



**POLYTECHNIQUE
MONTRÉAL**

UNIVERSITÉ
D'INGÉNIERIE

PHS3910 – TECHNIQUES EXPÉRIMENTALES ET INSTRUMENTATION

Équipe : Lundi 03

Suivi de particules

Fiche technique

Présenté à

Jean Provost
Lucien Weiss

Par :

Émile **Guertin-Picard** (2208363)
Philippe **Beaubois** (2211153)
Marie-Lou **Dessureault** (2211129)
Maxime **Rouillon** (2213291)

6 décembre 2024
Département de Génie Physique
Polytechnique Montréal

Table des matières

1 Description générale et spécifications	1
2 Rapports de tests	2
2.1 Suivi de particules	2
2.2 Paramètres utilisés	2
2.3 Magnification réelle	3
2.4 Caractérisation de la taille de particules	4
2.5 Étude statistique	6
2.6 Résolution de l'instrument	7
2.7 Limitations de l'appareil	8
2.8 Étude des coûts	8
3 Annexes	9
3.1 Programme de prise de données avec la caméra	9
3.2 Programme d'analyse des particules de 1 micron	11
3.3 Preuve de correction par Antidote	21

1 Description générale et spécifications

Cette fiche technique, à la demande du Gouvernement du Québec, présente les caractéristiques d'un dispositif de mesure de taille de microparticules contaminant l'environnement près de l'usine Polyfab. Les composantes principales sont un laser 405 nm (CPS405) pour illuminer les fluorophores dans les échantillons, un objectif de microscope (grossissement $M = 20$, aperture NA = 0.4), puis une lentille tube de 150 mm de focale (LA1433-A-ML) pour converger les rayons sortants de l'objectif sur le capteur d'une caméra CMOS (CS165MU) pour l'analyse [1]. Ce système présente une résolution théorique de 1.012 μm [2]. Afin d'analyser des particules d'environ cette taille, un traitement numérique est fait pour extraire la taille des particules du mouvement brownien filmé, avec une résolution sous-pixellaire. Avec ce procédé, l'appareil a été testé pour des tailles de particules dans la plage 1-10 μm . Suite à la caractérisation de particules de 1 μm , la valeur identifiée est de $1.6 \pm 0.3 \mu\text{m}$. Pour la caractérisation de particules de 10 μm , elle est de $6 \pm 1 \mu\text{m}$. Le système, monté sur table optique et utilisable dans le noir seulement, a des dimensions de $600 \times 90 \times 200$ mm, tel que présenté à la figure 1. La résolution de l'appareil post-traitement est de 0.3 μm , et son grossissement réel est de 17.14 ± 0.02 . Sans compter la table optique, le dispositif a pour coût total 1789.39 \$, prix qui peut être réduit par l'usage d'impression 3D. Les spécifications du dispositif sont rassemblées dans le tableau 1 ci-dessous.

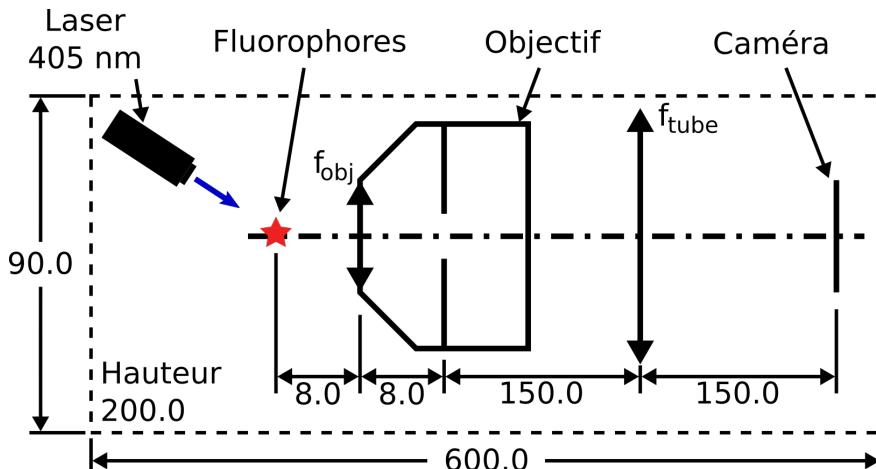


Figure 1 : Schéma du microscope avec les dimensions critiques. Toutes les dimensions sont en millimètres avec une tolérance de ± 1 mm.

Tableau 1 – Tableau des spécifications

Magnification réelle	Résolution (μm)	Dimensions (mm)	Coût \$CAD	Composantes
17.14 ± 0.02	0.3	L 600 × W 90 × H 200	\$1789.39	<ul style="list-style-type: none"> ▷ KM100S ▷ CS165MU ▷ 420FDL12 ▷ LA1433-A-ML ▷ CPS405 ▷ LMR1 ▷ TR3-P5 ▷ PH4 ▷ PH3 ▷ BA1 ▷ BA1S ▷ BA2 ▷ VC1

2 Rapports de tests

Cette section explique en détail le processus et les résultats pour les tests de caractérisation effectués sur le dispositif de suivi de particules.

2.1 Suivi de particules

Pour obtenir la taille d'une particule avec la plus grande précision possible, l'analyse de son mouvement brownien est utilisée. La position de la particule doit être déterminée sur au moins une cinquantaine d'instants successifs, avant de pouvoir procéder à l'estimation de son coefficient de diffusion et de sa taille. Des images sont prises pour chaque instant.

Pour positionner la particule avec la plus faible incertitude possible, un *fit* gaussien est effectué sur l'image pixelisée de la particule pour un moment précis. Pour que le *fit* s'effectue adéquatement, il faut restreindre la zone de *fit* à la particule. Pour ce faire, le suivi de la particule commence autour de la position précédente de la particule, par la recherche de maxima d'intensité surpassant un seuil. Les dimensions de la zone rectangulaire en question et le seuil utilisé sont des valeurs déterminées par essai-erreur et sont respectivement d'environ 100 pixels de côté et une intensité de 50 sur un maximum de 255. Une fois le nombre de maxima obtenu, soit le nombre de particules potentielles, la zone de *fit* est recentrée sur la particule la plus similaire à celle du départ.

Pour reconnaître la particule, un système de comparaison des particules potentielles en fonction de leur taille et leur luminosité est utilisé lorsque plus d'une particule est dans la même zone. La taille, soit le nombre de pixels au-dessus d'un seuil fixé, y a un rôle prépondérant, mais la luminosité, soit la moyenne d'intensité des pixels en question, a également un impact. Ces informations sont obtenues en centrant l'image sur la particule d'intérêt et en appliquant la fonction DBSCAN pour regrouper les maxima en des entités qui correspondent aux particules. La luminosité et la taille obtenues sont alors comparées à la luminosité et la taille de la première observation qui sert de référence. La première observation est obtenue comme les suivantes, avec comme seul différence que sa détection est faite par l'humain utilisant le système, qui clique initialement sur la particule que le système aura à suivre.

Une fois que la particule peut être suivie d'un instant au suivant, la localisation de la particule est obtenue à l'aide du *fit* gaussien, et l'incertitude sur la position correspond alors à l'écart-type de la gaussienne obtenue, qui est généralement plus faible que la taille d'un pixel. Des vérifications sont faites pour s'assurer que la nouvelle position est plausible et elle est alors enregistrée avant de servir de référence pour le traitement du cadre suivant.

Si la particule choisie sort de l'écran avant la fin des instants observés, le système s'arrête tout simplement à sa dernière position identifiable, soit environ 100 pixels avant le bord.

2.2 Paramètres utilisés

Certains paramètres du dispositif ont eu besoin d'être choisis pour optimiser sa performance. Entre autres, l'objectif du microscope choisi possède une magnification théorique de 20, et une ouverture numérique de 0.4. Après quelques essais, il a été conclu qu'une magnification de 20 offrait une plage de fonctionnement optimale, en considérant que pour des magnifications plus élevées, les plus grosses particules (de l'ordre de 10 microns) se déplaçaient souvent hors de la région observée. La figure 2 ci-dessous présente les images capturées pour trois différents temps d'exposition.

