Détection de petites cibles mobile dans des arrière-plans complexes en utilisant des capteurs optiques par traitement bio-inspiré.

‘’Approche Python’’



* SLT Boukary Derra
* LCL Tijeni Delleji

Année: 2022 – 2023

# Table des matières

[Table de matières 2](#_Toc135204890)

[1. Introduction 2](#_Toc135204891)

[2. La méthode \_\_init\_\_() 6](#_Toc135204892)

[3. Le noyau gaussien 7](#_Toc135204893)

[4. Photorécepteur 9](#_Toc135204894)

[5. Lipetz transformation 11](#_Toc135204895)

[6. Le filtre passe bas 11](#_Toc135204896)

[7. LMC 12](#_Toc135204897)

[8. ON / OFF channels 13](#_Toc135204898)

[9. FDSR 13](#_Toc135204899)

[10. SIGMA 15](#_Toc135204900)

[11. HWR 16](#_Toc135204901)

[12. Inhibition latérale 17](#_Toc135204902)

[12.1. Block based motion estimation 17](#_Toc135204903)

[12.2. Find optical motion vection 18](#_Toc135204904)

[1.4.3. get motion components 20](#_Toc135204905)

[1.4.4. create convolution kernel and convolve uv witdh h 21](#_Toc135204906)

[1.4.5. Calculate W 22](#_Toc135204907)

[13. Get\_delay and Final output 23](#_Toc135204908)

[14. Conclusion 24](#_Toc135204909)

[Annexe 25](#_Toc135204910)

[A1. Résultat des différents couche du frame 63 de V1 ; 25](#_Toc135204911)

[A2. Résultat des différents couche du frame 63 de v2; 31](#_Toc135204912)

# Tables des Figures

[Figure 1: Frame V1-63.: 4](#_Toc135204964)

[Figure 2: Frame V2-63. 5](#_Toc135204965)

[Figure 3: sortie photorécepteur de V-63 26](#_Toc135204966)

[Figure 4: sortie lipetz transformation de V-63 26](#_Toc135204967)

[Figure 5: sortie low pass filter de V-63 27](#_Toc135204968)

[Figure 6: sortie LMC de V-63 27](#_Toc135204969)

[Figure 7: LMC en OFF channel de V1-63 28](#_Toc135204970)

[Figure 8: LMC en ON channel de V1-63 28](#_Toc135204971)

[Figure 10: sortie OFF FDSR de V1-63 29](#_Toc135204972)

[Figure 11: Sortie ON FDSR de v1-63 29](#_Toc135204973)

[Figure 12: Sortie OFF SIGMA de V1-63 30](#_Toc135204974)

[Figure 13: Sortie ON SIGMA V1-63 30](#_Toc135204975)

[Figure 14: Sotie OFF HWR de V1-63 31](#_Toc135204976)

[Figure 15: Sortie ON HWR de V1-63 31](#_Toc135204977)

[Figure 16: sortie photorécepteur de V2-63 32](#_Toc135204978)

[Figure 17: sortie lipetz transformation de V2-63 32](#_Toc135204979)

[Figure 18: sortite low pass filter de V2-63 33](#_Toc135204980)

[Figure 19: sortie LMC de V2-63 33](#_Toc135204981)

[Figure 20: LMC en: OFF channel de V-63 34](#_Toc135204982)

[Figure 21: LMC en ON channel de V2-63 34](#_Toc135204983)

[Figure 22: sortie OFF FDSR de V2-63 35](#_Toc135204984)

[Figure 23: sortie ON FDSR de V2-63 35](#_Toc135204985)

[Figure 24: sortit OFF SIGMA de V2-63 36](#_Toc135204986)

[Figure 25: sortie ON SIGMA de V2-63 36](#_Toc135204987)

[Figure 26: sortit OFF HWR V2-63 37](#_Toc135204988)

[Figure 27: sortie ON HWR de V2-63 37](#_Toc135204989)

# Acronymes

**FDSR**: fast depolarization, slow repolarization

**HWR**: half wave rectifier

**LI**: Lateral Inhibition

**STMDs**: Small Target Motion Detector(s)

# Introduction

Dans cette section, nous allons créer une classe Python que nous appellerons STMD. Cette classe sera basée sur les différentes équations et couches des modèles STMD étudiés dans le rapport principal chap. 2. Notre tâche ici consistera à mettre en œuvre ces équations, à varier les différents paramètres pour obtenir des résultats intéressants. La classe STMD devrait permettre la détection de petits mouvements de cibles dans un arrière-plan relativement complexe et potentiellement en mouvement. Les différentes méthodes de la classe Python STMD représenteront les couches des détecteurs STMD.

Pour la suite, nous utiliserons l'éditeur ATOM pour écrire nos codes [25]; et Windows PowerShell pour les installations de packages, la gestion des environnements virtuels, et l'exécution des scripts [26]."

L'objectif étant d'évaluer les performances du modèle en fonction de la complexité de l'arrière-plan, nous allons utiliser deux vidéos pour les différents tests, chacune d'elles a été divisée en 151 images à l'aide du code [A-.]. La première vidéo montre le mouvement d’une petite cible dans un arrière-plan moins complexe et la seconde dans un arrière-plan très complexe et en mouvement.



Figure 1: Frame V1-63.:



Figure 2: Frame V2-63.

Toutes les ressources et les codes source sont disponibles sur notre compte GitHub.

<https://github.com/boukary-derra/pfe/tree/main/code>

Les sorties des différents couche se trouve dans l’annexe.

* Annexe A1 [Figure 3 à 15] pour le frame V1-63 ;
* Annexe A2 [Figure 16 à 27] pour le frame V2-16 .

Voici globalement comment se présentera le modèle :

# La méthode \_\_init\_\_()

Dans la méthode \_\_init\_\_(), nous commençons par une initialisation claire des variables et des contrôles de validité pour les entrées en utilisant np.ndarray de la bibliothèque numpy. Cette bibliothèque est un bon choix pour le traitement d'image en Python en raison de sa haute performance pour les opérations matricielles.

Ensuite nous initialisons une variété de paramètres et de constantes dans notre méthode, y compris un filtre gaussien, des paramètres pour une transformation de Lipetz, des paramètres pour un filtre passe-bas, et des constantes pour le modèle LMC.

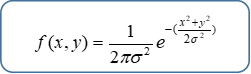


# Le noyau gaussien

La méthode get\_gaussian\_kernel()génère une matrice de taille (size, size) représentant un noyau de convolution gaussien, qui est souvent utilisé pour lisser une image.



Le noyau gaussien est généré en utilisant la formule de la densité de probabilité de la distribution normale (gaussienne), qui est : (voir le rapport principale Annexe A--):



Où

* sigma est l'écart-type de la distribution,
* x est la distance au centre du noyau, et
* e est la constante d'Euler (la base du logarithme naturel).

Une fois le noyau généré, il est normalisé en divisant chaque valeur par la somme de toutes les valeurs du noyau, de sorte que la somme de toutes les valeurs du noyau soit égale à 1. Cela garantit que l'intensité totale de l'image reste la même après le lissage.

Il est important de noter que la taille du noyau (size) doit être un nombre impair, de sorte qu'il y ait un pixel central unique.

Si une erreur se produit lors de la génération du noyau, notre méthode retourne self.w comme noyau par défaut et imprime l'erreur.

# Photorécepteur

La méthode get\_photoreceptor() applique un effet de flou à l'image d'entrée en utilisant un filtre gaussien. Et l’effet de flou ici a pour but de réduire le bruit et les détails dans chaque frame. Théoriquement, nous devons parcourir chaque pixel de l'image et appliquez le noyau gaussien aux pixels voisins (Eq. 1); cela reviendra à utiliser une boucle for imbriquée pour appliquer le filtre à l’image.



En Python, les boucles for peuvent être assez lentes, surtout pour des images de grande taille. Une approche plus efficace serait d'utiliser donc la fonction scipy.ndimage.filters.convolve de la bibliothèque scipy, qui peut effectuer une convolution 2D sur l’image avec le noyau gaussien. Cela éliminerait le besoin de boucles for imbriquées, ce qui pourrait considérablement accélérer notre code.



# Lipetz transformation

La methode get\_lipetz\_transformation()applique une transformation de Lipetz modifiée à l'image d'entrée. C'est une technique couramment utilisée en bio-inspiration, et plus précisément dans la modélisation de la réponse des photorécepteurs, pour transformer la luminance d'entrée en potentiel de membrane.

La transformation de Lipetz consiste à utiliser un filtre passe-bas sur l'image, puis à effectuer une transformation non linéaire sur le rapport entre l'image originale et l'image filtrée.



# Le filtre passe bas

La méthode get\_low\_pass\_filter() applique un filtre passe-bas à l'image transformée par Lipetz, ce qui donne un léger délai, comme nous l’avons noté en commentaire dans le code.



