

Traitement d'images en python

Photo séquence



ESIEE
PARIS

Céline NGUYEN
Ayane AHMED MOHAMED ALI
Agathe DENOUAIL
Céline SISAVANH

E2 groupe 2

Lien github : https://github.com/ayanea/Traitement_images

Prérequis:

- Télécharger la librairie ffmpeg à l'adresse suivante:
<https://www.google.com/url?q=https://ffmpeg.zeranoe.com/builds/&sa=D&source=hangouts&ust=1558784352806000&usg=AFOjCNE4MkEOIb4ALOATS1YtHT5l7WcY2g> version 20190524-1d74150
- Modifier les variables d'environnement du PATH en ajoutant une nouvelle variable avec le chemin d'accès du dossier bin situé dans le dossier télécharger, par exemple : C:\Users\DELL\Documents\E2\ffmpeg-20190524-1d74150-win64-static\bin
- Dans le cmd faire `pip install ffmpeg-python`
- Lancer photosequence.py avec la commande suivante (écrite dans le README):
`python photosequence.py -i nom_de_la_video.mp4 -fi 4`
par exemple : `python photosequence.py -i monsieur.mp4 -fi 4`

Une photo séquence c'est quoi ?

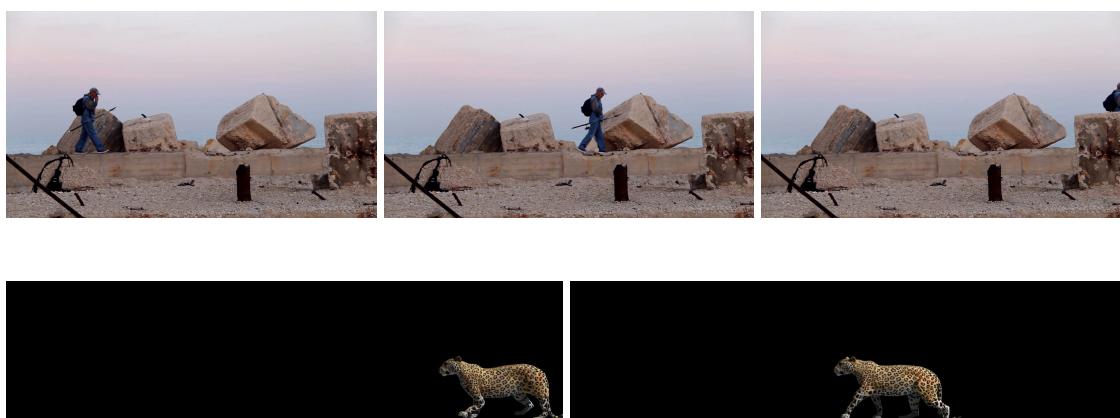
La photo séquence est le fait de prendre plusieurs photos d'un sujet en mouvement en gardant le même paysage et de les assembler sur une même image. Cela permet d'avoir la décomposition d'un mouvement et une impression de vitesse. C'est une technique particulièrement utilisée pour illustrer une figure sportive. Cela est très intéressant surtout pour les sports d'hiver rapides et acrobatiques.

Prise de vue :

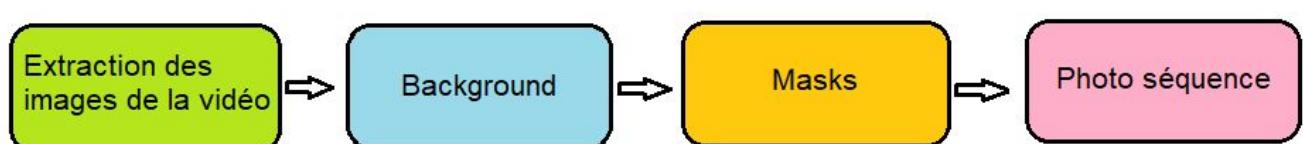
Tout d'abord, nous avons cherché des photos intéressantes sur internet afin de pouvoir tester nos fonctions dessus. Il y a plusieurs conditions à respecter pour pouvoir effectuer la photo séquence. La prise de vue doit avoir un cadrage large afin d'avoir le mouvement en intégralité et on doit éviter les superpositions du sujet d'une image à une autre. On a choisi de prendre une vidéo car cela nous a permis d'extraire plusieurs photos à plusieurs intervalles de temps très courts. À la suite de ça, nous avons extrait les images de cette vidéo avec la commande :

```
ffmpeg -i monsieur.mp4 -r 5 -f image2 image-%07d.png
```

