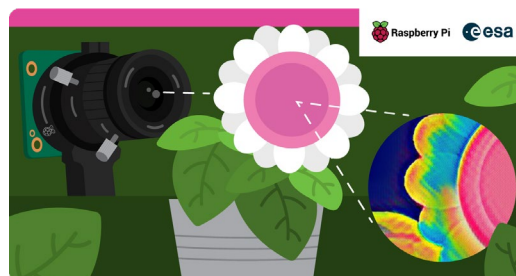




## Mesurer la santé des plantes avec NDVI et Raspberry Pi

Utilisez un Raspberry Pi et une caméra pour mesurer la santé des plantes à l'aide d'une lumière infrarouge.



### Étape 1 Introduction

---



## △ Module caméra Raspberry Pi - Notice du système d'exploitation

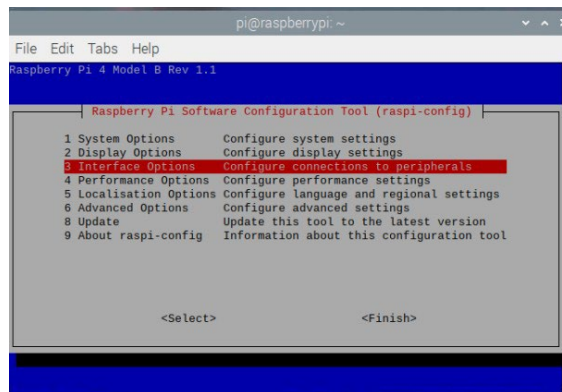
Le module Python Picamera n'est actuellement pas, par défaut, compatible avec la dernière version de Raspberry Pi OS (appelée Bullseye).

Pour utiliser le module Picamera, active la prise en charge de la caméra héritée.

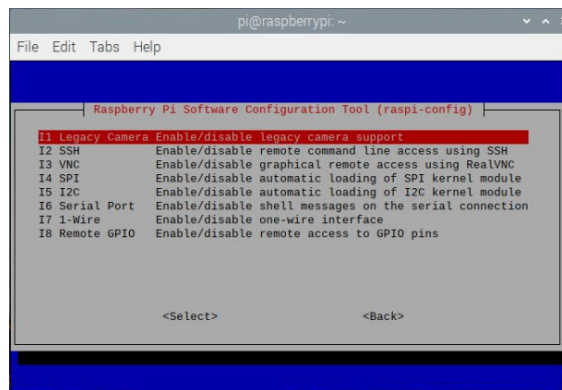
Ouvre une fenêtre de terminal et entre la commande suivante :

```
sudo raspi-config
```

Utilise les curseurs pour faire défiler vers le bas **Interface Options** et appuie sur la touche « Entrée ».

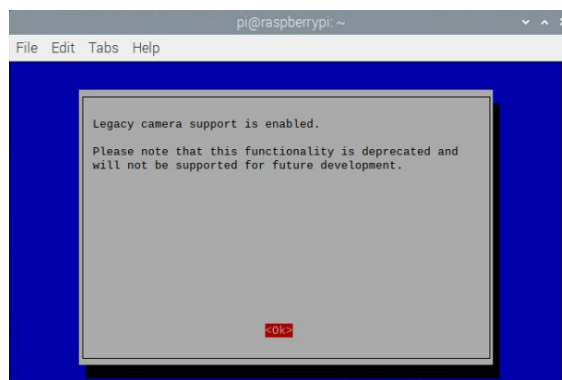


Assure-toi que « Activer Camera héritée/désactiver la prise en charge de caméra héritée » est sélectionné et appuie sur la touche « Entrée ».

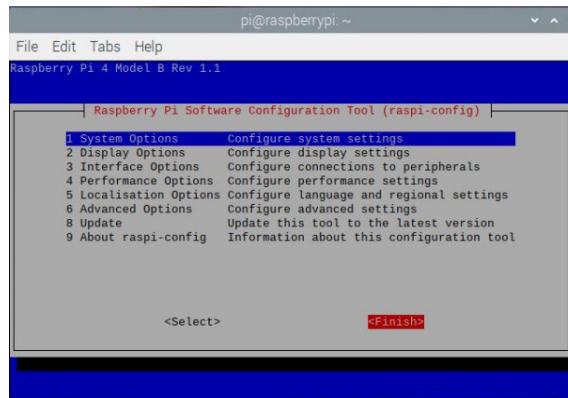


Utilise les clés du curseur pour sélectionner **<Yes>** et appuie sur la touche « Entrée ».

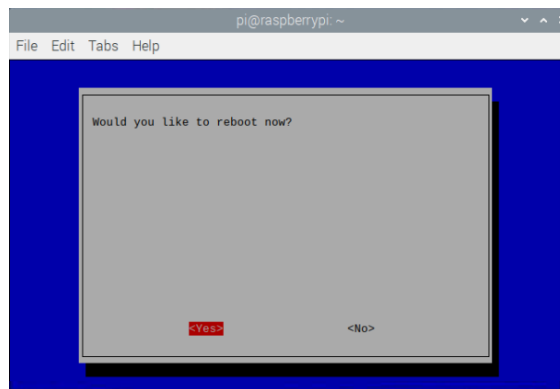
Appuie à nouveau sur « Entrée » pour confirmer



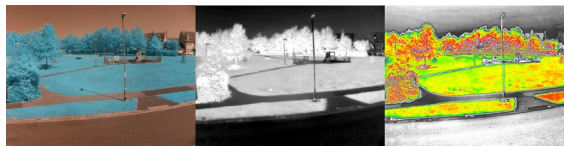
Utilise les touches curseur pour sélectionner <Finish>



Appuie sur « Entrée » pour redémarrer.