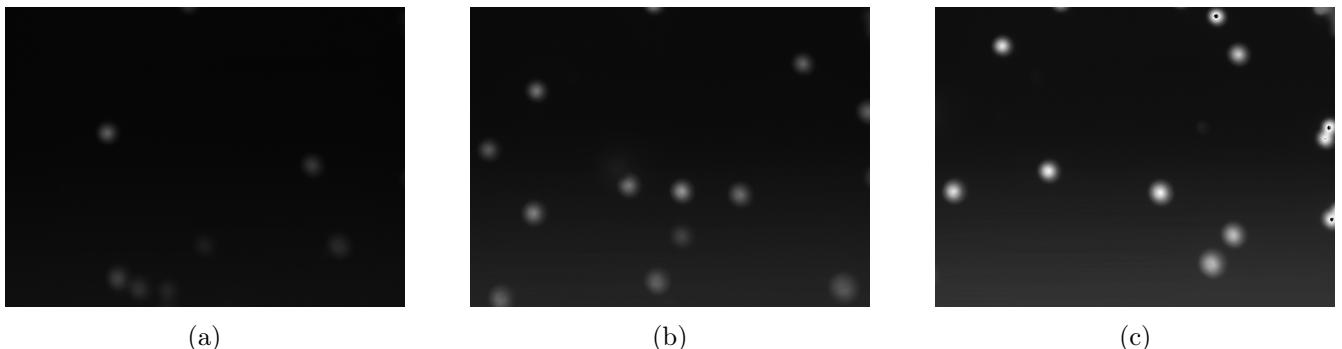


Figure 2 : Image capturée pour un temps d'exposition de : (a) 50 ms (b) 100 ms et (c) 150 ms.

Ici, l'image 2a est sous-exposée et l'image 2c est surexposée. Pour le cas sous-exposé, le contraste entre le bruit et les particules est réduit, ce qui peut affecter la justesse des *fits* gaussiens effectués pour suivre le déplacement des particules. Pour le cas surexposé, l'intensité des pixels de certaines particules est saturée, ce qu'on peut observer en haut à droite de l'image 2c. Par conséquent, un temps d'exposition de 100 ms, correspondant à l'image 2b, a été choisi.

La fréquence d'images f_{image} est dépendante au temps d'exposition choisi. Effectivement, celle-ci doit respecter la contrainte ci-dessous :

$$f_{image} \leq \frac{1}{\Delta t},$$

où Δt correspond au temps d'exposition. Pour un temps d'exposition de 0.100 s, on a : $f_{image} \leq 1/0.100 = 10$ images/s. Normalement, la fréquence d'images peut être gardée à sa valeur maximale, mais après quelques essais, il a pu été observé que de nombreuses images ne se sauvegardaient pas. Sachant que la position dans le temps de chaque capture d'image est importante pour le calcul de la MSD, la fréquence d'images a été réduite à deux images par secondes pour essayer de limiter le nombre d'images perdues.

Afin d'obtenir un calcul des premiers points de la MSD ayant des poids statistiques significatifs, le nombre d'images capturées a été posé à 75. Ce choix a été fait pour que le nombre d'images enregistrées soit de 50 ou plus, en ajoutant une marge de 25 images pour prendre en compte les images qui seront potentiellement perdues.

2.3 Magnification réelle

La magnification réelle du système a été déterminée à l'aide d'une cible ayant des dimensions connues. L'image de la cible, capturée par la caméra et suite au microscope, est présentée dans la figure 3. La distance réelle entre le centre de deux carrés blancs voisins dans le quadrillage est de 0.01 mm.

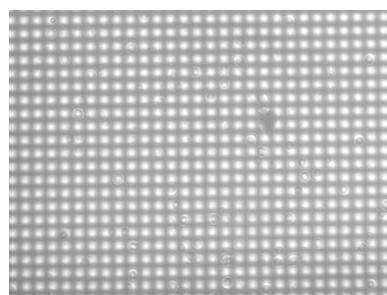


Figure 3 : Image de la cible au plan du capteur.

Afin de déterminer la distance entre les centres pour l'image de la cible, deux *fits* gaussiens ont été effectués sur une région recadrée de l'image, comprenant l'équivalent de deux carrés blancs. L'image recadrée est illustrée dans la figure 4 ci-dessous.



Figure 4 : Image recadrée de la cible.

Les deux *fits* gaussiens 2D ont été effectués avec la fonction Python `curve_fit` sur la moitié droite et sur la moitié gauche de l'image 4, ce qui a permis de trouver le centre de chaque carré blanc. La gaussienne 2D est décrite par :

$$f(x, y) = A \cdot \exp \left(- \left(\frac{(x - x_0)^2}{2\sigma_x^2} + \frac{(y - y_0)^2}{2\sigma_y^2} \right) \right) + B, \quad (1)$$

où A est l'amplitude, B est le décalage, x_0 et y_0 sont les moyennes et σ_x et σ_y sont les écarts-types. En ayant obtenu les coordonnées des centres des deux gaussiennes, on peut trouver la distance d_{im} entre les deux centres :

$$d_{im} = (x_{02} - x_{01}) \cdot d_{pixel} \approx 0.1714 \text{ mm}, \quad (2)$$

où $d_{pixel} = 3.45 \times 10^{-6}$ m est la dimension d'un pixel [3]. Ici, on approxime la distance en utilisant la distance en x parce que la composante $|y_{02} - y_{01}| \approx 0.6$ pixel est négligeable. Finalement, on retrouve la magnification réelle de l'objectif :

$$m = \frac{d_{im}}{d_{vraie}} \approx \frac{0.1714}{0.01} = 17.14. \quad (3)$$

L'incertitude sur m provient des incertitudes sur les paramètres du *fit* gaussien. On retrouve :

$$\sigma_m = \frac{\sqrt{(d_{pixel} * \sigma_{x_{02}})^2 + (d_{pixel} * \sigma_{x_{02}})^2}}{d_{vraie}} \approx 0.0169. \quad (4)$$

On obtient donc que la magnification réelle est de : 17.14 ± 0.02 . Celle-ci permettra de redimensionner les images d'une échelle en pixel à une échelle métrique.

2.4 Caractérisation de la taille de particules

Lors de l'acquisition, il a été observé que la caméra ne parvient pas à enregistrer chaque image. Afin de remédier ce problème, l'acquisition d'images remplace les images qui ne sont pas enregistrées par une image noire. Pour traiter les images et suivre les particules correctement, toutes les images noires sont premièrement ignorées pour produire une liste de positions. Les images de départ sont évaluées une par une, et à chaque fois qu'une image noire apparaît, un vecteur `[NaN, NaN]` est ajouté à la liste contenant toutes les positions. Cette procédure nous permet de savoir avec exactitude les temps des images qui n'ont pas pu être enregistrées. Une fois la liste de positions complétée, il est possible d'effectuer le calcul de la MSD avec la formule suivante :

$$MSD = \frac{1}{N_{\Delta t}} \sum_{i=0}^{N_{\Delta t}-1} (\mathbf{r}(i + \Delta t) - \mathbf{r}(i))^2, \quad (5)$$

$$\mathbf{r}(i) = x(i)\hat{x} + y(i)\hat{y} \quad (6)$$

où $N_{\Delta t}$ est le nombre de paires i disponibles pour un intervalle Δt et Δt qui varie de 1 jusqu'au nombre de positions (nombre d'images) moins une. L'incertitude sur ces valeurs est donnée par la formule suivante :

$$\alpha = A + B \quad (7)$$

$$A = \sqrt{\frac{(2(x(i + \Delta t) - x(i)) \cdot \Delta x(i + \Delta t))^2}{+ (2(x(i + \Delta t) - x(i)) \cdot \Delta x(i))^2}}$$

$$B = \sqrt{\frac{(2(y(i + \Delta t) - y(i)) \cdot \Delta y(i + \Delta t))^2}{+ (2(y(i + \Delta t) - y(i)) \cdot \Delta y(i))^2}}.$$