Voici un exemple d'images extraites de la vidéo :



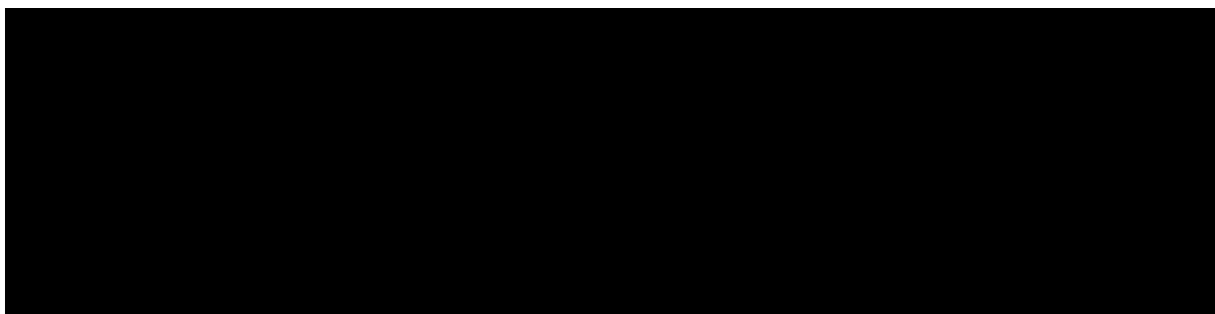
Voici les étapes à réaliser :



Création du background : [background.py](#)

La 1ère étape est de créer le background de l'image finale. Nous avons d'abord normaliser toutes les images afin qu'elles aient la même taille.

Ensuite, nous avons fait un filtre médian des pixels de toutes les photos afin d'éliminer les valeurs aberrantes (dans notre cas, ce sont les images des personnages) et donc d'obtenir le background.



Création des masques : [mask.py](#)

La 2ème étape est de créer les masks qui permettent d'extraire le sujet sur les images. On commence par créer un mask qui est une image sur fond blanc. Ensuite, on compare chaque image avec l'image background. La différence entre ces deux images représente le sujet.

On recopie la différence sur le mask en changeant la couleur des pixels en noir. Au final, le mask est une image avec un fond blanc et un sujet de couleur noire.

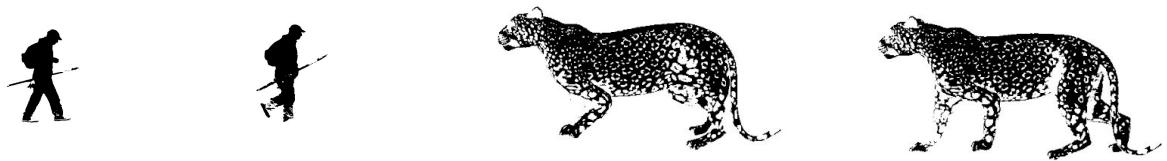
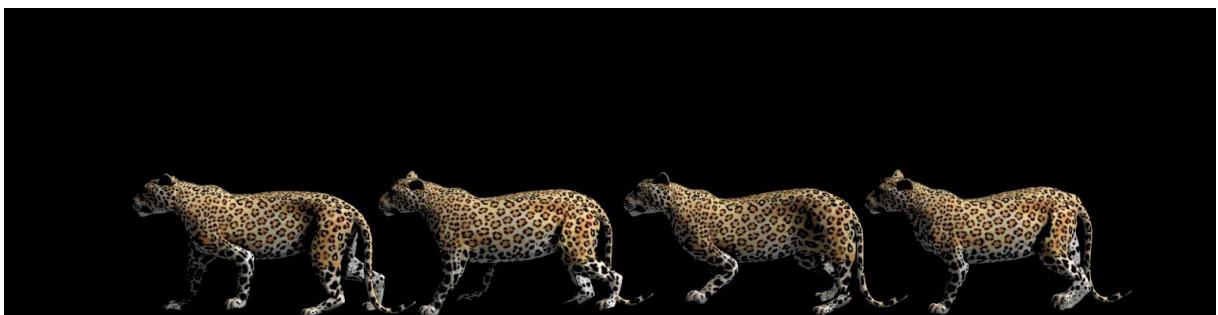


