




Besoins

Besoins Fonctionnels:

- Importation d'images 
- Saisie de l'utilisateur 
- Traitement des images et calcul 
- Enregistrement 


=> résultats fiables et précises

Besoins non fonctionnels:

- Validation des données 
- Stabilité de l'application 
- Performance de l'application (en cours)
- Résultat et Benchmark (en cours)

Besoins

Besoins Fonctionnels:

- Importation d'images 
- Saisie de l'utilisateur 
- Traitement des images et calcul 
- Enregistrement 

=> résultats fiables et précises

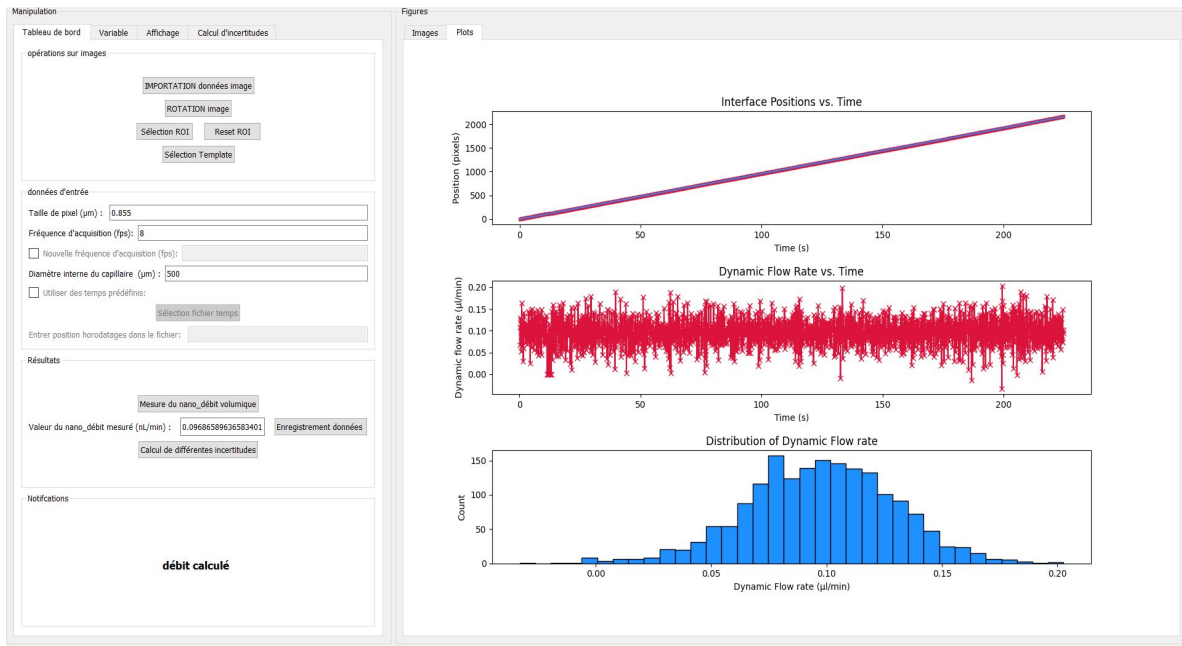
Besoins non fonctionnels:

- Validation des données 
- Stabilité de l'application 
- Performance de l'application (en cours)
- Résultat et Benchmark (en cours)

Résultats: besoins fonctionnels (calcul + output data)

données d'entrée	débit théorique	Résultat obtenu
diamètre: 500 um frame rate: 8 fps taille de pixel: 0.855 (zoom 25%)	100 nL/min = 0.1 uL/min	0.0969 uL/min
diamètre: 250 um frame rate: 2 fps taille de pixel: 0.855 (zoom 25%)	10 nL/min = 0.1 uL/min	0.0114 uL/min
diamètre: 250 um frame rate: 4 fps taille de pixel: 0.855 (zoom 25%)	20 nL/min = 0.1 uL/min	0.273 uL/min
diamètre: 500 um frame rate: 4 fps taille de pixel: 0.855 (zoom 25%)	70 nL/min = 0.1 uL/min	0.0678 uL/min
diamètre: 250 um frame rate: 2 fps taille de pixel: 0.546 (zoom 50%)	5 nL/min = 0.1 uL/min	0.000188 uL/min