On remarque que c'est ici que le traitement de toutes les images est important, même celles qui ont été perdues, car sinon le Δt n'aurait pas été cohérent avec la procédure de la MDS. La façon de traiter ce calcul avec les vecteurs $[NaN, NaN]$ tout en perdant le moins d'information possible a été de définir que dès que l'élément NaN est détecté dans un calcul, $(\mathbf{r}(i + \Delta t) - \mathbf{r}(i)) = 0$. De cette façon, les images non enregistrées ne peuvent pas interférer dans le calcul de la MDS. Dans la même idée, les incertitudes sur les positions sont calculées à l'aide de l'écart-type du *fit* gaussien, et dans le cas d'un vecteur identifié par $[NaN, NaN]$, on pose $\Delta x = 0$ et $\Delta y = 0$. Les images noires ne sont donc pas prises en compte non plus dans les calculs d'incertitude.

Une fois toutes les valeurs trouvées, un graphique des résultats de MSD en fonction du Δt est produit. Le calcul de la MSD est évalué en fonction de l'intervalle de temps qui sépare deux points. En considérant que plus l'intervalle est important, moins il existe de paires de positions, seuls les cinq premiers points de la MSD ont été choisis. Effectivement, ces points correspondent à un intervalle de temps $\Delta t = 1, 2, 3, 4, 5$, pour lesquels le poids statistique est beaucoup plus élevé que pour des intervalles proches du nombre de positions évaluées ($\Delta t = 50$ dans le cas ci-présent). Afin de bien caractériser le mouvement de la particule qui correspond au mouvement brownien, un *fit* quadratique sur les points de la MSD est déterminé. En utilisant un *fit* quadratique et non un *fit* linéaire, il est possible de prendre en compte le facteur de dérive du mouvement de la particule et de l'éliminer. Effectivement, on sait que la MSD est décrite par :

$$MSD(t) = \langle (\Delta x)^2 \rangle. \quad (8)$$

En considérant le facteur de dérive et le facteur du déplacement Brownien, on peut écrire :

$$\Delta x(t) = vt + \Delta x_{Brownien}(t), \quad (9)$$

où v est la vitesse des particules due à la dérive. On insère l'équation (9) dans l'équation (8) :

$$MSD(t) = \langle (vt + \Delta x_{Brownien}(t))^2 \rangle$$

$$= \langle (vt)^2 \rangle + \langle (\Delta x_{Brownien}(t))^2 \rangle + \langle 2vt \cdot \Delta x_{Brownien}(t)x_{Brownien}(t) \rangle,$$

où vt est indépendant de $\Delta x_{Brownien}(t)$, et $\langle \Delta x_{Brownien}(t) \rangle = 0$,

$$MSD(t) = v^2 t^2 + 2nDt + 2n\sigma^2, \quad (10)$$

où n est le nombre de dimensions pour lesquelles le mouvement est analysé, D est le coefficient de diffusion et σ est l'écart type de l'erreur de localisation [4]. L'équation (10) correspond à une quadratique avec les

coefficients $a = v^2$, $b = 2nD$ et $c = 2n\sigma^2$. Les incertitudes sur ces trois valeurs sont données par la racine carrée de la diagonalisation de la matrice de covariance donnée par la fonction `curve_fit`. C'est le coefficient b du *fit* qui sera utilisé pour trouver le coefficient de diffusion (D) et la taille de la particule (r). Les formules suivantes sont utilisées :

$$D = \frac{b}{2n} = \frac{b}{4} \quad r = \frac{k_B T}{6\pi\eta D}$$

L'incertitude sur chacune de ces valeurs est donnée par l'incertitude sur le coefficient b , car on considère que les autres paramètres sont des constantes et ne comportent donc pas d'incertitudes.

2.5 Étude statistique

Pour un échantillon contenant des particules de 1 μm , 12 particules ont été suivies pour déterminer, par une analyse statistique, leurs rayons ainsi que leurs coefficients de diffusion. Le même principe a été utilisé dans un échantillon contenant des particules de 10 μm . Dans ce cas-ci, 3 particules ont été jugées comme analysables. Les résultats suivants ont été obtenus :

Tableau 2 – Tableau des valeurs du coefficient de diffusion et du rayon des particules dans l'échantillon de 1 μm .

Particule	$D (\times 10^{-13} \text{ m}^2/\text{s})$	$r (\times 10^{-6} \text{ m})$
1	1.2 ± 0.7	2 ± 1
2	3.7 ± 0.2	0.60 ± 0.03
3	2.3 ± 0.2	1.0 ± 0.1
4	5.5 ± 0.9	0.40 ± 0.07
5	0.66 ± 0.09	3.4 ± 0.4
6	1.3 ± 0.5	1.7 ± 0.6
7	1.24 ± 0.02	1.77 ± 0.03
8	1.1 ± 0.3	2.1 ± 0.5
9	2.1 ± 0.4	1.1 ± 0.2
10	0.8 ± 0.1	2.9 ± 0.5
11	1.3 ± 0.3	1.7 ± 0.4
12	4.0 ± 0.3	0.55 ± 0.04
Moyenne	8 ± 1	1.6 ± 0.3

Tableau 3 – Tableau des valeurs du coefficient de diffusion et du rayon des particules dans l'échantillon de 10 μm .

Particule	$D (\times 10^{-13} \text{ m}^2/\text{s})$	$r (\times 10^{-6} \text{ m})$
1	0.42 ± 0.07	5.3 ± 0.9
2	0.4 ± 0.1	6 ± 2
3	0.39 ± 0.10	6 ± 1
Moyenne	1.6 ± 0.4	6 ± 1

On remarque donc que les valeurs finales ont été obtenues en faisant la moyenne de chaque valeur. l'incer-

titude liée à cette moyenne est donnée par l'équation suivante :

$$\Delta x = \sqrt{\frac{\sum_{i=1}^N \delta x_i^2}{N^2}},$$

où dans notre cas, x peut être remplacé par r ou D . On remarque ici qu'il s'agit d'une propagation de l'incertitude de chaque point sur la moyenne. Il aurait pu être considéré de prendre l'écart-type des valeurs dues à la théorie du théorème central limite, mais il a été jugé ici que le nombre de points analysables des échantillons était insuffisant pour appliquer cette théorie. Pour conclure, on rappelle que pour un échantillon de $r = 1 \mu\text{m}$, analysant 12 particules différentes, le traitement de données élaboré nous donne les résultats suivants : Coefficient de dérive : $(8 \pm 1) \times 10^{-13} \text{ m}^2/\text{s}$ rayon de la particule : $(1.6 \pm 0.3) \times 10^{-6} \text{ m}$. On remarque ici qu'une erreur de 16 % a été obtenue pour le coefficient de dérive et 21.41% sur la taille de la particule. De plus, connaissant maintenant la valeur de r et son incertitude, on peut conclure, en comparant à la valeur théorique, que l'analyse des petites particules retourne des valeurs cohérentes et logiques. Finalement, pour un échantillon de $r = 10 \text{ m}$, analysant 3 particules différentes, le traitement de données élaboré nous donne les résultats suivants : Coefficient de dérive : $(1.6 \pm 0.4) \times 10^{-13} \text{ m}^2/\text{s}$ rayon de la particule : $(6 \pm 1) \times 10^{-6} \mu\text{m}$. On a donc ici une erreur de 23,15% sur la valeur coefficient de dérive et une erreur de 23.40% sur le rayon de la particule. En comparant ici les erreurs relatives entre les deux échantillons, on peut considérer que dans les deux cas les tailles de rayon sont assez bien estimées. Cependant, il est important de remarquer que le nombre de particules analysable dans l'échantillon de $10 \mu\text{m}$ est assez faible, ce qui peut jouer sur la fiabilité de représentation de nos résultats. Les images ci-dessous représentent les deux échantillons.