Photo séquence : [photosequence.py](#)

La dernière étape est la création de la photo séquence. Ainsi, si les pixels de chaque mask sont noirs, les pixels du sujet sont mis dans l'image résultat. On initialise l'image résultat avec l'image du background.



Par la suite, nous avons fait en sorte que les masks et donc les images associées ne se superposent pas lors de la photo séquence. Pour cela, nous avons créé une fonction qui sélectionne les photos à un même intervalle de temps régulier. Cela permettrait d'avoir une image plus claire et plus jolie.

De plus, on pourra également appliquer différents filtres sur les images afin d'avoir un rendu plus stylisé.

Ajout d'autres fonctionnalités :

Photo séquence avec fading : [photosequence_fading.py](#)

Nous avons voulu réaliser une photo séquence avec fading c'est-à-dire avec un dégradé de transparence.

Pour ce faire, nous avons créé une fonction qui permet de récupérer un certain pourcentage de pixels du background ou de l'image. Plus le pourcentage du background est élevé et plus la transparence est élevée. Au contraire, plus le pourcentage de l'image est élevé et plus la transparence est faible. Nous avons ensuite appliqué cette fonction à notre photo séquence. Pour un pixel , sa couleur sera

$(pixel_image * transparence + (pixel_background * 100\%-transparence))$

Pour calculer le pourcentage, nous avons utilisé un ratio par rapport au nombre d'images de façon à ce que la première image soit la plus transparente et la dernière image la plus opaque.

Par exemple:

Pour 150 images, ratio = 100/ 150

Transparence image 1 = $100/150 * 1$

Transparence image 2 = $100/150 * 2$

Transparence image 150 = $100/150 * 150 = 100$

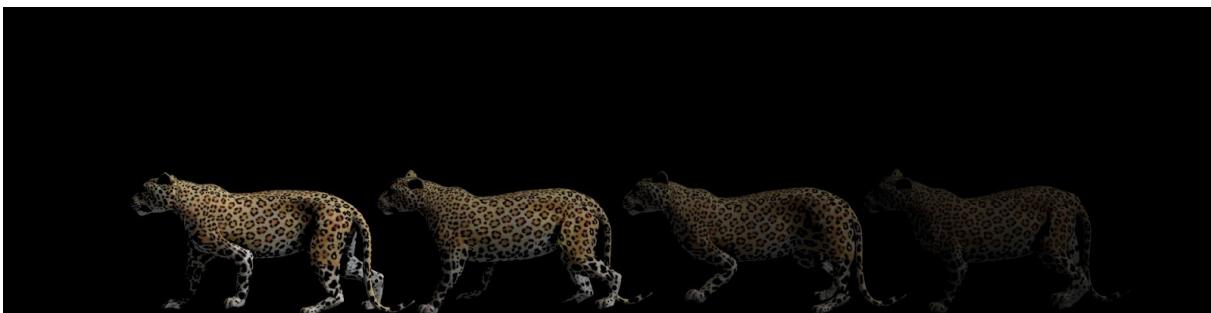


Photo séquence avec vidéo : [photosequence_video.py](#)

On utilise une image de fond dans laquelle on capture l'image du sujet toutes les 15 images. Pour ce faire, on commence par fusionner le background avec la première image puis cette image ainsi créée (composée ainsi de la première image du sujet) avec la deuxième image et ainsi de suite. Ensuite, on fusionne cette image de fond avec toutes les autres images pour avoir le sujet qui avance dans la vidéo.

