MO102 - Introduction à Matlab
Projet Robotique

Cartographie et localisation simultanées
Filtrage de Kalman

david.filliat@ensta-paristech.fr

Dans le cadre de ce projet, nous allons mettre en oeuvre une méthode de filtrage classique, le *filtrage de Kalman*, appliquée à un problème de robotique : la cartographie et la localisation simultanées. Cette méthode, qui est par ailleurs utilisées dans un très grand nombre de domaines de l'ingénierie, nous permettra de construire une carte de l'environnement (a priori inconnu) du robot et d'estimer en permanence la position du robot au sein de cette carte.

1 Description du projet

1.1 Problème : Cartographie et localisation simultanées

Nous allons mettre en oeuvre une méthode qui permet d'estimer la position d'un robot mobile (terrestre ou aérien comme dans la fin de ce projet) se déplaçant dans un environnement a priori inconnu dont nous construirons une carte.

Pour estimer sa position, ce robot dispose de deux types d'informations :

— une information sur son déplacement qui provient par exemple de son odométrie, c'est à dire de l'estimation de la distance qu'il a parcouru en mesurant le nombre de tour effectué par chacune de ses roues

— une information sur sa position par rapport à des points de repère de l'environnement (que nous appellerons *amers*), qui provient par exemple de la perception de ces amers par une caméra.

Ces deux sources d'information sont évidement bruitées : l'odométrie devient rapidement peu fiable car les roues du robot peuvent glisser au sol et les perceptions fournissent une mesure entachée d'erreur.

L'objectif du projet va être d'estimer simultanément, à partir de ces deux types d'information, la position des amers (la carte de l'environnement) et la trajectoire du robot. Ce problème est connu sous le nom de **SLAM** (Simultaneous Localization and Mapping).

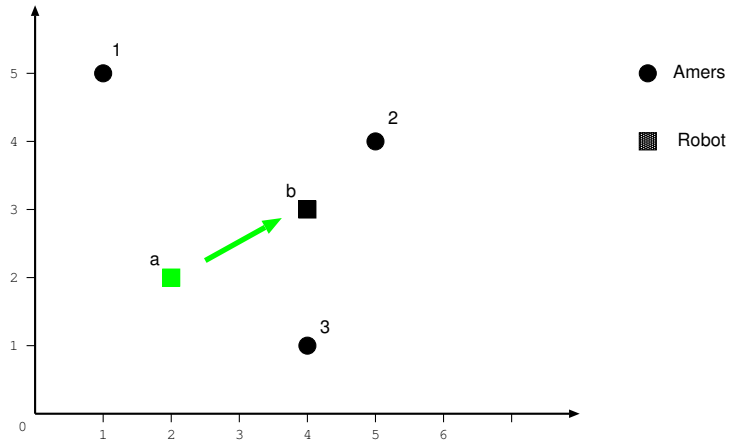

FIGURE 1 – Conventions utilisées dans le projet. La perception du robot depuis la position b sera le vecteur des positions relatives des amers par rapport au robot : $[-3\ 2\ 1\ 1\ 0\ \ -2]^T$. L'odométrie entre les deux positions a et b sera le vecteur de déplacement relatif $[2\ 1\ ]^T$. La carte correspondante (au moment ou le robot est en b) serait le vecteur des positions absolues du robot et des amers : $[4\ 3\ 1\ 5\ 5\ 4\ 4\ 1]^T$

Dans le cadre de ce TD, nous nous attaquerons à un problème très simplifié. Le robot et tous les amers sont supposés ponctuels et sans orientation. Les positions du robot et des amers seront données en coordonnées cartésiennes en 2 dimensions. L'odométrie sera une mesure du déplacement du robot dans le repère absolu. Les

observations seront une mesure de la position relative amers robot également dans le repère absolu (Figure 1). L'utilisation d'un unique repère absolu rend le problème très irréaliste (car la partie difficile réside, entre autres, dans l'estimation de l'orientation du repère du robot par rapport au repère global) mais permet de simplifier très nettement sa résolution en le rendant linéaire. La figure 2 illustre une sortie graphique des résultats du projet.