Dans ce projet, tu apprendras à utiliser des images prises avec une caméra équipée de filtres spéciaux pour mesurer la santé des plantes. Si tu as accès à une caméra haute qualité Raspberry Pi (<https://www.raspberrypi.org/products/raspberry-pi-high-quality-camera/>), tu apprendras également à la modifier pour qu'elle puisse être utilisée pour prendre ces photos. Si tu as accès à un module de caméra Pi NoIR (<https://www.raspberrypi.org/products/pi-noir-camera-v2/>), tu peux aussi l'utiliser.



**Les filtres** peuvent être utilisés sur les caméras pour empêcher certaines longueurs d'onde de la lumière d'atteindre le capteur. Par exemple, la plupart des caméras numériques disposent d'un filtre infrarouge pour empêcher la lumière infrarouge d'atteindre le capteur.

Tu vas :

- Apprendre ce que veut dire « Indice de végétation par différence normalisée » (Normalised Difference Vegetation Index, NDVI)
- Convertir une image prise avec une caméra modifiée afin qu'elle puisse être utilisée pour mesurer le NDVI
- Modifier un module de caméra Raspberry Pi, afin de pouvoir l'utiliser pour prendre des images NDVI

Il te faudra :

- un ordinateur Raspberry Pi
- en option, une caméra Raspberry Pi HQ modifiée ou un module de caméra Raspberry Pi NoIR

## Étape 2 Qu'est-ce que le NDVI ?

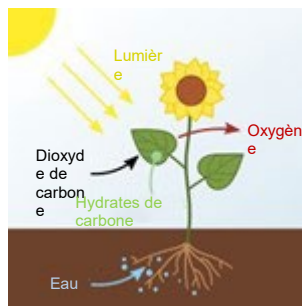
Le NDVI correspond à l'indice de végétation par différence normalisée, mais qu'est-ce que cela signifie ? Dans cette section, tu apprendras les bases du NDVI.

Les plantes sont généralement de couleur verte, mais t'es-tu déjà demandé pourquoi ?



([https://commons.wikimedia.org/wiki/File:Diversity\\_of\\_plants\\_image\\_version\\_5.png](https://commons.wikimedia.org/wiki/File:Diversity_of_plants_image_version_5.png))

La raison pour laquelle la plupart des feuilles sont vertes est qu'elles contiennent un produit chimique appelé chlorophylle. Ce produit chimique les aide à utiliser la lumière du soleil pour transformer le dioxyde de carbone et l'eau en produits chimiques utiles appelés hydrates de carbone, dans un processus appelé photosynthèse.

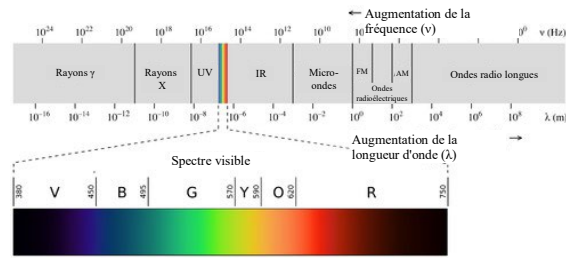


([https://commons.wikimedia.org/wiki/File:Photosynthesis\\_en.svg](https://commons.wikimedia.org/wiki/File:Photosynthesis_en.svg))

La chlorophylle ne peut pas utiliser toute la lumière du soleil. La lumière venant du soleil se présente sous différentes formes. En observant un arc-en-ciel, tu peux voir plusieurs couleurs de lumière différentes. Il existe des formes de lumière que les humains ne peuvent pas voir :

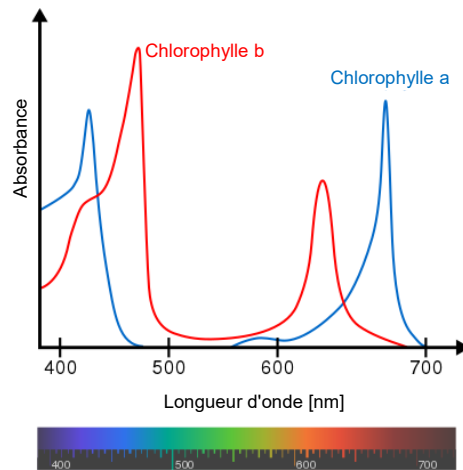
- La lumière ultraviolette (UV) est invisible pour les humains. C'est un type de lumière qui peut provoquer un coup de soleil.
- La lumière infrarouge (IR) est également invisible pour les humains, mais c'est la raison pour laquelle tu peux ressentir la chaleur quand tu places tes mains devant un feu.

Il existe aussi d'autres formes de lumière dont tu as peut-être entendu parler, telles que les micro-ondes, les ondes radio, les rayons X et les rayons gamma.



([https://commons.wikimedia.org/wiki/File:EM\\_spectrumrevised.png](https://commons.wikimedia.org/wiki/File:EM_spectrumrevised.png))

La chlorophylle ne peut utiliser qu'une partie de la lumière du soleil pour réaliser la photosynthèse. Elle ne peut pas utiliser la lumière verte, qui est donc réfractée, ce qui explique pourquoi les plantes sont vertes.



