Juan Ricardo DAHL Sandrine ZHENG

Rapport Projet HPC

Sommaire

Introduction **1**

Avant-Propos **1**

Détection de mouvement **2**

Frame-Difference 2

Définition 2

Performances & Optimisation 2

Qualité des résultats 3

Sigma-Delta 5

Définition 5

Performances & Optimisation 5

Qualité des résultats 6

Morphologie Mathématique **7**

Définition 7

Performances & Optimisation 7

Qualité des résultats 8

Test Unitaire **9**

Comparaison **10**

Conclusion **11**

I. Introduction

Dans le cadre du cours de Calcul Haute Performances, nous avons eu un projet qui avait pour but d'effectuer le plus rapidement possible de la détection de mouvement à partir d'un flux d'images. Nous présenterons les algorithmes utilisés pour la détection de mouvement: Frame-Difference, Sigma-Delta et les opérateurs de morphologie mathématique pour améliorer le traitement de l'image. De même, nous montrerons les optimisations apportées accompagnées de tests.

II. Avant-Propos

Dans ce projet, nous avons calculé les performances des algorithmes grâce à la macro "Chrono". Nous avons calculé la précision et l'accuracy des différents algorithmes pour différents paramètres afin de déterminer l'algorithme la plus efficace au niveau du temps et de la qualité de la détection de mouvement avec un taux d'erreur faible.

La précision est la fraction des instances pertinentes parmi les instances extraites. On peut voir sur la figure 1, comment est calculé la précision.

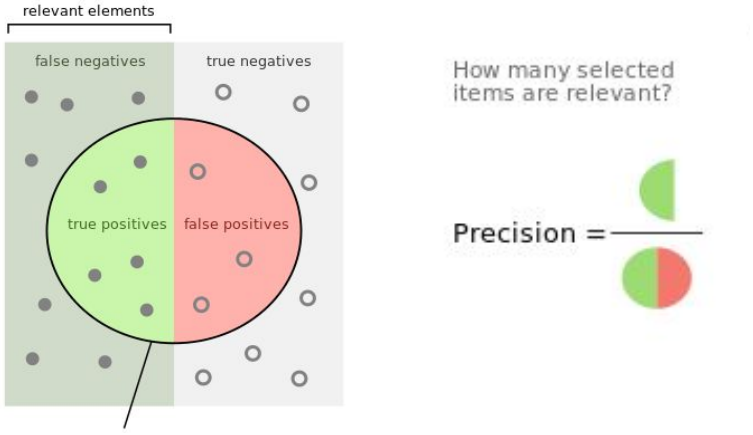

Figure 1: La précision

L'accuracy est la proportion des vrais résultats (vrais positifs et vrais négatifs) parmi le nombre total de cas examinés.

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn}$$

Avec **tp**: vrai positif (pixel en mouvement détecté en mouvement), **tn**: vrai négatif (pixel en mouvement détecté immobile), **fp**: faux positif (pixel du fond détecté en mouvement), **fn**: faux négatif (pixel du fond détecté immobile)

III. Détection de mouvement

A. Frame-Difference

1. Définition

Cet algorithme de détection de mouvement qui à partir d'une séquence d'images en niveaux de gris fourni une séquence d'images binaires indiquant quels sont les pixels en mouvement et immobiles. Pour un pixel donné, si la différence entre les valeurs (de luminosité) de ce pixel pour deux images successives est supérieure à un seuil fixe , alors ce pixel est en mouvement. Les pixels des images sont caractérisés par leur valeur de niveau de gris dans une matrice.

foreach *pixel* x **do** // step #1 : O_t computation
  $O_t(x) = |I_t(x) - I_{t-1}(x)|$
foreach *pixel* x **do** // step #2 : O_t thresholding and $\hat{E}_t$ estimation
  if $O_t(x) < \theta$ **then** $\hat{E}_t(x) \leftarrow 0$
  else $\hat{E}_t(x) \leftarrow 1$

2. Performances & Optimisation

Pour l'algorithme de différence de trames, nous avons implémenté des threads pour augmenter les performances. Pour obtenir une meilleure analyse, nous avons implémenté des threads pour le code écrit au format scalaire, puis pour le code optimisé avec SSE2.

