

TÉLÉCOM SUDPARIS

FILTRAGE PARTICULAIRE

Valentin SIX
Théo NIEMANN

11th February 2024

RAPPELS :

Faisons un bref rappel sur le filtrage et son principe : Soit (X_n) une chaîne de Markov cachée et (Y_n) des observations. Le filtrage de (X_n) consiste à s'intéresser à $X_n|Y_{0:n}$ et estimer cette variable aléatoire. On se place donc dans le cadre suivant :

Modèle

$$X_t = f(X_{t-1}) + U_t \quad \text{modèle dynamique}$$

$$Y_t = g(X_t) + V_t \quad \text{modèle d'observation}$$

Remarque: Ce cadre est plus général que celui du TP de Kalman où f et g étaient linéaires.

Prérequis

- Espérance Mathématique

$$E[h(X)] = \int h(x)p(x)dx$$

- Principe de Monte Carlo

$$E[h(X)] \approx \frac{1}{N} \sum_{i=1}^N h(X_i) \quad \text{p.s lorsque } N \rightarrow \infty$$

où $X \sim p$, X_1, \dots, X_N iid $\sim p$

On a

$$e = \int f(x)p(x)dx$$

On construit la mesure suivante :

$$\hat{p}(dx) = \frac{1}{N} \sum_{i=1}^N \delta_{X_i}(dx)$$

On remplace donc e dans l'intégrale

$$\hat{e} = \int f(x)\hat{p}(dx) = \int f(x) \left(\frac{1}{N} \sum_{i=1}^N \delta_{X_i}(dx) \right) = \frac{1}{N} \sum_{i=1}^N f(X_i)$$

car

$$\int f(x)\delta_{X_i}(dx) = f(X_i)$$

RAPPELS

PRINCIPE D'ÉCHANTILLONNAGE D'IMPORTANCE

On a

$$e = \int f(x) \frac{p(x)}{q(x)} q(x) dx = \mathbf{E}_q \left[f(x) \frac{p(x)}{q(x)} \right]$$

a

$$\hat{e} = \frac{1}{N} \sum_{i=1}^N \frac{f(z_i)p(z_i)}{q(z_i)} = \frac{1}{N} \sum_{i=1}^N w_i f(z_i)$$

où les z_i sont des réalisations de q avec

$$w_i = \frac{p(z_i)}{q(z_i)}$$

On a

$$\hat{p}(dx) = \frac{1}{N} \sum_{i=1}^N w_i \delta_{z_i}(dx)$$

avec

$$w_i = \frac{\tilde{w}_i}{\sum_{j=1}^N \tilde{w}_j}$$

où

$$\tilde{w}_i = \frac{p(z_i)}{q(z_i)}$$

RAPPELS

PRINCIPE DU FILTRAGE PARTICULAIRE

$\hat{X}_t \sim \hat{p}(X_t|Y_{0:t})$ par Monte-Carlo

et $\hat{X}_{t|t} = \frac{1}{N} \sum_{i=1}^N X_t^{(i)}$ avec $X_t^{(i)}$ iid selon $p(X_t|Y_{0:t})$

Néanmoins, on a un problème: On ne sait pas faire des tirages de la loi $p(X_t|Y_{0:t})$.
On suppose à l'instant $t - 1$ pour tout $i \in [1, N]$:

$$X_{t-1}^{(i)} \sim \hat{p}(X_{t-1}|Y_{0:t-1})$$

Selon la formule du filtre bayésien:

$$p(dx_t|Y_{0:t}) = \frac{p(Y_t|x_t) \int p(dx_t|X_{t-1})p(dx_{t-1}|Y_{0:t-1})}{p(Y_t|Y_{0:t-1})} \quad (*)$$

On remplace à présent la loi par son estimation.

On obtient :

$$\hat{p}(dx_{t-1}|Y_{0:t-1}) = \frac{1}{N} \sum_{i=1}^N \delta_{X_{t-1}^{(i)}}(dx_{t-1})$$

Donc on a:

$$\begin{aligned} (*) &\approx \frac{p(Y_t|x_t) \int p(dx_t|X_{t-1}) \frac{1}{N} \sum_{i=1}^N \delta_{X_{t-1}^{(i)}}(dx_{t-1})}{p(Y_t|Y_{0:t-1})} \\ &\approx \frac{p(Y_t|X_t) \frac{1}{N} \sum_{i=1}^N p(dx_t|X_{t-1}^{(i)})}{p(Y_t|Y_{0:t-1})} \end{aligned}$$

À présent, utilisons cette loi comme loi d'importance

$$p(X_t|X_{t+1}, p(X_{t+1}|Y_{0:t}), p(X_{t+1}, Y_{0:t}) \cdot dX_{t+1} = p(X_t|Y_{0:t+1})$$

RAPPELS

PRINCIPE DU FILTRAGE PARTICULAIRE

On effectue à présent les différentes étapes de l'algorithme du filtrage particulaire.