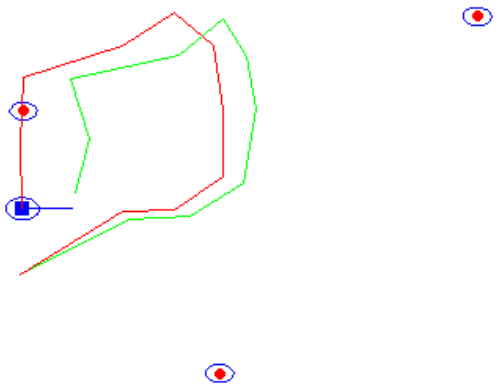

FIGURE 2 – Exemple de résultat du projet. Le robot est représenté par un carré, les amers par des cercles. La trajectoire estimée par l'odométrie seule est affichée en vert (clair), la trajectoire corrigée au moyen du filtre de Kalman est en rouge (foncé). Les ellipses autour des points indiquent l'incertitude dans l'estimation de la valeur réelle par l'algorithme.

1.2 Méthode : Le filtrage de Kalman

1.2.1 Rappel de probabilités

Dans ce TD, nous cherchons à estimer les valeurs de *variables aléatoires* que sont la position du robot et des amers. Nous supposerons que toutes ces variables suivent une *loi de probabilité gaussienne*, c'est a dire que la probabilité qu'une variable prenne une valeur donnée est une courbe en forme de 'cloche' (Figure 3) dont la formule est :

$$p = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{(x-\hat{x})^2}{2\sigma^2})$$

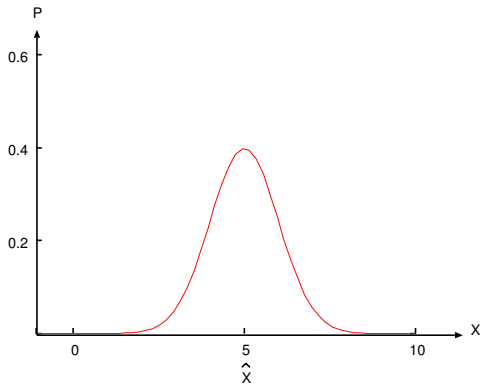

FIGURE 3 – Exemple de loi gaussienne de moyenne 5 et de variance 1.

En supposant ces lois gaussiennes, nous pouvons décrire toute notre connaissance de la variable par seulement deux valeurs : la moyenne $\widehat{x}$ et la variance σ^2 de cette gaussienne.

Dans le cadre de ce TD, les variables seront des vecteurs de taille N. La moyenne est alors un vecteur de même dimension, mais la variance devient une matrice de covariance Σ de taille $N \times N$. Cette matrice de covariance est toujours symétrique, définie positive.

Graphiquement en 2D, la loi gaussienne sur un vecteur est une 'cloche' dont les coupes horizontales sont des ellipses (Figure 4). Les directions des axes de cette ellipse sont les vecteurs propres de la matrice de covariance. La longueur des axes sont les valeurs propres de cette matrice.

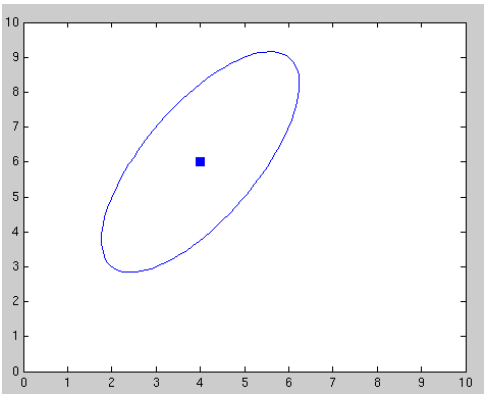

FIGURE 4 – Exemple de loi gaussienne de moyenne [4 6] et de variance [2 1 ; 1 3].

Dans le cadre de ce TD, nous représenterons donc toutes nos informations (position du robot et carte) sous la forme d'une variable gaussienne vectorielle que nous représenterons uniquement à l'aide de sa moyenne et de sa matrice de covariance. De plus, étant donné les simplifications, les ellipses correspondant aux matrices de covariances seront en fait des cercles.