Nous avons choisi les threads car le problème convient parfaitement pour tirer partie des avantages de la parallélisation.
Le tableau suivant montre les résultats obtenus:

Frame Difference	**Version Scalaire**		
	Précision (%)	**Accuracy (%)**	**Cycles par pixel**
Séquentiel	53.9474	97.2203	2544.07
Thread	52.3284	97.2135	2357.01
Thread + SSE2	53.9474	97.2203	568.38

Figure 2: Tableau comparant les performances et la qualité de la détection de mouvement avec des threads ayant 60 pour seuil

Tout d'abord, nous pouvons remarquer que les versions «Séquentiel» et «Thread», les cycles sont pratiquement identiques. En effet, avec les threads, nous allons utiliser

toutes les ressources (cœurs) de notre pc, mais essentiellement, le code où les instructions d'assemblages sont similaires. Même dans la version de thread, nous aurons plus d'instructions pour créer et scinder le code pour chaque thread.
Cependant, nous allons voir la meilleure optimisation de cette application, en analysant le temps. C'est pourquoi nous voyons de résultats similaires à ceux obtenus sans les appliquer.
Avec SIMD, l'algorithme est 4,47 fois plus rapide. Cela est faisable pour la totalité de la chaîne de traitement. On a ainsi un parallélisme de 16 en 128 bits.

3. Qualité des résultats

Afin de quantifier la qualité de la détection de mouvement, autrement que visuellement, nous avons calculé une matrice ROC qui consiste à comparer une image résultant d'un algorithme de détection à une image vérité terrain (image binaire ici, construite manuellement).

A partir de la matrice ROC, nous avons pu calculer la précision et l'accuracy pour différents paramètres de seuil. Ci-dessous la matrice ROC qui contient le pourcentage de vrai positif et de vrai négatif détectés le plus élevé possible.

ROC average =

0.001197% 0.026775%
0.001091% 0.970938%

Pour les différentes valeurs de seuil, nous avons trouvé les valeurs suivantes:

Seuil	Version Scalaire			Version SIMD		
	Précision (%)	Accuracy (%)	Cycles par pixel	Précision (%)	Accuracy (%)	Cycles par pixel
10	14.3178	91.2847	2112.1	14.2636	91.2532	66.07
20	14.3178	91.2847	3699.23	32.7759	96.6143	62.51
30	32.6545	96.6080	5193.15	45.7475	97.1414	62.54
40	45.3168	97.1345	6663.13	52.5539	97.2245	61.0
50	51.7175	97.2176	8186.39	53.94	97.2260	61.86
60	52.7108	97.2191	9664.67	53.9474	97.2203	61.46
70	52.3284	97.2135	11216.17	53.2654	97.2139	61.28
80	51.2021	97.2071	12667.19	52.3953	97.2091	63.56

Figure 3: Tableau analysant la qualité des résultat et la performance de l'algorithme Frame Difference pour différentes valeurs de seuil

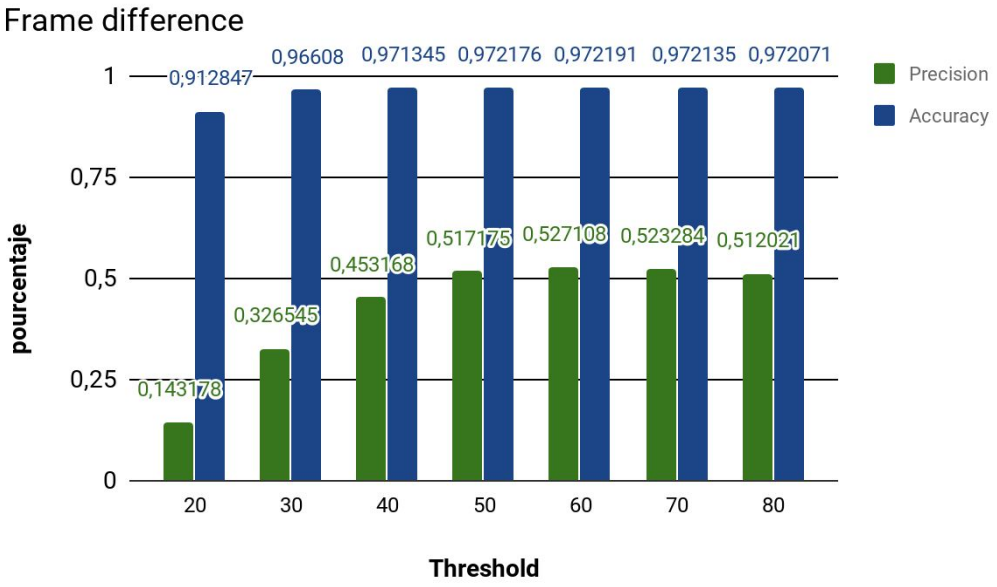

Figure 4: Représentation graphique de de la précision et de l'accuracy en fonction du seuil

