Contribution au SLAM avec caméra omnidirectionnelle

Cyril JOLY et Patrick RIVES

INRIA Sophia-Antipolis Méditerranée – École des Mines de Paris

31 juillet 2009

École d'été en traitement du signal et des images – Peyresq 2009

Contexte du sujet

Travaux de l'équipe ARobAS appliqués à la robotique

- Asservissement visuel
- Reconstruction de scène
- Localisation
- Navigation
- Commande

Dans la continuité des travaux de...

Alessandro Victorino : SLAM et commande référencée laser

Christopher Mei : SLAM et couplage entre laser et caméra omnidirectionnelle

Plan

Plan

Plan

1. Introduction
   - Le SLAM : définition
   - Notations
   - Variables
   - Equations
   - Objectifs

Le SLAM : définition

SLAM : **S**imultaneous **L**ocalization **A**nd **M**apping

Le SLAM : définition

SLAM : **S**imultaneous **L**ocalization **A**nd **M**apping

Intérêt : en robotique mobile

- Localisation du robot dans un environnement inconnu
- Exploration **autonome** en milieu hostile
- ...

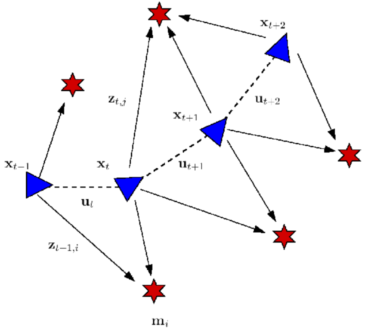
Notations

Etat du système

- $\mathbf{x}_t$: état du robot à l'instant t
- $\mathbf{m}_{(i)}$: état de l'amer (i)

Utilisation de :

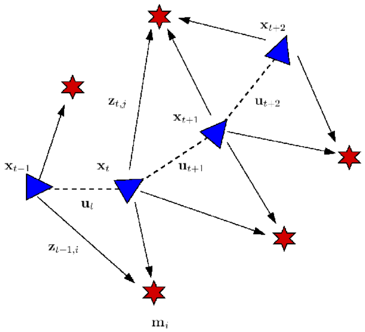
1. Equation de modèle impliquant $\mathbf{u}_t$
2. Mesures : $\mathbf{z}_t$



Notations

Présence de bruits :

- Sur le modèle
- Sur les mesures des entrées
- Sur les mesures des amers



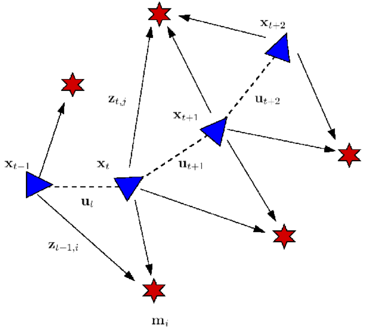
Notations

Présence de bruits :

- Sur le modèle
- Sur les mesures des entrées
- Sur les mesures des amers

Présence de bruit :

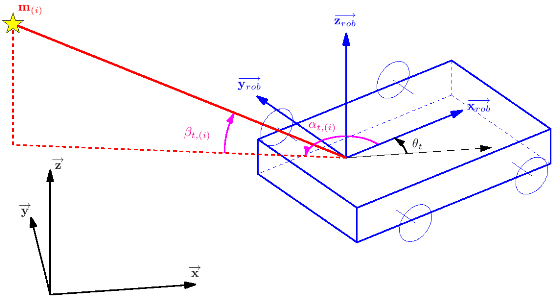
1. Valeurs des bruits **inconnues**
2. Caractéristiques des bruits : **connues**



