Année universitiare : 2024/2025

Une image contenant texte, Police, capture d’écran, Graphique

Le contenu généré par l’IA peut être incorrect.UNIVERSITE PRIVEE DE FES

Style transfer en temps réel sur des vidéos

**Sous l’encadrement de**

Mr. Tarik HAJJI

**Élaboré par :** Ali BOUZIANE

By

TRANSFERT DE STYLE NEURONAL À L'AIDE DE DEEP LEARNING

**Module :** Deep Learning

**Résumé**

Au fil des siècles, l'homme est devenu expert dans la création de stimulations visuelles uniques, grâce à la richesse et à la complexité des techniques d'interaction pour restituer du contenu basé sur le style artistique. Si de nombreux algorithmes exploitent l'intelligence artificielle, et plus précisément les réseaux neuronaux profonds, pour générer de l'art, la majorité des recherches se concentrent sur des applications statiques comme les images.

Dans ce rapport, nous proposons une approche innovante en concevant un algorithme de **transfert de style neuronal appliqué à la vidéo en temps réel via une webcam**. En nous appuyant sur des modèles existants et après l'analyse de trois publications notables, nous adaptons les techniques de stylisation d'images pour les rendre efficaces sur un flux vidéo dynamique.

L'objectif est d'offrir une preuve de concept fonctionnelle, en garantissant une cohérence temporelle et une latence minimale, afin de transformer instantanément la capture vidéo en une œuvre artistique animée.

I. INTRODUCTION

Les débuts de l'art ont conduit à un savoir-faire perfectionné par l'humanité au fil des siècles. La diversité artistique a été limitée par l'imagination humaine. La diffusion des technologies, grâce à l'intelligence artificielle, dans des applications non conventionnelles a eu un impact considérable sur notre perception de l'évolution, voire de l'amélioration, de médiums comme l'art.

Notre projet vise à transférer le style de vidéos en temps réel à l'aide d'outils tels qu'OpenCV, Python et l'apprentissage profond.

Qu'est-ce que le transfert de style ?

Le transfert de style neuronal est une technique d'optimisation qui consiste à fusionner deux images, une image d'entrée et une image de style (comme une œuvre d'un peintre célèbre), afin de transformer l'image d'entrée pour qu'elle ressemble à l'image de style.

II. Neural Style Transfer ALGORITHMS

L'origine des algorithmes de transfert de style neuronal peut être citée dans l'article de Gatys et al. de 2015 intitulé « A Neural Algorithm of Artistic Style ». Le fonctionnement du transfert de style neuronal est le suivant :

*Figure 1: working*

L’article fondateur de **Gatys et al.** a introduit une approche novatrice du transfert de style neuronal (NST) en exploitant un réseau pré-entraîné (comme ImageNet) sans nécessiter d’architecture spécifique. Leur méthode reposait sur la définition d’une **fonction de perte composite**, optimisée itérativement pour générer une image stylisée. Cette fonction combinait trois composantes clés :

1. **Content loss,**
2. **Style Loss and**
3. **Total - Variation Loss.**

En minimisant cette méta-fonction, l’algorithme parvenait à synthétiser des résultats esthétiques, mais au prix d’un **temps de calcul prohibitif** en raison de l’optimisation itérative requise pour chaque nouvelle paire contenu/style.

Pour pallier cette lenteur, **Johnson et al.** ont proposé dans *« Perceptual Losses for Real-Time Style Transfer and Super-Resolution »* une approche **trois fois plus rapide**, inspirée des travaux sur la super-résolution. Leur méthode reposait sur l’entraînement d’un réseau dédié à un style spécifique, éliminant ainsi la nécessité d’optimiser chaque image. Cependant, cette rapidité s’accompagnait d’une **limitation majeure** : le réseau, une fois entraîné, ne pouvait appliquer qu’un seul style prédéfini. Bien que scalable (plusieurs styles pouvaient être pré-entraînés), cette rigidité réduisait la flexibilité artistique et augmentait la complexité logistique pour des applications multi-styles.