Photo séquence avec vidéo et fading : [photosequence_video_fading.py](#)

Enfin, nous avons voulu réaliser une vidéo avec un effet fading. Le code est ainsi similaire à celui photosequence_fading associé à celui avec la vidéo.

Problèmes rencontrés :

1er problème :

Le premier problème est intervenu lors de la création du background.



L'oiseau ne bougeait pas assez lorsqu'il était sur l'arbre du coup une partie de son corps était considérée comme le background.



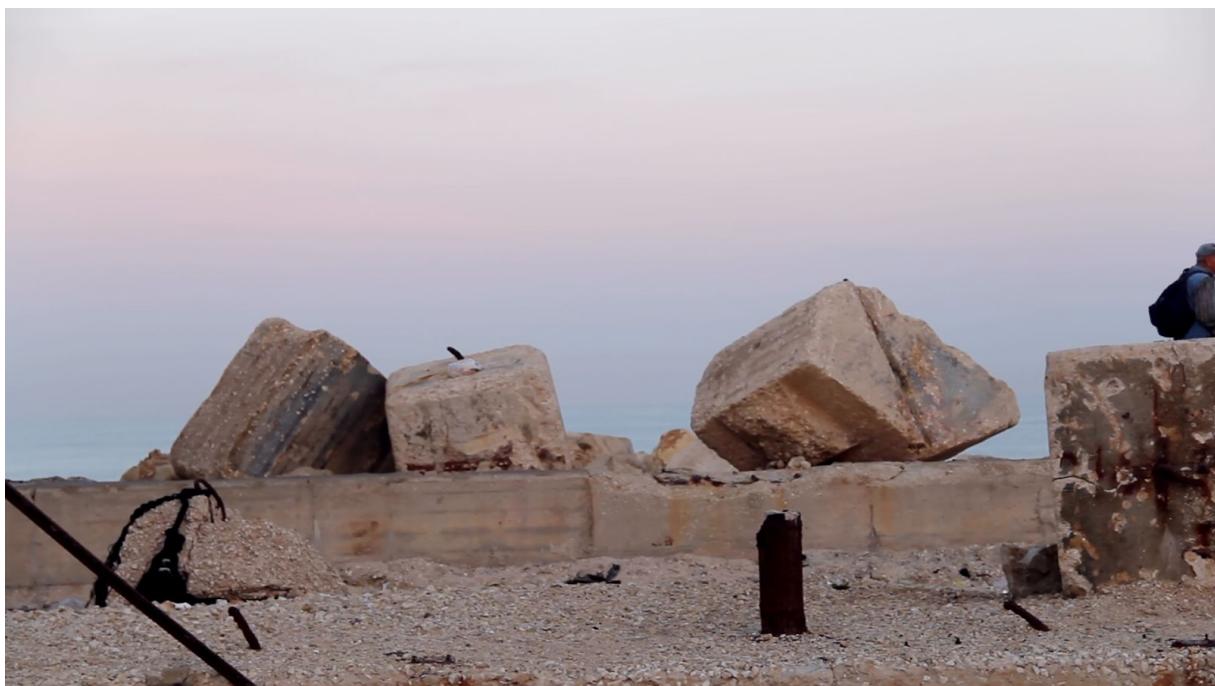
La caméra n'était pas stable donc l'image du background ne pouvait pas être nette.

Pour avoir un beau background il faut donc que la caméra ne bouge pas et que le sujet de la vidéo soit en mouvement constant.

2ème problème :

Le problème est que lors de la photo séquence, seul le dernier personnage a été affiché. En effet, dans la fonction `binary_merge`, si le pixel du mask est noir, on prend le pixel de l'image correspondant à ce mask et nous la mettons dans l'image finale. Sinon, nous prenons le pixel du background.

Tous les points de la première image et du mask sont correctement mis en place. Cependant, pour les autres images, la fonction va écraser les images précédentes (c'est-à-dire les remplacer par les pixels du background) .



Nous avons donc mis en paramètre de la fonction l'image résultat et nous l'avons initialisé avec l'image background en dehors de la fonction.