Variables

Robot

$$\mathbf{x}_t = \begin{bmatrix} x_t & y_t & \theta_t \end{bmatrix}^T$$



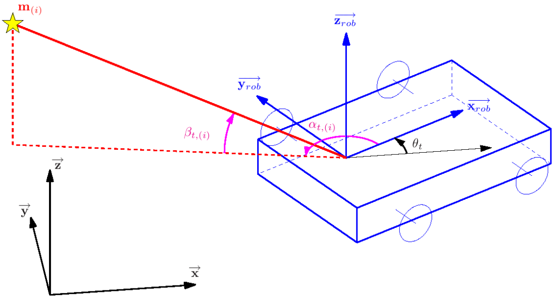
Variables

Robot

$$\mathbf{x}_t = \begin{bmatrix} x_t & y_t & \theta_t \end{bmatrix}^T$$

Commande

$$\mathbf{u}_t = \begin{bmatrix} v_t & \omega_t \end{bmatrix}^T$$



Variables

Robot

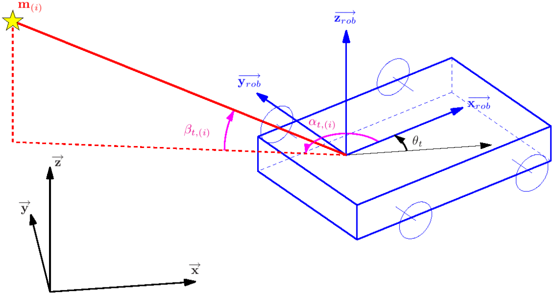
$$\mathbf{x}_t = \begin{bmatrix} x_t & y_t & \theta_t \end{bmatrix}^T$$

Commande

$$\mathbf{u}_t = \begin{bmatrix} v_t & \omega_t \end{bmatrix}^T$$

Amer (i)

$$\mathbf{m}_{(i)} = \begin{bmatrix} x_{(i)} & y_{(i)} & z_{(i)} \end{bmatrix}^T$$



Variables

Robot

$$\mathbf{x}_t = \begin{bmatrix} x_t & y_t & \theta_t \end{bmatrix}^T$$

Commande

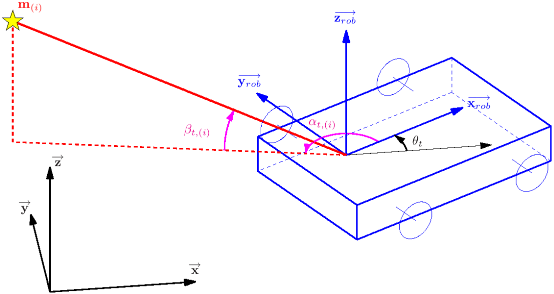
$$\mathbf{u}_t = \begin{bmatrix} v_t & \omega_t \end{bmatrix}^T$$

Amer (i)

$$\mathbf{m}_{(i)} = \begin{bmatrix} x_{(i)} & y_{(i)} & z_{(i)} \end{bmatrix}^T$$

Mesures amer (i)

$$\mathbf{z}_{t,(i)} = \begin{bmatrix} \alpha_{t,(i)} & \beta_{t,(i)} \end{bmatrix}^T$$



Equations

Equations de modèle

$$\begin{cases} x_t & = & x_{t-1} + v_t dt \mathrm{sinc}\frac{\omega dt}{2} \cos\left(\theta_{t-1} + \frac{\omega dt}{2}\right) \\ y_t & = & y_{t-1} + v_t dt \mathrm{sinc}\frac{\omega dt}{2} \sin\left(\theta_{t-1} + \frac{\omega dt}{2}\right) \\ \theta_t & = & \theta_{t-1} + \omega dt \end{cases}$$

Equations de mesure

$$\begin{cases} \alpha_{t,(i)} & = & \mathrm{atan2}\left(y_{(i)} - y_t, x_{(i)} - x_t \right) - \theta_t \\ \beta_{t,(i)} & = & \arctan\left(\frac{z_{(i)}}{\sqrt{(x_{(i)} - x_t)^2 + (y_{(i)} - y_t)^2}}\right) \end{cases}$$

Objectifs

But :

1. Estimer l'ensemble des états
   - Position(s) du robot
   - Positions des amers
2. Estimer “ au mieux ” l'incertitude : fournir une enveloppe d'erreur **consistante**
   - Ni trop optimiste (estimation erronée)
   - Ni trop pessimiste (résultat non exploitable)

Lien avec les problèmes inverses

Il s'agit ici de retrouver la position du robot et des amers ayant permis d'obternir les mesures effectuées par un capteur donné.