# LMC

La méthode get\_lmc() implémente une opération de type "Centre-surround", couramment utilisée dans les modèles de détection de mouvement pour réduire les informations redondantes et maximiser la transmission de l'information.

Dans cette méthode, nous appliquons encore un autre filtre passe-bas à l'image, puis nous soustrayons le résultat de l'image d'origine. C'est une manière courante de mettre en évidence les contours dans une image, car cela permet de soustraire la moyenne locale de chaque pixel, mettant ainsi en évidence les régions où l'intensité change rapidement (c'est-à-dire les contours).

A partir de cette couche déjà, nous commençons à avoir des résultats intéressants. Nous somme capable de détecter des petites cibles mobiles dans des arrière-plans moins complexes. L’implémentation des autres couche serait donc d’essayer de tenir compte de la complexité des font dans lesquels se déplace notre cible.



# ON / OFF channels

La méthode get\_on\_off\_channels() est utilisée pour séparer les canaux ON et OFF à partir du résultat de la couche LMC. C'est une approche courante dans les modèles de détection de mouvement bio-inspirés, où les canaux ON et OFF sont utilisés pour détecter respectivement les augmentations et les diminutions de l'intensité lumineuse.

Ici, nous effectuons cela en calculant la moitié de la somme (pour le canal ON) et la moitié de la différence (pour le canal OFF) de l'image LMC et de sa valeur absolue (Eq. 8 et Eq. 9). Cela a pour effet de séparer les valeurs positives et négatives de l'image LMC en deux images distinctes.



# FDSR

La méthode get\_fdsr()implémente un mécanisme de suppression des changements rapides de texture et d'amélioration des changements de contraste. Ce type de mécanisme est souvent utilisé dans les modèles de détection de mouvement pour filtrer les changements de texture qui ne sont pas pertinents pour la détection de mouvement.

Dans cette méthode, nous comparons les images ON et OFF actuelles avec celles du dernier cadre, et nous appliquons un flou gaussien différent en fonction de si l'intensité a augmenté ou diminué. Cela a pour effet de réduire la réponse aux changements de texture qui se produisent rapidement dans le temps, tout en améliorant la réponse aux changements de contraste.



# SIGMA

La méthode get\_sigma() calcule la différence entre le signal original (y\_on et y\_off) et le signal filtré (s\_on et s\_off). C'est une approche courante dans les modèles de détection de mouvement pour isoler les changements importants qui se sont produits entre les images.

Nous faisons cela en utilisant la fonction cv2.subtract pour soustraire les images filtrées des images originales. Cette fonction effectue une soustraction élément par élément entre les deux images, ce qui est exactement ce que nous voulons dans ce cas.

Cependant, nous devons nous assurer que les images sont du même type avant de les soustraire. C’est pour cela que nous les convertissons np.float32 avant la soustraction.



# HWR

La méthode get\_hwr() implémente une fonction de rectification demi-onde (half-wave rectification), qui est souvent utilisée dans les modèles de détection de mouvement pour supprimer les valeurs négatives.

Nous avons fait cela en créant une fonction max\_with\_zero() sur les images f\_on et f\_off. Cette fonction renvoie le maximum entre l'image et zéro, effectué pixel par pixel, ce qui aurait pour effet de remplacer toutes les valeurs négatives par zéro.



Où :



Alternativement, nous pourrions obtenir le même effet en utilisant la fonction np.maximum de numpy, qui effectue également une opération max élément par élément max\_image = np.maximum(image, 0). Cependant, l'utilisation de cv2.max est légèrement plus rapide pour de grandes images, car OpenCV est optimisé pour le traitement d'images.



# Inhibition latérale

## Block based motion estimation

La fonction block\_based\_motion\_estimation() divise l'image actuelle en blocs de taille donnée et trouve le vecteur de mouvement optimal pour chaque bloc en recherchant dans l'image précédente.



## Find optical motion vection

La fonction find\_optimal\_motion\_vector() calcule le vecteur de mouvement qui minimise la somme des différences absolues (SAD) entre le bloc actuel et chaque bloc déplacé dans l'image précédente. Cette fonction utilise une recherche exhaustive dans une plage de recherche donnée autour de la position actuelle du bloc.



En outre, l'utilisation d'une recherche exhaustive pour l'estimation du mouvement peut être très coûteuse en temps de calcul, surtout si la plage de recherche est grande. Il existe des méthodes plus efficaces pour l'estimation du mouvement basée sur des blocs, comme la recherche en spirale ou la recherche hiérarchique.



## 1.4.3. get motion components

L'extraction des composantes de mouvement à partir des vecteurs de mouvement est assez simple. Les vecteurs de mouvement sont généralement stockés dans un tableau 3D, où le premier axe correspond aux lignes de l'image, le deuxième axe correspond aux colonnes de l'image, et le troisième axe correspond aux composantes du vecteur de mouvement (généralement l'horizontale et la verticale).

La fonction get\_motion\_components() que nous avons définie fonctionne en extrayant simplement les tranches correspondantes du tableau 3D. En Python, l'opérateur « : » est utilisé pour sélectionner tous les éléments le long d'un axe. Par conséquent, motion\_vectors[:,:,0] sélectionne tous les éléments le long du premier et du deuxième axe (c'est-à-dire toutes les lignes et toutes les colonnes), et la composante 0 le long du troisième axe (c'est-à-dire la composante horizontale du vecteur de mouvement). De même, motion\_vectors[:,:,1] sélectionne la composante verticale du vecteur de mouvement.

L'implémentation de la fonction serait comme suite :



## 1.4.4. create convolution kernel and convolve uv witdh h

Ces deux fonctions créent un noyau de convolution, puis convoluent deux images (ou matrices) avec ce noyau.

La fonction create\_convolution\_kernel(p, q, a) crée un noyau de convolution de taille p par q avec la valeur a au centre et la valeur b aux bords. La valeur b est calculée comme -a / (4 \* (q - 1)) pour maintenir la somme de tous les éléments du noyau égale à zéro.

La fonction convolve\_uv\_with\_h(U, V, H) prend deux images (ou matrices), U et V, et un noyau de convolution H, puis convolue chaque image avec le noyau en utilisant la fonction convolve2d de la bibliothèque scipy. Le paramètre mode='same' indique que la sortie doit avoir la même taille que l'entrée.

Voici les implémentations de ces fonctions :





## 1.4.5. Calculate W

La fonction calculate\_w prend en entrée deux matrices U\_c et V\_c (qui peuvent être des images ou des composantes de vecteurs de mouvement), et calcule la norme euclidienne (ou distance euclidienne) pour chaque paire correspondante de points dans U\_c et V\_c. Cela se fait en élevant chaque matrice au carré, en les additionnant, puis en prenant la racine carrée de chaque élément du résultat. Le résultat w est une matrice de la même taille qu’U\_c et V\_c contenant les normes euclidiennes.

Voici l’implémentation de cette fonction :



Notons que cette fonction utilise le broadcasting de numpy pour effectuer les opérations d'élévation au carré, d'addition et de racine carrée sur chaque élément des matrices U\_c et V\_c.

Et enfin nous calculons le LI en utilisant les deux matrices F\_ON et F\_OFF (qui représentent les canaux ON et OFF d'une image), une matrice w (qui peut représente la magnitude du mouvement), et deux constantes k1 et k2. Chaque élément de F\_ON et F\_OFF est multiplié par w, puis additionné à lui-même. Nous obtenons deux nouvelles matrices F\_ON\_new et F\_OFF\_new qui sont ensuite renvoyées.

Une fois de plus, nous avons profité du broadcasting de numpy pour effectuer les opérations de multiplication et d'addition sur chaque élément des matrices F\_ON, F\_OFF, et w.



# Get\_delay and Final output

La méthode get\_delay() applique un filtre passe-bas à l'image f\_off\_li en utilisant tau\_delay comme paramètre du filtre. Si f\_off\_li n'est pas fourni, il génère une erreur.

La méthode get\_final\_output() multiplie les images f\_on\_li et lob\_off ensemble. Si l'une de ces images n'est pas fournie, il génère une erreur.



# Conclusion

# BIBLIOGRAPGIE

[1] B. Derra, T. Dalleji, “Bio-inspired small target motion detection against cluttered backgrounds using optical sensors.”, PFE, 2022-2023.

