

Interface *Shiny* pour l'analyse et la visualisation de séries temporelles de croissances de plantes

Mathis CORDIER

2019/2020

Projet annuel de Master 2

Tutoré par :
David ROUSSEAU



Table des matières

1 Objectifs	1
2 Architecture de l'application	2
2.1 Importation des données	2
2.2 Traitement d'images	4
2.3 Traitement de séquences d'images	4
3 Analyse d'images	5
3.1 Représentations par couche couleur	5
3.1.1 Image	5
3.1.2 Histogramme	5
3.2 Histogramme 3D : Décomposition Rouge-Vert-Bleu	6
3.3 Segmentation des plantes	7
4 Analyse de séquences d'images	10
4.1 Évolution de la couleur moyenne	10
4.2 Croissance entre 2 instants	11
4.3 Évolution de la surface de la canopée	12
5 Analyse de données de profondeur	13
5.1 Évolution de la hauteur moyenne	13
5.2 Analyse 3D de suivi quotidien	14
6 Interfaces Shiny	17
6.1 Structure et fonctionnement de <i>Shiny</i>	17
6.2 Structure et fonctionnement d'une application	18
6.2.1 Format du dossier de l'application	18
6.2.2 Programmation optimale	18
7 Pour aller plus loin	20
7.1 Inconvénients et difficultés	20
7.2 Pistes d'amélioration	20

1 Objectifs

L'objectif de ce projet est de permettre à un utilisateur spécialiste d'une autre discipline de pouvoir analyser ses séquences d'images issues du **phénotypage de plantes**. Le phénotypage est la caractérisation de l'ensemble du phénotype d'individus, résultat du génotype de ceux-ci.

Le but de ce projet est donc de proposer une **interface utilisateur** ergonomique et simple à utiliser pour qu'elle soit accessible à un maximum de personnes. Cette interface proposera un grand nombre d'outils de **visualisation de données**, allant de simples visualisations de la séquence à des outils de segmentation des plantes.

Les données, sous forme de **séquences d'images**, sont très volumineuses et l'interface devra alors permettre de faire ressortir uniquement les informations pertinentes associées à une quantification ou une caractérisation phénotypique de la croissance de la plante.

Typiquement, l'interface devra permettre à l'utilisateur de visualiser à la fois une étude du phénotype à un instant t et les évolutions de croissance de ces végétaux au cours du temps. Dans le cadre de ce projet, nous utiliserons comme **critères de croissance** la profondeur de l'image, la couleur des plantes et leur surface vue de dessus. Nous utiliserons notamment des outils de traitement de signal appliqués aux images de la séquence.

Ce rapport a pour but d'expliquer le fonctionnement de l'interface, ses outils de visualisation et le code qui permet de la faire tourner. Il permettra aussi de comprendre certains outils de visualisation nécessitant une explication plus théorique.

L'application est hébergée sur le serveur *shinyapps.io*, vous pouvez la retrouver sur <https://sithamfr.shinyapps.io/GrowthData/>. Les données utilisées dans le projet étant propriétaires, elles ne peuvent malheureusement pas être mises à disposition comme exemples pour tester l'application.



2 Architecture de l'application

L'application est codée sous R, avec la librairie *Shiny*. Elle se décompose en 2 grandes parties.

- La première partie concerne les images couleur au format RGB. Ce sont les images classiques où chaque pixel est le résultat de l'intensité du triplet Rouge-Vert-Bleu. Pour cette partie, nous avons les volets suivants :
 - Importation
 - Analyse d'images
 - Analyse de séquences d'images
- La seconde partie concerne les images de profondeur. Ces images sont en niveaux de gris, c'est-à-dire qu'elles ne sont codées que sur une valeur. Pour cette partie, nous avons les volets suivants :
 - Importation
 - Analyse de séquences d'images

L'architecture de l'interface peut être résumée par le schéma en Figure 1 :

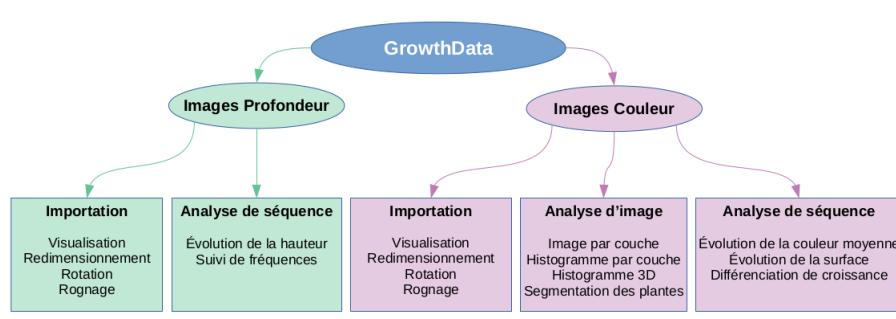


FIGURE 1 – Schéma de l'architecture de l'application

Dans cette partie, nous expliquerons chacun des volets définis précédemment.

2.1 Importation des données

Le premier volet de l'application consiste à demander à l'utilisateur d'importer ses données afin qu'elles soient traitées. Ce volet est le même que l'on soit dans le cas d'images 'Couleur' ou d'images 'Profondeur' à la différence près que l'on importe respectivement une séquence d'image RGB ou en niveaux de gris. L'application prend des séquences d'images au format TIFF comme structure de données. Ce format a l'avantage de conserver l'encodage de chaque pixel sans y appliquer de compression. De plus, il permet de stocker plusieurs images dans un même fichier. Ces atouts induisent également un coût, les fichiers *.tif* sont donc inévitablement plus lourds. Ce poids peut être trop

important pour que l'application ne le supporte. En effet, si l'application est exécutée localement, les fichiers seront mis sur la RAM du PC. La limite de l'application dépend donc des caractéristiques de la machine qui l'exécute. Si à l'inverse, l'application est exécutée sur le site d'hébergement, les restrictions de volume liées au serveur auront raison de fichiers trop volumineux.