Plan

Plan

2. Algorithme de filtrage
   - Données et hypothèses
   - SAM : Simultaneous Smoothing and Mapping

Données et hypothèses

Rappel du modèle

$$\begin{cases} \mathbf{x}_t = \mathbf{f}\left(\mathbf{x}_{t-1}, \mathbf{u}_t + \nu_t^{\mathbf{u}}\right) + \nu_t^{\mathbf{f}} \\ \mathbf{z}_t = \mathbf{h}\left(\mathbf{x}_t, \mathbf{m}\right) + \nu_t^{\mathbf{z}} \end{cases}$$

Données et hypothèses

Rappel du modèle et distribution des erreurs

$$\left\{ \begin{array}{lcl} \mathbf{x}_t & = & \mathbf{f}\left(\mathbf{x}_{t-1}, \mathbf{u}_t + \nu_t^{\mathbf{u}}\right) + \nu_t^{\mathbf{f}} \\ \mathbf{z}_t & = & \mathbf{h}\left(\mathbf{x}_t, \mathbf{m}\right) + \nu_t^{\mathbf{z}} \end{array} \right. \quad \text{avec} \quad \boxed{\begin{array}{lcl} \nu_t^{\mathbf{f}} & \sim & \mathcal{N}\left(\mathbf{0}, \mathbf{Q}^{\mathbf{f}}\right) \\ \nu_t^{\mathbf{u}} & \sim & \mathcal{N}\left(\mathbf{0}, \mathbf{Q}^{\mathbf{u}}\right) \\ \nu_t^{\mathbf{z}} & \sim & \mathcal{N}\left(\mathbf{0}, \mathbf{Q}^{\mathbf{z}}\right) \end{array}}$$

Données et hypothèses

Rappel du modèle et distribution des erreurs

$$\left\{ \begin{array}{lcl} \mathbf{x}_t & = & \mathbf{f}\left(\mathbf{x}_{t-1}, \mathbf{u}_t + \nu_t^{\mathbf{u}}\right) + \nu_t^{\mathbf{f}} \\ \mathbf{z}_t & = & \mathbf{h}\left(\mathbf{x}_t, \mathbf{m}\right) + \nu_t^{\mathbf{z}} \end{array} \right. \quad \text{avec} \quad \boxed{\begin{array}{lcl} \nu_t^{\mathbf{f}} & \sim & \mathcal{N}\left(\mathbf{0}, \mathbf{Q}^{\mathbf{f}}\right) \\ \nu_t^{\mathbf{u}} & \sim & \mathcal{N}\left(\mathbf{0}, \mathbf{Q}^{\mathbf{u}}\right) \\ \nu_t^{\mathbf{z}} & \sim & \mathcal{N}\left(\mathbf{0}, \mathbf{Q}^{\mathbf{z}}\right) \end{array}}$$