[2] Python: <https://www.python.org/>

[3] OpenCv: <https://opencv.org/>

[4] Scipy: <https://scipy.org/>

[5] Matplotlib: <https://matplotlib.org/>

[6] Numpy: <https://numpy.org/>

# Annexe

## A1. Résultat des différents couche du frame 63 de V1 ;



Figure 3: sortie photorécepteur de V-63

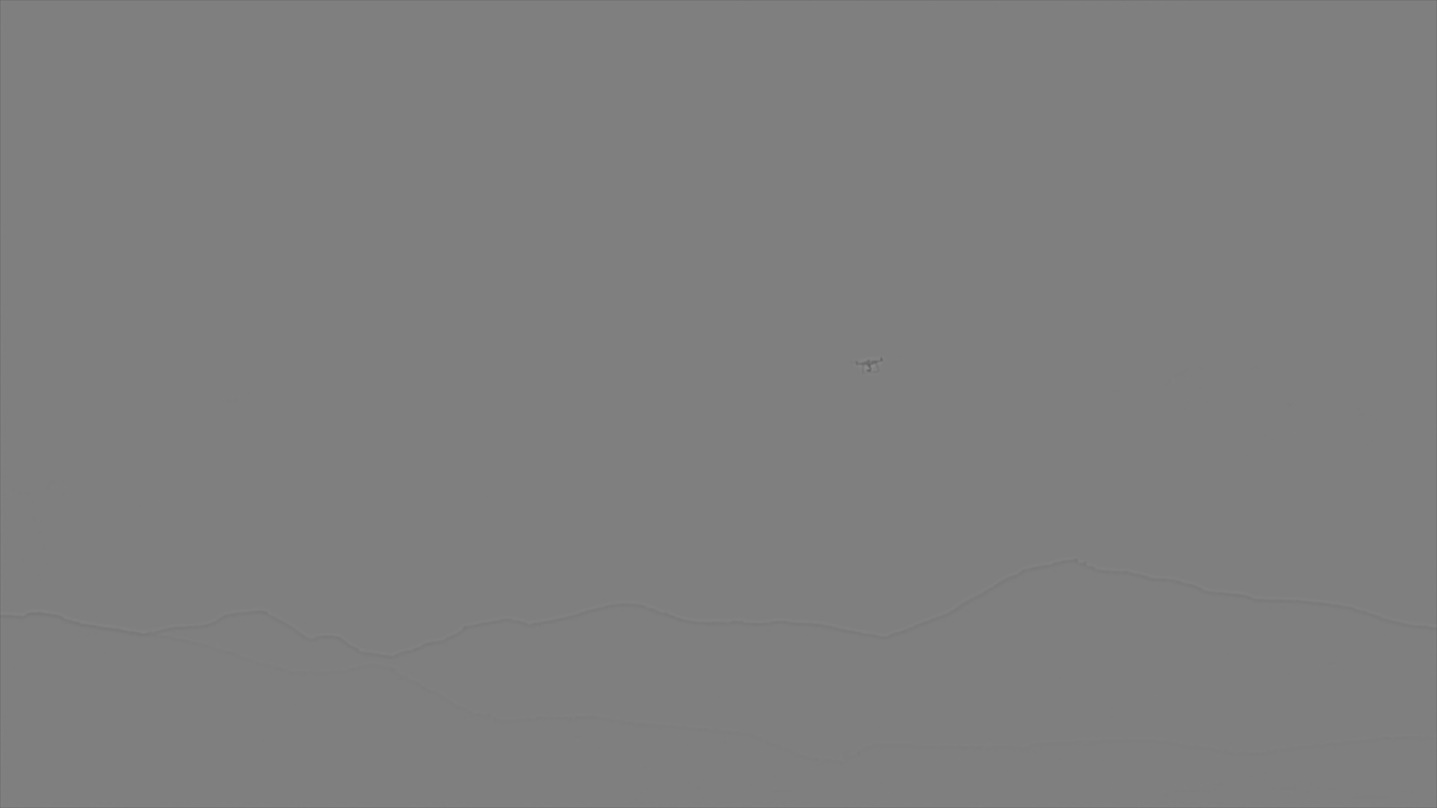


Figure 4: sortie lipetz transformation de V-63



Figure 5: sortie low pass filter de V-63

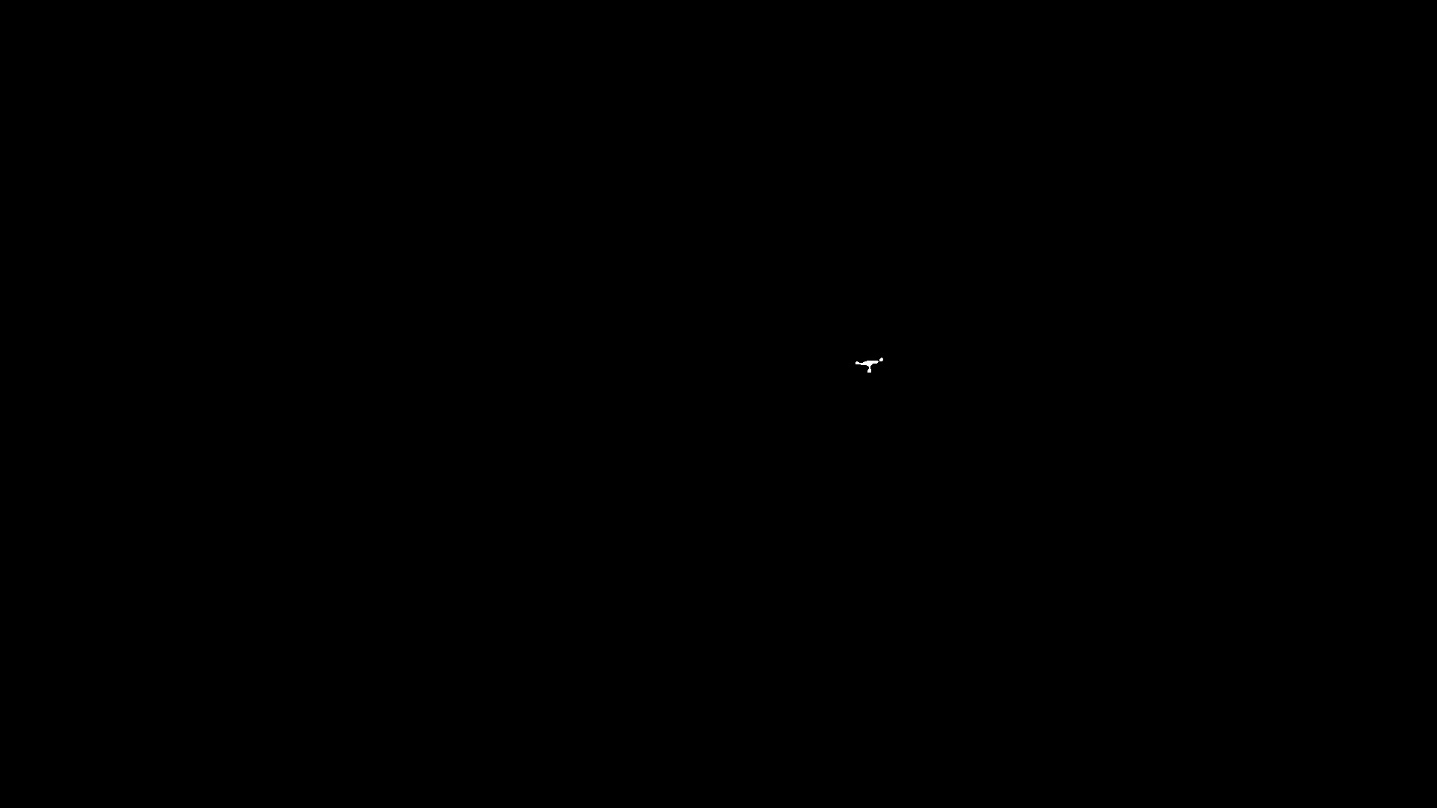


Figure 6: sortie LMC de V-63

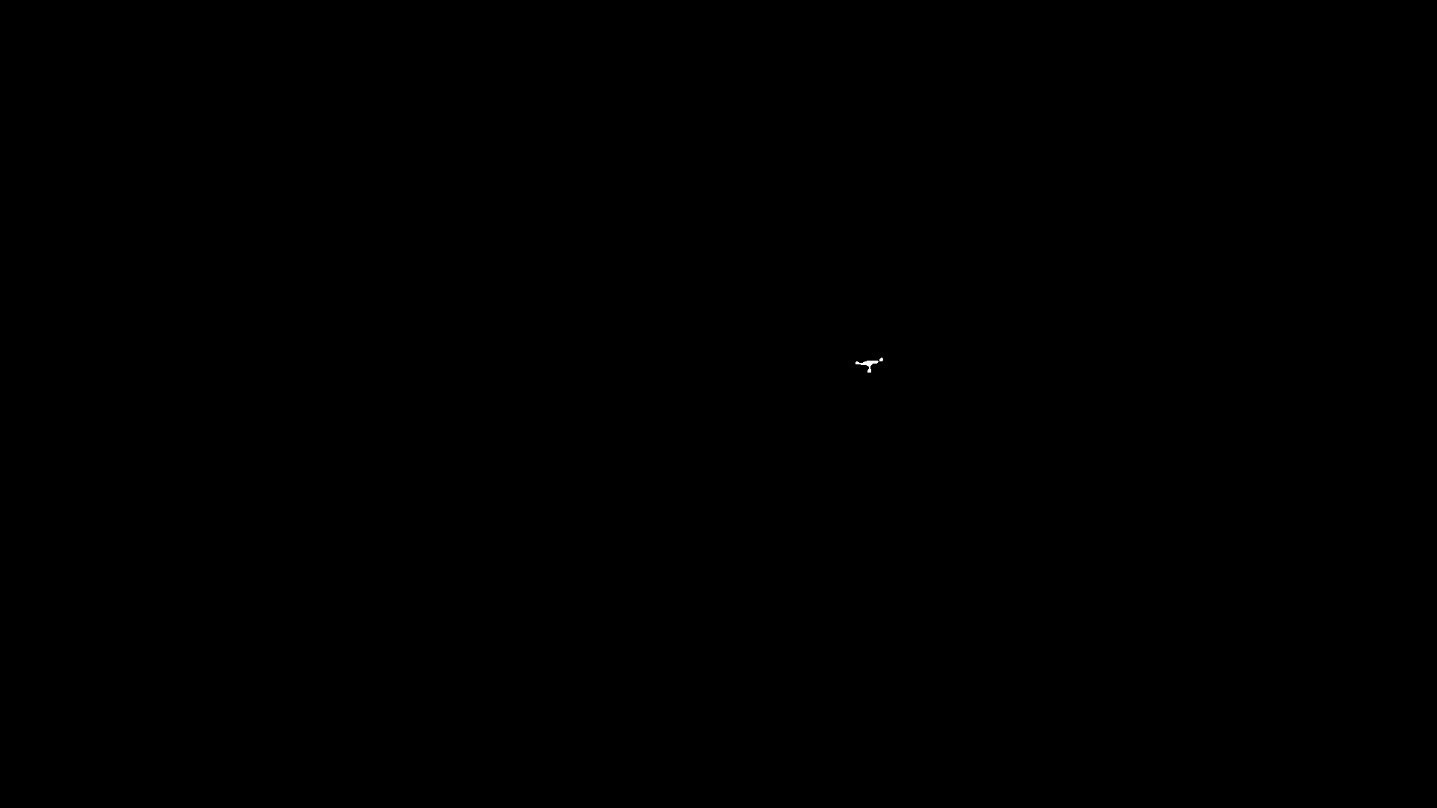


Figure 7: LMC en OFF channel de V1-63

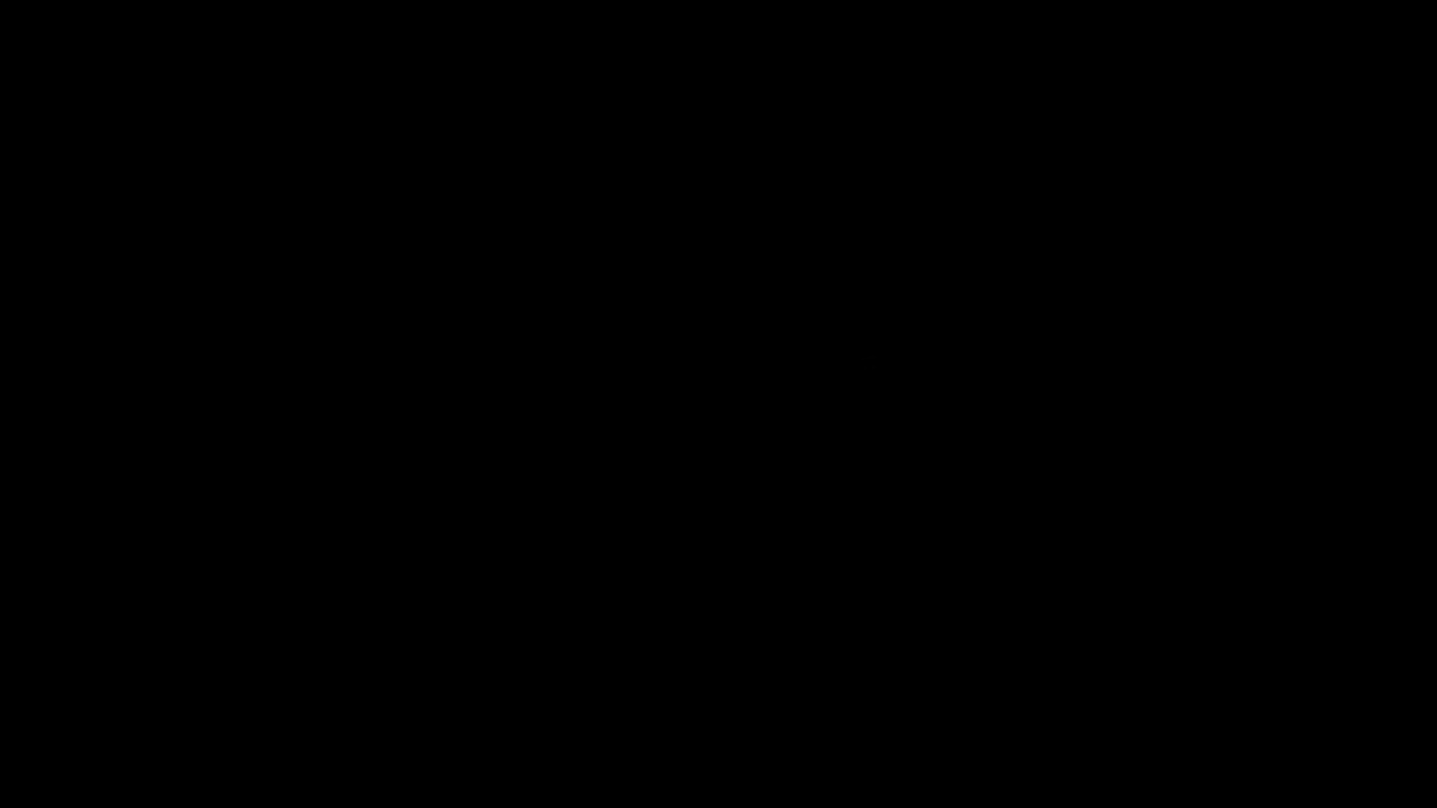


Figure 8: LMC en ON channel de V1-63

