Dezvoltarea unui SDK de Vedere Artificiala bazat pe driverele microcontrolerului Raspberry pi zero 2w/3b+/4

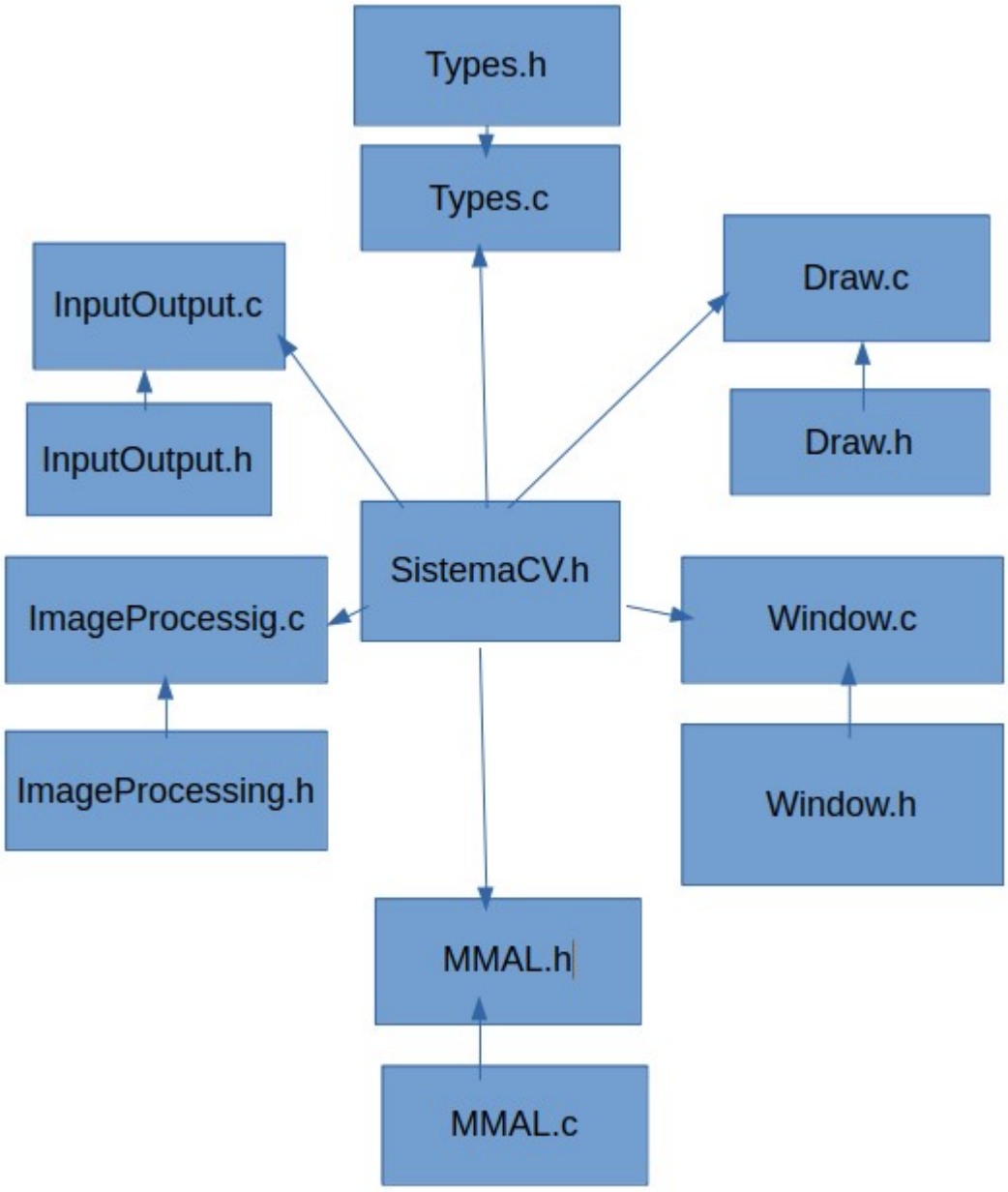
Tucudean Adrian-Ionut

1)Introducere

Motivul pentru care am ales sa dezvolt un SDK de computer vision specializat pentru familia de microcontrolere Raspberry pi este ca SDK-urile actuale folosesc la baza drivere generale care functioneaza pe mai multe tipuri de CPU si GPU si rezultatele lor in benchmark sunt mici in comparatie cu potentialul placilor respective.