([https://commons.wikimedia.org/wiki/File:Chlorophyll\\_ab\\_spectra-en.svg](https://commons.wikimedia.org/wiki/File:Chlorophyll_ab_spectra-en.svg))

Les plantes n'aiment pas non plus beaucoup les rayons infrarouges, car ils les font chauffer, de la même manière que tu ne mettrais pas tes mains devant un feu pendant trop longtemps. Les plantes ont évolué pour réfléchir autant de lumière infrarouge que possible.

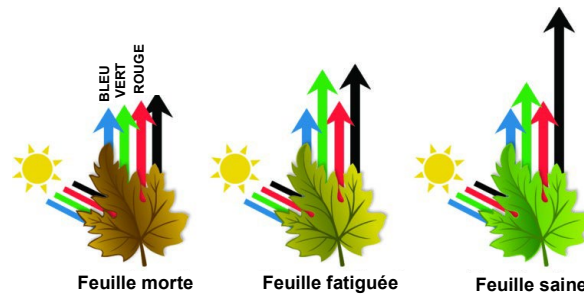
Les caméras modernes peuvent détecter de nombreux types de lumière. Parce qu'une photo prise par une caméra numérique, qui montrerait également la lumière infrarouge, serait un peu étrange pour l'œil humain, des filtres spéciaux ont été ajoutés à ces caméras, de sorte que la lumière infrarouge ne puisse pas atteindre le capteur. Tu peux voir ci-dessous une image d'un parc, prise en ayant retiré le filtre infrarouge.



Il existe différents types de lumière infrarouge. Les caméras capables d'imagerie thermique captent la lumière infrarouge à ondes longues. Celle-ci est différente de la lumière infrarouge à ondes courtes (infrarouge proche) qui doit être filtrée par des caméras numériques.

Cela s'avère très utile pour mesurer la santé des plantes. Si une plante est en bonne santé, elle réfléchira beaucoup de lumière infrarouge proche. Si une plante est en train de mourir, elle absorbera beaucoup de lumière infrarouge proche. La couleur bleu-vert de la photo signifie qu'une plus grande quantité de lumière infrarouge est réfractée.

Regarde cette image de feuilles réfléchissant la lumière. Tu peux voir que la lumière infrarouge est davantage réfléchi par les feuilles saines que par les feuilles fatiguées ou mortes.



L'utilisation d'une caméra sans filtre infrarouge nous permet de détecter la quantité de lumière infrarouge qui est réfractée par les plantes, ce qui permet de mesurer l'état de santé de la plante.



## Étape 3 Charger et afficher des images avec Python

Pour transformer une image sans filtre IR en une image NDVI, tu vas utiliser un module graphique appelé OpenCV.

Sur ton Raspberry Pi, clique sur une borne et maintiens les touches Ctrl et Alt enfoncées, puis appuie sur la touche t.

Ce projet nécessite des paquets Python supplémentaires pour effectuer des calculs sur les images.

Dans le terminal,

```
sudo pip3 install -U numpy
sudo pip3 install opencv-python sudo
apt install libatlas-base-dev
```

Les paquets devraient être installés après quelques minutes.

Ouvre Thonny depuis le menu Programming.

Pour commencer, charge simplement une image et affiche-la sur ton écran.

Fais un clic droit sur cette image et enregistre-la dans ton dossier personnel sur ton Raspberry Pi.



Ensuite, dans Thonny, commence par importer les deux modules dont tu auras besoin pour démarrer.

ndvi.py

```
1 import cv2
2 import numpy as np
```

L'étape suivante consiste à charger une image et à l'afficher sur l'écran.

- `cv2.imread` est utilisé pour charger une image
- `np.array(original, dtype=float)/float(255)` est utilisé pour convertir l'image en un tableau avec le type approprié
- `cv2.namedWindow` est utilisé pour créer une fenêtre d'affichage
- `cv2.imshow` est utilisé pour afficher une image dans une fenêtre
- `cv2.waitKey` empêche la fenêtre de disparaître, jusqu'à ce qu'une touche soit activée
- `cv2.destroyAllWindows()` ferme la fenêtre lorsque la touche a été activée

Voici le code dont tu auras besoin.



ndvi.py

```
1 import cv2
2 import numpy as np
3
4 image = cv2.imread('/home/pi/park.png') # load image
5 image = np.array(image, dtype=float)/float(255) #convert to an array
6 cv2.namedWindow('Original') # create window
7 cv2.imshow('Original', image) # display image
8 cv2.waitKey(0) # wait for key press
9 cv2.destroyAllWindows()
```

Maintenant, exécute ton code. Tu devrais voir l'image apparaître à l'écran. Lorsque tu appuies sur une touche, elle disparaît.




Il se peut que l'image soit trop grande pour ton écran, mais tu peux corriger cela en mettant l'image à l'échelle.