Après analyse de la figure 3 et 4, nous pouvons évidemment remarquer que le meilleur choix de seuil pour une meilleur qualité calcul de détection de mouvement est de choisir un seuil de 60 car sa précision accompagnée de son accuracy sont les valeurs les plus élevés. Si nous regardons niveau temps, on en déduit que plus nous augmentons le seuil, plus le nombre de cycles par pixel augmente. Tandis que pour la version SIMD, le nombres de cycles pour différents seuils ne varient pas énormément. Par la suite, nous utiliserons le seuil égale à 60 pour faire des meilleurs analyse.

B. Sigma-Delta

1. Définition

L'algorithme Sigma-Delta s'agit d'une détection de mouvement fondée sur une estimation des statistiques du fond statique. On fait l'hypothèse que le niveau de bruit (écart-type) peut varier en tout point et on modélise les niveaux de gris des pixels par une moyenne M et un écart-type V (variance).

foreach *pixel* x **do** // step #1 : M_t estimation
  if $M_{t-1}(x) < I_t(x)$ **then** $M_t(x) \leftarrow M_{t-1}(x) + 1$
  if $M_{t-1}(x) > I_t(x)$ **then** $M_t(x) \leftarrow M_{t-1}(x) - 1$
  otherwise do $M_t(x) \leftarrow M_{t-1}(x)$

[step #2 : difference computation]
foreach *pixel* x **do** // step #2 : O_t computation
  $O_t(x) = |M_t(x) - I_t(x)|$

foreach *pixel* x **do** // step #3 : V_t update and clamping
  if $V_{t-1}(x) < N \times O_t(x)$ **then** $V_t(x) \leftarrow V_{t-1}(x) + 1$
  if $V_{t-1}(x) > N \times O_t(x)$ **then** $V_t(x) \leftarrow V_{t-1}(x) - 1$
  otherwise do $V_t(x) \leftarrow V_{t-1}(x)$
  $V_t(x) \leftarrow max(min(V_t(x), V_{max}), V_{min})$ // clamp to $[V_{min}, V_{max}]$

foreach *pixel* x **do** // step #4 : $\hat{E}_t$ estimation
  if $O_t(x) < V_t(x)$ **then** $\hat{E}_t(x) \leftarrow 0$
  else $\hat{E}_t(x) \leftarrow 1$

2. Performances & Optimisation

Nous allons comparer les performances de l'algorithme Sigma Delta en mode non optimisé et optimisé:

Sigma Delta	Séquentiel	SSE2	Amélioration
Cycles par pixel	7853.62	575.20	13,65

Figure 5: Tableau représentant les performances de l'algorithme Sigma Delta

Similaire à l'algorithme de Frame-Difference, avec SIMD, l'algorithme est beaucoup plus rapide. Ceci est réalisable pour toute la chaîne de traitement. Nous avons un parallélisme de 16 en 128 bits.
De cette comparaison, on peut dire que la limite du parallélisme est de 16 mais que le code n'est pas totalement parallélisable.
C'est pourquoi nous avons une amélioration ($\frac{nombres\ de\ cycles\ par\ pixel\ du\ code\ séquentiel}{nombres\ de\ cyles\ par\ pixel\ du\ code\ en\ SSE2}$) de moins de 16.
Le code optimisé est ainsi donc 13,65 fois plus rapide.

3. Qualité des résultats

Nous procédons de la même manière pour calculer la précision et l'accuracy que le précédent algorithme mais cette fois-ci suivant différentes valeurs de VMIN et VMAX.