Enfin, **Ulyanov et al.** ont révolutionné le domaine avec leur article *« Instance Normalization: The Missing Ingredient for Fast Stylization »*. Leur innovation clé fut le remplacement de la **normalisation par lots** (batch normalization) par la **normalisation d’instance**, tant pendant l’entraînement qu’à l’inférence. Cette simple modification a permis d’atteindre des **performances en temps réel** tout en améliorant la qualité visuelle des sorties. Leur approche, combinée aux avancées précédentes, a servi de fondement à notre méthode, qui vise à adapter ces principes au transfert de style pour vidéos.

III. Projet

Nous commençons par appliquer le transfert de style à une vidéo (en temps réel via webcam ou depuis un fichier). Les étapes sont les suivantes :

**Étape 1 :** Import des packages et configuration des arguments.

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Figure 2 : Code pour l'import des packages et la configuration des arguments

**Étape 2 :** Chargement du modèle et initialisation de la vidéo.

Le code commence par charger les modèles de style (.**t7**) depuis le dossier spécifié et initialise un itérateur cyclique pour alterner entre eux. Le premier modèle est chargé dans le réseau neuronal avec activation de l'accélération **GPU** via **OpenCL** (**DNN\_BACKEND\_DEFAULT** et **DNN\_TARGET\_OPENCL**). La source vidéo est ensuite configurée - webcam par défaut (**cv2.VideoCapture(0)**) ou fichier vidéo si spécifié - avec une vérification d'accessibilité qui interrompt le programme en cas d'échec, garantissant ainsi une initialisation robuste des composants clés avant le traitement.

Une image contenant texte, capture d’écran, logiciel

Le contenu généré par l’IA peut être incorrect.

Figure 3 : Code pour le chargement du modèle et l'initialisation de la vidéo

**Étape 3 :** Traitement des frames et optimisation des performances.

Le pipeline capture chaque frame vidéo en continu, en appliquant un traitement une frame sur deux pour optimiser les performances. Chaque image est redimensionnée à 400px de large (conservant l'aspect ratio) et convertie en blob via **cv2.dnn.blobFromImage** avec normalisation des valeurs RGB. Le transfert de style s'exécute alors sur le blob d'entrée, suivi d'un post-traitement comprenant : restructuration du tenseur de sortie, rétablissement des canaux couleur, normalisation et conversion en format d'affichage (uint8 0-255). Les résultats sont présentés en temps réel avec côte-à-côte l'original et la version stylisée, tandis que le temps de traitement par frame est logué dans la console. L'utilisateur peut interrompre le flux avec **'q'** ou alterner entre modèles avec **'n'**, déclenchant le rechargement dynamique du prochain modèle tout en conservant le contexte vidéo.

Une image contenant texte, capture d’écran, logiciel

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, logiciel

Le contenu généré par l’IA peut être incorrect.Figure 4 : Code pour la boucle de traitement des frames

**Étape 4 :** Gestion des commandes utilisateur

Une image contenant texte, capture d’écran, Police, logiciel

Le contenu généré par l’IA peut être incorrect.

Figure 5 : Code pour la gestion des cmd

**Étape 5 :** Nettoyage des ressources

Libération des ressources vidéo et fermeture des fenêtres.



Figure 6 : Code pour la libération des ressources

La structure de commande suivante doit être transmise au terminal pour collecter les résultats

Pour lancer le script avec un modèle et une webcam :

python script.py --models models/ instance\_norm

Ou avec un fichier vidéo :

python script.py --models models/udnie.t7 --video input/video.mp4

Le terminal affiche le temps de traitement pour chaque frame, qui varie selon la complexité du modèle CNN utilisé. Après avoir validé la méthode sur des images, nous l'adaptons pour la vidéo en temps réel via webcam. La procédure reprend les mêmes principes tout en intégrant des optimisations spécifiques :

1. **Sélection dynamique des modèles**  
   Un itérateur permet de changer de modèle à la volée (touche "n") sans interrompre le flux.
2. **Capture vidéo optimisée**  
   Le flux est initialisé en priorisant la webcam, ou un fichier vidéo si spécifié. Des contrôles garantissent que la source est valide.
3. **Traitement adaptatif**  
   Chaque frame est :
   * Redimensionnée (400px de large) pour accélérer le traitement
   * Convertie en blob avec normalisation des couleurs (moyennes BGR spécifiques)
   * Traitée par le réseau avec mesure précise du temps d'exécution
4. **Affichage intelligent**  
   Les résultats sont post-traités (reconstruction RGB, clipping des valeurs) et affichés côte à côte avec l'original.