Vous pouvez d'ailleurs utiliser un facteur de redimensionnement afin que les images prennent moins de place en mémoire. Cela induit également une perte d'information lors de l'importation. Nous avons des fichiers sous forme de tableaux à 4 dimensions :

- **Hauteur** : Hauteur de chaque image
- **Largeur** : Largeur de chaque image
- **Couches** : Couches couleur RGB (3) ou couche de niveaux de gris (1) contenant l'intensité des pixels
- **Temps** : Nombre d'images de la séquence

Il ne semble pas intéressant de réduire, dès l'importation, l'information sur les dimensions de temps et de couches car la perte d'information serait trop importante. Cependant, si l'utilisateur possède des images à haute résolution, l'information perdue lors d'un redimensionnement est négligeable, à condition que le facteur choisi soit raisonnable. De plus, l'application ne pourra pas supporter des images à trop hautes résolutions.

Un point très important de l'interface concerne l'utilisation des données. Lorsque vous exécutez l'application localement, vous n'êtes pas concernés par le problème de collecte de données car vos fichiers restent sur votre machine. Lorsque vous l'exécutez sur le serveur d'hébergement de l'application, vous envoyez vos données au serveur pour qu'elles soient traitées par l'interface. L'application garantit que ces données ne sont pas conservées, ni par l'hébergeur, ni par le créateur.

Si vous souhaitez également tourner vos images ou les rogner, l'application le permet en cliquant sur le bouton correspondant dans la fenêtre de visualisation de la séquence. Sur l'application nous avons alors les outils suivants en Figure 2 :



FIGURE 2 – Outils de recadrage des images

On choisit ici l'angle de rotation, l'intervalle en largeur et en hauteur. On peut alors résumer les fonctionnalités de ce volet, par l'image en Figure 3.

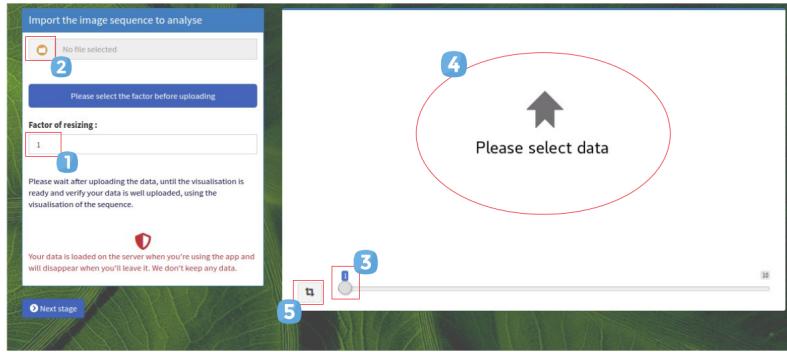


FIGURE 3 – Schéma d'utilisation du volet d'importation

- ① Sélection du facteur de redimensionnement
- ② Choix du fichier à importer
- ③ Choix du temps
- ④ Fenêtre de visualisation
- ⑤ Outil de rotation/rognage

2.2 Traitement d'images

Ce volet n'est présent que dans le cas des images 'Couleur'. Il permet d'analyser une image isolée de la séquence. Pour cela, elle donne à voir un grand nombre de visualisations :

- Scission des couches couleurs :
 - Visualisation de l'image par couche couleur
 - Histogramme de la couche couleur
- Histogramme 3D : représentation Rouge-Vert-Bleu
- Segmentation des plantes

Ces outils de visualisation seront détaillés dans la partie qui leur est dédiée.

2.3 Traitement de séquences d'images

Ce volet diffère si l'on utilise des images 'Couleur' ou des images 'Profondeur'.

- Dans le cas d'images 'Couleur' :
 - Évolution de la surface de la canopée (vue de dessus)
 - Évolution de la couleur moyenne
 - Croissance entre 2 instants
- Dans le cas d'images 'Profondeur' :
 - Évolution de la hauteur moyenne de la canopée
 - Évolution 3D : Point de vue fréquentiel

Ces outils de visualisation seront également détaillés dans la partie qui leur est dédiée.

3 Analyse d'images

Dans cette partie, nous présenterons les outils de visualisation proposés afin d'extraire les informations pertinentes des images de la séquence. Dans cette partie, nous fixons donc la dimension de temps. Il nous reste alors les 3 dimensions de hauteur, largeur et couches couleur.

3.1 Représentations par couche couleur

Nous pouvons alors visualiser différentes informations exprimées par opérations sur les couches couleur de l'image sélectionnée. Pour bien comprendre la manière dont sont formées les données, nous avons, pour chaque pixel de l'image, un triplet de valeurs codant la couleur.

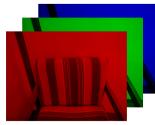


FIGURE 4 – Décomposition Rouge-Vert-Bleu d'une image

Ces valeurs sont par défaut des entiers compris entre 0 et κ_{Max} , où $\kappa_{Max} = 2^\beta$ et β est le nombre de bits sur lequel est codée l'image. Ainsi, pour une image codée sur 8 bits, nous aurons des intensités comprises entre 0 et 255. Ces intensités peuvent être divisées par κ_{Max} afin de standardiser les données entre 0 et 1. Cette opération a pour inconvénient de transformer des nombres de type entier en nombres de type réel qui prend plus de place en mémoire. Pour le programme codant l'interface, nous conservons 2 décimales après la virgule afin de minimiser cet impact négatif.

Nous obtenons alors la couleur par l'opération suivante sur les couches couleur :

$$[0, 1] \times [0, 1] \times [0, 1] \rightarrow \text{couleur}$$

Cette opération étant une bijection, un triplet composant les couches couleur engendre donc une unique couleur et inversement. De cette manière, nous pouvons décomposer une couleur en un triplet de valeurs.

3.1.1 Image

Ce type de représentation permet simplement de représenter chacune des couches couleur au choix. On peut ainsi distinguer quelles couleurs primaires permettent de distinguer ou extraire les plantes du fond de culture, voire d'éventuels objets pouvant être présents de manière récurrente au long de la séquence ou de manière exceptionnelle.

3.1.2 Histogramme

De cette manière, nous pouvons également donner la distribution des intensités de pixel pour la couleur sélectionnée. Ainsi, une fois les dimensions de

temps et de couche couleur fixées, nous obtenons une distribution d'intensités comprises entre 0 et 1. Plus la valeur est proche de 1, plus l'intensité de la couleur est importante sur le pixel. Nous représentons donc le nombre d'occurrences de chaque intensité de la couleur.