VMIN	VMAX	Précision	Accuracy
20	255	0.720708 %	0.980238 %
30	255	0.795019 %	0.980880 %
40	255	0.821407 %	0.980294 %

Figure 6: Tableau représentant la précision et l'accuracy de l'algorithme Sigma Delta pour des valeurs différentes de VMIN et VMAX

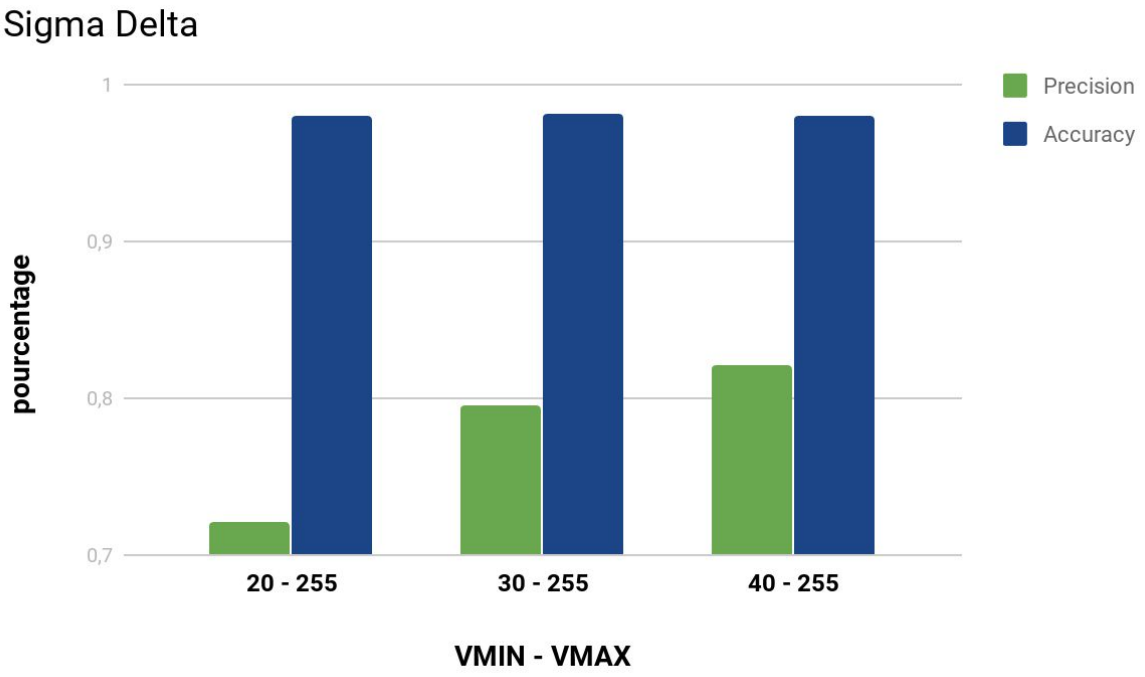

Figure 7: Représentation graphique de la figure 6

Nous pouvons observer qu'en augmentant la valeur de VMIN sans changer la valeur de VMAX, la précision du calcul de détection de mouvement devient plus importante sachant que l'accuracy reste constante.

IV. Morphologie Mathématique

1. Définition

La morphologie mathématique contient différents opérateurs qui permettent d'améliorer la chaîne de traitement d'image tels que:

- La dilatation
- L'érosion
- La fermeture
- L'ouverture

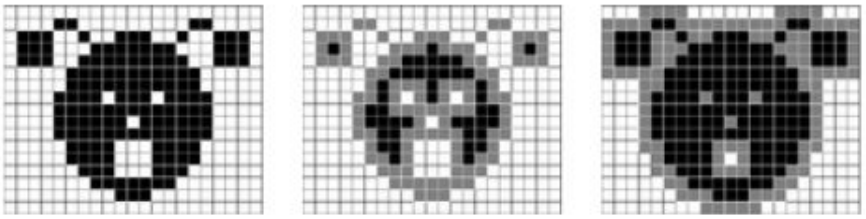

Figure 8: A gauche, image binaire initiale. Au centre, image érodée par un carré 3×3: les pixels gris sont supprimés de l'ensemble résultant. A droite, image dilatée par un carré 3×3: les pixels gris sont ajoutés à l'ensemble résultant.

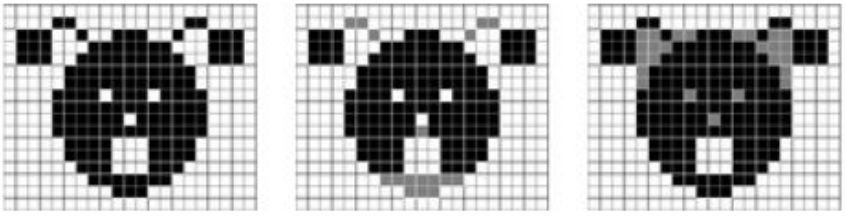

Figure 9: A gauche, image binaire initiale. Au centre, image ouverte par un carré 3×3: les pixels gris sont supprimés de l'ensemble résultant. A droite, image fermée par un carré 3×3: les pixels gris sont ajoutés à l'ensemble résultant.