On commence par calculer :

Le ratio des densités de probabilités

$$\frac{p(X_t|Y_{0:t})}{p(X_t|Y_{0:t+1})} = \frac{p(Y_{t+1}|X_t)}{p(Y_{t+1}|Y_{0:t})} \quad \text{on appelle cette formule le ratio d'importance.}$$

Puis on effectue l'étape suivante :

Le tirage suivant $p(X_t|Y_{0:t})$

On a

$$p(dx_t|Y_{0:t+1}) \propto p(dx_t|X_{t+1})p(X_{t+1}|Y_{0:t})$$

$$\hat{p}(dx_t|Y_{0:t+1}) \propto \frac{1}{N} \sum_{i=1}^N \delta_{X_t^{(i)}}(dx_t)$$

car

$$\hat{p}(dx_t|Y_{0:t+1}) = \frac{1}{N} \sum_{i=1}^N \delta_{X_t^{(i)}}(dx_t)$$

Cette formule constitue notre estimation à présent passons à l'étape suivante la propagation :

Propagation

$$X_t^{(i)} \sim \hat{p}(X_t|X_{t-1}^{(i)})$$

Passons au reweighting

Reweighting

$$w_t^{(i)} = \frac{1}{k} \frac{p(Y_{t+1}|X_t^{(i)})}{p(Y_{t+1}|Y_{0:t})}$$

et

$$w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^N \tilde{w}_t^{(j)}}$$

Ceci conclut nos rappels sur le principe du filtrage particulaire et les aspects théoriques en lien avec ce concept. Mettons en œuvre ce principe à présent.

CHAPTER 1

EXERCICE 1 : MODÈLE DE KITAGAWA

Commençons par étudier le modèle de Kitagawa. Il se définit de la manière suivante :

$$\begin{cases} X_n = 0.5X_{n-1} + \frac{25}{1+X_{n-1}^2} + 8\cos(1.2(n)) + U_n \\ Y_n = \frac{X_n^2}{20} + V_n \end{cases}$$

où $U_n \sim \mathcal{N}(0, Q)$ et $V_n \sim \mathcal{N}(0, R)$. Tous les bruits sont indépendants entre eux. On cherche à estimer la variable X_n à chaque instant.

Le modèle semble de prime abord difficile à estimer car la fonction $f(x) = x^2$ n'est ni linéaire ni bijective ainsi à partir de Y_n on a accès à X_n^2 mais pas X_n .

Les lois de $f_{n|n-1}(x_n|x_{n-1})$ et $g_n(y_n|x_n)$ sont données par :

$$\begin{aligned} f_{n|n-1}(x_n|x_{n-1}) &\sim \mathcal{N}\left(0.5X_{n-1} + \frac{25}{1+X_{n-1}^2} + 8\cos(1.2(n)), Q\right) \\ g_n(y_n|x_n) &\sim \mathcal{N}\left(\frac{X_n^2}{20}, R\right) \end{aligned}$$

Voici le résultat des observations à la suite de la création du modèle sur Python :

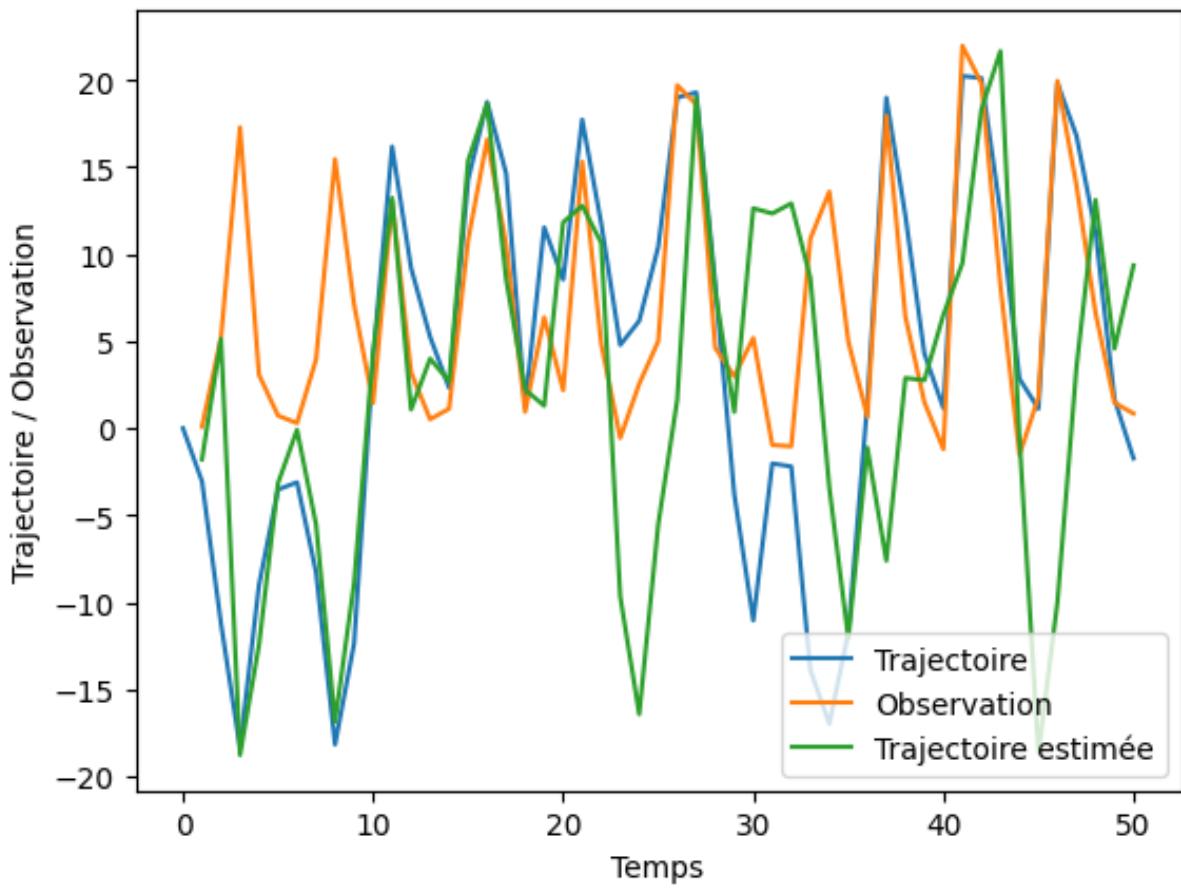


FIGURE 1.1
Courbe de nos observations, notre trajectoire et notre trajectoire estimée

On peut observer quelques anomalies, notamment dues à une qualité d'observations plutôt moyenne. Cela dit, l'erreur moyenne de notre trajectoire estimée oscille entre 7 et 10 et varie peu en fonction du nombre de particules.