1.2.2 Principe du SLAM

L'outil que nous allons utiliser pour résoudre le problème du SLAM est le filtre de Kalman. Ce filtre permet d'estimer l'*état* d'un système à partir d'une prédiction (bruitée) de son évolution au cours du temps et de mesures (bruitées) de cet état. Pour le SLAM en robotique mobile, l'état du système est la position **absolue** du robot et des amers, la prédiction de l'évolution proviendra des données odométriques **relatives** lors du déplacement du robot et les mesures proviendront de la perception **relative** des amers par le robot.

Le travail essentiel du filtre est donc de retrouver l'information de position absolue dans la carte en intégrant toutes les mesures relatives faites depuis le robot et en filtrant le bruit.

Dans notre exemple, l'état sera donc un vecteur composé de la position absolue en 2D du robot et des n amers $a_1 ... a_N$:

$$X_t = \begin{bmatrix} x \\ y \\ x_{a1} \\ y_{a1} \\ \vdots \\ x_{aN} \\ y_{aN} \end{bmatrix}$$

Le filtre de Kalman permet d'estimer cet état, mais estime aussi l'incertitude de cette estimation. Dans le cadre du filtre de Kalman, toutes les erreurs sur les différentes grandeurs sont supposées gaussiennes. L'incertitude d'une estimation est donc représentée par la variance de cette gaussienne. En plus du vecteur d'état, le filtre de Kalman utilise donc une *matrice de covariance*.

Ainsi le filtre donne à chaque instant une estimation $\hat{X}_t$ de la valeur de l'état réel X_t du système, ainsi qu'une estimation de la précision de cette estimation sous forme de sa matrice de covariance P_t. $P_t(1,1)$ est par exemple la variance de l'estimation de la coordonnée x de la position du robot.

Le fonctionnement d'une itération du filtre se déroule en quatre étapes (Figure 5) :

— **Prédiction de l'état** à l'instant courant X^*, ainsi que de sa covariance P_t^* à partir des mesures odométriques relatives u_t, de leur incertitude Q et de l'estimation de l'état au pas de temps précédent.

— **Prédiction des perceptions** relatives Y^* que le robot devrait obtenir depuis la position X^*

— **Observation** de l'état réel : on obtient, grâce au système perceptif du robot, une perception Y et on fournit une estimation de son erreur sous forme d'une matrice de covariance P_Y.

— **Correction de l'état prédit** pour fournir une estimation de l'état courant X_{t+1} à partir de l'écart entre perception prédite et perception réelle.

Ces quatre étapes sont utilisées à chaque nouvelle information d'odométrie et de perception, afin de mettre à jour l'estimation de l'état du système.

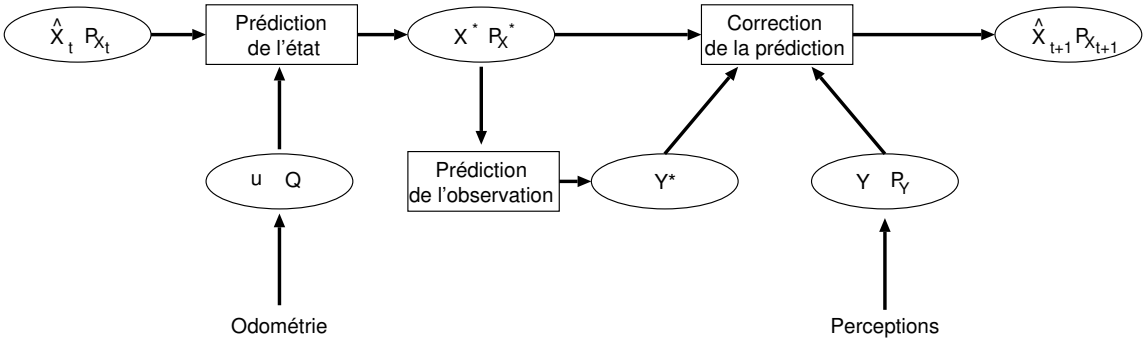

FIGURE 5 – Schéma de fonctionnement du filtre de Kalman.