Résultats: besoins fonctionnels (calcul + output data)



recap

Récapitulation	
Taille de pixel (μm) :	0.855
Diamètre interne du capillaire (μm) :	500.0
Fréquence d'acquisition (Hz) :	8.0
Vitesse d'écoulement (($\mu\text{m/s}$) :	8.222232652613195
débit volumique estimé ($\mu\text{l/min}$) :	0.09686589636583401
Process time((s) :	1099.28125
Position moyenne des pixels (pixels) :	1078.3921642297926
Nombre de positions utilisé :	1800
Nombre d'instants utilisé :	1800

Annuler Confirmer



Besoins

Besoins Fonctionnels:

- Importation d'images 
- Saisie de l'utilisateur 
- Traitement des images et calcul 
- Enregistrement 

=> résultats fiables et précises

Besoins non fonctionnels:

- Validation des données 
- Stabilité de l'application 
- Performance de l'application (en cours)
- Résultat et Benchmark (en cours)

Solution: Performance

Optimisation du programme existant:

- Optimisation des calculs

=> Solution Limitée

Optimisation en utilisant les Threads

- Traitements des plusieurs images au même temps

=> les résultats sont encourageantes

=> La solution est presque entièrement stable

=> on peut même l'optimiser encore plus :

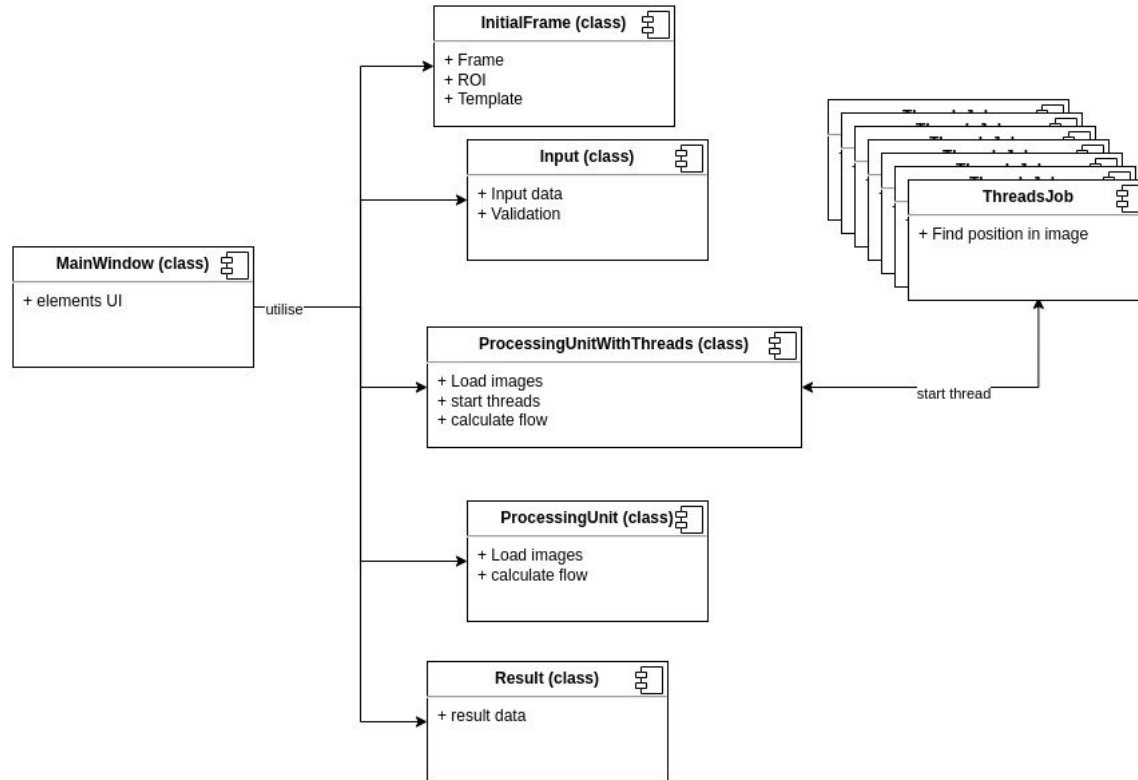