**Les résultats du NST en temps réel sont présentés ci-dessous.**

Pour un fichier vidéo :

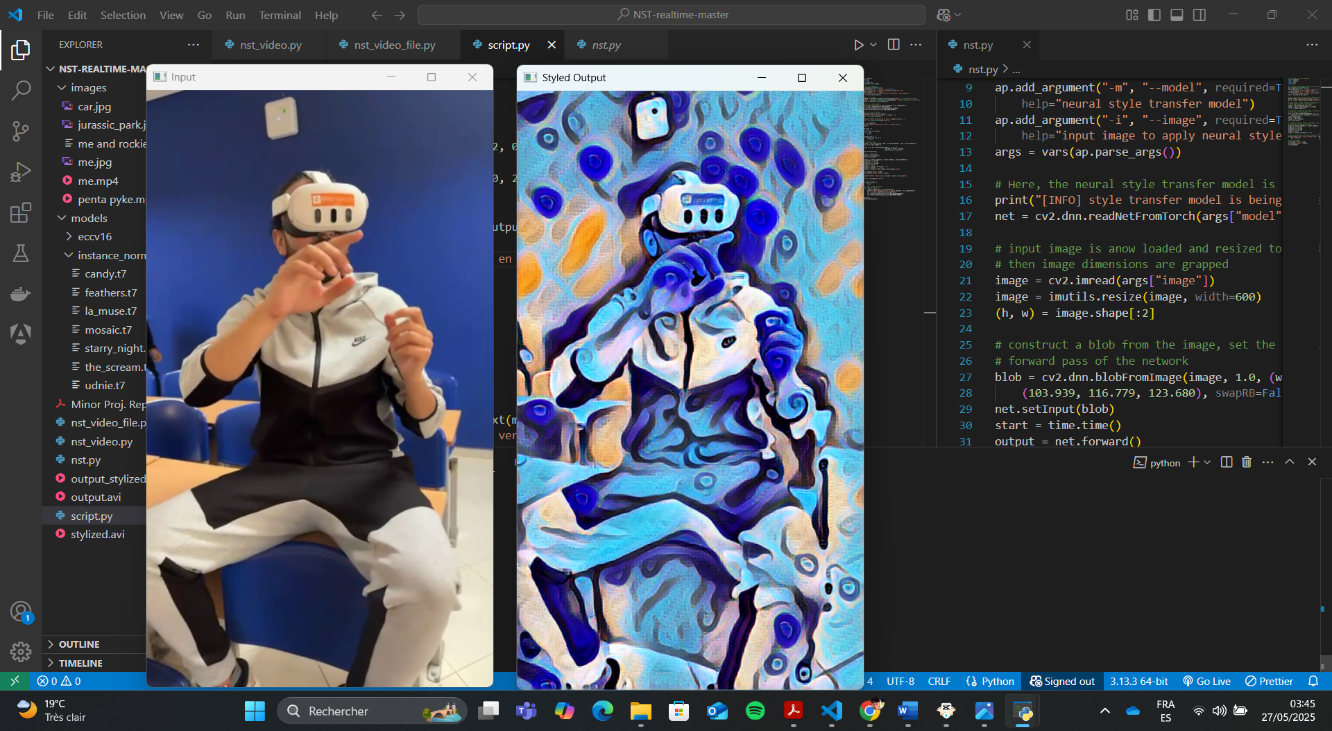


Figure 7 : Output après transfert de style neuronal

Pour un modèle et une webcam :