2. Performances & Optimisation

Analysons les performances de l'algorithme en code séquentiel et en code optimisé avec différentes combinaisons possible avec les algorithmes vu précédemment:

Sigma Delta	Séquentiel Cycles	SSE2 Cycles	Amélioration
FD + Ouverture 3x3	2571.17	479.31	5.36
FD + Fermeture 3x3	2568.66	571.73	4.49
SD + Ouverture 3x3	6765.53	974.14	6.94
SD + Fermeture 3x3	6959.53	958.48	7.26

Figure 10: Tableau représentant les performances de la morphologie mathématique du code séquentiel et SSE2

En analysant ces différents résultats, nous pouvons voir que la combinaison la plus performante niveau temps de calcul est FD + Fermeture et ce dernier est 1.6 fois plus rapide que SD + Fermeture

3. Qualité des résultats

Analysons les performances de l'algorithme morphologie mathématique avec la version scalaire:

Algorithme	Version Scalaire		
	Précision (%)	**Accuracy (%)**	**Cycles par pixel**
FD + Ouverture 3x3	87.0602	98.5562	2571.17
FD + Fermeture 3x3	87.1148	98.5573	2568.66
SD + Ouverture 3x3	85.5398	97.4899	6765.53
SD + Fermeture 3x3	85.5398	97.4899	6959.53
FD + Ouverture 5x5	87.0602	98.5562	2563.68
FD + Fermeture 5x5	87.0602	98.5562	2567.2
SD + Ouverture 5x5	85.5398	97.4899	6891.30
SD + Fermeture 5x5	85.5398	97.4899	6765.30

Figure 11: Tableau représentant la qualité du résultat de la morphologie mathématique

Après analyse du tableau, nous observons que la précision du calcul de l'algorithme est légèrement plus élevé si nous prenons l'algorithme Frame Difference comme base. De même, cette combinaison est 2.7 fois meilleure que celle ayant pour base Sigma Delta.

V. Test Unitaire

Dans cette partie, nous avons effectué des tests unitaires en SIMD pour les algorithmes Frame-Difference et Sigma-Delta pour pouvoir vérifier le bon fonctionnement de nos deux algorithmes.
Analysons tout d'abord le test unitaire de l'algorithme Frame-Difference:

```
---------------------------------------
-- test_unitaire_FrameDifference_SSE2 --
---------------------------------------
Ot Test          20  20  21  19  21  19  224  255  0  0  1  125  75  34  0  1
Ot result        20  20  21  19  21  19  224  255  0  0  1  125  75  34  0  1

Et Test          255  255  255  0  255  0  255  255  0  0  0  255  255  255  0  0
Et Result        255  255  255  0  255  0  255  255  0  0  0  255  255  255  0  0
```

Figure 12: *Résultat test unitaire de Frame-Difference*

Pour deux images sources en niveau de gris données et l'image de différence calculée **Ot Test** au préalable pour la vérification, nous obtenons bien la bonne image de différence en niveau de gris en sortie représentée par **Ot result**. De même, pour l'image d'étiquettes binaires **Et Test** calculées au préalable, on obtient un bon résultat en sortie **Et Result**.