(a) Échantillon avec particules de $10 \mu\text{m}$



(b) Échantillon avec particules de $1 \mu\text{m}$

Figure 5 : Images d'échantillons testés pour comparaison qualitative

2.6 Résolution de l'instrument

La résolution de l'instrument concernant la taille des particules observables est définie comme la plus petite différence qu'il est capable de discerner. Selon les tests effectués, cette plus petite différence est obtenue lors de l'identification des particules de $1 \mu\text{m}$ de diamètre. Elle correspond à l'incertitude obtenue sur la valeur estimée, ce qui se traduit par une résolution de $0,3 \mu\text{m}$.

2.7 Limitations de l'appareil

L'appareil peut distinguer et suivre des particules dans la gamme de 1 à 10 μm avec et sans dérive. Toutefois, pour estimer la taille des particules, une dérive trop forte peut affecter négativement les performances de l'appareil. La grande dérive des particules dans l'échantillon de particules de 10 μm est la raison pour laquelle leur étude statistique n'est faite qu'avec 3 données, puisque les tests effectués avec les autres particules de l'échantillon ne permettaient pas d'obtenir une valeur cohérente. La valeur obtenue était soit négative, soit erronée par plusieurs ordres de grandeur.

Un seuil critique survient lorsque la dérive est de quelques ordres de grandeur supérieure à la diffusion associée au mouvement brownien. Dans ce régime, malgré une compensation de la plus grande partie de la dérive, l'incertitude qu'elle vient ajouter, bien que faible relativement au mouvement de dérive, est grande par rapport à la diffusion. Celle-ci n'est alors plus adéquatement estimée par le modèle utilisé. L'autre facteur qui semble ici déterminant est l'angle de la dérive qui ne semble pas être constant. Cette dérive non-homogène dans l'espace échappe alors en partie à la suppression du coefficient quadratique et ses composantes viennent renforcer le déplacement latéral attribué à la diffusion, ce qui augmente la mobilité de la particule et fausse les résultats.

Il y a plusieurs sources potentielles pour ces types de dérive. D'abord, pour la grande dérive, une fuite due à un mauvais calfeutrage de l'échantillon peut causer un courant dans le liquide qui englobe les particules. Ce courant peut être la cause de la dérive observée. Sans qu'il y ait de fuite, le scotch utilisé pour tenir les lames de verre ensemble peut absorber du liquide par capillarité, ce qui peut également entraîner des courants dans diverses directions. Enfin, le laser utilisé pour éclairer l'échantillon peut également y provoquer des différences de températures, entraînant un effet de convection dans le liquide. Cet effet peut provoquer lui-aussi provoquer un dérive qui varie selon la position dans l'échantillon.

En tenant compte de ces limitations, la plage de particules identifiables avec cet appareil est de 1 à 10 μm lorsque la dérive n'empêche pas la détermination adéquate de la taille. Les tailles obtenues dans cette plage de détection sont toutefois obtenues à un facteur 2 près, c'est-à-dire de 0,5 μm à 2 μm pour des particules de 1 μm , et de 5 μm à 20 μm pour celles de 10 μm . Ainsi, sans donner exactement la taille de la particule, l'appareil fournit un estimé du bon ordre de grandeur et à au plus un facteur 2.

2.8 Étude des coûts

Une étude des coûts a été faite pour le microscope construit afin de voir s'il y a possibilité de construire un appareil aux performances similaires, mais à moindre coût. Le tableau 4 présente la liste exhaustive des pièces avec leur prix en dollars canadiens avant taxes.

Quelques éléments sont à souligner. Premièrement, étant donné que l'objectif de microscope a été fourni par le gouvernement du Québec, le coût qui y est associé a été estimé selon leurs informations. Ces objectifs sont de seconde main, et ils ont été achetés en ensemble. Cela mène au faible coût d'environ 10\$, qui pourrait être difficile à retrouver pour la construction d'un seul microscope. Toutefois, pour la construction d'un ensemble d'appareils spécialisés pour différentes tailles de particules, donc ayant besoin de différents objectifs à différents grossissements, le prix d'ensemble est idéal. Deuxièmement, un coût important a été omis dans le tableau de calcul, soit celui de la table optique elle-même. Cette dernière est très dispendieuse, mais elle a été omise, car elle est déjà présente dans plusieurs laboratoires d'optique, ou si non, elle peut être acheté à plus faible coût si les dimensions de la table achetée se limitent aux dimensions de l'appareil. Par exemple, chez Thorlabs, la plus petite table optique pour tenir ce microscope est la B1824F, avec des dimensions de 18" × 24" [1]. Avec un coût de 1319.08 CAD, presque l'entièreté du coût du reste de l'appareil, le total monte à 3108.47 \$. Troisièmement, selon le tableau 4, les pièces

Tableau 4 – Liste des pièces et coûts totaux pour le microscope sur table optique [1] [5].

ID pièce	Description	Qté	\$ CAD	Total ind.
KM100S	Montage ajustable pour échantillon	1	\$130.45	\$130.45
M-TSX-1S	Platine de translation	1	\$305.80	\$305.80
CS165MU	Caméra CMOS monochrome	1	\$667.01	\$667.01
-	Objectif de microscope	1	\$10.00	\$10.00
420FDL12	Filtre passe-long	1	\$36.29	\$36.29
LA1433-A-ML	Lentille tube $f = 150.0$ mm	1	\$71.42	\$71.42
CPS405	Laser bleu 405 nm	1	\$312.07	\$312.07
LMR1	Trou taraudé pour lentilles	2	\$22.89	\$45.79
TR3-P5	5 tiges 3 po pour optiques	1	\$38.50	\$38.50
PH4	Base pour tiges d'optique 4 po	4	\$14.83	\$59.33
PH3	Base pour tiges d'optique 3 po	1	\$13.37	\$13.37
BA1	Pied de montage optique	1	\$8.42	\$8.42
BA1S	Pied de montage optique	2	\$7.83	\$15.65
BA2	Pied de montage optique	1	\$11.26	\$11.26
VC1	Pince en V	1	\$64.04	\$64.04
			Total :	\$1789.39

ayant les coûts les plus considérables sont celles directement liées à l'optique et à l'alignement. Le reste des pièces sont pour des éléments de construction du montage, et leur total s'élève à 1039.11 \$.

Il est donc possible de recommander fortement l'utilisation de l'impression 3D pour une production à plus grand volume de ce microscope. Cela permettra de sauver l'argent sur les pièces de construction et sur la table optique elle-même, aiderait à l'alignement des éléments optiques, permettrait plus de flexibilité sur les composantes à ajuster tel que le montage de l'échantillon pour s'occuper du focus, et enfin pourrait rendre l'appareil plus portatif pour des essais sur le terrain. Comme les éléments optiques seraient tous réutilisables dans un design d'impression 3D, les performances du dispositif resteraient identiques. Il est aussi possible de recommander l'usage d'un différent laser de 405 nm, car ils sont fréquemment vendus en seconde main pour beaucoup moins cher.