3.2 Histogramme 3D : Décomposition Rouge-Vert-Bleu

Afin de visualiser un maximum d'informations, on peut également représenter un histogramme en 3 dimensions. Pour cela, nous scindons les couleurs pour obtenir le triplet de valeurs codant. Nous divisons ensuite le cube unité en 1 millions de cube plus petit, soit 100 intervalles par couche couleur, afin de compter le nombre de pixels dont la couleur appartient à l'intervalle concerné.

L'utilisateur choisit ensuite le seuil qu'il veut appliquer. On trace alors les points dont le nombre d'occurrences est supérieur au seuil choisi. La taille des points dépend alors du nombre d'occurrences.

Le seuillage permet de ne faire ressortir que les informations vraiment nécessaires mais également de maintenir un espace mémoire raisonnable. En effet, lorsque le seuil est à 0, c'est-à-dire que nous n'appliquons pas de seuillage, nous devons tracer tous les points dans le cube unité 3D. Le graphique devient alors très lourd avec l'utilisation de *plotly* pour pouvoir tourner ou zoomer dans la figure.



FIGURE 5 – Choix du seuil d'occurrences de couleurs dans l'image

Une fois le seuil défini (Figure 5), on représente la décomposition obtenue en un nuage de points 3D. Cela permet de visualiser directement (Figure 6), les couleurs importantes de l'image. Typiquement, en faisant évoluer le temps dans nos séquences de croissance de plantes, on s'attend à obtenir une croissance du volume englobant du nuage de points qui contient les teintes vertes.

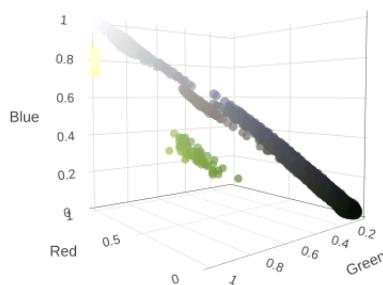


FIGURE 6 – Histogramme 3D des couleurs de l'image

3.3 Segmentation des plantes

Afin d'analyser la manière dont évoluent les plantes de la canopée, il est important de pouvoir isoler ces dernières du fond de culture. Pour cela, nous utiliserons des opérations sur les couches couleur de l'image afin d'isoler les plantes du reste de l'image. Ainsi, nous souhaitons obtenir un **masque de segmentation binaire** des plantes de l'image.

Pour commencer, il est nécessaire d'effectuer quelques opérations élémentaires sur les images. En effet, un ajustement pertinent de la luminosité et du contraste permet de tout de suite séparer les teintes souhaitées du reste de l'image. Tout d'abord, il est important de bien expliquer les opérations auxquelles correspondent des modifications de luminosité et de contraste.

- Luminosité : $\forall p_{xy} \in [0, 1]^3$ pixel en position (x, y) , $\phi(p_{xy}) = p_{xy} + \begin{pmatrix} \gamma \\ \gamma \\ \gamma \end{pmatrix}$
- Contraste : $\forall p_{xy} \in [0, 1]^3$ pixel en position (x, y) , $\psi(p_{xy}) = \delta \times p_{xy}$

Nous pouvons donc définir la fonction d'ajustement linéaire de l'image notée $\varphi = \psi \circ \phi$. Alors, $\forall p_{xy} \in [0, 1]^3$ pixel en position (x, y) ,

$$\varphi(p_{xy}) = \delta \times (p_{xy} + \begin{pmatrix} \gamma \\ \gamma \\ \gamma \end{pmatrix}).$$

Soit,

$$\varphi \begin{pmatrix} p_{xy}^{(1)} \\ p_{xy}^{(2)} \\ p_{xy}^{(3)} \end{pmatrix} = \begin{pmatrix} \delta \cdot (p_{xy}^{(1)} + \gamma) \\ \delta \cdot (p_{xy}^{(2)} + \gamma) \\ \delta \cdot (p_{xy}^{(3)} + \gamma) \end{pmatrix}$$

Cependant, la difficulté réside dans le fait que les images sont prises à différents moments avec un éclairage ou une exposition au soleil qui peuvent être fluctuants. Ainsi, le réglage optimal des paramètres de modification de la luminosité et du contraste ne sont pas les mêmes à tous les instants. Le choix de ces paramètres est donc difficile, il faut les choisir afin qu'ils soient efficaces à tous les instants. Pour les données que nous étudions actuellement, nous avons choisi $\gamma = -0.4$ et $\delta = 2$.

$$\text{Nous obtenons alors } \varphi \begin{pmatrix} p_{xy}^{(1)} \\ p_{xy}^{(2)} \\ p_{xy}^{(3)} \end{pmatrix} = \begin{pmatrix} 2 \cdot (p_{xy}^{(1)} - 0.4) \\ 2 \cdot (p_{xy}^{(2)} - 0.4) \\ 2 \cdot (p_{xy}^{(3)} - 0.4) \end{pmatrix}.$$

Cela implique que $\varphi(p_{xy}) \in [-0.8, 1.2]^3$ car la transformation linéaire translate et dilate l'espace de départ. Par un seuillage dur, nous revenons dans l'espace de départ $[0, 1]^3$, i.e.

$$\forall i \in \{1, 2, 3\}, \forall p_{xy}^{(i)} = \begin{cases} 0 & \text{si } p_{xy}^{(i)} < 0 \\ 1 & \text{si } p_{xy}^{(i)} > 1 \\ p_{xy}^{(i)} & \text{sinon} \end{cases}$$

Cette transformation permet d'isoler les pixels recherchés. Ainsi, on obtient le schéma de transformation suivant en Figure 7 :

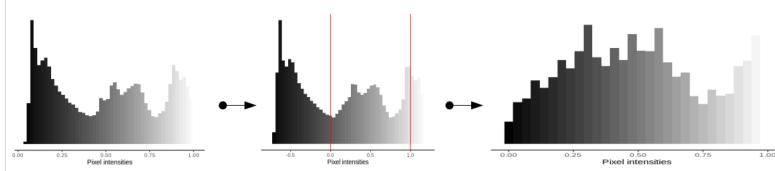


FIGURE 7 – Transformation Contraste-Luminosité de l'image

Une fois l'ajustement de l'image effectué, il faut définir des critères de sélection des pixels correspondant aux plantes. La difficulté est d'être capable de trouver une combinaison de critères permettant d'extraire les plantes du fond de culture mais aussi d'éventuels objets dont la présence peut être permanente sur certaines caméras ou momentanée. La couleur des plantes est majoritairement composée de vert et de rouge avec très peu de bleu. Sur nos données, nous avons un fond de culture noir, des murs blancs et des étiquettes dont la tige est bleue et le bout est blanc. Afin de séparer ces objets de l'objet 'Plante' que l'on veut extraire, nous définissons les critères suivants :