diviser le traitement de chaque image

Optimisation des images avant traitement

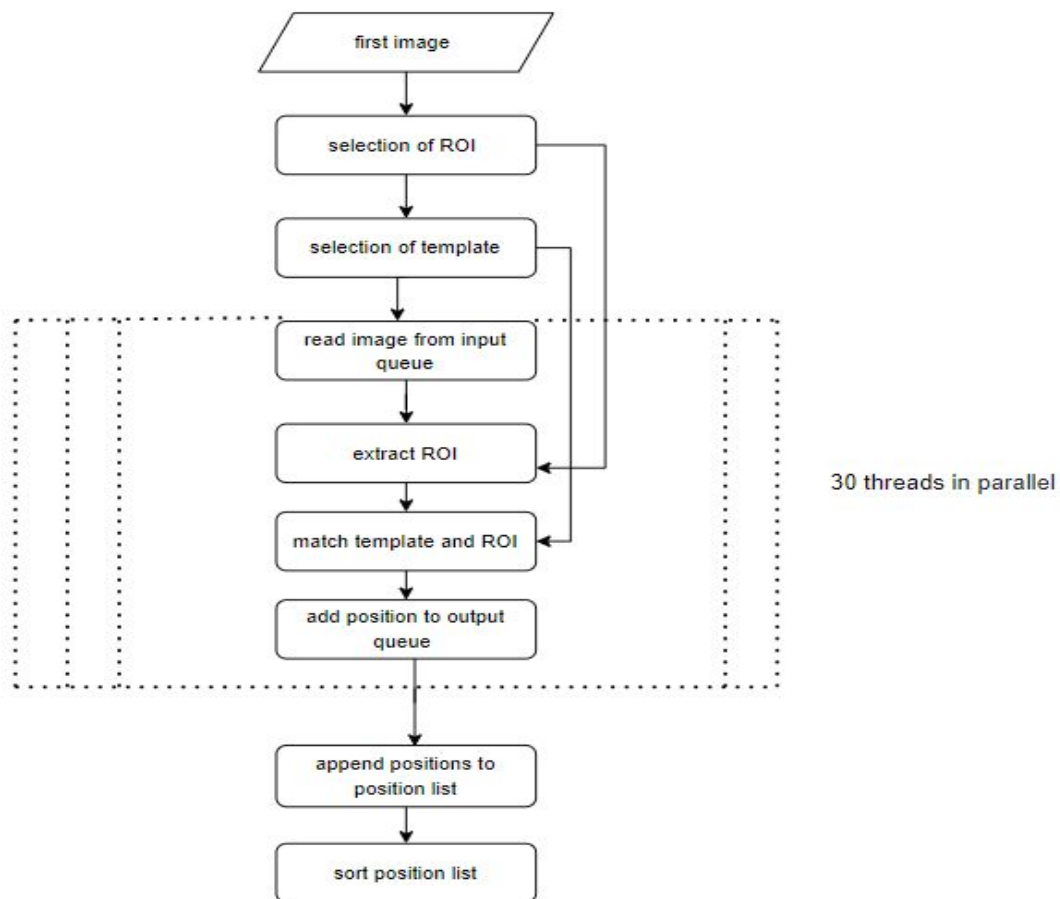
- Résolution
- Channels

=> Solution à tester par rapport aux performances et précision.

Solution: Nouvelle architecture



Threads job positions calculation



Démo

- input: 40 images de 100 nL

avec threads: Total execution time: 20.3 secs

sans threads: Total execution time: 23.9 secs

Benchmark et Résultat

nombres d'images	sample	hardware	temps d'exécution avec threads (30 threads)	temps d'exécution sans threads	Résultat avec threads
2176 images (100 nL)	résolution: 4096 x 872 taille: 3.41 Mo	CPU: i3-6500u 2 cores 4 threads 2.3GHZ	936secs (15 mins)	1493 secs (24.8 mins)	0.997 uL/min
		autre			
1804 images (20 nL)	résolution: 4096 x 416 taille: 1.62 Mo	CPU: i3-6500u 2 cores 4 threads 2.3GHZ	649.527 secs (10.8 mins)	entre 850 et 984 secs (16.4 mins)	0.201 uL/min
		autre			

to do list

- stabilisation de l'application avec threads
- tester l'effet de la diminution de la résolution de l'image par rapport à la précision des résultats
- tester l'effet des images 1 channels par rapport à la précision des résultats