Tu dois d'abord obtenir la largeur et la hauteur de l'image que tu utilises, puis réduire ces valeurs. Dans l'exemple ci-dessous, la hauteur et la largeur de l'image sont divisées par 2, mais tu peux utiliser une valeur différente pour les mettre plus ou moins à l'échelle. Ajoute le code en surbrillance ci-dessous.




ndvi.py

```
4 image = cv2.imread('/home/pi/park.png') # load image
5 image = np.array(original, dtype=float)/float(255) #convert to an array
6 shape = original.shape
7 height = int(shape[0]/2)
8 width = int(shape[1]/2)
9 cv2.namedWindow('Original') # create window
```

Maintenant, redimensionne l'image avant de l'afficher, en ajoutant le code en surbrillance. 

ndvi.py

```
6 shape = original.shape
7 height = int(shape[0] / 2)
8 width = int(shape[1] / 2)
9 image = cv2.resize(image, (width, height))
10 cv2.namedWindow('Original') # create window
```

Comme tu souhaites également afficher d'autres images, tu peux transformer le code que tu as écrit en une fonction et l'appeler. La fonction utilisera un objet image ainsi qu'une chaîne qui décrit l'image. 

Voici le code complet jusqu'ici.

ndvi.py

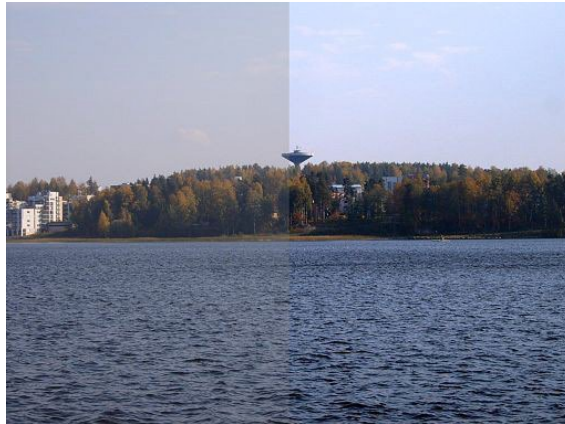
```
1 import cv2
2 import numpy as np
3
4 original = cv2.imread('/home/pi/park.png')
5
6
7 def display(image, image_name):
8     image = np.array(image, dtype=float)/float(255)
9     shape = image.shape
10    height = int(shape[0] / 2)
11    width = int(shape[1] / 2)
12    image = cv2.resize(image, (width, height))
13    cv2.namedWindow(image_name)
14    cv2.imshow(image_name, image) cv2.waitKey(0)
15    cv2.destroyAllWindows()
16
17
18 display(original, 'Original')
19
```




Enregistre ton projet.

## Étape 4 Augmenter le contraste

Le contraste désigne la différence entre la luminosité ou la couleur d'une image. L'image ci-dessous a été divisée de sorte que la moitié gauche présente un faible contraste et la moitié droite un fort contraste. Tu dois augmenter le contraste de ton image pour le NDVI.



([https://commons.wikimedia.org/wiki/File:Photo\\_editing\\_contrast\\_correction.jpg](https://commons.wikimedia.org/wiki/File:Photo_editing_contrast_correction.jpg))

Crée une nouvelle fonction appelée `contrast_stretch`. Cette fonction aura un paramètre unique qui sera l'objet image cv2 chargé. La fonction doit venir après que l'image ait été chargée. 

ndvi.py

```
1 import cv2
2 import numpy as np
3
4 original = cv2.imread('/home/pi/park.png')
5
6
7 def display(image, image_name):
8     image = np.array(image, dtype=float)/float(255)
9     shape = image.shape
10    height = int(shape[0] / 2)
11    width = int(shape[1] / 2)
12    image = cv2.resize(image, (width, height))
13    cv2.namedWindow(image_name)
14    cv2.imshow(image_name, image) cv2.waitKey(0)
15    cv2.destroyAllWindows()
16
17 def contrast_stretch(im):
18
```

L'objet image appelé `park` n'est qu'une grande liste de chiffres. L'étape suivante consiste à trouver le plus grand et le plus petit de ces nombres, en utilisant `numpy`.

Ajoute ces deux lignes à ta fonction pour trouver la luminosité supérieure des pixels de l'image dans les 5 % supérieurs et les 5 % inférieurs de ton image.



ndvi.py

```
19 def contrast_stretch(im):
20     in_min = np.percentile(im, 5)
21     in_max = np.percentile(im, 95)
```

Tu dois maintenant définir la luminosité maximale et la luminosité minimale de la nouvelle image que tu vas créer. La couleur la plus vive d'un pixel est 255, et la plus faible est 0.

Ajoute ces lignes à ta fonction.



ndvi.py

```
19 def contrast_stretch(im):
20     in_min = np.percentile(im, 5)
21     in_max = np.percentile(im, 95)
22
23     out_min = 0.0
24     out_max = 255.0
```

Il faut maintenant effectuer quelques calculs pour modifier tous les pixels de l'image, de sorte que l'image présente la gamme complète des contrastes de 0 à 255.

Ajoute ces lignes pour élargir la valeur des pixels et obtenir l'image



contrastée. ndvi.py

```
19 def contrast_stretch(im):
20     in_min = np.percentile(im, 5)
21     in_max = np.percentile(im, 95)
22
23     out_min = 0.0
24     out_max = 255.0
25
26     out = im - in_min
27     out *= ((out_min - out_max) / (in_min - in_max))
28     out += in_min
29
30     return out
```

Maintenant que tu disposes d'une fonction permettant de renforcer le contraste de l'image, tu peux convertir ton image et l'afficher à l'écran.