Soient I_R, I_G et I_B , les intensités respectives des couches rouge, verte et bleue d'un pixel. On définit alors les critères de sélection suivants :

- Critère 1 : $2 * I_G - I_R - I_B > 0.6$



FIGURE 8 – Image d'intensités $2I_G - I_R - I_B$ et son seuillage par scission en 0.6

- Critère 2 : $I_G - I_R \in [0.04, 0.4]$

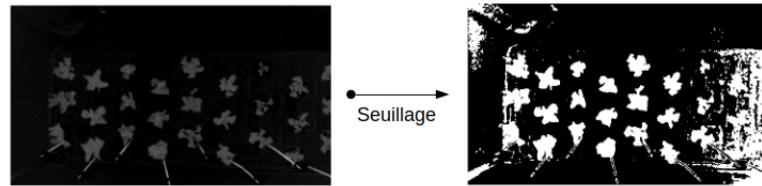


FIGURE 9 – Image d'intensités $I_G - I_R$ et son seuillage par isolation de l'intervalle en $[0.04, 0.4]$

Par ces transformations sur I_R, I_G et I_B de :

- $[0, 1]^3 \rightarrow [-2, 2]$ (Critère 1)
- $[0, 1]^2 \rightarrow [-1, 1]$ (Critère 2)

Nous obtenons des images en niveaux de gris (1 dimension). Une fois ces niveaux d'intensités obtenus, nous seuillons afin d'extraire l'objet voulu. On voit que certains objets comme le tuyau d'arrosage, l'arrosoir, des reflets ou des étiquettes peuvent être éliminés par ces critères. La majorité de ces objets peuvent d'ailleurs être éliminés par le rognage par l'utilisateur dans la partie importation des données. En effet, ces objets sont majoritairement présents hors du banc de culture vu de dessus. D'autres critères peuvent être ajoutés au modèle afin d'adapter le modèle à d'éventuels objets n'étant pas éliminés par ceux déjà présents. Cependant, l'objectif est de minimiser le nombre de critères afin d'optimiser l'efficacité du modèle et de réduire le coût de calcul.

Résultat final

Par combinaison de ces 2 critères sur tous les pixels de l'image, on obtient la transformation suivante en Figure 10 :



FIGURE 10 – Image originale et son masque de segmentation associé

On peut ensuite appliquer le masque de segmentation binaire des plantes sur l'image originale afin de ne tracer que la canopée et obtenir la figure finale comme sur l'image suivante en Figure 11 :



FIGURE 11 – Masque de segmentation appliqué à l'image originale

Ainsi, nous retrouvons l'outil de visualisation implémenté dans l'interface sans que l'utilisateur n'ait à paramétriser le modèle utilisé. Des opérations morphologiques de base sur le masque binaire peuvent compléter le rendu. En effet, la coupe vue de dessus de la canopée expose les plantes à d'éventuels obstacles dans la prise de vue. C'est le cas ici avec les étiquettes notamment. On pense donc typiquement à une fermeture, c'est-à-dire une dilatation de la surface de l'objet détectée (binaire) suivie d'une érosion. Ces opérations sont à effectuer avec parcimonie car elles transforment la forme obtenue par le masque. De plus, lorsque la taille des plantes devient importante, une fermeture engendrerait une erreur dans le masque lorsque l'intervalle entre deux plantes devient faible.

4 Analyse de séquences d'images

Dans cette partie, nous présenterons les outils de visualisation proposés afin d'extraire les informations pertinentes de la séquence en travaillant sur la dimension de temps. L'analyse de séquences d'images dépend grandement de celle effectuée sur les images dans la partie précédente. En effet, nous utilisons la sélection d'informations effectuée antérieurement image par image afin de distinguer d'éventuelles évolutions dans la séquence.

4.1 Évolution de la couleur moyenne

Tout d'abord, il est utile d'étudier la couleur moyenne des images au cours du temps car nous supposons que la teinte des plantes évolue au cours de sa croissance. Cependant, le fond de culture apporte un biais assez important sur cette couleur moyenne si nous prenons la totalité de l'image en compte dans le calcul. Ainsi, nous étudions l'évolution de cette couleur moyenne à la fois sur la totalité de l'image et également sur les pixels de l'image correspondant au masque de segmentation des plantes. Une couleur étant représentée par son triplet codant Rouge-Vert-Bleu, nous représentons cette couleur moyenne dans l'espace 3 dimensions correspondant. Nous obtenons donc deux évolutions de couleur moyenne au cours du temps que nous représentons de la manière suivante :

- La couleur des lignes correspond au type de calcul associé :
 - Toute l'image
 - Image segmentée
- La couleur des points correspond à la couleur moyenne associée
- En survolant les points, nous obtenons la valeur de temps et le type de calcul associés au point

Nous obtenons alors la représentation suivante en Figure 12 :

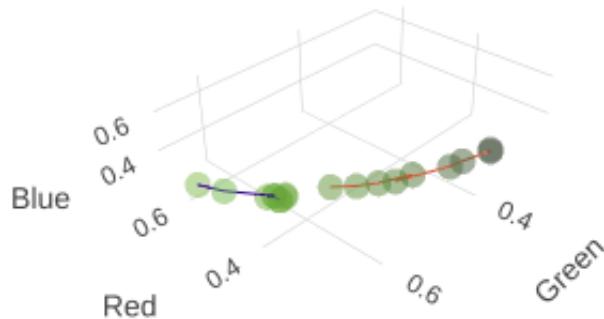


FIGURE 12 – Évolution de la couleur moyenne en fonction du temps

4.2 Croissance entre 2 instants

Grâce au masque de segmentation obtenu précédemment, nous pouvons différencier la croissance entre 2 instants de la séquence. Pour cela, nous calculons le masque aux 2 instants sélectionnés par l'utilisateur (Figure 13) et nous représentons les 2 masques superposés afin de comparer leur surface (Figure 14).



FIGURE 13 – Choix de l'intervalle de temps à étudier

Ainsi, nous représentons la superposition des deux masques avec en rouge la valeur de temps la plus faible et en bleu la plus élevée.