C'est donc une estimation pertinente. Le temps d'exécution croît avec le nombre de particules ce qui est cohérent (2 secondes pour 50000 particules contre 1 centième de seconde pour 50 particules).

Nous allons maintenant nous attaquer à une tâche plus ardue, tenter de faire des prédictions pour effectuer un suivi de visage.

CHAPTER 2

EXERCICE 2 : SUIVI DE VISAGE SUR UNE SÉQUENCE VIDÉO

À présent, on cherche à effectuer un suivi de visage sur une séquence vidéo.

Le modèle proposé est le suivant :

$$X_n = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \times X_{n-1} + U_n$$

où $U_n \sim \mathcal{N}(0_{2 \times 2}, \begin{pmatrix} C_1 & 0 \\ 0 & C_2 \end{pmatrix})$.

C'est un modèle simple à vitesse constante, qui suppose que la position de l'objet évolue linéairement avec le temps. En effet, en regardant la matrice d'état, on observe des 1 sur la diagonale, indiquant que la position à l'état suivant est la même que l'état actuel auquel on ajoute un bruit. On peut supposer que c'est une approximation raisonnable si le mouvement de l'objet sur la séquence vidéo est fluide et sans changements brusques.

Cela dit, ce modèle pourrait ne pas convenir aux scénarios dans lesquels le mouvement de l'objet est plus complexe, impliquant une accélération ou des changements soudains de direction. Pour améliorer le modèle d'état, on peut tenter d'incorporer l'accélération dans le vecteur d'état, conduisant à un modèle dynamique plus complexe pouvant prendre en compte les changements de vitesse et de direction.

On peut aussi utiliser un modèle capable de s'adapter aux changements dans la dynamique de l'objet, en appliquant des techniques de machine learning si on possède une grande quantité de données d'entraînement. Comme expliqué précédemment, si le mouvement de l'objet possède une forte accélération alors notre modèle simple à vitesse constante peut ne pas réussir à prédire avec précision la nouvelle position de l'objet. Les particules peuvent se disperser trop largement ou ne pas suivre l'objet de près, entraînant une perte de suivi.

Pour résoudre ce problème, on peut tenter d'augmenter le bruit du processus dans l'étape de prédiction pour permettre une plus large propagation de particules, ce qui pourrait capturer des mouvements plus dynamiques. On pourrait utiliser un modèle de bruit s'adaptant aux mouvements rapides détectés dans les images précédentes et augmentant ou diminuant la variance en conséquence. Enfin, si on utilise les modèles plus précis mentionnés auparavant qui prennent en compte l'accélération, alors notre nouveau

modèle devrait pouvoir prédire la position de l'objet avec une meilleure précision.

Nous avons donc choisi la séquence 2 car le mouvement de l'étudiant évolue très linéairement. Notons que, si l'on ne suit pas l'ordonnancement des images (découpe de la vidéo en image) et donc qu'il y a des changements brusques de position alors nos particules et notre rectangle ont beaucoup de mal à suivre la cible.