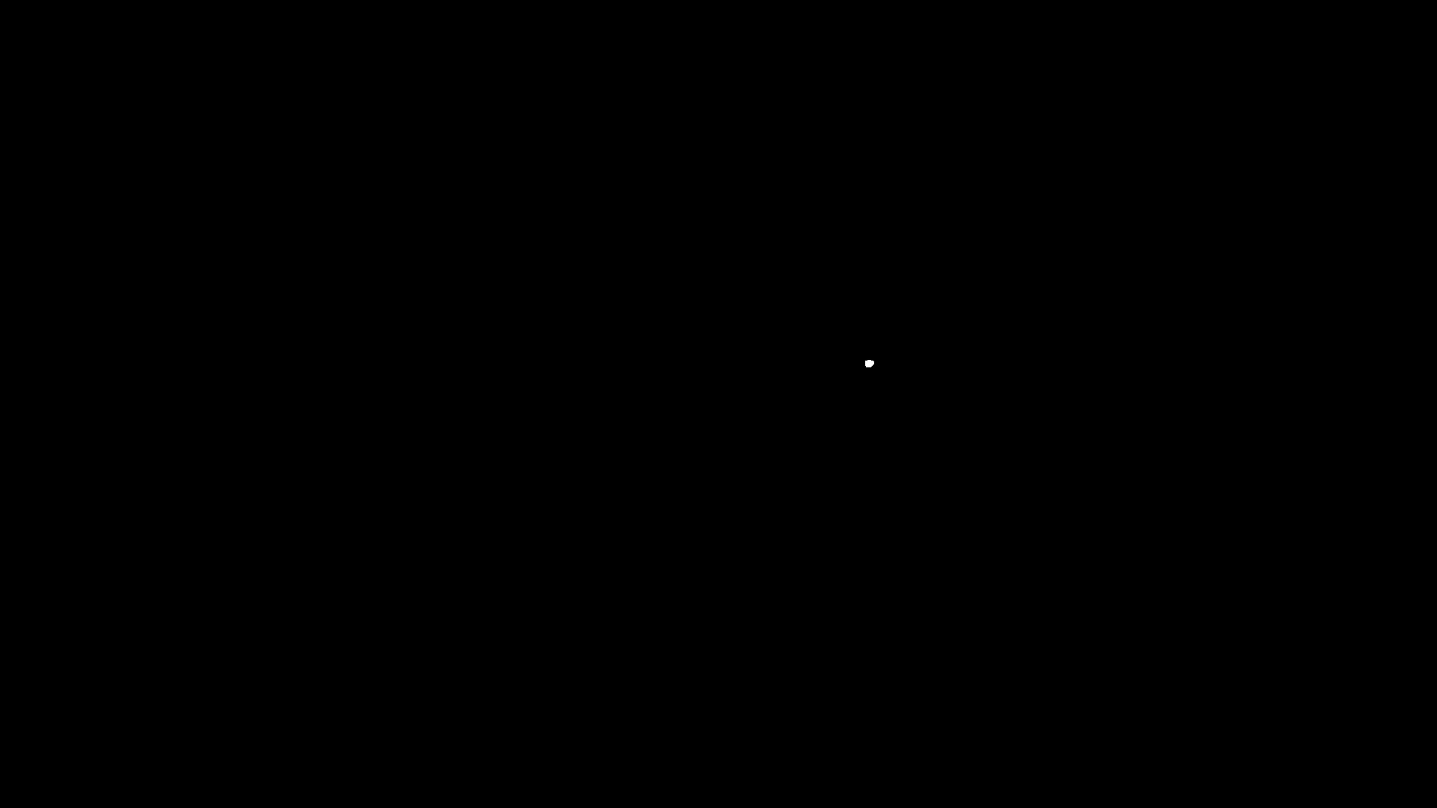


Figure 10: sortie OFF FDSR de V1-63

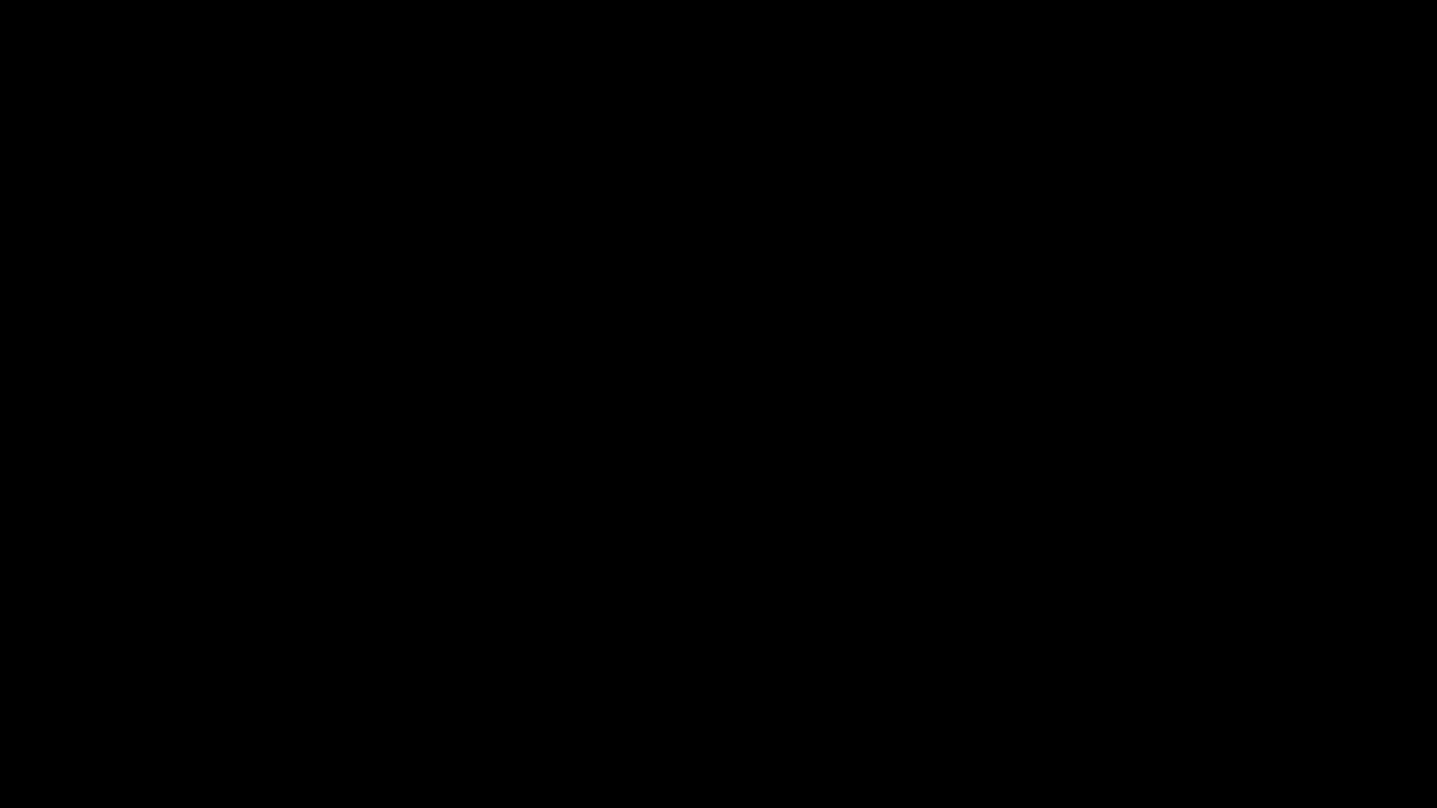


Figure 11: Sortie ON FDSR de v1-63

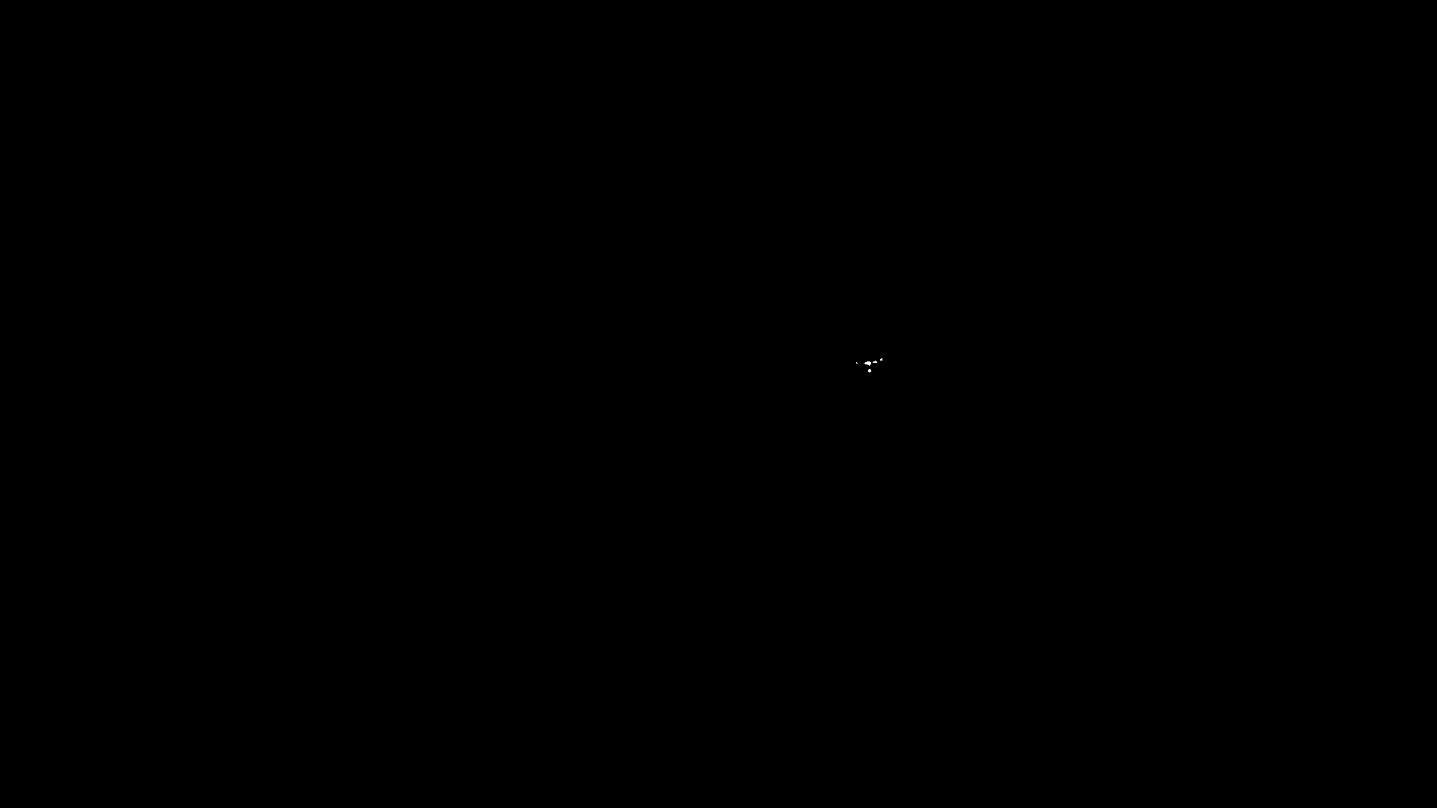


Figure 12: Sortie OFF SIGMA de V1-63

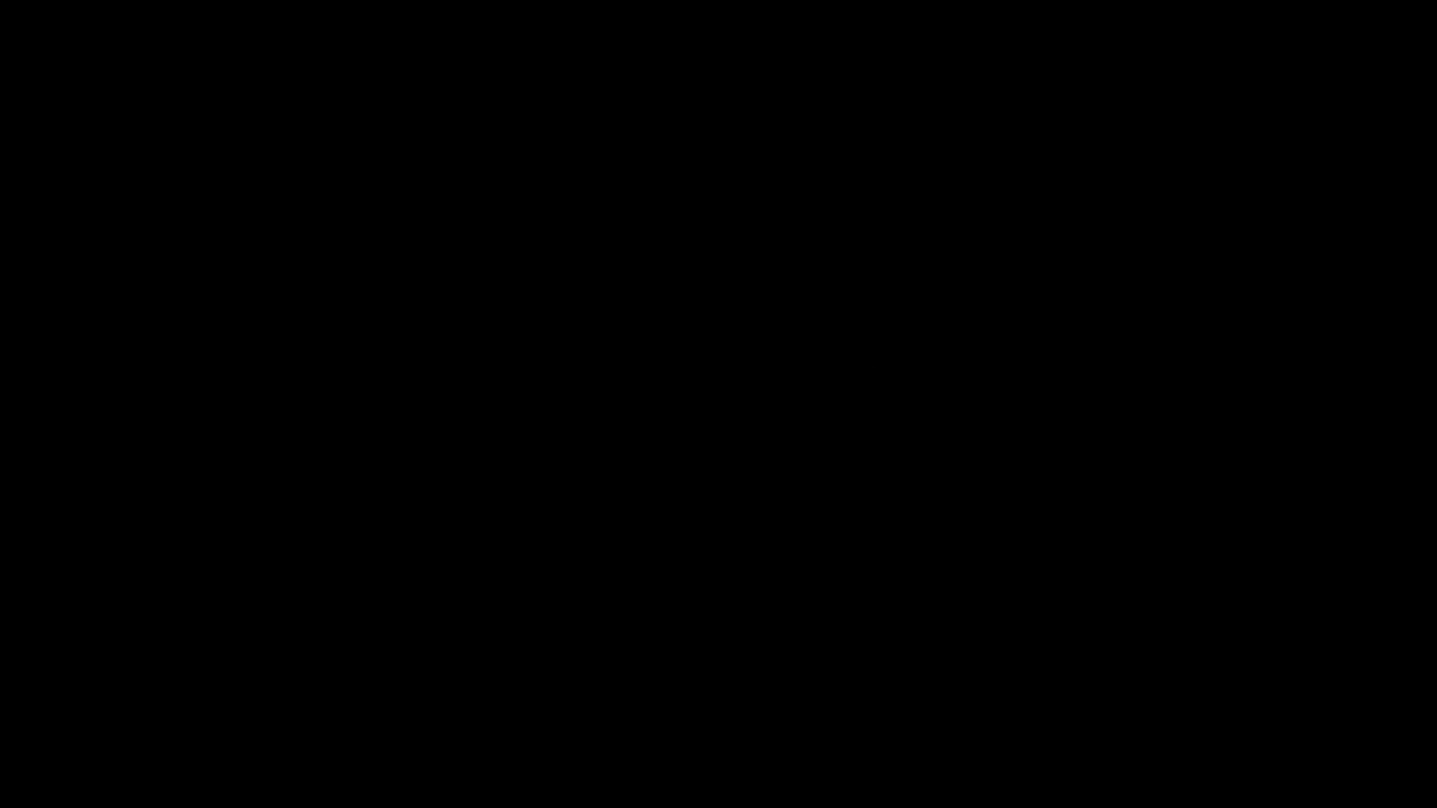


Figure 13: Sortie ON SIGMA V1-63

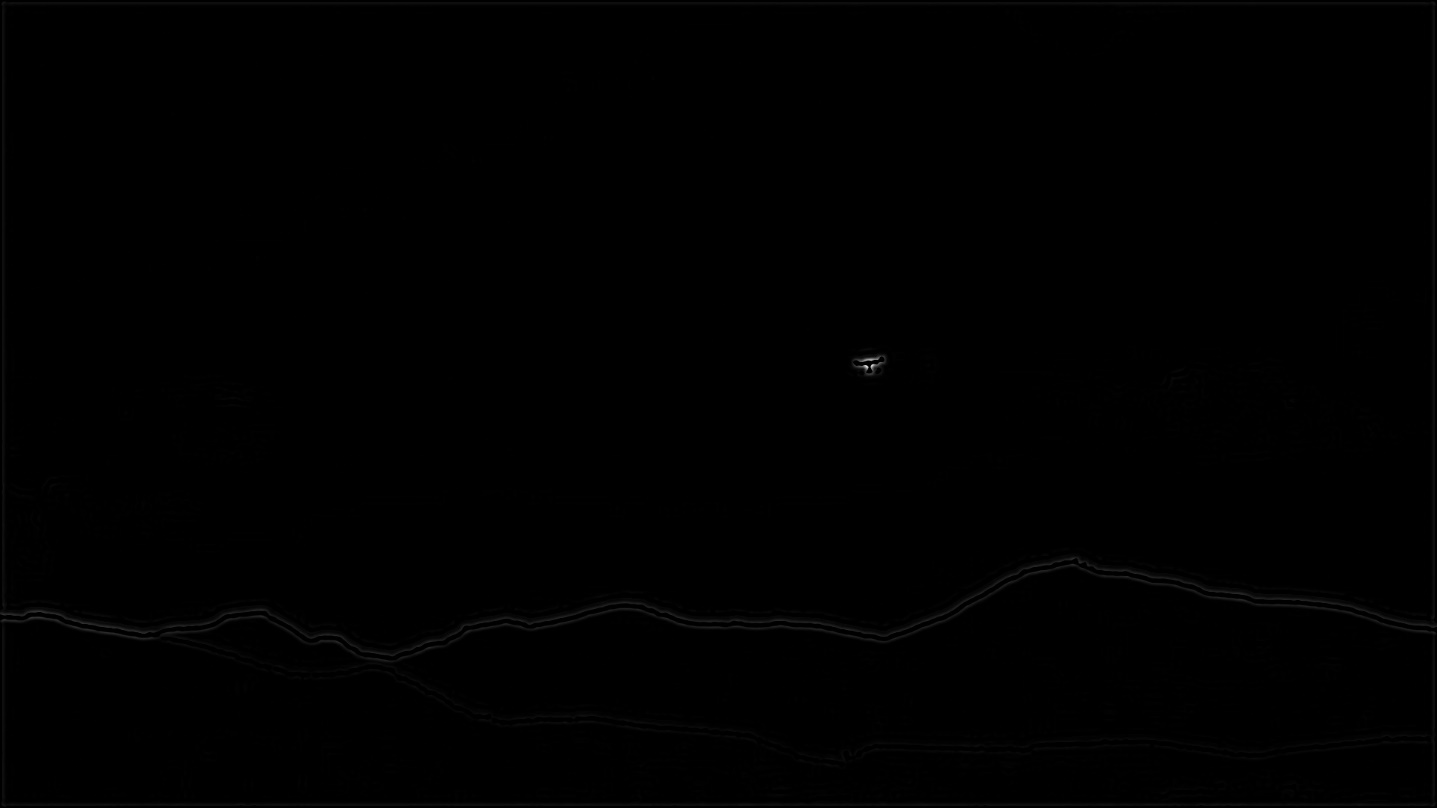


Figure 14: Sotie OFF HWR de V1-63

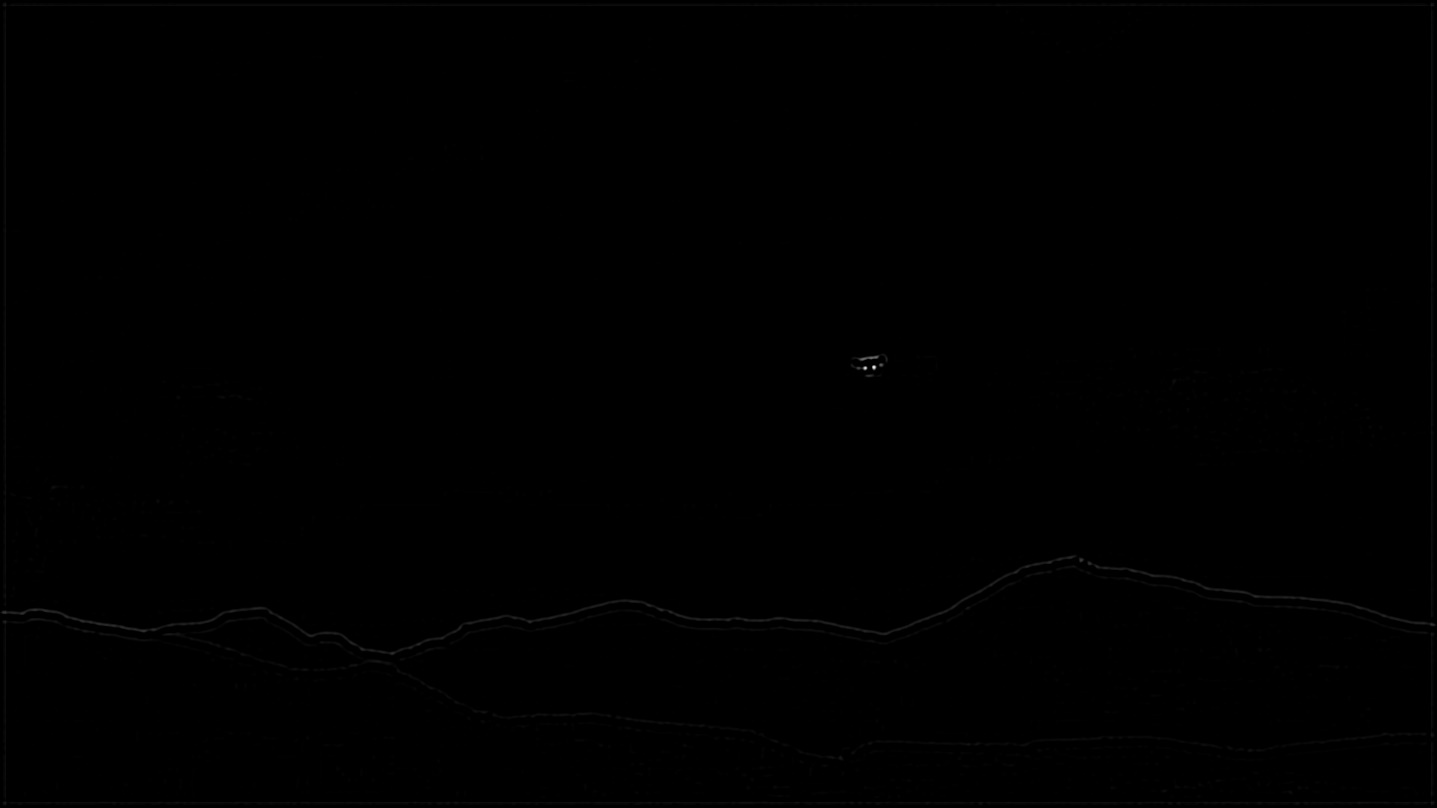


Figure 15: Sortie ON HWR de V1-63

## A2. Résultat des différents couche du frame 63 de v2;



Figure 16: sortie photorécepteur de V2-63



Figure 17: sortie lipetz transformation de V2-63



Figure 18: sortite low pass filter de V2-63