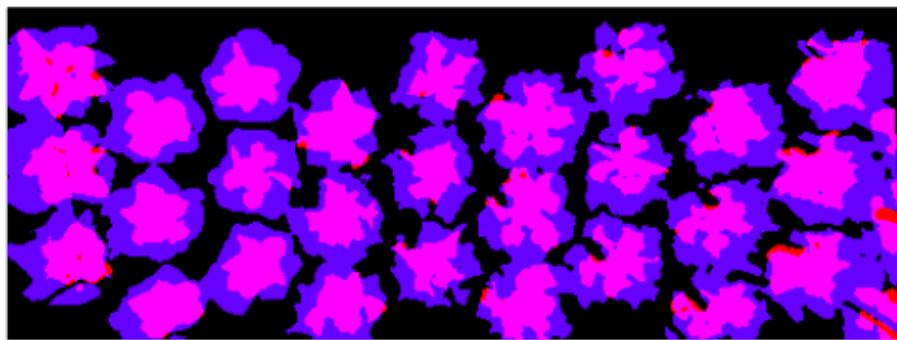


FIGURE 14 – Comparaison de croissance entre deux instants

Pour accompagner l'analyse de l'utilisateur, un indicateur de croissance (Figure 15) permet à l'utilisateur de visualiser le taux de croissance entre les deux instants, ainsi que le nombre de pixels au temps de départ et de fin.

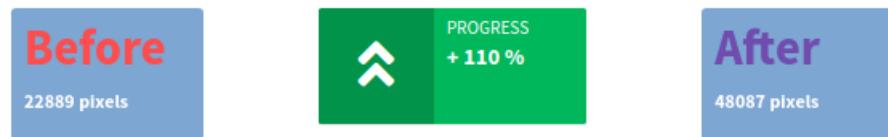


FIGURE 15 – Indicateur de croissance

Cet indicateur évolue en fonction du taux de croissance calculé. En effet, l'utilisateur obtient tout de suite visuellement les informations sur la croissance des plantes de sa séquence avec une représentation évolutive intuitive des couleurs et des icônes.



FIGURE 16 – Différents exemples d’indicateurs

Les résolutions des images des séquences importées pouvant être différentes, cet outil s’adapte à ces disparités car il ne dépend pas du nombre de pixels en lui-même mais du ratio entre le nombre de pixels aux temps nommés *After* et *Before*.

4.3 Évolution de la surface de la canopée

En utilisant ce même masque de segmentation obtenu dans la partie d’analyse d’images, nous pouvons récupérer le nombre de pixels y étant allumés et les sommer. Ainsi, en procédant sur l’ensemble des images de la séquence, nous déduisons la surface de la canopée en fonction du temps. Ces surfaces sont exprimées en pourcentage de la surface totaleⁱ des images de la séquence.

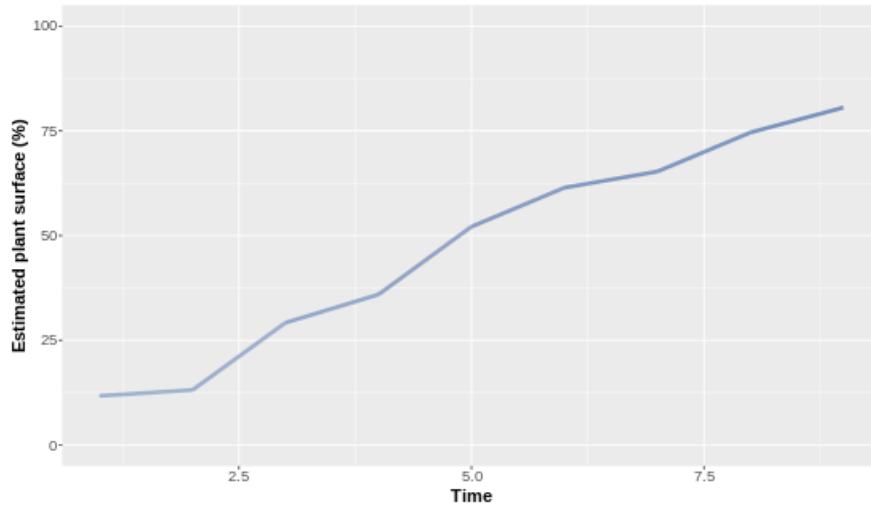


FIGURE 17 – Surface de la canopée en fonction du temps

Il est alors évident de déduire une croissance sur la Figure 17, en surface vue du dessus, des plantes de la canopée.

i. Nombre de pixels de l’image

5 Analyse de données de profondeur

De la même manière qu'avec les images couleur étudiées précédemment, nous pouvons analyser les images traduisant des intensités de profondeur. Ces images peuvent être obtenues de différentes manières : caméras stéréoscopiques, triangulations laser... Représentant la distance à la caméra, elles sont faites de niveaux de gris (1 dimension). Ainsi, plus un pixel est foncé, plus il est proche de la caméra et inversement.

5.1 Évolution de la hauteur moyenne

Afin d'établir une possible croissance en hauteur des plantes de la canopée, nous utilisons les images de profondeur.

Ces images comportant directement les distances à la caméra, nous récupérons la distance moyenne à la caméra pour chaque image de la séquence et nous obtenons l'évolution au cours du temps de ces distances. On obtient alors des croissances de la forme de la Figure 18 :

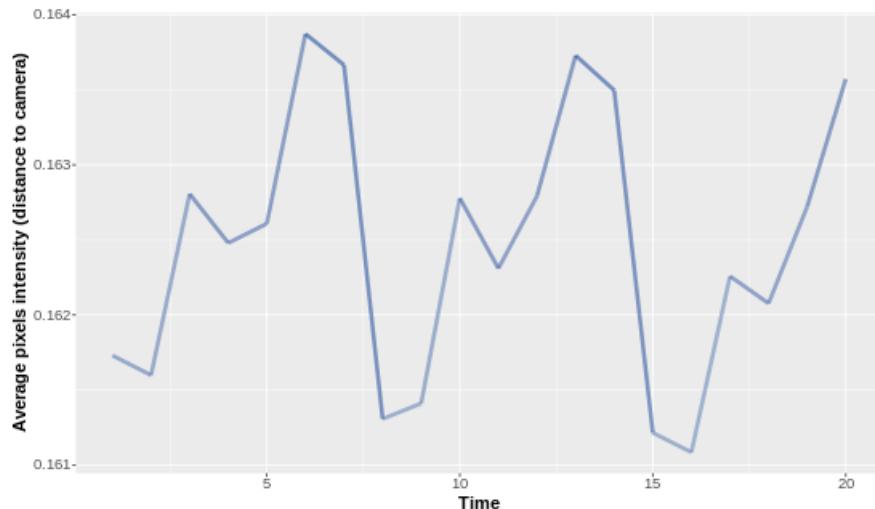


FIGURE 18 – Distance à la caméra en fonction du temps

En effet, si les prises de vue sont faites régulièrement tous les δ_x temps (en minutes) et que $1440^{ii} \bmod \delta_x = 0$, nous voyons ressortir une périodicité due aux cycles circadiens.