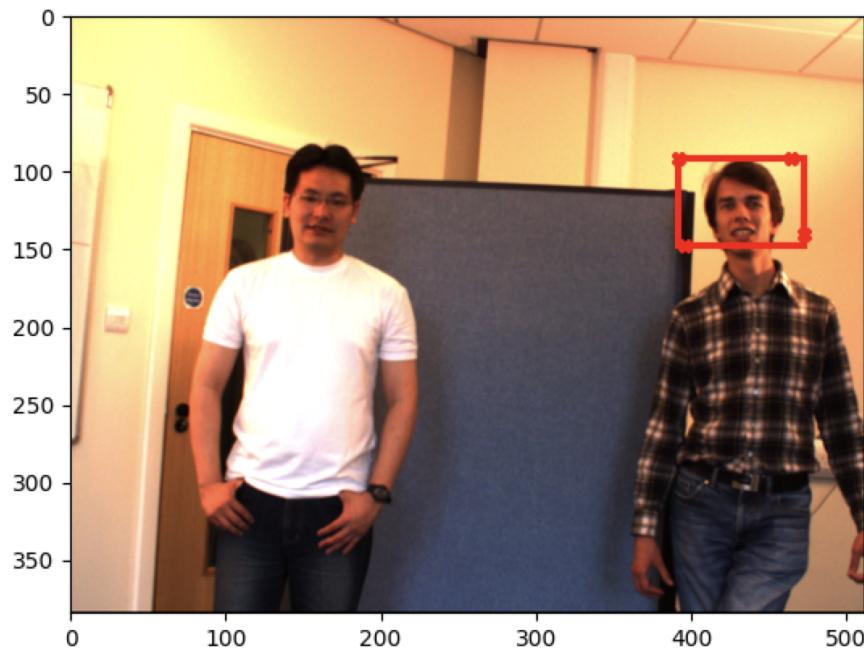


FIGURE 2.1
Début du suivi, on choisit nos points et on a ce rectangle

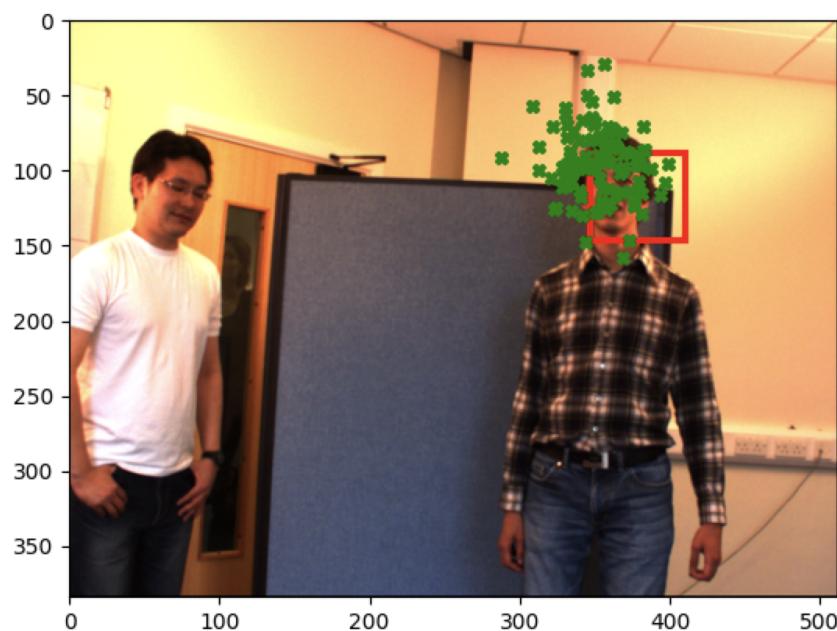


FIGURE 2.2
Suivi après une dizaine d'images

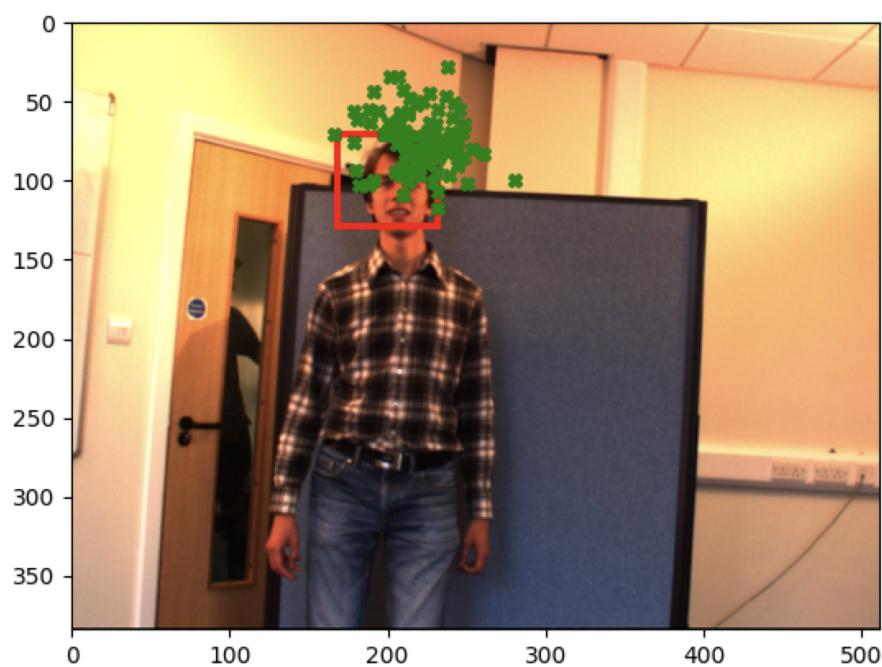


FIGURE 2.3
Suivi après une vingtaine d'images

