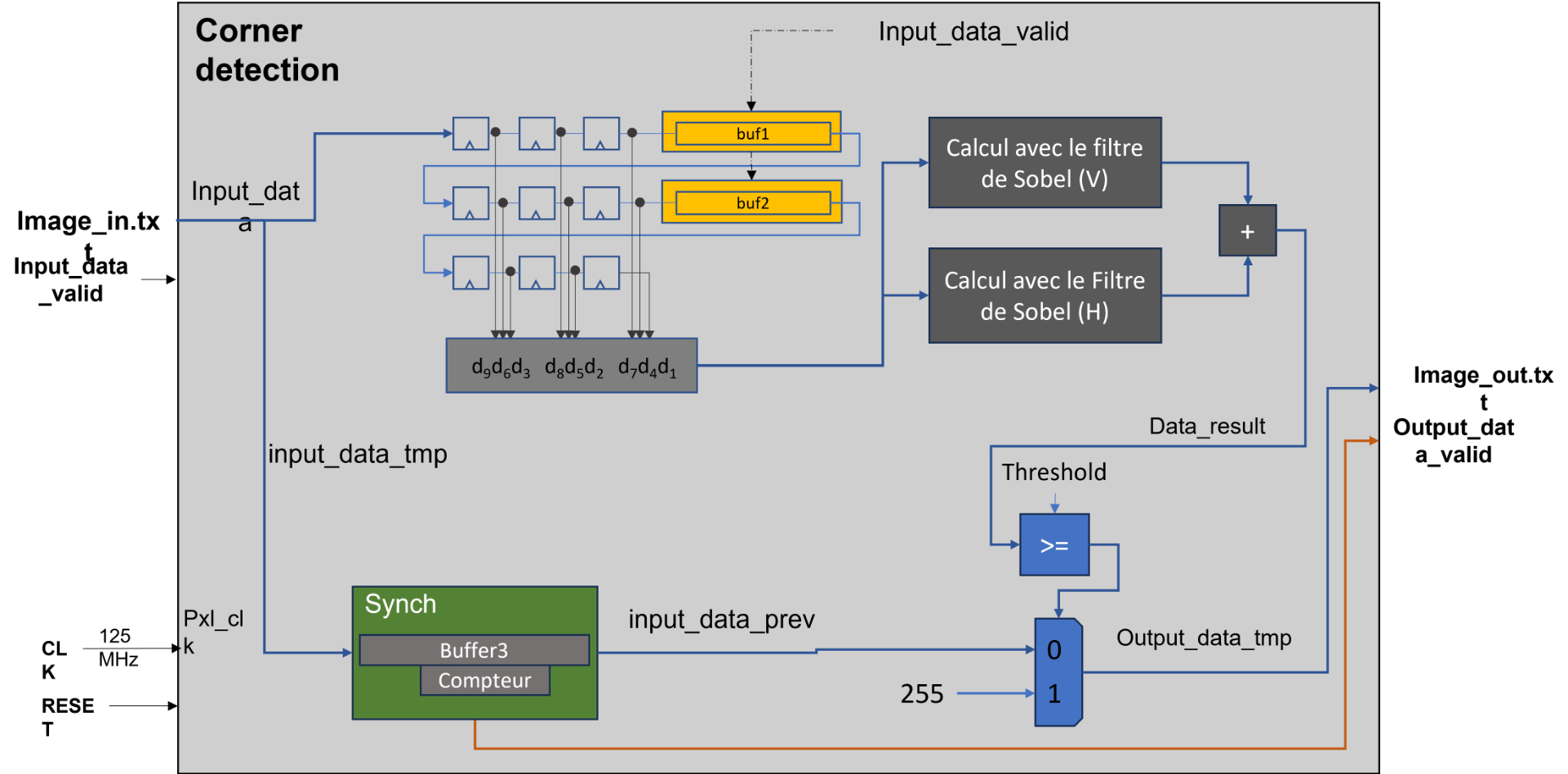
1. **Présentation du système**
2. **Fenêtre coulissante**

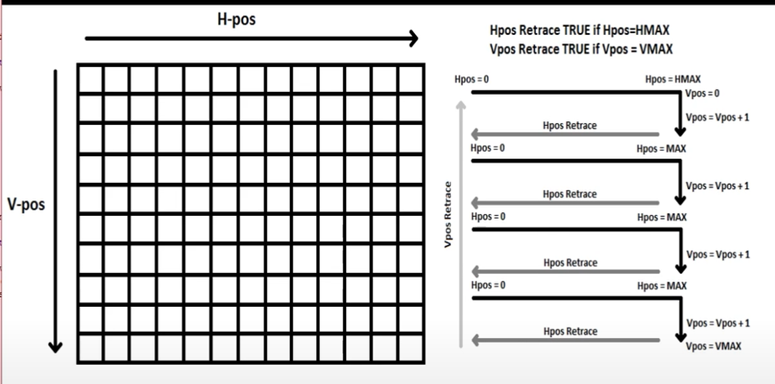
Afin de mettre en œuvre notre système, nous avons exploité le précédent projet «interface vidéo VGA», pour utiliser l’architecture de la fenêtre glissante développée lors de ce projet.