Du point de vue calculatoire, la prédiction de l'état à l'instant courant X_t^*, est réalisée par une équation matricielle linéaire de la forme :

$$X_t^* = A.\hat{X}_t + B.u_t \tag{1}$$

Dans notre cas, comme l'odométrie est un vecteur colonne qui donne directement le déplacement du robot dans le repère global, l'équation se simplifie donc :

$$\mathbf{X_t^* = \hat{X}_t + B.u_t} \tag{2}$$

ou B est une matrice qui permet simplement de compléter le vecteur u_t par des 0 pour qu'il ait la même taille que X.

La covariance de l'état est également prédite par la formule :

$$P_t^* = A.\hat{P}_t.A^T + B.Q.B^T \tag{3}$$

ce qui devient :

$$\mathbf{P_t^* = \hat{P}_t + B.Q.B^T} \tag{4}$$

Q étant une estimation à fournir du bruit de la mesure d'odométrie.

La prédiction de l'observation à partir du vecteur d'état se fait simplement par l'équation :

$$\mathbf{Y_t^* = H.X_t^*} \tag{5}$$

la matrice H, ou matrice d'observation, permet simplement de soustraire la position du robot à la position des amers pour donner dans le vecteur Y_t^* la position relative des amers par rapport au robot.

Enfin la correction de l'état prédit, qui constitue le coeur du filtre de Kalman se réalise par les formules :

$$\hat{X}_{t+1} = X_t^* + K(Y - Y_t^*) \tag{6}$$

$$\hat{P}_{t+1} = P_t^* - KHP_t^* \tag{7}$$

ou K est le gain de Kalman, calculé pour minimiser l'erreur d'estimation au sens des moindres carrés et donné par la formule :

$$K = P_t^* H^T.(H.P_t^*.H^T + P_Y)^{-1} \tag{8}$$

P_Y est une matrice carrée de la taille de Y relatant la précision du système de perception.

1.2.3 Traitement des perceptions

Le bon fonctionnement de la méthode de SLAM présentée dépend très fortement de la qualité du traitement des perceptions effectué. Il est en effet crucial d'être capable d'associer un amer perçu par le robot avec l'amer correspondant dans la carte pour réaliser correctement la mise à jour du vecteur d'état. Le moindre échec de cette phase d'*association de données* peut conduire à un échec du filtrage de Kalman. Dans le cadre de ce projet, nous allons traiter ce problème de manière très simplifiée.

— Dans un premier temps, nous supposerons que tous les amers sont perçus à chaque étape, dans un ordre fixé. Ainsi le premier amer perçu correspondra au premier amer de la carte etc.

— Pour rendre le problème un peu plus réaliste, nous supposerons par la suite que seulement une partie des amers pourra être perçue et dans un ordre inconnu. Il faudra donc retrouver la correspondance entre les amers perçus et les amers de la carte. Cette mise en correspondance pourra se faire de deux manières. Dans un premier temps, nous nous baserons uniquement sur la position des amers en associant un amer perçu (Y) avec l'amer de la perception prédite (Y*) le plus proche (Figure 6). Un seuil sur cette distance permettra de décider si l'amer est nouveau ou non : un amer perçu suffisamment loin de tous les amers de la carte sera nouveau.