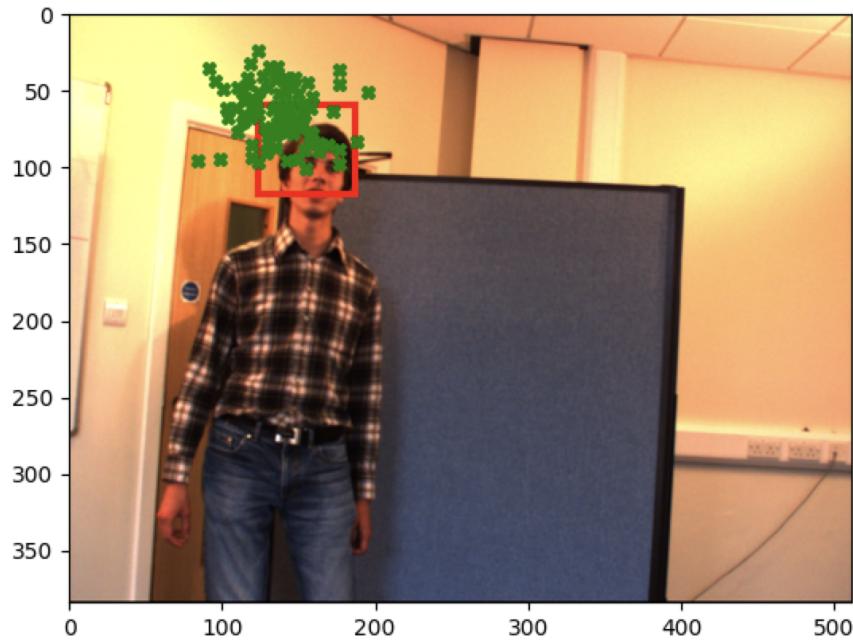


FIGURE 2.4
Suivi après une trentaine d'images

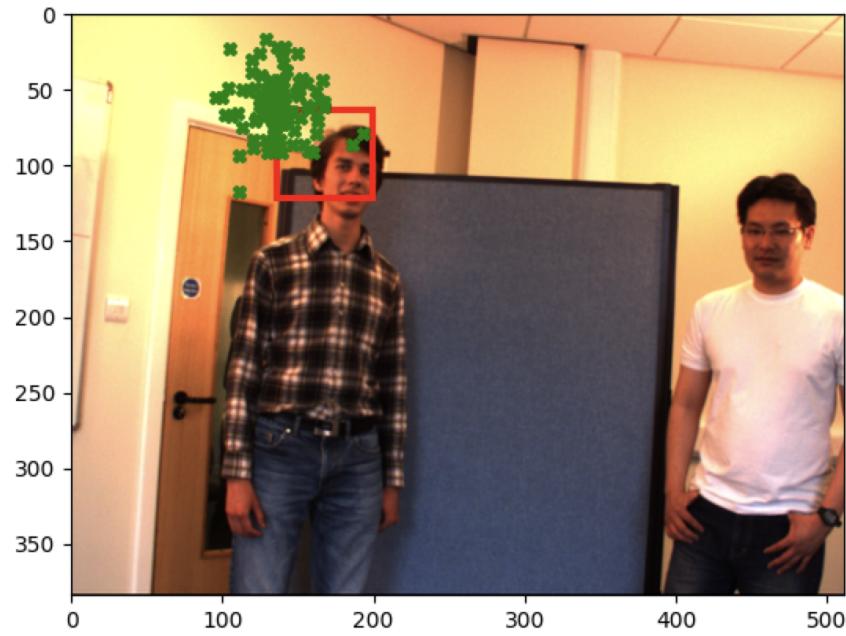


FIGURE 2.5
Suivi à la fin des images

Si nous choisissons la séquence 3 où il y a bien plus de personnes qui se déplacent et qui se croisent alors on peut observer les phénomènes suivants :

