detect\_window\_sizes((uint8\_t \*)gray.buf, (uint32\_t)img->w, (uint32\_t)img->h, coordinate, integral\_image, MODE\_BRIGHT);

//the image is freed

image\_free(&gray);

// Display the marker location and center-lines.

int px = coordinate[0] & 0xFFFe;

int py = coordinate[1] & 0xFFFe;

//checks if the response code is less than 92.

if (response < 92) {

for (int y = 0; y < img->h - 1; y++) {

Img(px, y) = 65;

Img(px + 1, y) = 255;

}

for (int x = 0; x < img->w - 1; x += 2) {

Img(x, py) = 65;

Img(x + 1, py) = 255;

}

uint32\_t temp = coordinate[0];

temp = temp << 16;

temp += coordinate[1];

blob\_locator = temp;

}

return NULL;

}

struct image\_t \*cv\_blob\_locator\_func(struct image\_t \*img);

struct image\_t \*cv\_blob\_locator\_func(struct image\_t \*img)

{

if (!blob\_enabled) { //checks if the blob is enabled

return NULL;

}

// sets the parameters of color filter with lum,blue chromiance,red chromiance range values

struct image\_filter\_t filter[2];

filter[0].y\_min = color\_lum\_min;

filter[0].y\_max = color\_lum\_max;

filter[0].u\_min = color\_cb\_min;

filter[0].u\_max = color\_cb\_max;

filter[0].v\_min = color\_cr\_min;

filter[0].v\_max = color\_cr\_max;

// Output image

struct image\_t dst;

//image is created by using the width,height and type and is stored in dst

image\_create(&dst,

img->w,

img->h,

IMAGE\_GRADIENT);

// Labels

uint16\_t labels\_count = 512; //indicates the total number of connected components in an image

struct image\_label\_t labels[512];

// Blob finder

//The connected components of an image are found and each one is labeled

image\_labeling(img, &dst, filter, 1, labels, &labels\_count);