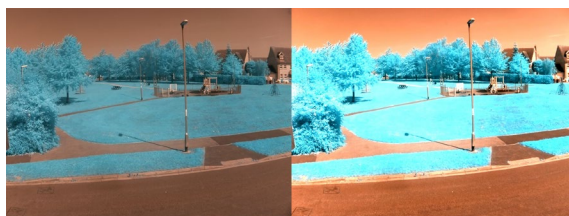
Appelle la fonction `contrast_stretch` puis modifie-la et affiche-la à l'aide de ta fonction



`affichage.ndvi.py`

```
33 | display(original, 'Original')
34 | contrasted = contrast_stretch(original)
35 | display(contrasted, 'Contrasted original')
```

Il devrait y avoir une différence notable dans le contraste des deux images maintenant.



Tu peux sauvegarder ton image ayant le contraste le plus élevé en ajoutant une seule ligne à la fin de ton code, afin de pouvoir comparer les deux images dans ton navigateur de fichiers. Elle s'appellera `contrasted.png`.



`ndvi.py`

```
33 | display(original, 'Original')
34 | contrasted = contrast_stretch(park)
35 | display(contrasted, 'Contrasted original')
36 | cv2.imwrite('contrasted.png', contrasted)
```



Enregistre ton projet.

## Étape 5 Calculer le NDVI

Maintenant que tu disposes d'une image à fort contraste, il est temps d'effectuer les calculs de NDVI. Cela rendra tous les pixels bleus plus clairs, et tous les pixels rouges plus foncés, laissant une image en noir et blanc. Les pixels les plus clairs de l'image indiquent des plantes saines, et les pixels les plus sombres indiquent des plantes en mauvaise santé ou une absence de plantes.



Crée une nouvelle fonction et appelle-la `calc_ndvi`. Elle prendra une image `cv2`



comme paramètre. `ndvi.py`

```

1  def contrast_stretch(im):
9      in_min = np.percentile(im, 5)
2      in_max = np.percentile(im, 95)
0
2
1      out_min = 0.0
2      out_max = 255.0
2
2      out = im - in_min
3      out *= ((out_min - out_max) / (in_min - in_max))
2      out += in_min
4
2      return out
5
2
6
2  def calc_ndvi(image):

```

Pour ajuster les pixels de l'image et ne travailler que sur le rouge et le bleu, l'image doit être divisée en trois canaux distincts. `r` pour le rouge, `g` pour le vert, et `b` pour le bleu.

Ajoute cette ligne à ta fonction.



```

32  def calc_ndvi(image):
33      b, g, r = cv2.split(image)

```

Maintenant, les canaux rouge et bleu doivent être additionnés et stockés sous `bottom`. On peut ensuite soustraire le canal rouge au canal bleu (rappelle-toi que le rouge signifie que les plantes ne sont pas en bonne santé ou qu'il n'y a pas de plantes), puis diviser ce dernier par le résultat du calcul `bottom`.

Puisque nous effectuons une division, nous devons également nous assurer qu'aucun de nos diviseurs n'est égal à 0, sinon il y aura une erreur.

Ajoute ces lignes à ta fonction pour effectuer le calcul.



```
32 def calc_ndvi(image):
33     b, g, r = cv2.split(image)
34     bottom = (r.astype(float) + b.astype(float))
35     bottom[bottom==0] = 0.01
36     ndvi = (b.astype(float) - r) / bottom
37     return ndvi
```

Maintenant que tu as une fonction pour calculer le NDVI, tu peux passer à l'image contrastée, l'afficher et l'enregistrer.

ndvi.py



```
40 display(original, 'Original')
41 contrasted = contrast_stretch(park)
42 display(contrasted, 'Contrasted original')
43 cv2.imwrite('contrasted.png', contrasted)
44 ndvi = calc_ndvi(contrasted)
45 display(ndvi, 'NDVI')
46 cv2.imwrite('ndvi.png', ndvi)
```

Si tu regardes ton image NDVI, elle sera probablement assez sombre, bien que tu puisses distinguer des taches de pixels plus lumineux. Pour améliorer à nouveau l'image, il est possible de la soumettre à la fonction `contrast_stretch`.



ndvi.py

```
40 display(original, 'Original')
41 contrasted = contrast_stretch(original)
42 display(contrasted, 'Contrasted original')
43 cv2.imwrite('contrasted.png', contrasted)
44 ndvi = calc_ndvi(contrasted)
45 display(ndvi, 'NDVI')
46 ndvi_contrastrated = contrast_stretch(ndvi)
47 display(ndvi_contrastrated, 'NDVI Contrastrated')
48 cv2.imwrite('ndvi_contrastrated.png', ndvi_contrastrated)
```

Maintenant, tu peux observer la végétation saine grâce à la luminosité des pixels de l'image `ndvi_contrastrated.png`.



Enregistre ton projet.



## Étape 6 Cartographier les couleurs

Tu disposes maintenant d'une image NDVI, mais il est difficile pour les humains de distinguer les différentes nuances de gris. Tu peux soumettre l'image à un processus de cartographie des couleurs qui transformera les pixels très clairs en couleur rouge et les pixels plus sombres en couleur bleue.



Une carte colorimétrique convertit les pixels d'une image d'une couleur en une autre. L'image NDVI est actuellement en niveaux de gris, ce qui signifie qu'elle est en noir et blanc, avec toutes les nuances intermédiaires.

L'image étant en niveaux de gris, chaque pixel peut être représenté par un seul nombre compris entre 0 et 255.