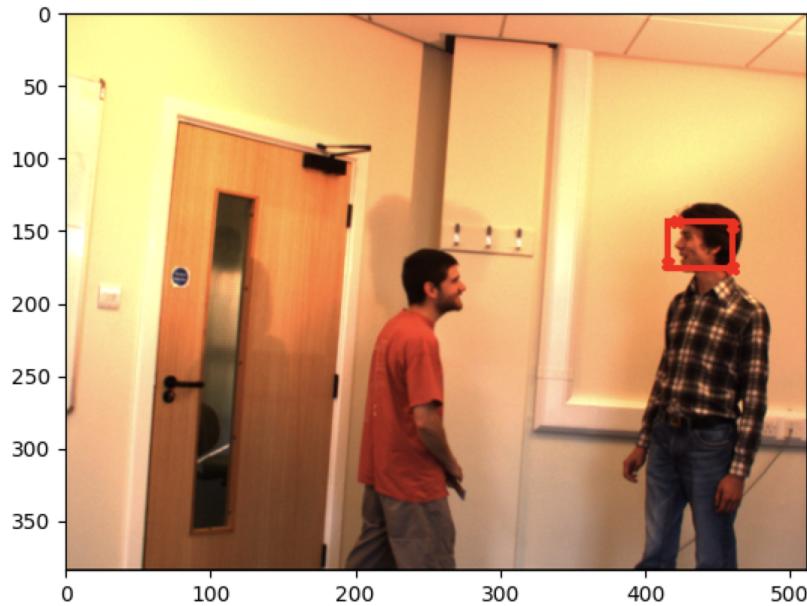


FIGURE 2.6
Début avec deux personnes

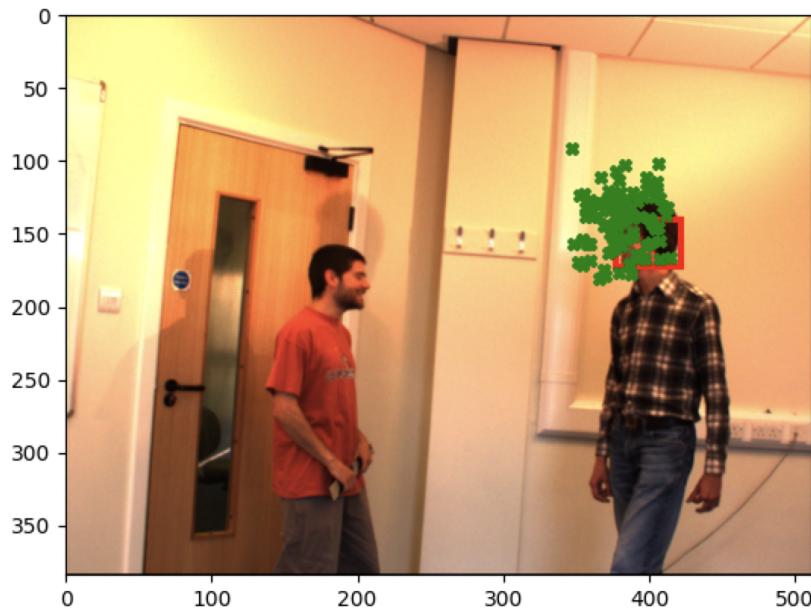


FIGURE 2.7
Début du suivi

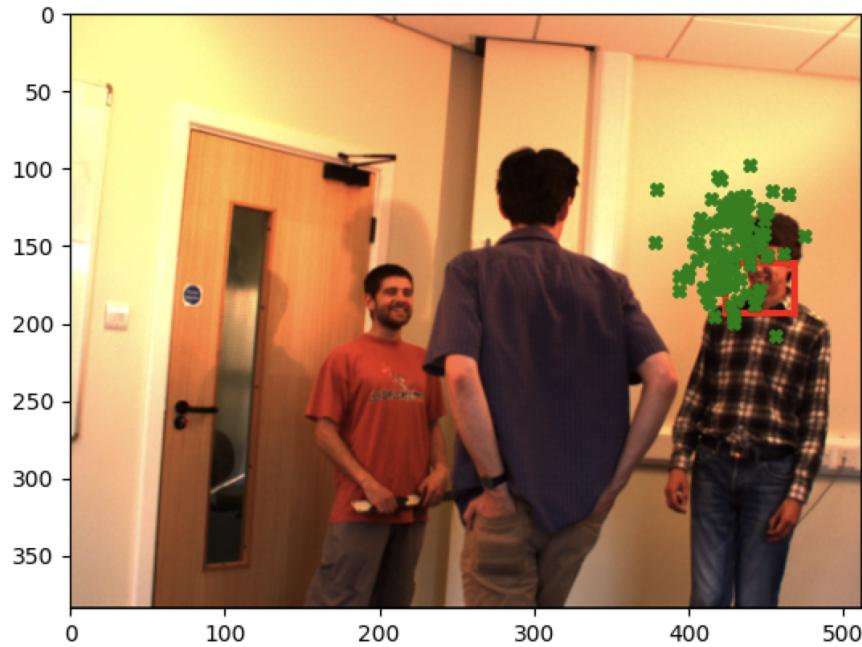


FIGURE 2.8
Juste avant le croisement

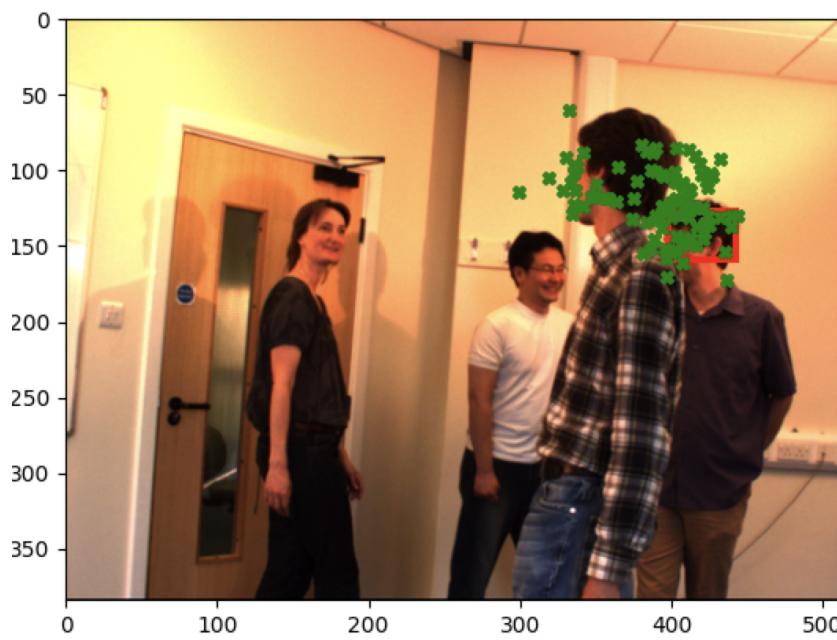


FIGURE 2.9
Premier croisement

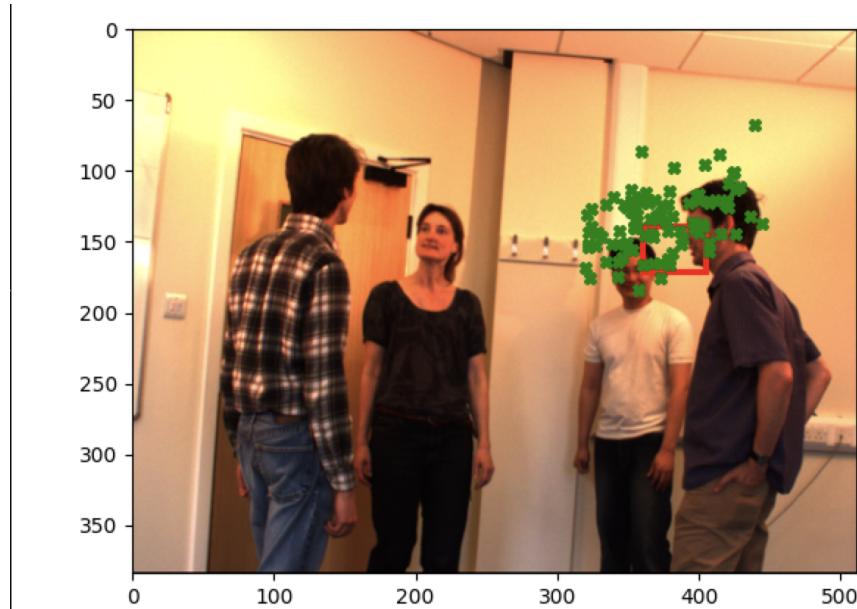


FIGURE 2.10
Deuxième croisement

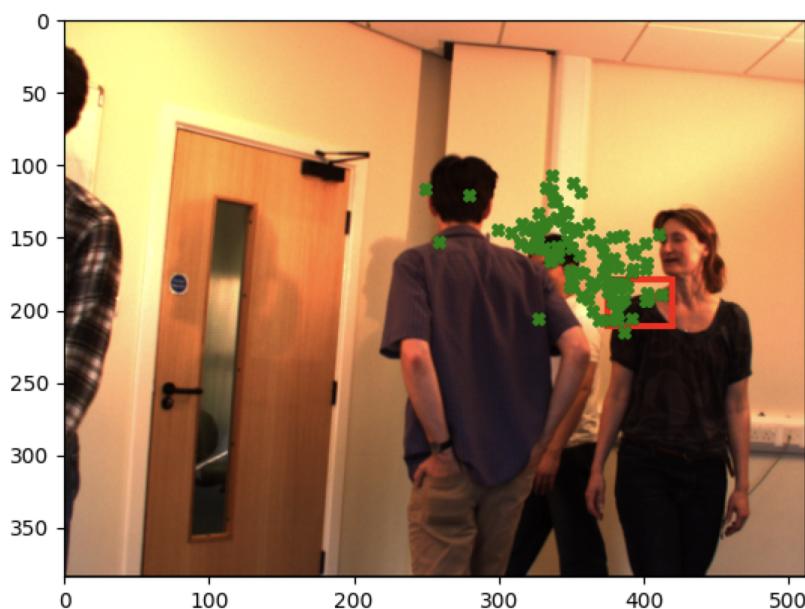


FIGURE 2.11
Troisième croisement

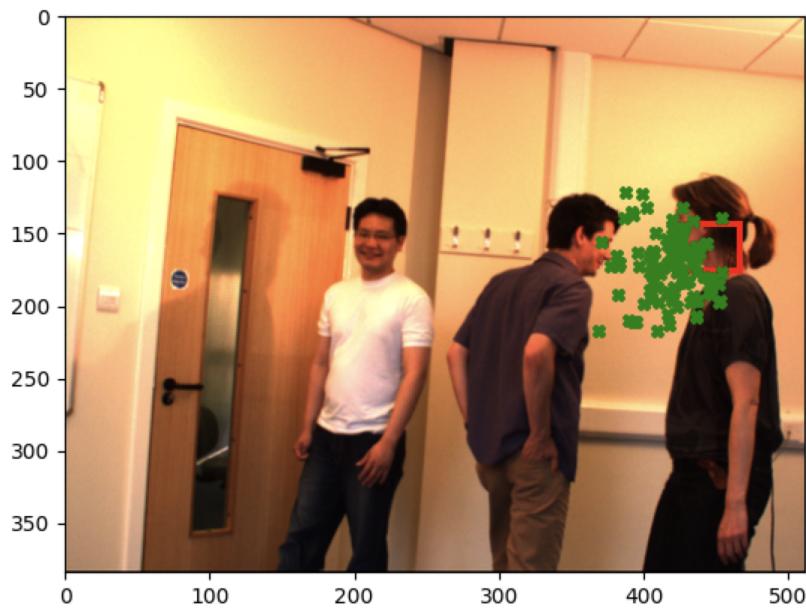


FIGURE 2.12
Avant le dernier croisement

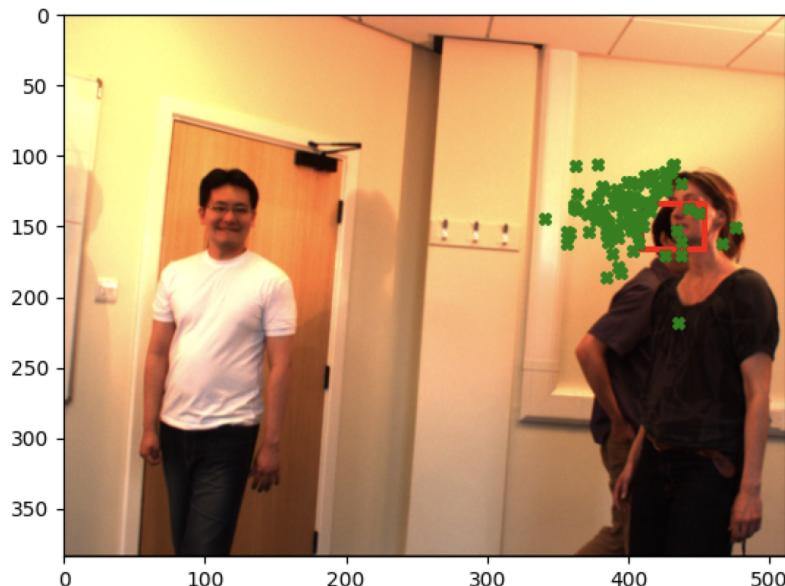


FIGURE 2.13
Dernier croisement

Comme on peut le constater, l'algorithme réagit mal aux croisements et change successivement de cibles, il n'est pas adapté aux changements soudain comme mentionné précédemment. Néanmoins on peut remarquer que chaque visage se situant proche du rectangle reste suivi pendant un moment avant les croisements (ici on a sélectionné que quelques photos des 220 qui ont défilé).

La différence majeure entre notre algorithme et un appareil photo réside dans le fait que nous choisissons le

visage qui va devoir être suivi alors que l'appareil doit être capable de "reconnaître" dans un environnement quelconque le visage. Des techniques de machine learning et d'IA sont donc nécessaires (classification) pour permettre à l'appareil d'effectuer de la détection de visage (traits du visages, critères etc...).

A présent, tentons de rajouter une mise à niveau, un algorithme s'adaptant à l'échelle et donc aux effets de profondeur de la cible suivie.