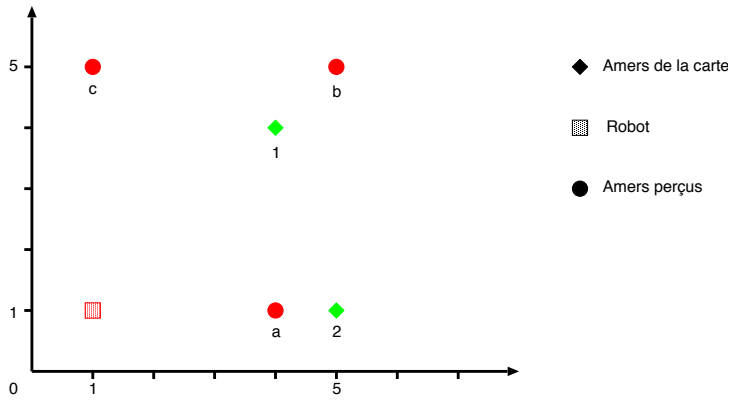

FIGURE 6 – Exemple d'association de données. La carte contient deux amers (1,2) dont on prédit la position par rapport au robot. Le robot perçoit trois amers (a, b et c). En associant les couples les plus proches, on déduit que a correspond à 2 et b à 1. c est un nouvel amer.

— Enfin nous pourrons utiliser des amers extraits d'images pour réaliser la cartographie. Dans ce cas, il est possible d'utiliser la position des amers pour les identifier, mais il est plus sûr d'utiliser leur apparence. Ainsi un amer pourra être un coin détecté dans une image et caractérisé par une petite image prise autour de ce coin (Figure 7). Pour réaliser l'association de données, il faudra alors comparer les imagettes prises autour des amers perçus avec les imagettes mémorisées pour les amers de la carte. Cette technique permet ainsi de traiter des problèmes pour lesquels l'information odométrique est très mauvaise.

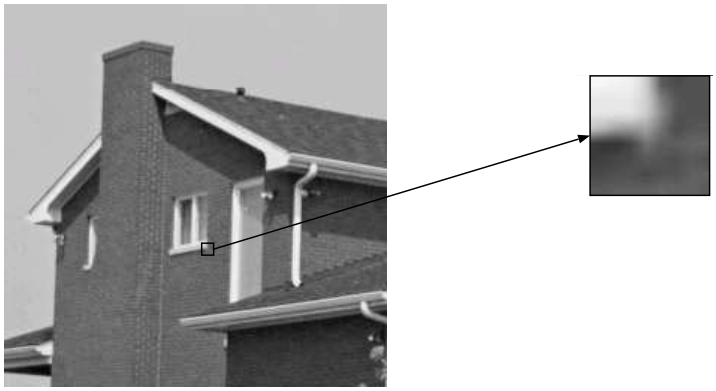

FIGURE 7 – Exemple d'amer caractérisé par une imagette prise autour d'un coin.

2 Organisation du travail

Une organisation des programmes est proposée comme fil conducteur pour le développement du projet. Cet ordre est choisi pour une découverte progressive des fonctionnalités de Matlab, mais vous êtes libres de l'améliorer à votre guise. Chacune des sous sections correspond à une fonction Matlab. Les données à utiliser sont sur la page `http://perso.ensta-paristech.fr/~filliat/Courses/`

2.1 Observations simplifiées

Dans les premières séances, les amers sont supposés tous détectés et dans le même ordre. Les vecteurs d'observation Y sont donc de taille constante et les lignes de Y correspondent directement aux lignes du vecteur d'état (moins les deux premières qui concernent le robot).

2.1.1 Covariance de l'observation

Ecrire une fonction qui prend en entrée un vecteur d'observation Y et construit la matrice de covariance P_Y associée. Cette matrice représente les erreurs de perception du robot réel. Pour simplifier, toutes les composantes seront supposées indépendantes (la matrice sera diagonale). La valeur des covariances pour un amer sera proportionnelle à la distance robot amer. Le coefficient de proportionnalité est un paramètre de la fonction qui sera ajusté plus tard.

2.1.2 Covariance de l'odométrie

Ecrire une fonction qui prend en entrée un vecteur d'odométrie u et renvoie la matrice de covariance Q associée. Pour simplifier, toutes les composantes seront supposées indépendante (la matrice sera diagonale). La valeur des covariances sera proportionnelle à la distance parcourue. Le coefficient de proportionnalité est un paramètre de la fonction qui sera ajusté plus tard.