3 Annexes

3.1 Programme de prise de données avec la caméra

```

1 import numpy as np
2 import matplotlib as plt
3 import pycromanager
4 from pycromanager import Core
5 import time
6 from typing import Any
7 from PIL import Image
8 import pylablib as pll
9 from pylablib.devices import Thorlabs
10 from thorlabs_tsi_sdk.tl_camera import TLCameraSDK, OPERATION_MODE, TLCameraError
11
12 pll.par["devices/dlls/thorlabs_tlcam"] = r"C:\Users\marie\Documents\Scientific Camera
  Interfaces\SDK\Python Toolkit\dlls\64_lib"
13

```

```
14 #Numero de serie de la camera
15 printThorlabs.list_cameras_tlcam())
16
17 try:
18     # Initialiser le SDK
19     sdk = TLCameraSDK()
20     print("SDK initialise avec succes.")
21
22     # Decouvrir les cameras disponibles
23     available_cameras = sdk.discover_available_cameras()
24     print(f"Cameras disponibles : {available_cameras}")
25
26 if not available_cameras:
27     print("Aucune camera trouvée. Vérifiez la connexion.")
28     sdk.dispose()
29     exit()
30
31 # Essayer d'ouvrir la première camera disponible
32 try:
33     with sdk.open_camera(available_cameras[0]) as cam1:
34         print(f"Camera connectée avec succès : {available_cameras[0]}")
35
36         # Configurer les paramètres de la camera
37         cam1.exposure_time_us = 125000 # Exemple de temps d'exposition de 50 ms (
38         # tester sur Thorlab)
39         cam1.black_level=1
40         cam1.gain=0
41         cam1.frames_per_trigger_zero_for_unlimited = 1 # Mode de capture continue :
42         0 ; Mode de capture ponctuel : 1
43         cam1.arm(10) # Préparez la camera avec 2 tampons
44         nombre_image_par_seconde=2
45         print("Camera armée et prête pour capturer des images.")
46
47         output_tiff_path = r"C:\Users\marie\Documents\GitHub\super_res_microscopy\
48 acquisition\video_output_carac_125ms_2im_poussieres.tiff"
49
50         # Capture and save images
51         images = [] # List to store frames for TIFF
52         for picture_number in range(5): # Capture 100 frames
53             cam1.issue_software_trigger()
54             time.sleep(1 / nombre_image_par_seconde) # Maintain frame rate
55
56             frame = cam1.get_pending_frame_or_null()
57             if frame is not None:
58                 print(f"Image successfully captured: {frame.frame_count}")
59                 # Convert the image buffer to a NumPy array
60                 image = np.array(frame.image_buffer, dtype=np.uint8).reshape(
61                     (cam1.image_height_pixels, cam1.image_width_pixels)
62                 )
63             else:
64                 image = np.zeros((cam1.image_height_pixels, cam1.image_width_pixels),
65 , dtype=np.uint8)
66                 print("No image captured.")
67
68                 # Add the image to the list as a Pillow Image
69                 images.append(Image.fromarray(image))
70
71                 # Save all frames as a multi-page TIFF
72                 if images:
```

```

69         images[0].save(
70             output_tiff_path,
71             save_all=True,
72             append_images=images[1:], # Save as multi-page TIFF
73             compression="tiff_deflate" # Optional: Apply compression
74         )
75         print(f"Multi-page TIFF saved at: {output_tiff_path}")
76     else:
77         print("No frames to save.")

78
79     # Desarmer et fermer la camera apres utilisation
80     cam1.disarm()
81     print("Camera desarmee et fermee.")

82
83
84 except TLCameraError as e:
85     print(f"Erreur lors de la connexion de la camera : {e}")
86     print("Verifier que la camera est correctement connectee et non utilisee par un autre programme.")

87
88 finally:
89     # Assurez-vous que le SDK est libere pour liberer les ressources
90     sdk.dispose()

91
92 except TLCameraError as e:
93     print(f"Erreur SDK : {e}")

94
95 finally:
96     # Nettoyer le SDK correctement
97     if 'sdk' in locals():
98         sdk.dispose()

```

3.2 Programme d'analyse des particules de 1 micron

```

1 import matplotlib
2 matplotlib.use('Qt5Agg') # Utiliser le backend Qt5Agg pour Windows
3 import matplotlib.pyplot as plt
4 import matplotlib.animation as animation
5 from scipy.optimize import curve_fit
6 import cv2
7 import numpy as np
8 import os
9 from PIL import Image
10 import lmfit
11 #import tifffile
12 #%matplotlib tk #pour Linux
13
14 import scipy.ndimage as ndimage
15 from sklearn.cluster import DBSCAN
16
17 path_to_tiff = os.path.join("../", "acquisition", "essai_10um_2im_50ms.tiff")
18 tiff = Image.open(path_to_tiff)
19
20 with Image.open(path_to_tiff) as img:
21     frame_number = 0
22     actual_frames = 0
23     try:
24         while True:

```

```
25         frame_number += 1
26         if np.sum(np.array(img)) != 0:
27             actual_frames += 1
28
29         img.seek(frame_number)
30     except EOFError:
31         print("All frames processed.")
32
33 frame_index = 0
34 first_frame = 0
35 tiff.seek(frame_index)
36 original_image = np.array(tiff)
37
38 while np.sum(original_image) == 0:
39     frame_index += 1
40     tiff.seek(frame_index)
41     original_image = np.array(tiff)
42     first_frame += 1
43     print(frame_index)
44
45 clahe = cv2.createCLAHE(clipLimit=10.0, tileGridSize=(30, 30))
46 preprocessed = clahe.apply(original_image)
47
48 blurred = cv2.medianBlur(preprocessed, 115)
49 preprocessed2 = cv2.subtract(preprocessed, blurred)
50
51 # Apply Non-Local Means Denoising
52 img = cv2.fastNlMeansDenoising(preprocessed2, None, 15, 7, 41)
53
54 def crop(img, x, y, crop_size_x, crop_size_y):
55     x_start = int(x - crop_size_x // 2)
56     x_end = int(x + crop_size_x // 2)
57     y_start = int(y - crop_size_y // 2)
58     y_end = int(y + crop_size_y // 2)
59
60     if x_start < 0:
61         x_start = 0
62     if x_end > 1440:
63         x_end = 1440
64     if y_start < 0:
65         y_start = 0
66     if y_end > 1080:
67         y_end = 1080
68
69     return img[y_start:y_end, x_start:x_end]
70
71 fig, ax = plt.subplots()
72 ax.imshow(img, origin='lower', cmap='gray')
73 plt.title(f"Frame {frame_index}: Select a point")
74
75 # Selection de points
76 print("Please click on the point you want to select.")
77 x, y = plt.ginput(1)[0]
78 print(f"Selected point: ({x}, {y})")
79 plt.close()
80
81 crop_sze_x = 250
82 crop_sze_y = 300
83
```

```
84 def visionneur(frame, titre):
85     plt.figure(figsize=(10, 5))
86     plt.clf()
87     plt.imshow(frame, origin='lower', cmap='gray')
88     plt.title(titre)
89     plt.colorbar()
90     plt.show()
91
92 def prepare_data(x, y, z):
93     return (x.flatten(), y.flatten(), z.flatten())
94
95 def gaussian_2d(xy, amplitude, x0, y0, sigma_x, sigma_y, offset):
96     x, y = xy
97     a = 1 / (2 * sigma_x**2)
98     b = 1 / (2 * sigma_y**2)
99     return offset + amplitude * np.exp(- (a * (x - x0)**2 + b * (y - y0)**2))
100
101 def localisateur_gaussien(intensity_grid, centre):
102     x = np.arange(intensity_grid.shape[1])
103     y = np.arange(intensity_grid.shape[0])
104     X, Y = np.meshgrid(x, y)
105
106     # Preparer les donnees pour le fit
107     (xdata, ydata), zdata = prepare_data(X, Y, intensity_grid)
108     model = lmfit.Model(gaussian_2d)
109     max_idx = np.unravel_index(np.argmax(intensity_grid), intensity_grid.shape)
110
111     initial_x0 = x[max_idx[1]]
112     initial_y0 = y[max_idx[0]]
113
114     # Definir les parametres du modele
115     params = model.make_params(
116         amplitude=np.max(intensity_grid),
117         x0=initial_x0,
118         y0=initial_y0,
119         sigma_x=5,
120         sigma_y=5,
121         offset=3
122     )
123
124     # Ajouter des bornes sur les parametres x0 et y0
125     params['x0'].min = 0      # Limite inferieure pour x0
126     params['x0'].max = intensity_grid.shape[1] - 1 # Limite superieure pour x0
127     params['y0'].min = 0      # Limite inferieure pour y0
128     params['y0'].max = intensity_grid.shape[0] - 1 # Limite superieure pour y0
129
130     # Ajouter des bornes sur les parametres sigma
131     params['sigma_x'].min = 1 # Limite inferieure de sigma_x
132     params['sigma_x'].max = 40 # Limite superieure de sigma_x
133     params['sigma_y'].min = 1 # Limite inferieure de sigma_y
134     params['sigma_y'].max = 40 # Limite superieure de sigma_y
135
136     # Effectuer l'ajustement
137     result = model.fit(zdata, params, xy=(xdata, ydata))
138
139     x_position = result.params['x0'].value + centre[0] - crop_sze_x//4
140     y_position = result.params['y0'].value + centre[1] - crop_sze_y//4
141
142     return [x_position, y_position], result.params['sigma_x'].value, result.params['
```

```
sigma_y'].value

143
144 def denoise(image):
145     clahe = cv2.createCLAHE(clipLimit=10.0, tileGridSize=(30, 30))
146     preprocessed = clahe.apply(image)
147
148     blurred = cv2.medianBlur(preprocessed, 115)
149     preprocessed2 = cv2.subtract(preprocessed, blurred)
150
151     return cv2.fastNlMeansDenoising(preprocessed2, None, 15, 7, 41)
152
153 def is_within_bounds(position):
154     x, y = position
155
156     # Definir les bornes du cadre
157     x_min, x_max = 100, 1300
158     y_min, y_max = 100, 980
159
160     # Vérifier si la position est dans les bornes du cadre
161     if x_min <= x <= x_max and y_min <= y <= y_max:
162         return True
163     else:
164         return False
165
166 def empreinte_digitale(image, threshold=50, eps=3, min_samples=1):
167     # 1. Observer chaque particule approximativement
168     visionneur(image, f'particule choisie')
169
170     maxima = (image > threshold)
171
172     coordinates = np.column_stack(np.where(maxima))
173     if coordinates.shape[0] == 0:
174         print("Aucune particule détectée.")
175         return [], image
176
177     dbscan = DBSCAN(eps=eps, min_samples=min_samples)
178     labels = dbscan.fit_predict(coordinates)
179     unique_labels = np.unique(labels)
180     size=0
181
182     # 2. Déterminer l'empreinte digitale de notre particule
183     for label in unique_labels:
184         if label != -1: # -1 correspond au bruit dans DBSCAN
185             cluster_points = coordinates[labels == label]
186             plt.scatter(cluster_points[:, 1], cluster_points[:, 0], label=f'Particule {label}')
187
188             if cluster_points.shape[0]>size:
189                 size = cluster_points.shape[0]
190                 luminosity = np.mean(image[cluster_points[:, 0], cluster_points[:, 1]])
191
192                 seuil=np.max(image)*0.5
193                 print(f'seuil:{seuil}, taille:{size} et luminosité: {luminosity}')
194
195     # 3. Afficher les résultats et filtrer les particules selon leur taille et
196     # luminosité
197     plt.imshow(image, origin='lower', cmap='gray')
198     plt.title(f"Empreinte digitale de la particule, lum={luminosity} et taille={size}")
199     plt.legend()
```

```
199 plt.colorbar()
200 plt.show()
201
202     return size, luminosity, seuil
203
204 def detect_nbre_particles(image, threshold, eps=9, min_samples=1):
205     # 1. Appliquer un filtre de voisinage pour detecter les maxima locaux
206     neighborhood_size = 3 # Taille du voisinage pour detecter les maxima locaux
207     local_max = ndimage.maximum_filter(image, size=neighborhood_size)
208     # 2. Comparer l'image originale et les maxima locaux pour identifier les vrais
209     # maxima
210     maxima = (image == local_max) & (image > threshold)
211
212     # 3. Extraire les coordonnees des maxima (particules)
213     coordinates = np.column_stack(np.where(maxima)) # Extraire les indices des pixels
214     maximaux
215     if coordinates.shape[0] == 0:
216         print("Aucune particule detectee.")
217         visionneur(image, f'detection du nombre de particules')
218
219     return [], image
220
221     # 4. Appliquer DBSCAN pour regrouper les particules proches (cluster les maxima
222     # detectes)
223     dbSCAN = DBSCAN(eps=eps, min_samples=min_samples)
224     labels = dbSCAN.fit_predict(coordinates)
225
226     #5. Determiner le nombre de particules
227     unique_labels = np.unique(labels)
228     nb_particules = len(unique_labels)
229
230     positions=[]
231     for label in unique_labels:
232         if label != -1: # -1 correspond au bruit dans DBSCAN
233             cluster_points = coordinates[labels == label]
234             positions.append((np.mean(cluster_points[:, 1]), np.mean(cluster_points[:, 0])))
235
236     # Retourner les coordonnees des particules filtrées
237     return nb_particules, positions
238
239
240 def identificateur_particles(image, ref_size, ref_luminosity, threshold=50, eps=9,
241 min_samples=1):
242     # 1. Obtenir le plus d'info sur la particule
243     maxima = (image > threshold)
244
245     coordinates = np.column_stack(np.where(maxima))
246     if coordinates.shape[0] == 0:
247         print("Mauvais re-crop empêche identification de la particule.")
248         return [], image
249
250     dbSCAN = DBSCAN(eps=eps, min_samples=min_samples)
251     labels = dbSCAN.fit_predict(coordinates)
252     unique_labels = np.unique(labels)
253
254     # 2. Parcourir chaque particule et identifier laquelle est celle qu'on suit
255     distances = []
256     position_particules = []
257
258     for label in unique_labels:
```

```

254     if label != -1: # -1 correspond au bruit dans DBSCAN
255         cluster_points = coordinates[labels == label]
256         plt.scatter(cluster_points[:, 1], cluster_points[:, 0], label=f'Particule {label}')
257
258         size = cluster_points.shape[0]
259         luminosity = np.mean(image[cluster_points[:, 0], cluster_points[:, 1]])
260         size_distance = abs(size - ref_size)
261         luminosity_distance = abs(luminosity - ref_luminosity)
262
263         combined_distance = size_distance + luminosity_distance
264         distances.append(combined_distance)
265         position_particules.append((np.mean(cluster_points[:, 1]), np.mean(
266             cluster_points[:, 0])))
267
268         ecart = np.min(distances)
269         position_best_particle = position_particules[np.argmin(distances)]
270
271     return position_best_particle, ecart
272
273 def particle_tracker_simple(image, x, y):
274     image = denoise(image)
275
276     cropped_img = crop(image, x, y, crop_sze_x, crop_sze_y)
277
278     #Gerer plus qu'une particule
279     cropped_img = np.array(cropped_img)
280     max_index = np.argmax(cropped_img)
281     max_coords = np.unravel_index(max_index, cropped_img.shape)
282
283     nouveau_x = x - crop_sze_x // 2 + max_coords[1]
284     nouveau_y = y - crop_sze_y // 2 + max_coords[0]
285
286     second_crop = crop(image, nouveau_x, nouveau_y, crop_sze_x, crop_sze_y)      # Re-crop
287     # autour d'une seule particule
288
289     result_fit = localisateur_gaussien(second_crop, [nouveau_x, nouveau_y])
290
291     x_new, y_new = result_fit[0][0], result_fit[0][1]
292
293     return [result_fit, cropped_img, (x_new, y_new), (result_fit[1], result_fit[2])]
294
295 #Places ici pour pouvoir y faire reference dans particule_tracker
296 position_list=[[x,y]]
297 sigma_list = []
298 crop_frames = []
299 big_frames = []
300
301 def particle_tracker(img, x, y, taille_initiale, luminosite_initiale, seuil):
302     # 1. Ce qu'on voit pres de l'ancienne position
303     image = denoise(img)
304
305     cropped_img = crop(image, x, y, crop_sze_x, crop_sze_y)
306
307     # 2. Determiner combien il y a de particules et laquelle est la notre
308     # Utiliser une reconnaissance de particules et si plus qu'une, alors clic pour
309     # choisir
310     nb_part, coordonnees = detect_nbre_particles(cropped_img, seuil)