Les cycles circadiens sont les cycles de 24 heures consécutives qui dépendent de la lumière du jour et de la nuit. En effet, la nuit la plante aura tendance à se tasser et se redresser ou s'ouvrir plus ou moins en fonction du moment de la journée et de son exposition à la lumière.

On voit alors apparaître l'importance de prendre en compte ces différences dues à la périodicité dans l'étude de la croissance en hauteur des plantes.

ii. 1440 est le nombre de minutes par jour

5.2 Analyse 3D de suivi quotidien

Dans cette sous-partie, l'objectif est d'étudier les séries chronologiques de la croissance en hauteur de la séquence jour par jour. Pour cela, nous utiliserons la périodicité associée aux **cycles circadiens**. Ces cycles évoluent notamment en fonction de la hauteur de la plante et peuvent être caractéristiques de différents types de croissance. Par transformée de Fourier, nous obtenons les valeurs dans le **domaine fréquentiel**, nous pouvons alors définir les critères que nous utiliserons ensuite. Ce domaine, se distinguant du domaine temporel habituellement utilisé, est bien plus adapté pour certains types de calcul.

La **transformée de Fourier**, notée \mathcal{F} , permet en effet de décrire le spectre fréquentiel d'un signal à valeurs dans \mathbb{R}^n en décomposant ce signal en une somme infinie de signaux sinusoïdaux. Elle est également invariante par translation, ce qui signifie que notre série n'a pas besoin de débuter à un moment particulier de la journée. Cependant, l'utilisateur devra faire attention, lors de l'importation de la séquence, à bien faire débuter sa séquence sur la première prise de vue de la journée, c'est-à-dire sur le premier instant de la période. L'objectif dans cette partie est de caractériser par les moyens suivants l'évolution du signal période par période.

Notons notre signal $s(t)$ de taille N et de période T . Soit \div la division entière, nous avons alors $N_T = N \div T$ le nombre de périodes dans le signal. Notons alors $\forall t \in [nT, (n+1)T], u(t, n) = s(t) - \mathcal{G}(n, t)$ le signal correspondant à la n -ième période à laquelle nous soustrayons sa tendance linéaire. \mathcal{G} désigne ici la tendance linéaire associée à cette période. Sa pente nous indique donc la croissance associée à la journée (période) concernée. Nous conservons l'information portée par la pente pour la visualisation finale.

Harmonique

Les harmoniques sont les composantes d'un signal périodique, dont la fréquence est un multiple d'une fréquence fondamentale ou première harmonique. La première harmonique est donc la fréquence la plus basse de toutes les harmoniques, celle qui est associée à la plus grande longueur d'onde que le signal puisse émettre. La fréquence fondamentale d'un signal est celle qui le caractérise le mieux dans le sens où elle définit le comportement du signal à la plus grande échelle. Ici, elle constitue une bonne approximation de l'amplitude des cycles circadiens.

Pour calculer cette fréquence fondamentale, nous utilisons la transformée de Fourier de u pour chaque période du signal. Nous n'avons ici besoin que de l'amplitude donc nous pouvons simplement effectuer les calculs suivants. Étant

dans un cadre discret, nous avons :

$$a(n) = \frac{2}{T} \cdot \sum_{k=nT}^{(n+1)T-1} u(k, n) \cos\left(\frac{2\pi k}{T}\right),$$

$$b(n) = \frac{2}{T} \cdot \sum_{k=nT}^{(n+1)T-1} u(k, n) \sin\left(\frac{2\pi k}{T}\right).$$

Nous définissons alors $c(n) = \sqrt{a(n)^2 + b(n)^2}$ le résultat recherché.

$c(n)$ correspond l'amplitude associée au cycle circadien de la n-ième période du signal u .

Énergie

L'énergie d'un signal x à valeurs dans I , notée E_x permet en plus des résultats précédents de capturer l'énergie relative au phénomène de réPLICATION des feuilles qui est à l'origine des motifs spectraux de hautes fréquences. Cette énergie se définit généralement de la manière suivante :

$$E_x = \int_I |x(t)|^2 dt.$$

De plus, l'**égalité de Parseval** nous dit que l'énergie d'un signal ne dépend pas du domaine choisi : temporel ou fréquentiel. Cela signifie que nous obtiendrons le même résultat avec une intégrale sur la transformée de Fourier de ce même signal.

Ici, nous sommes dans un cadre discret avec des valeurs positives, nous avons :

$$E_u(n) = \sum_{k=nT}^{(n+1)T-1} u(k, n)^2.$$

Parfois, nous pouvons également trouver l'énergie comme étant la moyenne et non la somme mais chaque période contenant le même nombre de valeur, le facteur somme/moyenne est le même pour chacune d'elles.

Taux de distorsion harmonique

Le taux de distorsion harmonique, noté *HDR* (*Harmonic Distortion Rate*), permet de quantifier la qualité d'interpolation associée à la première harmonique du signal étudié.