2.1.3 Initialisations

Ecrire une fonction qui prend en entrée un premier vecteur d'observation y et qui renvoie le vecteur d'état X contenant les positions initiales du robot (supposée égale à $[0, 0]^T$ et des amers, la matrice de covariance P et les matrices A, B et H, tous initialisés. La matrice P est initialisée avec des valeurs reflétant la confiance que l'on a dans les valeurs initiale : pour les deux premières composante, on suppose que la position du robot est connue (prise comme référence) son incertitude sera faible. L'incertitude sur les amers sera celle fournie par P_Y avec la fonction écrite précedemment.

2.1.4 Affichage graphique

Ecrire une fonction qui prend un vecteur X et une matrice P en entrée et affiche le robot et les amers avec leur incertitude associée (voir figure 2 pour un exemple). Dans le cadre de ce projet, les incertitudes des positions seront toutes représentées par des cercles, dont le rayon est la covariance associée à la position considérée.

2.1.5 Prédiction de l'état

Ecrire une fonction qui prend en entrée des vecteurs X et u et des matrices A, B et P et Q et renvoie un vecteur X^* et un matrice P^*.

2.1.6 Prédiction de l'observation

Ecrire une fonction qui prend en entrée un vecteur X^* et une matrice H et renvoie le vecteur d'observation Y^* en sortie.

2.1.7 Correction de l'état

Écrire une fonction qui prend en entrée X^*, P^*, Y^*, Y, H et P_Y et qui renvoie un vecteur X et un matrice P correspondant à la mise à jour de la prédiction par le filtre de Kalman.

2.1.8 Script principal

Ecrire un script principal lisant dans un fichier (fourni sur ma page web) des séquences d'observation et d'odométrie et appliquant le filtre de Kalman. Vous utiliserez la fonction MATLAB `textscan`. Le fichier `FullObservation.data` contient une succession de mesures d'odométries et de perceptions sous la forme :

```
percep : -1.911445 3.046930 1.995198 -4.880126 6.935804 5.808414
odom : 1.224257 0.095414
percep : -2.975497 3.083787 0.875721 -5.050758 6.197317 5.831181
...
```

La première perception doit servir à initialiser la carte. Ensuite l'odométrie doit servir à prédire l'état et la perception suivante à le corriger. Le fonctionnement global est le suivant :

- Ouvrir le fichier
- Lire la première perception dans le fichier
- Initialiser les variables du filtre
- Afficher la carte (pause)
- Tant que le fichier n'est pas vide :
  - Lire une odométrie
  - Effectuer la prédiction de l'état
  - Afficher la carte (pause)
  - Lire une perception
  - Effectuer la prédiction de l'observation
  - Effectuer la correction
  - Afficher la carte (pause)

Jouer avec les différents paramètres (Q, P_Y ...) jusqu'à obtenir un résultat correct.

2.1.9 Affichage graphique amélioré

Afficher la trajectoire du robot prédite par l'odométrie et celle corrigée par le filtre de Kalman. Vous devez créer des variables supplémentaires qui enregistrent les trajectoires et les affichent en entier à chaque appel de la fonction d'affichage de la carte. Vous pouvez également afficher le contenu de la matrice de covariance dans une sous-figure au coté de la carte et de la trajectoire pour observer les corrélations qui apparaissent entre les amers.

2.2 Amers partiellement observés

A partir de maintenant, nous allons considérer que les amers sont vus en partie et sans connaître leur ordre.

2.2.1 Observation partielle

Modifier le script principal pour tenir compte d'un nombre variable d'amers perçus dans le désordre. La taille du vecteur Y va alors être variable, de même que la taille et le contenu de la matrice H qui changera a chaque perception. Le vecteur d'état X et la matrice de covariance P vont également grandir chaque fois qu'un nouvel amer est découvert. Créez une nouvelle fonction qui prendra en entrée X^* et Y et qui produira en sortie la matrice H correspondant aux amers revus et des vecteur Y_{known} correspondant aux amers de la carte et Y_{new} correspondant aux nouveaux amers. Il faut faire le calcul de mise en correspondance basée sur la distance entre amers prédits à partir de la carte et amers perçus. Le reste du code du filtre de kalman ne devrait pas changer, il suffit de créer la matrice H juste avant la prédiction de l'observation, de réaliser la prédiction d'état avec H, la correction avec Y_{known} et d'ajouter les nouveaux amers à la fin du filtre avec Y_{new}. Créez pour cela une fonction qui prendra en entrée X, P , A, B et Y_{new} qui ajoutera les nouveaux amers dans X et P et re-dimensionnera A et B.