Entrées de l'algorithme

$p\left(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t\right)$

$p\left(\mathbf{z}_t | \mathbf{x}_t, \mathbf{m}\right)$

Données et hypothèses

Rappel du modèle et distribution des erreurs

$$\left\{\begin{array}{lcl} \mathbf{x}_t & = & \mathbf{f}\left(\mathbf{x}_{t-1}, \mathbf{u}_t + \nu_t^{\mathbf{u}}\right) + \nu_t^{\mathbf{f}} \\ \mathbf{z}_t & = & \mathbf{h}\left(\mathbf{x}_t, \mathbf{m}\right) + \nu_t^{\mathbf{z}} \end{array}\right. \quad \text{avec} \quad \boxed{\begin{array}{lcl} \nu_t^{\mathbf{f}} & \sim & \mathcal{N}\left(\mathbf{0}, \mathbf{Q}^{\mathbf{f}}\right) \\ \nu_t^{\mathbf{u}} & \sim & \mathcal{N}\left(\mathbf{0}, \mathbf{Q}^{\mathbf{u}}\right) \\ \nu_t^{\mathbf{z}} & \sim & \mathcal{N}\left(\mathbf{0}, \mathbf{Q}^{\mathbf{z}}\right) \end{array}}$$

Entrées de l'algorithme

$$\begin{array}{lcl} p\left(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t\right) & = & \mathcal{N}\left(\mathbf{f}\left(\mathbf{x}_{t-1}, \mathbf{u}_t\right), \mathbf{Q}_t\right) \\ p\left(\mathbf{z}_t | \mathbf{x}_t, \mathbf{m}\right) & = & \mathcal{N}\left(\mathbf{h}\left(\mathbf{x}_t, \mathbf{m}\right), \mathbf{R}_t\right) \end{array}$$

Données et hypothèses

Rappel du modèle et distribution des erreurs

$$\left\{ \begin{array}{lcl} \mathbf{x}_t & = & \mathbf{f}\left(\mathbf{x}_{t-1}, \mathbf{u}_t + \nu_t^{\mathbf{u}}\right) + \nu_t^{\mathbf{f}} \\ \mathbf{z}_t & = & \mathbf{h}\left(\mathbf{x}_t, \mathbf{m}\right) + \nu_t^{\mathbf{z}} \end{array} \right. \quad \text{avec} \quad \boxed{\begin{array}{lcl} \nu_t^{\mathbf{f}} & \sim & \mathcal{N}\left(\mathbf{0}, \mathbf{Q}^{\mathbf{f}}\right) \\ \nu_t^{\mathbf{u}} & \sim & \mathcal{N}\left(\mathbf{0}, \mathbf{Q}^{\mathbf{u}}\right) \\ \nu_t^{\mathbf{z}} & \sim & \mathcal{N}\left(\mathbf{0}, \mathbf{Q}^{\mathbf{z}}\right) \end{array}}$$

Entrées de l'algorithme

$$\begin{array}{lcl} p\left(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t\right) & = & \mathcal{N}\left(\mathbf{f}\left(\mathbf{x}_{t-1}, \mathbf{u}_t\right), \mathbf{Q}_t\right) \\ p\left(\mathbf{z}_t | \mathbf{x}_t, \mathbf{m}\right) & = & \mathcal{N}\left(\mathbf{h}\left(\mathbf{x}_t, \mathbf{m}\right), \mathbf{R}_t\right) \end{array} \qquad \begin{array}{l} \mathbf{Q}_t = \mathbf{Q}^{\mathbf{f}} + \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \mathbf{Q}^{\mathbf{u}} \frac{\partial \mathbf{f}}{\partial \mathbf{u}}^T \\ \mathbf{R}_t = \mathbf{Q}^{\mathbf{z}} \end{array}$$

Données et hypothèses

Rappel du modèle et distribution des erreurs

$$\left\{ \begin{array}{lcl} \mathbf{x}_t & = & \mathbf{f}\left(\mathbf{x}_{t-1}, \mathbf{u}_t + \nu_t^{\mathbf{u}}\right) + \nu_t^{\mathbf{f}} \\ \mathbf{z}_t & = & \mathbf{h}\left(\mathbf{x}_t, \mathbf{m}\right) + \nu_t^{\mathbf{z}} \end{array} \right. \quad \text{avec} \quad \boxed{\begin{array}{lcl} \nu_t^{\mathbf{f}} & \sim & \mathcal{N}\left(\mathbf{0}, \mathbf{Q}^{\mathbf{f}}\right) \\ \nu_t^{\mathbf{u}} & \sim & \mathcal{N}\left(\mathbf{0}, \mathbf{Q}^{\mathbf{u}}\right) \\ \nu_t^{\mathbf{z}} & \sim & \mathcal{N}\left(\mathbf{0}, \mathbf{Q}^{\mathbf{z}}\right) \end{array}}$$