En effet, il permet d'établir un taux d'explication du signal par la fréquence fondamentale en comparant l'énergie totale du signal avec sa fréquence fondamentale. La qualité d'interpolation associée à cette fréquence est d'autant plus grande que le taux est proche de 0. Ainsi, nous définissons la fonction *HDR*, telle que :

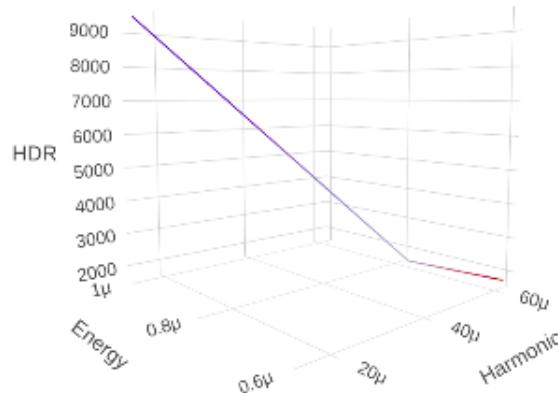
$$HDR(n) = 100 \times \sqrt{\frac{E(n) - \frac{1}{2}c(n)^2}{\frac{1}{2}c(n)^2}}$$

Ce taux peut être interprété comme étant le taux d'erreur moyenne au carré entre le signal sur une période et l'amplitude de son cycle circadien.

Représentation

On peut alors définir un type de représentation qui dépend du temps, permettant d'évaluer l'évolution de la pente, de la fréquence fondamentale, de l'énergie et du taux de distorsion harmonique, jour après jour.

Cette représentation, en Figure 19, doit permettre de visualiser ces 5 informations en un même graphique. Afin de visualiser au mieux l'information totale, nous représentons les points dans un espace 3D formé du triplet suivant : (c, E, HDR) . La pente est représentée par un gradient de couleur sur les lignes rejoignant ces points. Pour finir, la dimension de temps est présente lors du survol des points du graphique, nous y trouvons en effet toutes les informations correspondant au point ciblé.



Color lines depend on the average growth for each period
Average growth on periods : Blue is minimum and red is maximum

FIGURE 19 – Évolution des critères de croissance en fonction du temps

6 Interfaces Shiny

6.1 Structure et fonctionnement de *Shiny*

Une interface shiny se construit en 2 partie distinctes :

- **UI** : Interface utilisateur
- **Serveur** : Système de calculs

L'interface utilisateur a pour but de donner à l'utilisateur les capacités d'interagir avec le système de calcul qu'est le serveur. Ainsi, elle prend en charge les *inputs* et les transmet au serveur. Le serveur traite ces *inputs* et à partir de ceux-ci renvoie des *outputs*. Les *outputs* sont alors récupérés par la partie UI, permettant à l'utilisateur de visualiser le résultat.

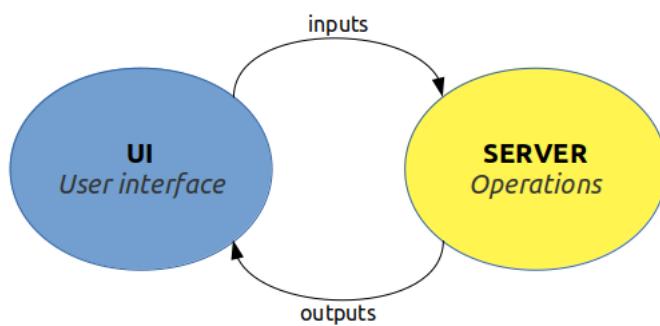


FIGURE 20 – Schéma d'interactions UI-Serveur

C'est sur cette base d'interactivité entre l'UI et le serveur qu'est conçu le package *Shiny*, comme un grand nombre d'assistants à la création d'interfaces. Ce type d'interfaces a pour but de permettre à des utilisateurs ne nécessitant pas de connaissances particulières en la matière de disposer d'un **pipeline** de traitement de données.

Ce pipeline peut ensuite être mis à disposition du plus grand nombre en étant hébergé sur des serveurs gérant *Shiny*. Ainsi, parmi les plus connus et simples à utiliser nous retrouvons les serveurs shinyapps.io mis à disposition par *RStudio* mais il est aussi possible d'utiliser des *dockers* ou *AWS* d'Amazon. Cela fonctionne alors de la manière suivante :

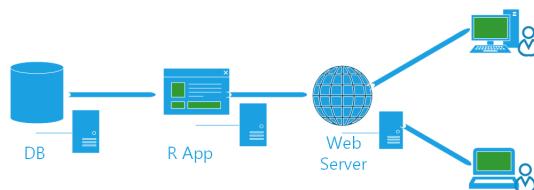


FIGURE 21 – Schéma de fonctionnement d'une application *Shiny* hébergée

Ce déploiement permet de mettre à disposition le pipeline à plusieurs utilisateurs à la fois qui n'ont pas besoin de télécharger quoi que ce soit.

Cependant, les temps de traitement et d'importation des fichiers sont plus longs avec ce procédé.

6.2 Structure et fonctionnement d'une application

Certains aspects de la programmation avec le package *Shiny* nécessitent d'être utilisés avec précaution.

6.2.1 Format du dossier de l'application

Une application *Shiny* est composée d'un dossier dont le nom est le même que celui de l'application. Dans ce dossier, nous retrouvons les codes *R* de l'application ainsi que tous les éventuels fichiers nécessaires à la compilation des codes. L'architecture à l'intérieur du dossier doit respecter un certain format afin d'être correctement interprétés par *Shiny*. Ainsi, lors de la compilation de l'application les fichiers doivent être rangés de l'une des manières suivantes :

- ◦ **ui.R** : Script R contenant la fonction *ui* de *Shiny*
- **server.R** : Script R contenant la fonction *server* de *Shiny*
- **global.R** : Script R contenant les packages à utiliser dans l'application, joignant les fichier ui.R et server.R, et définissant d'éventuelles variables globales
- **Data** : Dossier contenant les fichiers d'accompagnement (tableur, fichier texte...)
- **www** : Dossier contenant les images d'accompagnement (logo, fond d'écran...)

- ◦ **app.R** : Script R contenant l'ensemble du script R (UI, serveur, fonctions, variables globales)
- **Data** : Dossier contenant les fichiers d'accompagnement (tableur, fichier texte...)
- **www** : Dossier contenant les images d'accompagnement (logo, fond d'écran...)

6.2.2 Programmation optimale

Le programme du serveur étant composé d'une accumulation de fonctions, nous avons parfois besoin de garder en mémoire des variables ou des résultats afin de ne pas avoir à les recalculer à chaque fois. Cela pourrait être fait par l'opérateur "`<-`" de R qui permet de communiquer la valeur d'une variable dans une fonction à l'ensemble de l'environnement de travail. Cependant en *Shiny*, cela peut être la base d'erreurs et ce n'est absolument pas la méthode recommandée. Tout particulièrement lorsque l'application est hébergée, une variable globale restera alors en mémoire dans l'environnement du serveur après utilisation. Ainsi, un utilisateur qui se connecterait au site aurait déjà accès aux données importées mises en variables globales de l'utilisateur précédent.

