RAPPORT SEMAINE 09 JANVIER

PARTIE A – Adrien

Les lignes modifiées : 192 à 300 (fonction read_pic)

842 à 847 (boucle for du main)

Explications :

-Prototype de la fonction : void read_pic(int n_image, int *tab_size, int *tab_width, int *tab_length, uint8_t *global_tab)

-La fonction read_pic permet de lire une image au format ppm. L'image à lire est définie par l'entier n_image (on viendra

lire l'image 1.ppm, 2.ppm,... en fonction de la valeur de n_image). Le format ppm contiens en particulier un header avec

le mot clé P3 ainsi que les informations sur les dimenssions de l'image (hauteur/largeur). On commence donc par vérifier le

mot clé puis on lis les dimensions que l'on disposent dans les n_image ième éléments des tableaux tab_width et tab_length.

Le n_image ième élément du tableau tab_size est quant à lui initialisé par le produit largeur*hauteur (!attention ce n'est

pas la taille totale en octets car on a le RGB! (donc x3)). On vient ensuite lire les pixels pour les placer dans le tableau

global global_tab à la bonne position.

->Le début de la n_image ième image dans le tableau est en (n_image-1)*DISPLAY_IMAGE_SIZE * 3

->!Attention, étant donné le code de la partie B, on en déduit que n_image va de 1 à 14 et non pas 0 à 13 d'ou le -1 pour les

indices des tableaux.

->Les fonctions de lectures des fichiers ne sont pas les fonctions courantes, ce sont les fonctions de :

---------->FatFs-Generic FAT Filesystem Module (f_open et non pas fopen; f_read et non pas fread; etc...)

-Pour le main rien de particulier, on lance 14 fois la fonction read_pic pour initialiser le tableau global_tab avec les 14

images (ainsi que les tableaux tab_size, tab_width, tab_length).

PARTIE B – Danielle et Adrien

Les lignes à modifier : 305 (convert_to_greyscale)

654 (img_to_tensor)

642 (normalizing_tensor)

608 (my_resizing)

876 (boucle for du main)

Autres lignes à modifier : -la fonction normalizing (ligne 632) n'est à priori pas utiliser (on utilise à la place les fonctions img_to_tensor et normalizing_tensor)

-décomenter #include <math.h> en ligne 7 (la fonction de normalisation utilise la racine carrée du module

math, cependant, ce n'est probablement pas la bonne méthode, il faut surement compléter les fonctions __ieee754_sqrtf ou __ieee754_sqrt pour implémenter la racine, à voir...)

Explications :

-convert_to_greyscale permet de générer une image en nuance de gris à partir d'une image RGB en effectuant une moyenne pondérée des composantes RGB

-img_to_tensor permet de transformer une image (enchainement des pixels RGBRGB...) en un tenseur id l'enchainement des pixels est maintenant RRR...GGG...BBB...

-normalizing_tensor permet de normaliser le tenseur (voir la formule de mancini pour le CNN)

-my_resizing permet d'effectuer une diminution de la taille d'une image (l'algo choisis est l'INTER_AREA de open_cv). Pour cela on commence par calculer les facteurs de rétrécissements selon les deux axes, si les multiples ne sont pas entiers, on vien couper l'image initiale puis on moyenne l'image initiales sur des zones rectangulaire facteur_x * facteur_y pour obtenir les pixels de l'image rétraicie.

PARTIE C – Mehdi

Les lignes modifiées : 705 à 805 (fonctions display et on_screen)

892 à 914 (boucle while du main)

Explications :

1-Prototype de la fonction display : void display(int img_in_number, filter_type filter_nb, uint8_t previous_imageSel, uint8_t previous_filterSel):

2-Prototype de la fonction on_screen : void on_screen(int mode, int class, uint8_t *img)

PARTIE D – Igor

Les lignes modifiées : 729 (case EDGE_DETECTOR de on_screen)

867 à 870 (boucle for du main pour le filtrage)

Explications :

-Prototype de la fonction : convolution_filter(uint8_t image[CONV_READ_SIZE_PGM], KERNEL_CONV_FIXED_FORMAT kernel[3 * 3 * 1], BIAISES_CONV_FIXED_FORMAT biaises[1], uint8_t output[CONV_CONV_TOTAL_SIZE])

->image[CONV_READ_SIZE_PGM] étant le tableau dans lequel on prend les images d'entrée

sur lesquelles on applique la convolution avec CONV_READ_SIZE_PGM les dimensions de l'image (largeur et hauteur)

->output[CONV_CONV_TOTAL_SIZE] le tableau où on range les résultats de la convolution

-La fonction convolution_filter permet de faire la convolution sur les images déjà redimensionnées avec les valeurs

du kernel et du bias qui correspondent à la détection de contours et stocke les valeurs des pixels résultantes dans un tableau global en sortie.

-Dans le main, notre boucle for nous permet d'appliquer NB_IMAGES_TO_BE_READ fois la convolution sur les images

de TAB_GS et de stocker chacune d'elle dans TAB_GS_FILTERED

- on peut afficher l'image avec la détection de contours en utilisant la fonction on_screen(EDGE_DETECTOR, 11, diplay_ptr_filtered)

déjà définie en mettant les bons paramètres

-> EDGE_DETECTOR = mode

-> 11 = la classe ou l'indice 11 correspondant à l'overlay du edge detector

-> diplay_ptr_filtered = le pointeur sur l'image sélectionnée

PARTIE F – Nicolas – Thibaut

Nicolas

Modification des lignes :

323 - 331

405 - 452

Première partie :

Comme le TP, on initialise simplement tous les registres demandé à 0 avec la fonction write_csr

Deuxième partie : external_interrupt

- claim est la valeur de l'interruption de priorité la plus élevé, on va donc lui donner la valeur contenu dans le PLIC à l'adresse contenant l'interruption à la priorité la plus élevé donc PLIC_HART0_CLAIM_COMPLETE_ADDR

- pour chaque valeur de claim càd :

claim == 1 --> décrementation du sélecteur d'image, si on est au minimum du nombre d'image (càd si l'incrémentation va un cran en dessous du minimum), on le repasse au maximum

claim == 2 --> incrémentation du sélecteur d'image, si on est au maximum du nombre d'image (càd si l'incrémentation va un cran au dessus du maximum), on le repasse au minimum

claim ==3 et claim == 4 même chose pour le sélecteur de filtre en admettant qu'il y en 3 différents

- une fois tous cela fait on redonne la valeur de claim au registre correspondant pour terminer l'interruption

Thibaut

Modification des lignes 374 - 393 (enable_plic_interrupts)

Modification des lignes 893 - 906 (Boucle while(1))

Ajout des différentes priorité à 1

Mise en place du threshold à 0 pour que les priorité puissent fonctionner

Mise en place des ENABLE_INTERRUPT des différents boutons à l'aide de leurs masques

Mise a 1 les bits MIP_MEIP et MSTATUS_MIE des registres mie et mstatus

Au niveau du main, refait ce que Mehdi avait déjà fait donc pas de grandes modifications en réalité

ID des boutons :

- W : 1
- E : 2
- S : 3
- N : 4