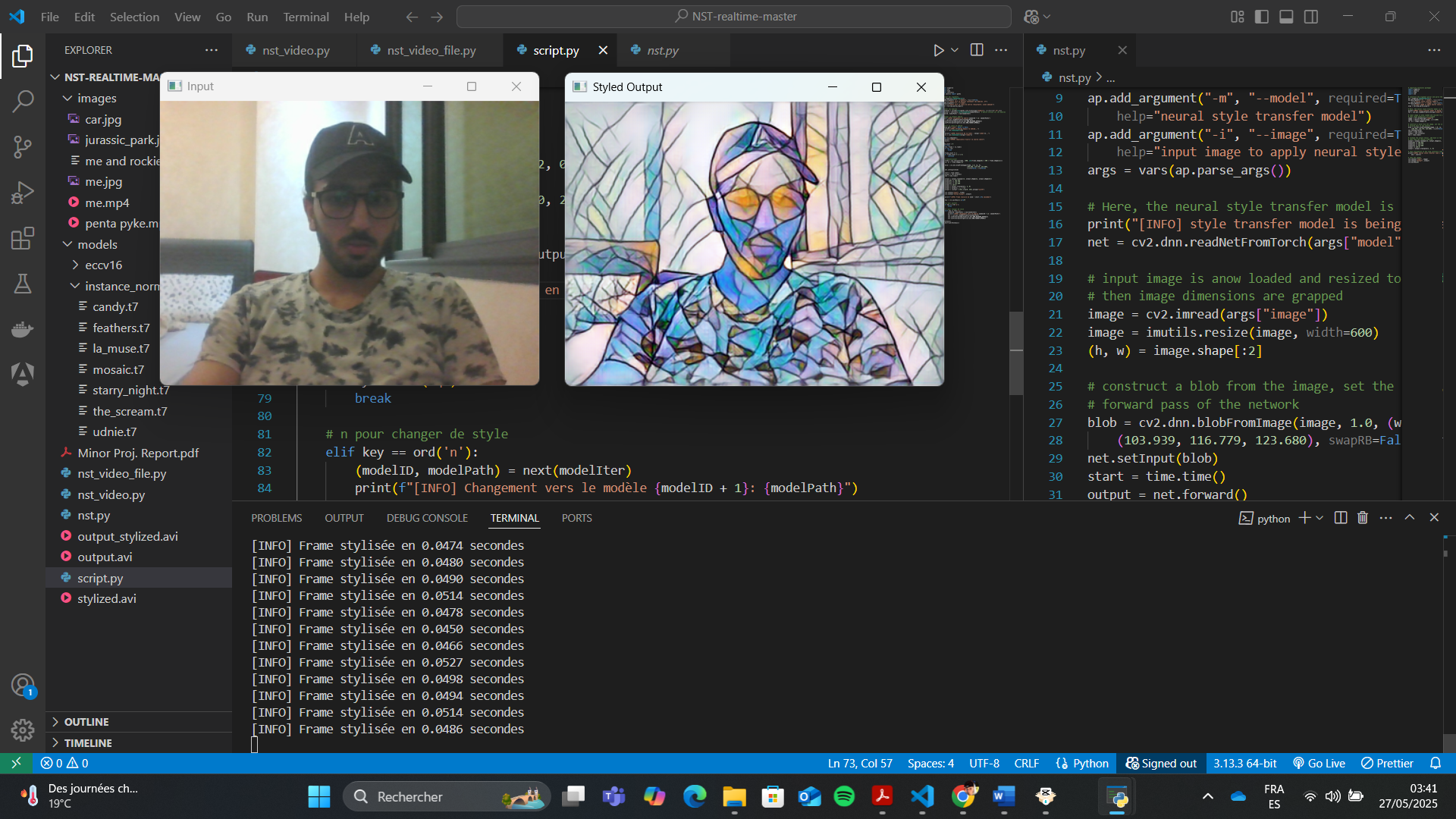


Figure 8 : Output transfert de style neuronal

Une image contenant texte, Logiciel multimédia, capture d’écran, logiciel

Le contenu généré par l’IA peut être incorrect.