Figure 19: sortie LMC de V2-63



Figure 20: LMC en: OFF channel de V-63

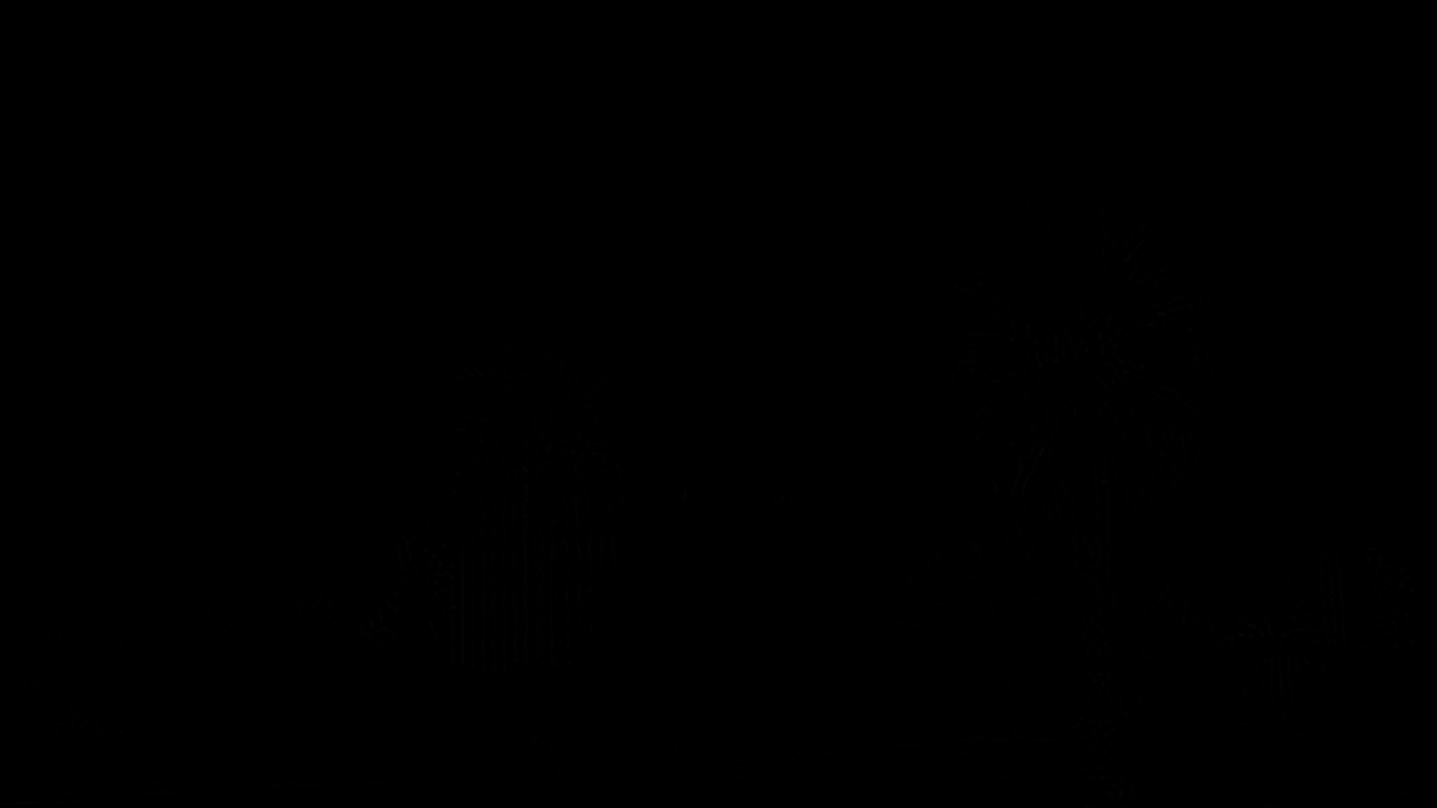


Figure 21: LMC en ON channel de V2-63



Figure 22: sortie OFF FDSR de V2-63



Figure 23: sortie ON FDSR de V2-63



Figure 24: sortit OFF SIGMA de V2-63



Figure 25: sortie ON SIGMA de V2-63



Figure 26: sortit OFF HWR V2-63



Figure 27: sortie ON HWR de V2-63

## A3. Paquages et Installations

### A.3.1. Python

Python est un langage de programmation polyvalent et convivial, largement utilisé dans le domaine du traitement d'images et de la vision par ordinateur. Sa syntaxe simple et expressive en fait un choix idéal pour développer des algorithmes de détection du mouvement de petites cibles. Dans mon projet, j'ai utilisé Python pour écrire le code principal, gérer les flux d'images, manipuler les données et interagir avec les autres bibliothèques. Python dispose également d'une vaste communauté de développeurs qui fournissent des ressources et des packages supplémentaires, ce qui facilite la mise en œuvre de solutions de traitement d'images avancées. Pour installer Python, vous pouvez télécharger la dernière version depuis le site officiel de Python (<https://www.python.org/>) et suivre les instructions d'installation.



### A.3.2. OpenCV

OpenCV, abréviation de « Open Source Computer Vision », est une bibliothèque populaire et puissante utilisée pour le traitement d'images et la vision par ordinateur. Elle offre un large éventail de fonctionnalités pour manipuler et analyser des images, y compris la détection du mouvement. Dans mon projet, j'ai utilisé la bibliothèque OpenCV (cv2) pour capturer les images en temps réel, extraire les contours et effectuer l'analyse du mouvement des petites cibles. OpenCV m'a offert des outils robustes pour traiter les images, notamment pour l'extraction de caractéristiques et la segmentation. Pour installer OpenCV, vous pouvez utiliser la commande pip install opencv-python dans votre environnement Python.