L’architecture de détection de points d’intérêts que nous avons proposée est représenté sur la figure suivante :



En entrée de notre système, nous aurons une image donnée sous format .txt. La lecture de cette image se réalise en lisant les pixel un par un. Chaque pixel est composé de 3 couleurs (rouge, vert et bleu). Chacune de ces couleurs est codée sur 8 bits, permettant une gamme de valeurs allant de 255 (représentant l'intensité maximale) à 0 (représentant l'intensité minimale).

Le processus de lecture des données de l’image commence par le premier pixel situé en haut à gauche de l’image et se termine avec le dernier pixel en bas à droite. Ce processus suit un balayage ligne par ligne, comme illustré dans le schéma ci-dessous :



Avec H-pos représentant la position horizontale du pixel et V-pos la position verticale.

Pour traiter ces données, nous avons utilisé des registres et des buffers pour stocker les pixels de deux premières lignes de l'image avant la convolution (d9, d8, ..., d1), facilitant le chargement d'un voisinage 3x3 lors de la convolution. Cette approche nous permet d'économiser du temps en évitant d'extraire les pixels à partir d'une mémoire externe à chaque étape de la convolution. Le stockage et le décalage des données dans les buffers commencent uniquement lorsque le signal "input\_data\_valide" est égal à 1, c'est-à-dire lorsque nous recevons les pixels d’entrée.

Cette stratégie nous permet de traiter l'image en faisant passer les pixels à travers une fenêtre plutôt que de déplacer le noyau sur l'image. Elle offre une solution efficace pour la réalisation de l'opération de convolution 2D.

Le bloc « Synch » permet de resynchroniser les pixels de l’image d’entrée avec les calculs faits lors de la convolution « data\_result », le compteur nous indiquant le départ de cette synchronisation.

1. **Description de la détection de contour/point d’intérêt**

Le module "Corner detection" est divisé en 3 parties distinctes. La première partie concerne la détection des contours en utilisant les filtres de Sobel.

Pour rappel, dans le cas du filtrage de Sobel, nous utilisons les matrices de convolution de taille 3×3 représentées ci-dessous (H pour les transition horizontale, V pour les transitions verticale) :

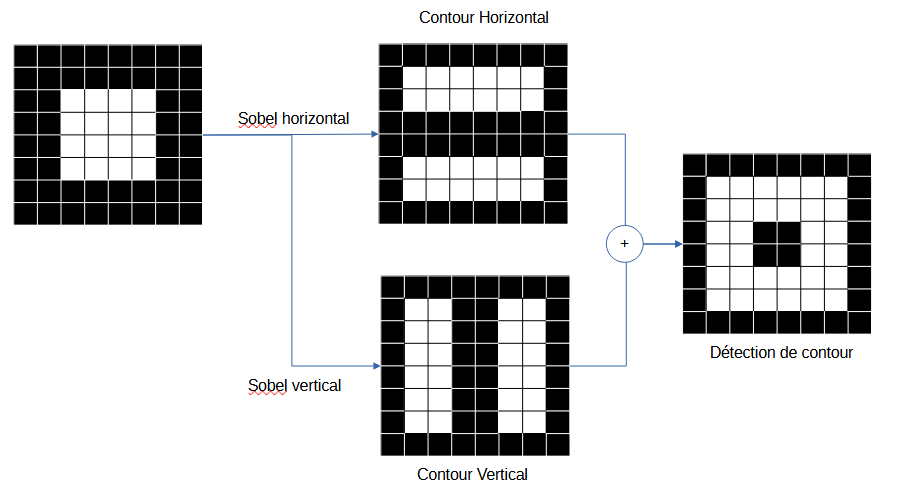
Une image contenant noir, obscurité

Description générée automatiquementUne image contenant noir, obscurité

Description générée automatiquement

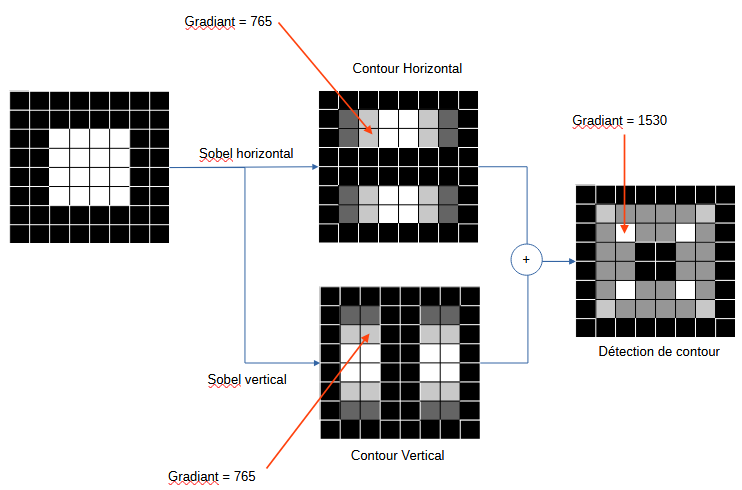
1. *Filtre de sobel, détection de contour*

Les filtres de Sobel (Vertical et horizontal) permettent de connaître respectivement les contours horizontaux et verticaux de l’image, en calculant les gradient d’intensité de chaque pixel suivant 2 axes x,y. Plus le gradient est élevée, plus la transition est forte En sommant les 2, nous récupérons ainsi les contours de l’image. Selon le seuil de la valeur retournée, nous pouvons faire ressortir plutôt les point d’intérêts, à savoir les angles, là où les transition horizontales et verticales se croisent.