```

```

309     if nb_part == 1:
310         cropped_img = np.array(cropped_img)
311         max_index = np.argmax(cropped_img)
312         max_coords = np.unravel_index(max_index, cropped_img.shape)
313
314         nouveau_x = x - crop_sze_x // 2 + max_coords[1] #Oui c'est inverse a cause du
unravel juste avant
315         nouveau_y = y - crop_sze_y // 2 + max_coords[0]
316     elif (nb_part == 0) or (nb_part == []):
317         # choix de point
318         fig, ax = plt.subplots()
319         ax.imshow(denoise(img), origin='lower', cmap='gray')
320         plt.title(f"Frame {len(position_list)-1}: Select a point")
321
322         # Ajouter une croix rouge a la position donnee
323         cx, cy = position_list[len(position_list)-1]
324         ax.plot(cx, cy, 'rx', markersize=10)
325
326         # Afficher l'image et demander a l'utilisateur de cliquer
327         print("Please click on the point you want to select.")
328         nouveau_x, nouveau_y = plt.ginput(1)[0] # Attente du clic de l'utilisateur (1
point)
329
330         # Fermer l'affichage apres selection
331         plt.close()
332     else:
333         imperfections, emplacements = [], []
334         for num_particule in range(nb_part):
335             x_identifier=x - crop_sze_x // 2 + coordonnees[num_particule][0]
336             y_identifier=y - crop_sze_y // 2 + coordonnees[num_particule][1]
337             crop_pour_identifier = crop(image, x_identifier, y_identifier, crop_sze_x
//2, crop_sze_x//2)
338             emplacement, ecart = identificateur_particles(crop_pour_identifier,
taille_initiale, luminosite_initiale, seuil)
339             imperfections.append(ecart)
340             emplacements.append(emplacement)
341             qualite = np.min(imperfections)
342             print(f'ecart:{qualite}')
343             max_coords = emplacements[np.argmin(imperfections)]
344             nouveau_x = x_identifier - crop_sze_x // 4 + max_coords[0]
345             nouveau_y = y_identifier - crop_sze_y // 4 + max_coords[1]
346
347         # Verification s'il fout le camp
348         dis = np.sqrt((position_list[-1][0]-nouveau_x)**2+(position_list[-1][1]-nouveau_y)
**2)
349         if dis>100:
350             # Display the image and let the user select a point interactively
351             fig, ax = plt.subplots()
352             ax.imshow(denoise(img), origin='lower', cmap='gray')
353             plt.title(f"Frame {len(position_list)-1}: Probleme d'identification, nouvelle
pos a {dis}, select a point")
354
355             print('probleme d''identification')
356             # Ajouter une croix rouge a la position donnee
357             cx, cy = position_list[-1]
358             ax.plot(cx, cy, 'rx', markersize=10)
359
360             # Afficher l'image et demander a l'utilisateur de cliquer
361             print("Please click on the point you want to select.")

```

```

362     nouveau_x, nouveau_y = plt.ginput(1)[0] # Attente du clic de l'utilisateur (1
363     print(f"Selected point: ({nouveau_x}, {nouveau_y})")
364
365     # Fermer l'affichage apres selection
366     plt.close()
367
368     # 3. Crop autour de notre particule et fit dessus
369     second_crop = crop(image, nouveau_x, nouveau_y, crop_sze_x//2, crop_sze_y//2)      #
370     # Re-crop autour d'une seule particule
371
372     result_fit = localisateur_gaussien(second_crop, [nouveau_x, nouveau_y])
373
374     x_new, y_new = result_fit[0][0], result_fit[0][1]
375
376     return [result_fit, second_crop, (x_new, y_new), (result_fit[1], result_fit[2])]
377
378 taille_initiale, luminosite_initiale, seuil = empreinte_digitale(crop(img,x,y,crop_sze_x
379 *0.8,crop_sze_y*0.5))
380 print(f'position initiale: {position_list}')
381
382 for i in range(first_frame,actual_frames):
383     tiff.seek(i)
384     if np.sum(np.array(tiff))==0:
385         position_list.append(np.array([np.nan,np.nan]))
386         sigma_list.append(np.array([np.nan,np.nan]))
387         print(f'frames vide:{i}')
388     else:
389         data = particle_tracker(img, x, y, taille_initiale, luminosite_initiale, seuil)
390         position_list.append(data[2])
391         x,y = position_list[i-first_frame+1]
392         sigma_list.append(data[3])
393         tiff.seek(i)
394         img = np.array(tiff)
395         big_frames.append(img)
396         crop_frames.append(data[1])
397         if is_within_bounds(position_list[-1]):
398             pass
399         else:
400             break
401     print(f'liste des positions frame {i}:{position_list}')
402
403 pixel_size = 3.45 # Taille absolue
404 f2 = 150 # Focale de L2
405 M_theo = 20 # Magnification
406 pxl = pixel_size / (f2 * M_theo / 160) # Pixel size en um dans le plan de la particule
407
408 sigma_arr=np.array(sigma_list)* p xl * 10**(-6)
409 position_arr=np.array(position_list)* p xl * 10**(-6)
410 print(position_arr)
411
412 def calculate_msd_with_uncertainty(position_arr, delta_x, delta_y):
413     """
414     Calcule les MSD avec propagation des incertitudes analytiques.
415     """
416     if position_arr[0:].shape[0] != delta_x[0:].shape :
417         position_arr=position_arr[:len(position_arr)-1]
418
419     # Traitement Lucien

```

```

418 diff_pair = np.zeros_like(position_arr[1:])
419 dx = np.zeros_like(delta_x[1:])
420 dy = np.zeros_like(delta_y[1:])
421
422 # Masque pour detecter les NaN
423 nan_mask_pos = np.isnan(position_arr[1:]) | np.isnan(position_arr[:-1])
424 nan_mask_delta = np.isnan(delta_x[1:]) | np.isnan(delta_x[:-1])
425
426 # Calculer les differences uniquement ou nan_mask est False
427 valid_mask_pos = ~nan_mask_pos.any(axis=1) # Inverser le masque pour obtenir les
428 position_arr valides
429 valid_mask_delta = ~nan_mask_delta
430
431 diff_pair[valid_mask_pos] = position_arr[1:][valid_mask_pos, :] - position_arr[:-1][
432 valid_mask_pos, :]
433 # Calculer les differences en x uniquement (prendre la premiere colonne des
434 positions)
435 print(len(diff_pair))
436 vitesse_moyenne = np.mean(diff_pair, axis=0)
437
438 temps = np.arange(0, len(position_arr)) # vecteur de temps
439
440 # Appliquer la soustraction terme a terme
441 nouvelles_donnees = position_arr - vitesse_moyenne * temps[:, np.newaxis]
442 print(f'vitesse moyenne d\'f'environ: {vitesse_moyenne}')
443
444 msd = []
445 uncertainties = []
446 for d in range(1, len(nouvelles_donnees)):
447     diff_pairs = np.zeros_like(nouvelles_donnees[d:])
448     dx = np.zeros_like(delta_x[d:])
449     dy = np.zeros_like(delta_y[d:])
450
451     # Masque pour detecter les NaN
452     nan_mask_pos = np.isnan(nouvelles_donnees[d:]) | np.isnan(nouvelles_donnees[:-d])
453     nan_mask_delta = np.isnan(delta_x[d:]) | np.isnan(delta_x[:-d])
454
455     # Calculer les differences uniquement ou nan_mask est False
456     valid_mask_pos = ~nan_mask_pos.any(axis=1) # Inverser le masque pour obtenir
457     les position_arr valides
458     valid_mask_delta = ~nan_mask_delta
459
460     diff_pairs[valid_mask_pos] = nouvelles_donnees[d:][valid_mask_pos, :] -
461     nouvelles_donnees[:-d][valid_mask_pos, :]
462
463     distances_squared = np.sum(diff_pairs**2, axis=1)
464     msd.append(np.mean(distances_squared))
465
466     dx[valid_mask_delta] = delta_x[d:][valid_mask_delta] + delta_x[:-d][
467     valid_mask_delta]
468     dy[valid_mask_delta] = delta_y[d:][valid_mask_delta] + delta_y[:-d][
469     valid_mask_delta]
470     term_x = 2 * (diff_pairs[:, 0]**2) * (dx**2)
471     term_y = 2 * (diff_pairs[:, 1]**2) * (dy**2)
472
473     # Propagation des incertitudes
474     total_uncertainty = np.mean(term_x + term_y)
475     uncertainties.append(np.sqrt(total_uncertainty))