Analysons maintenant le test unitaire de l'algorithme Sigma-Delta:

```
**********************************
** test_unitaire_SigmaDelta_SSE2 **
**********************************
*********** Step 1 ***********
It       255  0  1  0  162  10  158  23  200  255  254  70  56  30  7  98
Mt0      255  0  0  1  161  8  160  26  200  254  255  65  57  35  24  90

Mt       255  0  1  0  162  9  159  25  200  255  254  66  56  34  23  91
Mt attendu 255  0  1  0  162  9  159  25  200  255  254  66  56  34  23  91

*********** Step 2 ***********
It       120  130  120  130  120  130  244  255  15  223  158  35  180  254  10  2
Mt       100  150  141  149  99  111  20  0  15  223  159  160  255  220  10  1

Ot       20  20  21  19  21  19  224  255  0  0  1  125  75  34  0  1
Ot attendu 20  20  21  19  21  19  224  255  0  0  1  125  75  34  0  1

*********** Step 3 update ***********
Ot               20  20  21  19  21  19  224  255  0  0  1  125  75  34  0  1
2*Ot             40  40  42  38  42  38  255  255  0  0  2  250  150  68  0  2
NTimesOt attendu 40  40  42  38  42  38  255  255  0  0  2  250  150  68  0  2

*********** Step 3 clamping ***********
Vt_Init          80  0  20  19  18  54  98  158  210  251  255  130  101  129  201  125
Vt calculé       80  20  20  20  20  54  98  158  210  240  240  130  101  129  201  125
Vt attendu       80  20  20  20  20  54  98  158  210  240  240  130  101  129  201  125

*********** Step 4 ***********
Ot       10  108  202  64  128  60  90  255  0  255  0  187  209  80  18  166
Vt       10  107  203  64  128  50  50  0  255  255  0  177  208  88  19  166
Et       255  255  0  255  255  255  255  255  0  255  255  255  255  0  0  255
Et attendu 255  255  0  255  255  255  255  255  0  255  255  255  255  0  0  255
```

Figure 13: *Résultat test unitaire de Sigma-Delta*

Pour la première étape de l'algorithme, nous obtenons bien la bonne image de fond **Mt** (image de moyenne) avec celle calculée au préalable. Pour la deuxième étape, nous obtenons de même de bon résultat pour l'image de différence en niveau de gris **Ot**. Pour la 3 troisième étape , le calcul de N par Ot se fait avec succès. De même, pour l'image de variance **Vt**. Et enfin la dernière étape, qui retrouve bien l'image binaire **Et**.

VI. Comparaison

Maintenant, nous allons comparer les performances et la qualité du détection de mouvement des différents algorithmes en prenant le meilleur seuil qui est de 60, vu précédemment.

Algorithme	Précision (%)	Accuracy (%)	Version Scalaire (Cycles par píxel)	Version SIMD (Cycles par píxel)	Amélioration
FD	52.7108	97.2191	1942.15	114.10	17.03
SD	82.1407	98.0294	7842.83	625.83	12.54
FD + Ouverture 3x3	87.0602	98.5562	2571.17	479.31	5.36
FD + Fermeture 3x3	87.1148	98.5573	2568.66	571.73	4.49
SD + Ouverture 3x3	85.5398	97.4899	6765.53	974.14	6.94
SD + Fermeture 3x3	85.5398	97.4899	6959.53	958.48	7.26

En analysant les résultats, nous voyons que la combinaison des algorithmes FD et SD avec la morpho donne de meilleurs résultats par rapport aux images de vérité. En comparant les deux algorithmes, il est préférable de choisir FD + morpho car la qualité de la solution est similaire mais, en terme de temps, FD est beaucoup plus rapide.
Au niveau performance, par exemple, FD + fermeture est 1,67 fois plus rapide que SD + fermeture.
A notre avis, le choix le plus judicieux est de choisir la combinaison signalée en vert sur le tableau compte tenu des résultats.

VII. Conclusion

Ce projet nous a permis d'apprendre différents outils pour améliorer les performances de nos algorithmes.
D'autre part, il était utile et intéressant d'apprendre différents algorithmes de détection de mouvement avec leur qualité de solution respective.
L'utilisation de git a été un bon choix qui nous a aidé à travailler en collaboration. Cela facilitait l'intégration des différentes tâches, même s'il était nécessaire d'établir une bonne communication constante entre le binôme.
Nous avons pu voir que la précision et l'accuracy sont généralement très proches pour un même algorithme avec une implémentation différente. C'est ce qui est effectivement attendu, car l'algorithme est le même, nous changeons simplement la manière de les implémenter.
Après avoir effectué différentes comparaisons, nous avons pu voir l'amélioration apportée par les instructions SIMD qui est bien évidemment beaucoup plus performant que la version scalaire.
Enfin, il est important de noter que l'accuracy et l'analyse de précision ont été effectuées avec seulement 30 images de vérité. Nous pensons que si nous voulions avoir plus de précision dans l'analyse, ce nombre pourrait être augmenté. Cependant, avec 30 images, les résultats étaient suffisamment précis pour effectuer l'analyse présentée dans ce rapport.