De aceea am ales sa dezvolt o biblioteca de Computer Vision in C numita SystemaCV care are la baza driverele standard prezente in kernelul de linux si driverele specializate pentru utilizarea cipului VideoCore IV/VI care se ocupa cu procesarea grafica (in cazul nostru pentru operatii matriciale).

2)Arhitectura Software



2.1) InputOutput.c

Se ocupa cu citirea de intrarilor de la :
-camera picamera sau webcam prin usb sau connectat la orice pini periferici de pe placa
-imagine png,jpg sau jpeg
-citirea unui video de tip mp4
si scrierea iesirilor prelucrate in in format jpg, jpeg , png in cazul imaginilor sau mp4 in cazul unui video

2.1.1. Functiile Apartenente

void **ReadWebcam**(char*adresa_port_camera,struct Conditii_initiale*ci,struct image *destinatie);

Cu aceasta functie v-om citi valorile citite de la un webcam dupa conditiile initale stabilite de structura Conditii_initiale

void **ReadPicam**(struct Conditii_initiale*ci,struct image *destinatie);

Cu aceasta functie v-om citi valorile citite de la o camera picamera dupa conditiile initale stabilite de structura Conditii_initiale

void **ReadImage**(char*adresa_imaginii,struct image *destinatie)

Cu aceasta functie citesc imaginea si o incarc in variabila buffer din structura Image

void **VideoRead**(char*adresa_inregistrarii_mp4,struct image *destinatie)

2.2)ImageProcessing.c si ImageProcessing.h

Contine functiile care stau la baza procesarii de imagine. Acestea se pot impartii in urmatoarele categorii

a)Functii de conversie:

void **convertYUV_to_RGB**(struct Image*sursa,struct image *destinatie)

Aceasta functie are rolul de a citi imaginiea sursa de tip YUV si de a o transforma in imagine RGB.

void **convertYUV_to_BGR**(struct Image*sursa,struct image destinatie)

Aceasta functie are rolul de a citi imaginiea sursa de tip YUV si de a o transforma in imagine BGR.

void **convertYUV_to_GRAY**(struct Image*sursa,struct Image*destinatie)

Aceasta functie are rolul de a citi imaginiea sursa de tip YUV si de a o transforma in imagine GRAY cu un singur canal.

void **convertYUV_to_HSV**(struct Image*sursa,struct Image*destinatie)

Aceasta functie are rolul de a citi imaginiea sursa de tip YUV si de a o transforma in imagine HSV.

void **convertYUV_to_YcrCb**(struct Image*sursa,struct Image*destinatie)

Aceasta functie are rolul de a citi imaginiea sursa de tip YUV si de a o transforma in imagine YcrCb.

void **convertRGB_to_GRAY**(struct Image*sursa,struct image*destinatie)

Aceasta functie are rolul de a citi imaginiea sursa de tip RGB si de a o transforma in imagine de tip GRAY cu un singur canal.

b)functii de transformare al mastii:

void **Dilate**(struct Image*sursa,struct*image destinatie)
void **Erode**(struct Image*sursa,struct*image destinatie)
void **Open**(struct Image*sursa,struct*image destinatie)
void **Close**(struct Image*sursa,struct*image destinatie)

c)functii de blurare

void **GausianBlur**(struct Image*sursa,struct*image destinatie)
void **MedianBlur**(struct Image*sursa,struct*image destinatie)
void **CustomBlur**(struct Image*sursa,struct image *filtru_custom ,struct image *destinatie)