3ème problème :

Le troisième problème est intervenu lors de la photo séquence. Au départ, nous avons voulu créer une seule image avec tous les masks.



L'erreur est la même que précédemment, seule la dernière image du sujet est conservée. Nous avons donc mis en place un mask pour une tous les masks du sujet ensemble.

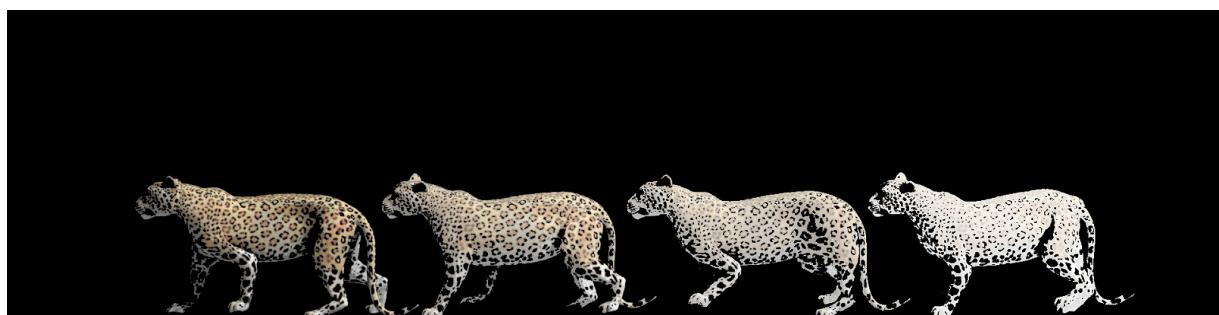
4ème problème :

Nous avons d'abord converti nos images en RGBA afin d'obtenir le canal alpha et donc la transparence. Puis, nous avons créé une fonction fading qui prend un pourcentage de transparence en paramètre et qui renvoie l'image avec la transparence correspondante.

Voici le résultat pour une image avec 50 % de transparence :



Nous avons ensuite appliqué une transparence croissante à toutes nos images.



Cependant, le résultat n'est pas très satisfaisant.

Comment peut-on améliorer notre projet ?

- Pour avoir différents résultats :

- Effet de disparition :

L'idée serait de travailler sur la vidéo photo séquence avec fading. Le personnage avance et plus il avance plus les images créant la photo séquence précédemment deviennent transparentes et finissent par s'effacer.

Par exemple, si on a les 4 premiers personnages qui apparaissent, lorsque le personnage arrive jusqu'au 4ème, le premier personnage s'efface et ainsi de suite.

- Travailler sur la netteté/contraste des images :

Rendre les images de plus en plus floues en fonction de l'avancement du sujet. Mettre des images avec des contrastes différents pour que certaines images ressortent plus que d'autres.

- Amélioration code :

- Faire en sorte que le programme trouve automatiquement combien de frames il doit prendre pour avoir une belle photo séquence (actuellement, c'est à nous de rentrer le nombre lorsque l'on lance la ligne de commande).
 - Faire en sorte que les programmes compilent et s'exécutent plus rapidement.

Indications au sujet des programmes et des instructions :

Les fichiers `background.py`, `mask.py` et `util.py` se trouvent dans le dossier lib. Les images ainsi que les photo séquence en vidéo se trouvent dans le dossier tpm. Les fichiers `photosequence.py`, `photosequence_fading.py`, `photosequence_video.py` et `photosequence_video_fading.py` se trouvent dans le dossier courant.

Voici les instructions à taper dans la ligne de commande pour lancer les programmes :

```
python photosequence.py -i monsieur.mp4 -fi 4
```

```
python photosequence.py -i jaguar.mp4 -fi 6
```

```
python photosequence_fading.py -i monsieur.mp4 -fi 4
```

```
python photosequence_fading.py -i jaguar.mp4 -fi 6
```

```
python photosequence_video.py -i monsieur.mp4 -fi 1 -vsi 15
```

```
python photosequence_video.py -i jaguar.mp4 -fi 1 -vsi 20
```