La carte colorimétrique `fastiecm` ([https://storage.googleapis.com/publiclab-production/public/system/images/photos/000/006/146/original/NDVI\\_VGYRM-lut.txt](https://storage.googleapis.com/publiclab-production/public/system/images/photos/000/006/146/original/NDVI_VGYRM-lut.txt)) transforme les pixels sombres en pixels blancs. Plus les pixels d'origine sont brillants, plus les couleurs sont décalées le long du spectre. Ainsi, les pixels gris foncé deviennent bleus, tandis que les pixels blancs brillants deviennent rouges.

Télécharge la carte colorimétrique `fastiecm.py` (<https://projects-static.raspberrypi.org/projects/astropi-ndvi/236334cedbdcb0841551dc04819de7160af62c5/en/images/fastiecm.py>). Sauvegarde-la au même emplacement que ton fichier `ndvi.py`.

Importe le fichier que tu viens de télécharger dans ton programme Python.

`ndvi.py`

```
1 import cv2
2 import numpy as np
3 from fastiecm import fastiecm
```


L'image actuelle, que tu as enregistrée sous `ndvi_contrasted` n'est pas adaptée pour la cartographie des couleurs. Les nombres stockés dans le tableau `numpy` sont actuellement tous des `floats` ou ce qu'on appelle communément des nombres décimaux. Ils doivent tous être convertis en nombres entiers, ou `integers` compris entre 0 et 255. Heureusement, la bibliothèque `numpy` peut le faire pour nous.

Ajoute la ligne en surbrillance pour convertir ton tableau.

`ndvi.py`

```
41 display(original, 'Original')
42 contrasted = contrast_stretch(original)
43 display(contrasted, 'Contrasted original')
44 cv2.imwrite('contrasted.png', contrasted)
45 ndvi = calc_ndvi(contrasted)
46 display(ndvi, 'NDVI')
47 ndvi_contrasted = contrast_stretch(ndvi)
48 display(ndvi_contrasted, 'NDVI Contrasted')
49 cv2.imwrite('ndvi_contrasted.png', ndvi_contrasted)
50 color_mapped_prep = ndvi_contrasted.astype(np.uint8)
```

L'image peut maintenant être convertie grâce à une cartographie des couleurs `cv2` et être transcrite dans un nouveau fichier.

Ajoute les lignes en surbrillance ci-dessous, pour convertir l'image à l'aide de la carte colorimétrique `fastiecm`, puis pour l'afficher et écrire un nouveau fichier. 

`ndvi.py`

```
50 color_mapped_prep = ndvi_contrasted.astype(np.uint8)
51 color_mapped_image = cv2.applyColorMap(color_mapped_prep, fastiecm)
52 display(color_mapped_image, 'Color mapped')
53 cv2.imwrite('color_mapped_image.png', color_mapped_image)
```

Tu devrais maintenant voir l'image cartographiée entièrement en couleur dans ton navigateur de fichiers, enregistrée sous `color_mapped.png`.



Enregistre ton projet.

## Étape 7 Crée tes propres images NDVI

Pour cette étape, tu auras besoin d'une caméra Raspberry Pi HQ et de quelques filtres, ou du module de caméra Raspberry Pi NoIR. Tu peux ensuite prendre tes propres images de NDVI ou même accéder au Mission Space Lab et prendre des photos NDVI depuis la Station spatiale internationale.



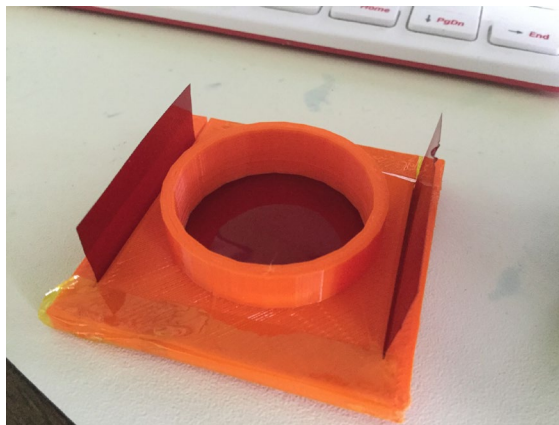
Conversion de ta caméra Raspberry Pi HQ avec un filtre ROUGE + NIR

- Tu devras acheter un filtre Rouge et un filtre NIR 850 nm (<https://midopt.com/filters/db660850/>).
- Suis ce guide ([https://www.raspberrypi.org/documentation/hardware/camera/hqcam\\_filter\\_removal.md](https://www.raspberrypi.org/documentation/hardware/camera/hqcam_filter_removal.md)) pour retirer le filtre IR de ta caméra HQ Raspberry Pi. Cela annulera la garantie.
- Suis ce guide vidéo pour installer ton filtre dans ta caméra.