On remarque dans un premier temps que le tracker a du mal à suivre ces effets mais après quelques images la trajectoire estimée devient bien meilleure. On peut observer l'algorithme à travers la séquence 4 : Tout d'abord l'étudiant est adossé contre le tableau puis il va venir bouger et se rapprocher puis se décaler vers la droite. L'algorithme répond bien puisqu'il suit à la bonne échelle l'étudiant comme on peut le constater dans les images suivantes :

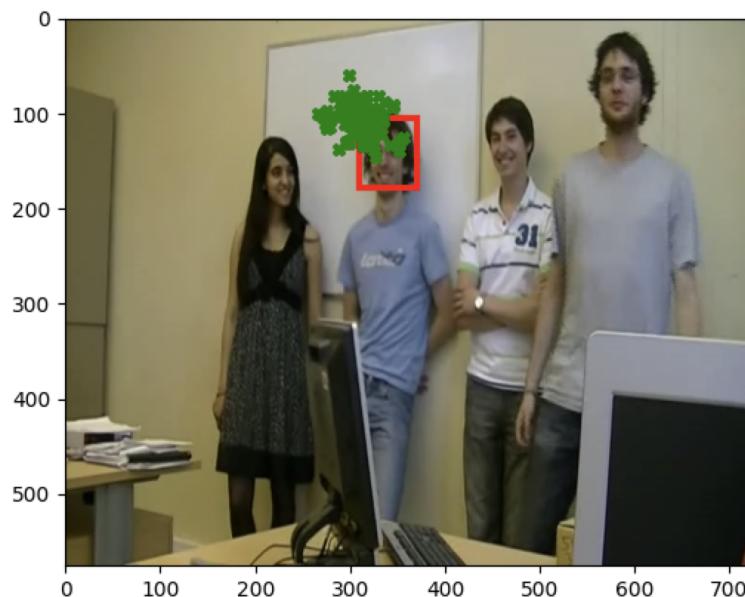


FIGURE 2.14
L'étudiant adossé

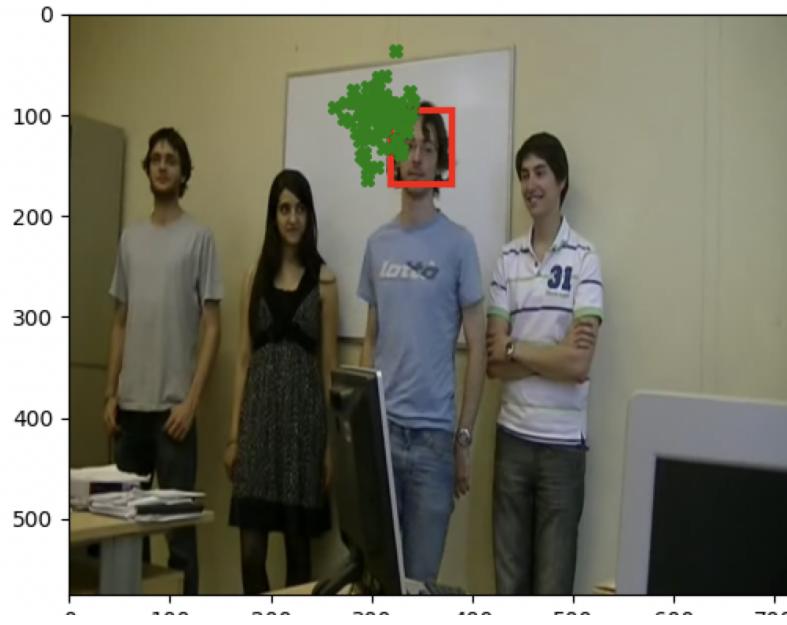


FIGURE 2.15
Un peu plus proche

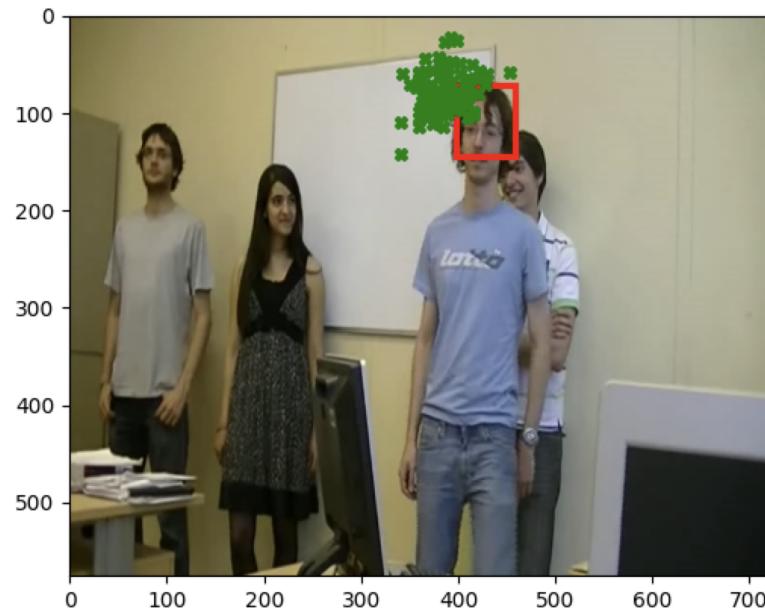


FIGURE 2.16
Plus proche et mouvement vers la droite

En conclusion, bien que notre modèle soit simpliste, il permet d'effectuer du suivi d'image sur séquence vidéo tant que les changements de position de la cible ne sont pas trop brusques (vitesse constante et position linéaire) et qu'il n'y a pas de croisement.

Remarque : lorsque la cible croise un élément mais que cet élément change de position brusquement alors notre prédiction reste sur la cible (voir figure 2.14 et 2.15, l'étudiant de droite est passé rapidement de droite à gauche mais notre étudiant visé est resté cible contrairement aux figures 2.8 à 2.13 où les changements de position sont très lents d'une image à l'autre donc à vitesse presque constante et c'est pourquoi notre cible change).

Enfin si l'on souhaite aller plus loin, il faudrait considérer bien d'autres paramètres, comme la vitesse de la cible, certains traits de mouvements, le cas où notre cible sort momentanément du cadre de l'image etc... mais notre équation dynamique deviendrait de plus en plus complexe et donc l'algorithme pourrait avoir un problème de complexité temporel.