```

```
469     return np.array(msd), np.array(uncertainties)
470
471 calculate_msd_with_uncertainty(position_arr, sigma_arr[:, 0], sigma_arr[:, 1])
472
473 msd_values, uncertainties = calculate_msd_with_uncertainty(position_arr, sigma_arr[:, 0], sigma_arr[:, 1])
474
475 time_intervals = np.arange(1, len(msd_values) + 1) * 0.5
476
477
478 # Plot MSD
479 plt.figure(figsize=(8, 6))
480 plt.errorbar(time_intervals, msd_values, yerr=uncertainties, fmt='o', label='MSD', color='blue')
481 plt.title("MSD en fonction de l'intervalle de temps")
482 plt.xlabel("Intervalle de temps")
483 plt.ylabel("MSD")
484 plt.grid(True)
485 plt.legend()
486 plt.show()
487
488 # Fonction quadratique pour l'ajustement
489 def quadratic(x, a, b, c):
490     return a * x**2 + b * x + c
491 def line(x, b,c):
492     return b*x+c
493
494 msd_values, uncertainties = calculate_msd_with_uncertainty(position_arr, sigma_arr[:, 0], sigma_arr[:, 1])
495
496 cropped_msd = msd_values[:5]
497 cropped_inc = uncertainties[:5]
498
499 time_intervals = np.arange(1, len(cropped_msd) + 1)
500
501 # curve fit quadratique
502 bounds = ([-np.inf, -np.inf, -np.inf], [np.inf, np.inf, np.inf])
503 popt, pcov = curve_fit(line, time_intervals, cropped_msd, sigma=cropped_inc)
504
505 # coefficients optimaux
506 coeffs = popt
507
508 # Incertitudes sur les coefficients sont les elements diagonaux
509 coeff_uncertainties = np.sqrt(np.diag(pcov))
510
511 # Affichage des coefficients separement
512 print("Coefficients du polynome ajuste :")
513 print(coeffs)
514
515 # Affichage de la matrice de covariance separement
516 print("Matrice de covariance des coefficients :")
517 print(pcov)
518
519 # Creation de la fonction quadratique de l'ajustement
520 quadratic_fit = np.poly1d(coeffs)
521
522 # Generation des valeurs ajustees pour afficher la courbe
523 fitted_msd = quadratic_fit(time_intervals)
```

```

525 # Affichage du MSD avec les barres d'erreur
526 plt.figure(figsize=(8, 6))
527 plt.errorbar(time_intervals, cropped_msd, yerr=cropped_inc, fmt='o', label='MSD', color=
528     'blue')
529 plt.plot(time_intervals, fitted_msd, label='Quadratic Fit', color='red', linestyle='--')
530 plt.title("MSD en fonction de l'intervalle de temps")
531 plt.xlabel("Intervalle de temps")
532 plt.ylabel("MSD")
533 plt.grid(True)
534 plt.legend()
535 plt.show()

536 # Affichage des incertitudes sur les coefficients
537 print(f"Incertitudes sur les coefficients : {coeff_uncertainties}")

538 # Calcul du coefficient de diffusion et de la taille de la particule
539 r = (4 * 1.38 * 10**-23 * 300 / (6 * np.pi * 10**(-3) * coeffs[0])) # Coefficient de
540     diffusion
541 r_uncertainty = (4 * 1.38 * 10**(-23) * 300 * coeff_uncertainties[0] / (6 * np.pi *
542     10**(-3) * (coeffs[0])**2)) # Coefficient de diffusion
543
544 # Affichage de la taille de la particule avec incertitude
545 print(f"Taille de la particule (m): {r:.2e} m, avec une incertitude : {r_uncertainty:.2e}
546 }")

```

3.3 Preuve de correction par Antidote

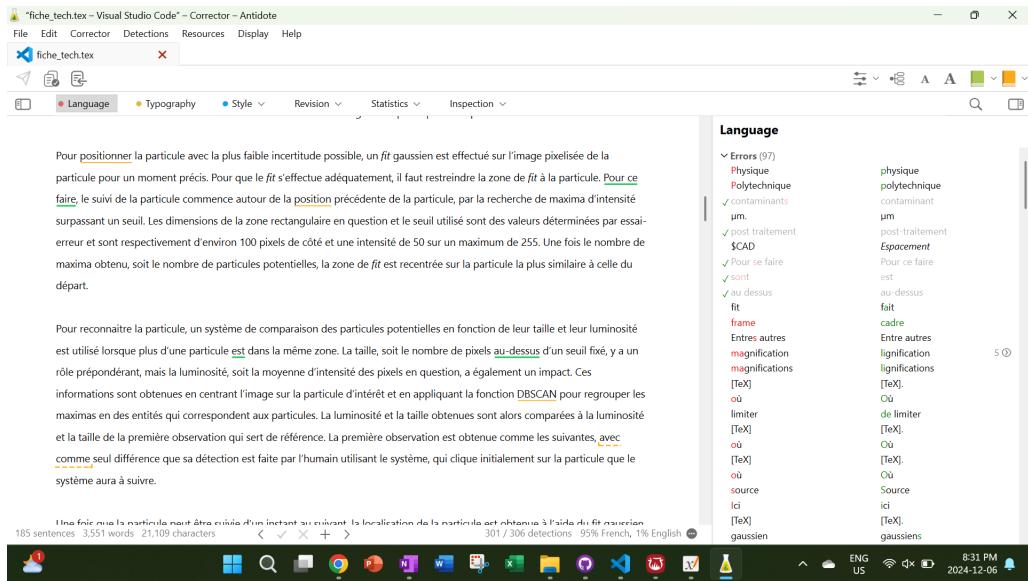


Figure 6 : Correction par Antidote

Références

- [1] Thorlabs - Photonics Products & Solutions, 2024. URL <https://www.thorlabs.com>.
- [2] Maxime Rouillon, Émile Guertin-Picard, Marie-Lou Dessureault, and Philippine Beaubois. Travail préparatoire - Microscope, 2024.
- [3] Thorlabs - CS165MU Zelux 1.6 MP Monochrome CMOS Camera, 1/4-20 Taps, 2024. URL <https://www.thorlabs.com/thorproduct.cfm?partnumber=CS165MU>.
- [4] Lucien Weiss and Jean Provost. Mandat 3 - Suivi de particules, 2024.
- [5] M-TSX-1D Dovetail Linear Stage, 2024. URL <https://www.newport.com/p/M-TSX-1D>.