Conversion de ta caméra Raspberry Pi HQ avec un filtre Rouge R26

- Tu devras acheter une feuille de Rosculux 26 : Filtres rouge clair (<https://www.pnta.com/expendables/gels/roscolux/roscolux-26-light-red/>).
- Suis ce guide ([https://www.raspberrypi.org/documentation/hardware/camera/hqcam\\_filter\\_removal.md](https://www.raspberrypi.org/documentation/hardware/camera/hqcam_filter_removal.md)) pour retirer le filtre IR de ta caméra HQ Raspberry Pi. Cela annulera la garantie.
- Le filtre rouge peut être fixé à l'aide de ruban adhésif à l'avant de l'objectif de ta caméra HQ ou maintenu en place à l'aide d'une pièce imprimée en 3D.
- Tu peux télécharger le STL ([https://projects-static.raspberrypi.org/projects/astropi-ndvi/236334cedbdcbb0841551dc04819de7160af62c5/en/images/lens\\_holder.stl](https://projects-static.raspberrypi.org/projects/astropi-ndvi/236334cedbdcbb0841551dc04819de7160af62c5/en/images/lens_holder.stl)) OBJ ([https://projects-static.raspberrypi.org/projects/astropi-ndvi/236334cedbdcbb0841551dc04819de7160af62c5/en/images/lens\\_holder.obj](https://projects-static.raspberrypi.org/projects/astropi-ndvi/236334cedbdcbb0841551dc04819de7160af62c5/en/images/lens_holder.obj)) ou SVG ([https://projects-static.raspberrypi.org/projects/astropi-ndvi/236334cedbdcbb0841551dc04819de7160af62c5/en/images/lens\\_holder.svg](https://projects-static.raspberrypi.org/projects/astropi-ndvi/236334cedbdcbb0841551dc04819de7160af62c5/en/images/lens_holder.svg)) et imprimer ou découper toi-même la pièce au laser.



### Utilisation du module de caméra Raspberry Pi NoIR

Tu peux utiliser le module de caméra Raspberry Pi NoIR pour les images NDVI, cependant, tu devras modifier une de tes lignes de code. En effet, la caméra Raspberry Pi NoIR utilise un filtre bleu au lieu d'un filtre rouge. Le filtre rouge est plus adapté à la capture d'images NDVI, mais le même type d'effet peut être obtenu avec une Raspberry Pi NoIR. Au lieu de soustraire le canal rouge de l'image, il faut soustraire le canal bleu.

La ligne est mise en surbrillance et commentée dans le script ci-dessous.

ndvi.py

```

32 def calc_ndvi(image):
33     b, g, r = cv2.split(image)
34     bottom = (r.astype(float) + b.astype(float))
35     bottom[bottom==0] = 0.01
36     ndvi = (r.astype(float) - b) / bottom # THIS IS THE CHANGED LINE
37     return ndvi

```

Si tu souhaites savoir comment connecter un module caméra au Raspberry Pi, et apprendre les bases de l'utilisation du module PiCamera, tu peux jeter un coup d'œil à notre guide Premiers pas avec le module caméra (<https://projects.raspberrypi.org/en/projects/getting-started-with-picamera>).

Lorsque ton Raspberry Pi est éteint, connecte ta caméra, puis redémarre ton Raspberry Pi.



La première étape consiste à configurer la caméra pour prendre une photo. Les résolutions des différentes caméras varient. Si tu utilises la caméra Raspberry Pi NoIR, tu dois choisir une résolution de 1920x1080. Si tu utilises la caméra Raspberry Pi HQ, tu peux utiliser une résolution de 2582x1952 .

Ajoute ces lignes à ton code pour configurer et utiliser la caméra Raspberry Pi. Commente la ligne qui permet de charger l'image park.png.



ndvi.py

```

1 import cv2
2 import numpy as np
3 from fastiecm import fastiecm
4 from picamera import PiCamera
5 import picamera.array
6
7 cam = PiCamera()
8 cam.rotation = 180
9 # cam.resolution = (1920, 1080) # Uncomment if using a Pi Noir camera
10 cam.resolution = (2592, 1952) # Comment this line if using a Pi Noir camera
11
12 # original = cv2.imread('park.png') #Comment out this line, as no longer used
13
14

```

Plutôt que de simplement capturer une image avec l'appareil photo et l'enregistrer sur la carte SD, l'image sera capturée sous la forme d'un tableau de données de pixels, afin qu'elle puisse être utilisée par `numpy` et `OpenCV`.

Capture un flux et enregistre-le sous forme de tableau.



ndvi.py

```

7 cam = PiCamera()
8 cam.rotation = 180
9 # cam.resolution = (1920, 1080) # Uncomment if using a Pi Noir camera
10 cam.resolution = (2592, 1952) # Comment this line if using a Pi Noir camera
11 stream = picamera.array.PiRGBArray(cam)
12 cam.capture(stream, format='bgr', use_video_port=True)
13 original = stream.array
14 # original = cv2.imread('park.png') #Comment out this line, as no longer used
15
16

```

Au lieu d'exécuter la fonction `contrast_stretch` sur l'objet image `original` que tu as chargé, elle sera maintenant exécutée sur le flux enregistré, qui est aussi appelé `original`.

Ajoute une ligne vers la fin de ton code pour que tu puisses aussi établir le tableau original



dans un fichier. ndvi.py

```
61 | display(color_mapped_image, 'Color mapped')
62 | cv2.imwrite('color_mapped_image.png', color_mapped_image)
63 | cv2.imwrite('original.png', original)
```

Cette image montre toutes les prises d'un plant de basilic mourant. Tu peux voir qu'à la base de la plante, les feuilles sont soit mourantes soit mortes, alors que près du sommet, il y a encore quelques feuilles saines.



Débogage - les images de ma caméra ressortent rouges

Il s'agit d'un problème connu, mais il existe une solution simple pour y remédier.