Le package prévoit un système de mise en mémoire subordonnée à la sessionⁱⁱⁱ. Il faut alors définir les variables de la manière suivante dans le dossier de l'application :

```
global_values_for_session =  
reactiveValues(var1=NULL, var2=NULL) iv
```

On modifie ensuite ces variables dans les fonctions du serveur selon les choix de l'utilisateur de la manière suivante :

```
global_values_for_session$var1 = X
```

Il faut tout de même savoir trouver un compromis entre ce que l'on décide de conserver et ce que l'on décide de recalculer en prenant en compte le coût que cela représente soit de charger sur la mémoire soit de refaire le calcul.

iii. Une session est ouverte à chaque connexion et fermée définitivement à la déconnexion
iv. reactiveValues est une fonction du package *Shiny*

7 Pour aller plus loin

7.1 Inconvénients et difficultés

L'utilisation du package *Shiny* implique une implémentation du programme en *R*. Cependant, ce n'est ni le langage de programmation le plus adapté ni le plus développé pour le traitement d'images. En effet, il est nécessaire d'utiliser plusieurs packages pour traiter les images, un seul ne suffisant pas. De plus, ces packages ne proposent pas de traitements réellement mathématiques sur les images mais plus des modifications basiques de type filtres, rotations, rognage... On regrette donc les packages *python* tels que *skimage* ou *numpy* et *scipy* qui ont une partie développée particulièrement pour le traitement d'images. Les opérations mathématiques ont donc dû être recodées en prenant comme valeur les intensités des pixels soit dans le triplet RGB pour les images couleur soit les niveaux de gris pour les images de profondeur.

Le traitement d'images nécessite également une capacité de mise en mémoire importante. *Shiny* est capable de traiter des flux d'information importants mais l'utilisation de l'application est délicate car elle est limitée dans la taille des fichiers importés. En local, cette limite dépend de la RAM de la machine sur laquelle l'application est lancée. Hébergée sur le serveur *shinyapps.io*, cette limite dépend de la formule souscrite par le compte hébergeant l'application. Ainsi, avec la formule gratuite, nous sommes limités à un total de 1 Go tout compris (Taille du programme, packages importés, fichiers d'accompagnement, mise en mémoire par l'utilisateur). Cela peut être assez restrictif, surtout si plusieurs personnes sont connectées en même temps sur l'application. Il y aura alors une mise en mémoire des utilisateurs qui peut saturer l'application et couper l'accès au serveur.

L'utilisation du package *plotly* implique également une mise en mémoire importante. En effet, une marge de mémoire disponible est nécessaire afin que l'utilisateur utilise une des représentations effectuée par ce package. Cette surcharge de la mémoire est due à la prise en charge des mouvements en 3D : rotation, zoom, sélection... Le package nécessite également *OpenGL* et qu'une carte graphique soit intégrée à la machine lorsque l'application est lancée en local.

7.2 Pistes d'amélioration

On pourrait imaginer par la suite une modélisation 3D de la canopée afin, par exemple, d'en calculer le volume. Cela n'a pas pu être fait car elle nécessite que l'acquisition des images couleur et profondeur soit faite en même temps et que le cadrage soit également identique. Ce n'est malheureusement pas le cas des données que nous possédons actuellement.

Nous procéderions alors de la manière suivante :

- Calcul du masque de segmentation binaire sur l'image couleur
- Application du masque binaire sur l'image profondeur

On obtiendrait alors 0 partout hors des plantes et la hauteur réelle sur les plantes. Ainsi, nous aurions un champ d'intensités différentes qui modéliseraient la canopée à la manière de la Figure 22 :

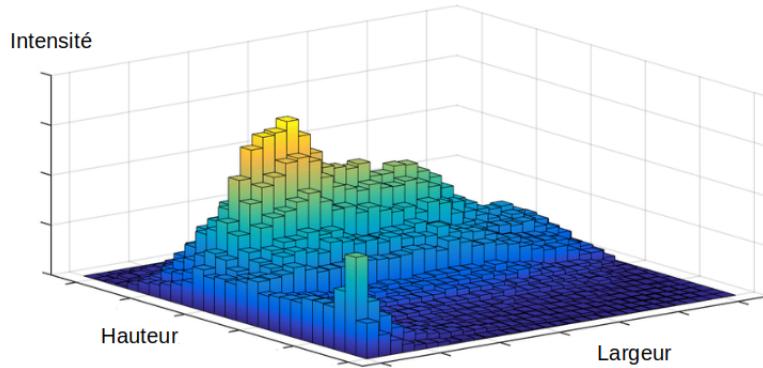


FIGURE 22 – Schéma théorique de modélisation d'une canopée

Cette modélisation ne serait pas parfaite car on ne prendrait pas en compte le fait qu'il peut y avoir du vide^v sous les feuilles de la plante mais elle serait sûrement optimale avec la manière dont sont captées les données actuellement.

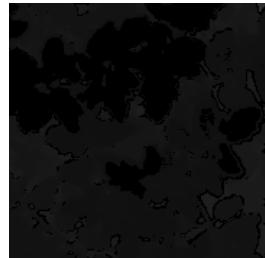
Nous pourrions également envisager un hébergement plus stable de l'application soit sur un docker soit en utilisant une formule de *shinyapps.io* plus poussée.

v. On considère comme vide l'espace non occupé par la plante

Les données

Nous avons travaillé sur ce projet avec deux répertoires de données différents.

1) Le premier concernait l'étude de différents types de croissance d'hortensias (normale, sous stress hydrique ou sous stress salin). Ces données étaient sous forme de séquences d'images de profondeur :



2) Le second concernait l'étude de croissances de salades. Ces données étaient sous forme de séquences d'images couleur et profondeur prises sur 4 caméras différentes :



Packages utilisés

- Interface : shiny, shinydashboard, shinyWidget, shinycssloaders
- Images : ijtiff, raster, imager
- Visualisation : ggplot2, hrbrthemes, plotly, scales
- Tableau de données : data.tables