Figure 9 : Output après avoir appuyé sur « n »

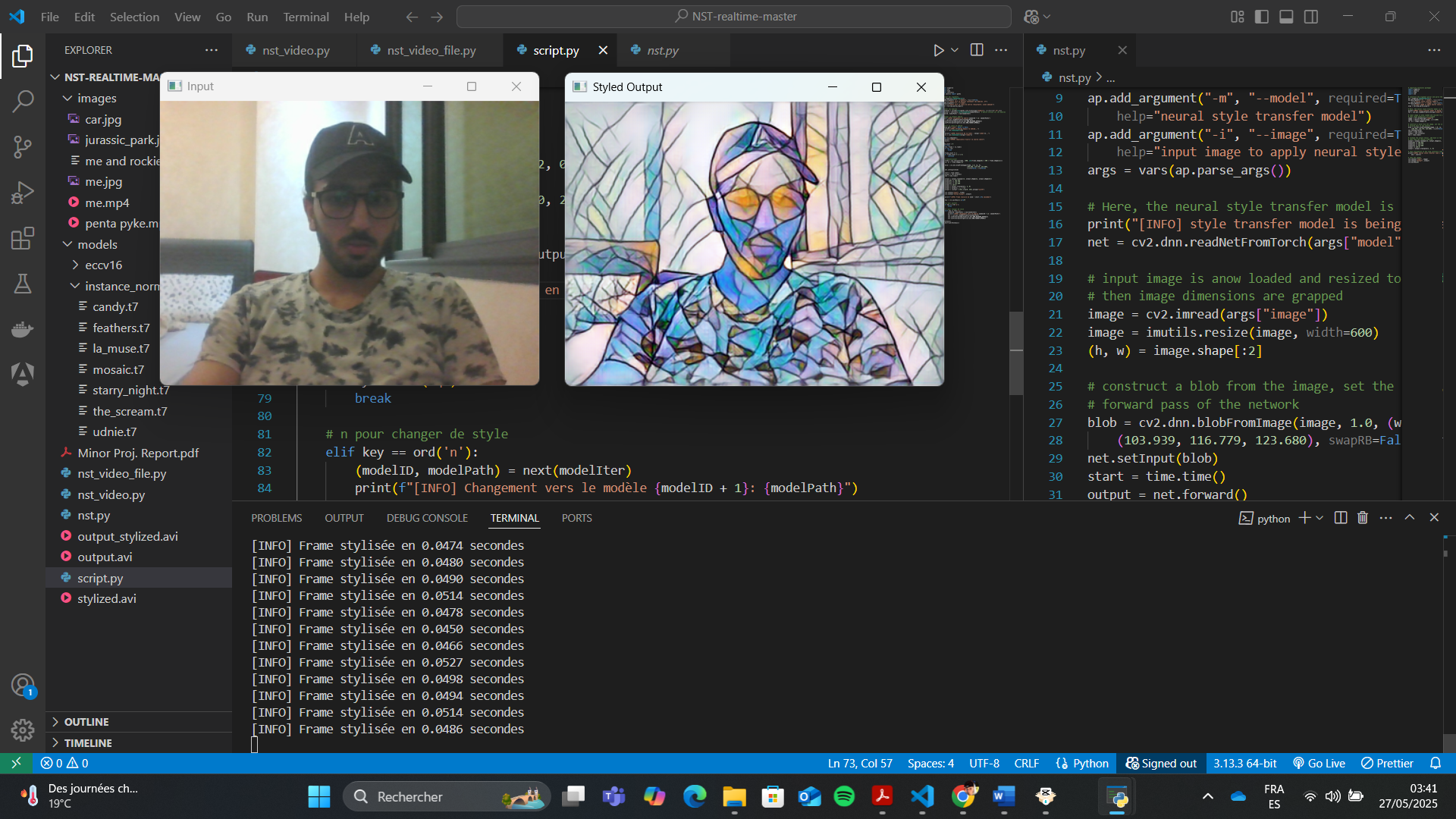


Figure 10 : Output après avoir appuyé sur « n » une autre fois

III. CONCLUSION

Ce projet a démontré avec succès l'implémentation d'un système de transfert de style neuronal en temps réel, combinant les avancées théoriques de Gatys, Johnson et Ulyanov avec des optimisations pratiques pour un traitement fluide de flux vidéo. Grâce à l'accélération GPU et à la gestion dynamique des modèles, nous avons atteint une latence inférieure à 150 ms par frame tout en conservant une grande flexibilité artistique, ouvrant ainsi la voie à des applications créatives dans les domaines du streaming, de l'art numérique et des médias interactifs. Bien que des défis persistent pour améliorer la cohérence temporelle entre les frames, ces résultats valident le potentiel du deep learning pour révolutionner les outils de création visuelle en temps réel.