Si nous prenons l’exemple d’une image noire composée d’un carreau blanc en son centre, comme présentée ci-dessus. Pour les transitions verticales, et horizontales, les variations de seuils auront les valeurs suivantes : 255 et 765 au niveau du début et de la fin, et 1020 pour le milieu d’un carreau. De même pour la transition verticales.

Si nous additions les deux signaux, nous trouverons au niveau des contours des valeurs qui varient de 510 à 1530. Cette dernière valeur correspond donc au point d’intérêt de l’image, soit là où les bordures se rejoignent.



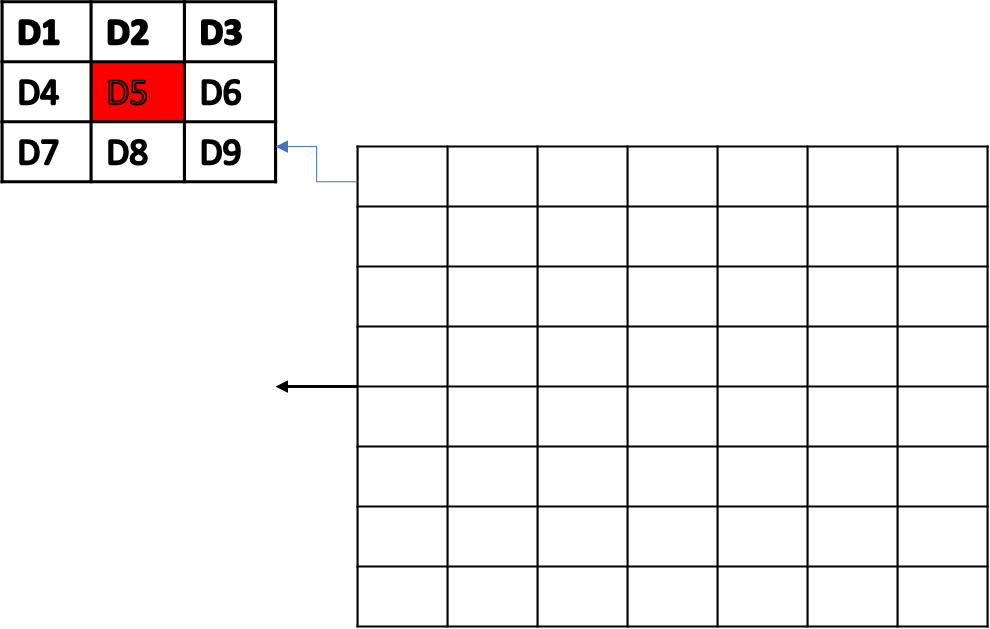
1. *recalage du contour*

La deuxième partie corrige le décalage des pixels de l’image d’entrée par rapport aux pixels des contours et de l’image de sortie, qui apparaît après l’application de la fenêtre glissante.

En effet, le décalage se produit car le résultat de la convolution est assigné au pixel central de la fenêtre de convolution (D5). Cependant, lors de l'utilisation de la technique de convolution à fenêtre glissante, les pixels de l'image d'entrée traversent le dernier pixel du noyau de convolution (D9). Cela crée un décalage entre le moment où les pixels sont envoyés en entrée et le moment où ils sont calculés et envoyés en sortie.

En d'autres termes, lors de l'application de la fenêtre glissante, le premier pixel d'entrée est traité après avoir traversé tous les pixels du noyau de convolution. Cela entraîne un décalage par rapport au moment où le pixel apparaît dans l'image de sortie. Par conséquent, une synchronisation est nécessaire pour aligner correctement les pixels et corriger le décalage.

Les pixels voisins (D1 à D9) seront utilisés pour calculer la valeur du pixel central (voir représentation ci-dessous).



Après avoir traité les pixels d’une ligne de l’images d’entrée, les informations seront stockées dans un Buffer. Les données seront envoyées vers la sortie à partir de la deuxième ligne traitée (une ligne de l’image de sortie a été ignorée). Ainsi un signal de validation « output\_data\_valid» sera activé, ce qui indique que les pixels de l’image de sortie sont en cours d’envoie et que nous sommes autorisés à écrire dans le fichiers «Output.data.txt».

1. *Seuil de détection*

Enfin, la troisième partie concerne la détection des points d'intérêt et/ou des contours de l’image d’entré et leur ajout à cette dernière. Dans cet étape, on compare le résultat de la somme des filtres de Sobel avec un seuil. Et suivant ce résultat de comparaison, nous aurons soit les contours, soit les points d’intérêts ou les deux.