Voici un exemple de sorties de la première fonction sur les données de la figure 6 :

$$X = \begin{bmatrix} 1 & 1 & 4 & 4 & 5 & 1 \end{bmatrix}^T$$

$$Y = \begin{bmatrix} 3 & 0 & 4 & 4 & 0 & 4 \end{bmatrix}^T$$

donneront :

$$H = \begin{bmatrix} -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 \\ -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$Y_{known} = \begin{bmatrix} 3 & 0 & 4 & 4 & \end{bmatrix}^T \; Y_{new} = \begin{bmatrix} 0 & 4 \end{bmatrix}^T$$

Le fichier `PartialObservation.data` contient les données correspondantes. Le fonctionnement global deviendra le suivant :

- Ouvrir le fichier
- Lire la première perception dans le fichier
- Initialiser les variables du filtre
- Afficher la carte (pause)
- Tant que le fichier n'est pas vide :
  - Lire une odométrie
  - Effectuer la prédiction de l'état
  - Afficher la carte (pause)
  - Lire une perception
  - Déterminer Y_{known}, Y_{new} et construire la matrice H
  - Effectuer la prédiction de l'observation Y^*_{know} à partir de X^* et H
  - Effectuer la correction en fonction de Y^*_{know} et Y_{know}
  - Ajouter les nouveaux amers et mettre à jour les matrices du filtre en fonction de Y_{new}
  - Afficher la carte (pause)

2.2.2 Simulateur

Ecrire une fonction qui prend en entrée une trajectoire (absolue) du robot et la position absolue réelle des amers et écrit un fichier contenant la séquence bruitée correspondante d'observations et d'odométries. Le bruit sur les valeurs odométriques doit être un bruit gaussien sur la distance et la direction du déplacement. Le bruit sur les amers est un bruit gaussien sur la position relative amer/robot. Tester votre programme avec de nouvelles séquences produites par votre simulateur.

2.3 Observation à partir d'images

2.3.1 Cartographie à partir d'images

Modifier vos scripts afin que les perceptions utilisées soient extraites d'images. Le répertoire `Images` fournit contient une suite d'images et de valeur de l'odométrie correspondant au déplacement du robot entre les prises d'images. La caméra (simulée) est supposer pointer vers le bas et filmer une scène plane. L'axe des x va vers la droite, l'axe des y vers le bas (cf Figure 8). Les amers seront des points extraits des images par le détecteur de coins fourni (fichier `corner.m`). Dans un premier temps, les amers seront caractérisés par leur position relative par rapport au robot. Modifier ensuite la méthode d'association de données pour que les amers soient caractérisés par une imagette autour du point correspondant. Utiliser ces imagettes pour déterminer à quel amer de la carte correspond un amer perçu.

2.3.2 Mosaïque d'image

Utiliser les positions du robot calculées par le SLAM pour réaliser une mosaïque d'images en assemblant toutes les images utilisées pour les perceptions en une seule grande image.

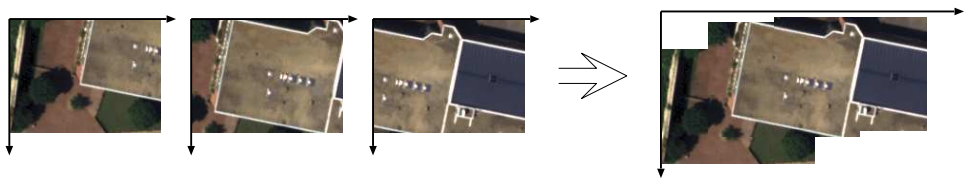

FIGURE 8 – Exemple de Mosaïque d'images. Le SLAM permet de calculer la position relative des images avant de les fusionner dans une image commune.