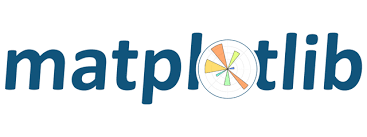
### A.3.3. Scipy

SciPy est une bibliothèque puissante et polyvalente utilisée pour le calcul scientifique et l'analyse de données en Python. Elle fournit des fonctionnalités avancées pour l'optimisation, l'algèbre linéaire, l'intégration, la statistique et bien plus encore. Dans mon projet, j'ai utilisé SciPy pour effectuer des opérations mathématiques complexes, telles que la transformation de Fourier discrète (DFT), qui est utile pour analyser les signaux de mouvement dans les images. SciPy m'a également permis d'exploiter des fonctionnalités statistiques pour analyser les données de mouvement. Pour installer SciPy, vous pouvez utiliser la commande pip install scipy dans votre environnement Python.



### A.3.4. Matplotlib

Matplotlib est une bibliothèque de visualisation de données en Python. Elle permet de créer des graphiques, des diagrammes, des histogrammes et des visualisations personnalisées pour analyser et présenter les résultats. Dans mon projet, j'ai utilisé Matplotlib pour afficher les images capturées en temps réel avec les contours détectés et les zones de mouvement identifiées. Cette visualisation m'a aidé à comprendre et à valider les résultats de détection du mouvement. Pour installer Matplotlib, vous pouvez utiliser la commande pip install matplotlib dans votre environnement Python.



### A.3.5. Numpy

NumPy est une bibliothèque fondamentale pour le calcul scientifique en Python. Elle offre des structures de données efficaces pour la manipulation de tableaux multidimensionnels, ainsi que des fonctions mathématiques de haut niveau pour effectuer des opérations sur ces tableaux. Dans mon projet, j'ai utilisé NumPy pour manipuler les images capturées, extraire des régions d'intérêt, effectuer des opérations matricielles et analyser les données de mouvement. Les performances optimisées de NumPy permettent d'accélérer les calculs et de gérer efficacement les grands ensembles de données. Pour installer NumPy, vous pouvez utiliser la commande pip install numpy dans votre environnement Python. Assurez-vous d'avoir Python préalablement installé avant d'installer NumPy.



### A.3.6. ATOM

ATOM est un éditeur de code open-source, léger et personnalisable, qui offre une expérience de développement agréable pour les programmeurs Python. J'ai utilisé ATOM pour écrire et organiser mon code Python de détection du mouvement. L'éditeur offre des fonctionnalités pratiques telles que la coloration syntaxique, la numérotation des lignes, le pliage du code et l'intégration avec Git. Pour installer ATOM, vous pouvez visiter le site officiel d'ATOM (<https://atom.io/>) et télécharger le package d'installation correspondant à votre système d'exploitation.



### A.3.7. Windows PowerShell

Windows PowerShell est un shell de ligne de commande et un langage de script développé par Microsoft, intégré aux systèmes d'exploitation Windows modernes. J'ai utilisé Windows PowerShell pour exécuter mes scripts Python et effectuer des tâches de gestion des fichiers, de navigation dans le système de fichiers et d'automatisation des tâches. PowerShell offre une grande flexibilité et une grande puissance pour interagir avec le système d'exploitation Windows. Il est préinstallé sur les systèmes Windows récents, donc aucune installation supplémentaire n'est nécessaire.



### A.3.8. Environnement virtuel «(venv )

Un environnement virtuel est un outil essentiel pour isoler et gérer les dépendances spécifiques d'un projet Python. Il permet de créer un environnement de développement autonome où vous pouvez installer des packages et des bibliothèques sans interférer avec d'autres projets Python sur votre système. Dans mon projet de traitement d'images, j'ai utilisé un environnement virtuel (venv) pour maintenir une configuration propre et spécifique à mon projet. Cela m'a permis de gérer les dépendances requises, d'installer des bibliothèques tierces comme OpenCV, NumPy et autres, sans risquer de conflits avec d'autres projets ou systèmes. Pour créer un environnement virtuel avec venv, vous pouvez exécuter les commandes suivantes dans votre invite de commandes ou votre terminal : python -m venv mon\_env.

Cela créera un nouvel environnement virtuel nommé mon\_env dans le répertoire actuel. Ensuite, pour activer l'environnement virtuel, vous devez exécuter la commande correspondante en fonction de votre système d'exploitation :

* Sur Windows : mon\_env\Scripts\activate
* Sur Linux/MacOS : source mon\_env/bin/activate

Une fois l'environnement virtuel activé, vous pouvez installer les packages spécifiques à votre projet en utilisant la commande pip, sans affecter votre système global.

**Avantages de l'environnement virtuel (venv) :**

L'utilisation d'un environnement virtuel présente plusieurs avantages. Tout d'abord, il vous permet de maintenir un environnement de développement propre et bien organisé, en isolant les dépendances de chaque projet. Cela facilite la gestion des versions des packages et garantit que votre projet utilise les versions spécifiques dont il a besoin. De plus, l'environnement virtuel offre une portabilité, vous permettant de partager facilement votre projet avec d'autres développeurs sans vous soucier des conflits de dépendances. Enfin, l'utilisation de venv facilite également le processus de déploiement en fournissant un moyen reproductible de créer et de gérer l'environnement de production de votre projet.

**Installation de venv :**

L'environnement virtuel (venv) est une fonctionnalité intégrée à Python à partir de la version 3.3. Par conséquent, il est généralement inclus dans l'installation standard de Python. Pour créer un environnement virtuel, vous devez exécuter la commande python -m venv nom\_de\_l\_environnement dans votre invite de commandes ou votre terminal, comme indiqué précédemment. Assurez-vous d'avoir Python installé correctement sur votre système avant de créer un environnement virtuel.

### A.3.9. Git et GitHub

**Git :**

Git est un système de contrôle de version décentralisé, largement utilisé dans le développement de logiciels pour gérer les modifications du code source. Il permet de suivre l'historique des modifications, de collaborer avec d'autres développeurs, de gérer les branches de développement et de fusionner les modifications de manière efficace. Dans mon projet de traitement d'images, j'ai utilisé Git pour suivre et enregistrer les différentes versions de mon code, ce qui m'a permis de revenir à des versions antérieures en cas de besoin et de travailler en collaboration avec mon encadrant. Pour installer Git, vous pouvez visiter le site officiel de Git (<https://git-scm.com/>) et télécharger le package d'installation correspondant à votre système d'exploitation. Suivez les instructions d'installation fournies pour configurer Git sur votre machine. Après l'installation, vous pouvez vérifier si Git est correctement installé en exécutant la commande git --version dans votre invite de commandes ou votre terminal. Assurez-vous d'avoir une connexion Internet active pour télécharger les fichiers d'installation et configurer Git correctement.

**GitHub :**

GitHub est une plateforme d'hébergement de code source basée sur Git. Elle offre des fonctionnalités supplémentaires telles que le suivi des problèmes, la gestion des projets et la collaboration en équipe. Dans mon projet, j'ai utilisé GitHub pour héberger mon code et le rendre accessible à d'autres développeurs. Cela m'a permis de partager facilement mon travail, de recevoir des commentaires et des contributions, et de synchroniser les mises à jour entre différents environnements de développement. Pour utiliser GitHub, vous devez créer un compte sur le site officiel de GitHub (<https://github.com/>) et configurer un dépôt pour votre projet. Vous pouvez ensuite utiliser des commandes Git telles que **git clone**, **git commit** et **git push** pour interagir avec votre dépôt distant et gérer les versions de votre code.

**Avantages de Git et GitHub :**

L'utilisation de Git et GitHub présente de nombreux avantages pour la gestion de projet et la collaboration. Tout d'abord, Git permet de garder une trace précise des modifications apportées au code, ce qui facilite la gestion des versions et le suivi des erreurs. De plus, Git facilite la collaboration en permettant à plusieurs développeurs de travailler simultanément sur différentes branches de développement et de fusionner les modifications de manière contrôlée. GitHub ajoute une couche supplémentaire de fonctionnalités en fournissant un espace centralisé pour héberger et partager le code, ainsi que des outils de gestion de projet pour suivre les problèmes, les demandes de fusion et les tâches. L'intégration de Git et GitHub dans le flux de travail de développement offre une meilleure traçabilité, une collaboration efficace et une facilité de déploiement.



Toutes les ressources et les codes source du projet sont disponibles sur notre compte GitHub.

<https://github.com/boukary-derra/pfe/tree/main/code>