- Appuie sur `Ctrl + Alt + t` pour ouvrir une fenêtre de terminal.
- Tape ce qui suit pour modifier ton fichier `config.txt` :

```
sudo nano /boot/config.txt
```

- Ajoute la ligne suivante au bas du fichier `awb_auto_is_greyworld=1`.
- Appuie sur `Ctrl + O` pour enregistrer le fichier et sur `Ctrl + X` pour quitter nano.
- Tu peux maintenant fermer la fenêtre du terminal, redémarrer ton Raspberry Pi et essayer de prendre des photos à nouveau.



Enregistre ton projet.

## Étape 8 Mission Space Lab

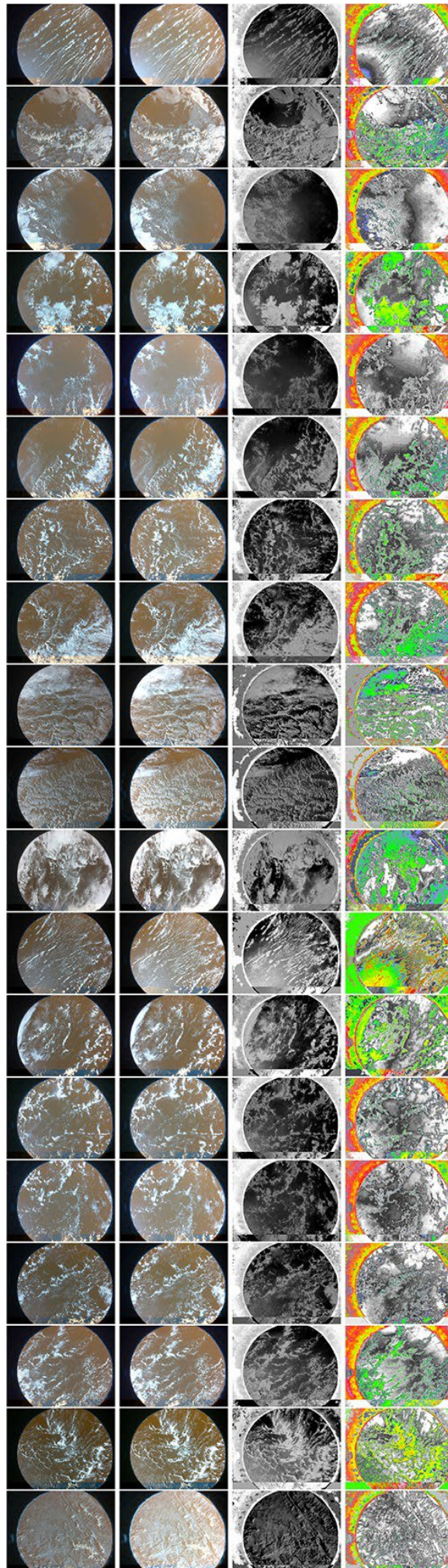
---

Mission Space Lab (<https://astro-pi.org/mission-space-lab/> ou <https://esero.fr/projets/astro-pi/>) fait partie du European Astro Pi Challenge, un projet éducatif de l'ESA mené en collaboration avec la Raspberry Pi Foundation, destiné aux jeunes de 19 ans et moins des États membres de l'ESA (<https://astro-pi.org/mission-space-lab/eligibility>), de Slovénie, de Lettonie, de Lituanie, du Canada et de Malte.



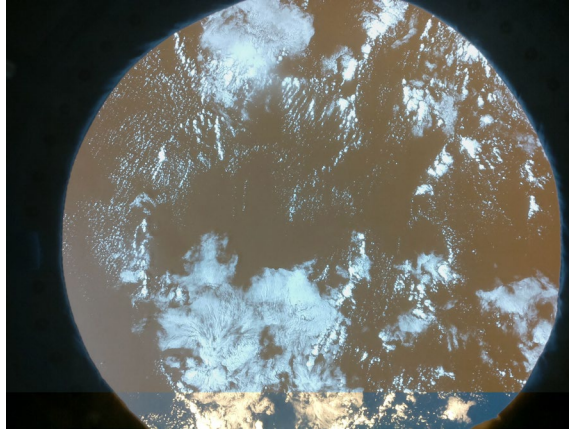
Mission Space Lab consiste à écrire un programme informatique qui sera exécuté sur l'une des deux unités Astro Pi de la Station spatiale internationale, afin de mener une expérience scientifique sur la vie sur Terre ou dans l'espace. NDVI est un excellent moyen d'utiliser la caméra de l'Astro Pi pour mesurer la santé des plantes sur Terre.

Voici un exemple d'images NDVI provenant de l'unité Astro Pi appelée Izzy. Izzy est équipée d'une caméra Raspberry Pi NoIR dirigée vers la Terre à travers le hublot de l'ISS, qui peut être utilisée pour capturer des images de la Terre.





Si tu souhaites t'entraîner à utiliser le NDVI sur des images prises depuis l'ISS, tu peux utiliser l'image ci-dessous, ou regarder les images dans le dossier de ressources de ce projet (<https://rpf.io/p/en/astropi-ndvi-go>).



---

Publié par Raspberry Pi Foundation (<https://www.raspberrypi.org>) sous une licence Creative Commons (<https://creativecommons.org/licenses/by-sa/4.0/>).

Voir le projet et la licence sur GitHub (<https://github.com/RaspberryPiLearning/astropi-ndvi>)