d)functii de calcul gradient

void **Sobel**(struct Image*sursa,struct*image destinatie)

```
        void Roberts(struct Imagine*sursa,struct *imagine destinatie)
        void Canny(struct Imagine *sursa,struct *imagine destinatie)
        void convolve( struct Imagine *sursa,struct Imagine *masca,int paddingH,int paddingV,
struct *imagine destinatie)
        void calculHistograma(struct Imagine *masca,int paddingH,int paddingV, int **destinatie)
```

e)functii de prag

```
        void Treshold(struct Imagine *sursa,struct Imagine *destinate, int valoare de prag)
        void TresholdInverted(struct Imagine *sursa,struct Imagine *destinate, int valoare de prag)
```

2.3)Types.c si Types.h

Contine structurile de date necesare pentru operatiile de intalnite in computer vision:

```
struct Buffer{void *data;
                MMAL_BUFFER_T *data_GPU;
                long int length;
               }
 struct Image{struct Buffer b;
               int w,h,c;
               char*format;
              }
 struct Pixel { unsigned char*valori
               int dimensiune;
              }
  struct Color{ Pixel p;
                unsigned char*culoare noua;
               }
  struct Vector{
                int *v;
                }
   struct VideoIO{ int width,heigth,channel;
                   unsigned char *data;
                   FILE *pipeIn;
                   FILE *pipeOut;

   }
    struct ImageIO{ int width,heigth,channel;
                   unsigned char *data;
                   FILE *pipeIn;
                   FILE *pipeOut;

   }
```

Contine functiile necesare pentru alocarea de memeorie si eliberarea acesteia:

```
void VideoClose(struct VideoIO*v);
void VideoRead(struct Image *i, struct VideoIO*v);
void VideoIO_init(struct VideoIO*v,int width,int height,int channel);
void VideoWrite(struct Image*i,struct VideoIO*v);
void ImageClose(struct ImageIO*v);
void ImageVideoRead(struct Image *i, struct ImageIO*v);
void ImageIO_init(struct ImageIO*v,int width,int height,int channel);
void ImageWrite(struct Image*i,struct ImageIO*v);
void init_color(struct Color*,unsigned char r,unsigned char g,unsigned char b);
void init_point(struct Point*,int,int);
```

```
void init_image(struct Image*,int,int,int);
void init_buffer(struct buffer*,long int);
void kill_buffer(struct buffer*);
```

2.4)Window.c si Window.h

Reprezinta o colectie de functii pentru afisarea si imaginii pe ecran. In prezent sunt utilizate 2 metode de afisare : metoda prin biblioteca SDL daca dorim afisare grafica la nivel inalt sau driverul gasit in locatia Linux/fb care proceseaza portul de framebuffer al ecranului gasit in locatia *dev*/fb0 si suprascrie pixeli pe ecran cu pixelii imaginii dorite in format RGBA

Contine urmatoarele functii

```
void SDLWindow_init(char*nume_fereastra, struct Image *sursa imagine)
void SDLWindow_render()
void FrameBuffer_init(fb*fbc)
void FrameBuffer_render()
```

2.5)Draw.c si Draw.h

Reprezinta cumulul de functii care se ocupa cu plotarea pe imagine:

```
void putText( struct Image *, const char *,int x,int y,struct Color TEXT,struct Color Background);
void rectangle(struct Image *pic,struct Point a,struct Point b,struct Color*c,int fill);
void line(struct Image*pic,struct Point a,struct Point b,struct Color*c);
void circle(struct Image*pic,struct Point a,struct Point b,int r,struct Color*c,int fill);
```

2.6)MMAL.c si MMAL.h

Contine functii de conversie de la tipurile de date standard la cele definite in driverul mmal.Aici avem functiile de procesare de imagine care pun in mod normal stress pe CPU si deci le mutam pe GPU.

Aceste functii de mai jos sunt destinate versiunilor de raspberry pi zero2w,3* deoarece acestea au ca si unitate de procesare grafica drept VideoCore IV.

```
Void MMAL_VC4_Sobel( struct Image*sursa,struct*image destinatie)
void MMAL_VC4_Roberts(struct Imagine*sursa,struct *imagine destinatie)
void MMAL_VC4_Canny(struct Imagine *sursa,struct *imagine destinatie)
void MMAL_VC4_convolve( struct Imagine *sursa,struct Imagine *masca,int paddingH,int paddingV, struct *imagine destinatie)
void MMAL_VC4_Dilate( struct Image*sursa,struct*image destinatie)
void MMAL_VC4_Erode( struct Image*sursa,struct*image destinatie)
void MMAL_VC4_Open( struct Image*sursa,struct*image destinatie)
void MMAL_VC4_ Close( struct Image*sursa,struct*image destinatie)
void MMAL_VC4_GausianBlur( struct Image*sursa,struct*image destinatie)
void MMAL_VC4_MedianBlur( struct Image*sursa,struct*image destinatie)
void MMAL_VC4_CustomBlur( struct Image*sursa,struct image *filtru_custom ,struct image *destinatie)
```

Raspberry pi 4 are ca si unitate de procesare grafica drept VideoCore VI deci urmatoarele functii sunt aferente modificarii hardware si al driverului MMAL

```
void MMAL_VC6_Sobel( struct Image*sursa,struct*image destinatie)
void MMAL_VC6_Roberts(struct Imagine*sursa,struct *imagine destinatie)
void MMAL_VC6_Canny(struct Imagine *sursa,struct *imagine destinatie)
void MMAL_VC6_convolve( struct Imagine *sursa,struct Imagine *masca,int paddingH,int paddingV, struct *imagine destinatie)
void MMAL_VC6_Dilate( struct Image*sursa,struct*image destinatie)
void MMAL_VC6_Erode( struct Image*sursa,struct*image destinatie)
```

MMAL_VC6_Open(struct Image*sursa,struct*image destinatie)
MMAL_VC6_ Close(struct Image*sursa,struct*image destinatie)
MMAL_VC6_GausianBlur(struct Image*sursa,struct*image destinatie)
MMAL_VC6_MedianBlur(struct Image*sursa,struct*image destinatie)
MMAL_VC6_CustomBlur(struct Image*sursa,struct image *filtru_custom ,struct image *destinatie)

3) Rezultate obtinute:

Am executat un algorithm de calcul al gradientului de tip canny pe video si am afisat pe ecran dupa in rezultatul obtinut. Aici se afla un tabel cu testele generate:

Raspberry	fps	Limbaj de programare	SDK	Stress CPU %	Memorie ram consumata	System de operare	Driver de afisare grafica al imaginii
Pi 4	30	Python3.5x	Opencv	~30%	20MB	Raspbian BullsEye 64 bits	OpenGLES
Pi 3B+	21	Python3.5x	Opencv	~45%	34MB	Raspbian Buster 32biti	OpenGLES
Pi zero2w	10	Python3.5x	Opencv	~55%	34MB	Raspbian Buster 32biti	OpenGLES
Pi 4	120	C99	Systema CV	0-1%	3.2MB	Raspbian Buster 32biti	SDL2.0
Pi 3B+	90	C99	Systema CV	0-1%	3.2MB	Raspbian Buster 32biti	SDL.2.0
Pi zero2w	50	C99	Systema CV	0-1%	3.2MB	Raspbian Buster 32biti	SDL.2.0
pi4	130	C99	Systema CV	0-1%	1.9MB	Raspbian Buster 32biti	Linux/framebuffer
Pi 3B+	94	C99	Systema CV	0.1%	1.9MB	Raspbian Buster 32biti	Linux/framebuffer
Pi zero2w	54	c99	Systema CV	0-1%	1.9MB	Raspbian Buster 32biti	Linux/framebuffer

Din aceste teste reiese ca opencv chiar daca este cross-platform si de asemenea foloseste la baza lui drivere cross-platform , nu realizeaza in timp util operatiile matriciale. Stresul pe CPU realizat este dat de conversia de date care este realizata foarte frecvent la functiile de opencv.