Entrées de l'algorithme

$$\begin{array}{lcl} p\left(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t\right) & = & \mathcal{N}\left(\mathbf{f}\left(\mathbf{x}_{t-1}, \mathbf{u}_t\right), \mathbf{Q}_t\right) \\ p\left(\mathbf{z}_t | \mathbf{x}_t, \mathbf{m}\right) & = & \mathcal{N}\left(\mathbf{h}\left(\mathbf{x}_t, \mathbf{m}\right), \mathbf{R}_t\right) \end{array} \qquad \begin{array}{l} \mathbf{Q}_t = \mathbf{Q}^{\mathbf{f}} + \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \mathbf{Q}^{\mathbf{u}} \frac{\partial \mathbf{f}}{\partial \mathbf{u}}^T \\ \mathbf{R}_t = \mathbf{Q}^{\mathbf{z}} \end{array}$$

SAM : Simultaneous Smoothing and Mapping (1/2)

Principe

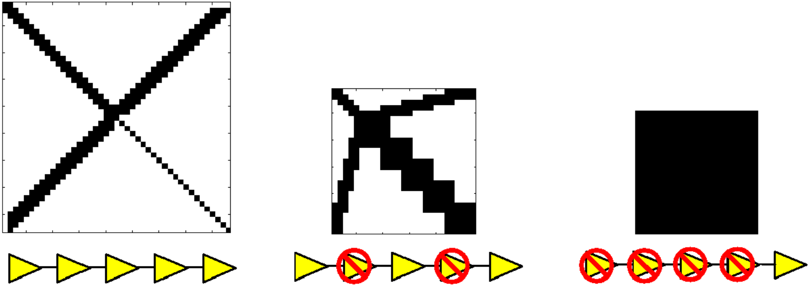
- Calculer la densité de la **trajectoire** du robot et de la carte : $\underbrace{p\left(\mathbf{x}_{1:t}, \mathbf{m} \middle| \mathbf{x}_0, \mathbf{z}_{0:t}, \mathbf{u}_{1:t}\right)}_{\text{Toute la trajectoire}}$ au lieu de $\underbrace{p\left(\mathbf{x}_{t}, \mathbf{m} \middle| \mathbf{x}_0, \mathbf{z}_{0:t}, \mathbf{u}_{1:t}\right)}_{\text{Dernière position (EKF)}}$
- Travailler dans l'espace matrice d'information / vecteur d'information

Comparaison avec l'EKF :

	Consistance	Calculs / Mémoire
EKF	Ne remet pas en cause la trajectoire. Une inconsistance à un instant donné sera propagée pour toute la suite	Quadratique avec le nombre d'amers
SAM	Remet pas en cause la trajectoire. Permet de limiter les inconsistances	Dimension de l'état très importante mais matrice d'information éparse Temps de calculs comparables

SAM : Simultaneous Smoothing and Mapping (2/2)

Propriété de la matrice d'information



(a)No marginalisation (b)Partial marginalisation (c)Full marginalisation

Plan

Plan

3. Construire des cartes locales
   - Intérêt
   - Quand démarrer une nouvelle carte ?
   - Gestion des cartes locales

Intérêt :

Objectif : réduire les incertitudes autour du robot

- Améliorer la sécurité des tâches à court terme effectuées par le robot (planification, contrôle).
- Garder un contrôle sur la qualité de l'estimation non linéaire (limiter les problèmes d'inconsistance)

Solutions envisagées

1. Carte robocentrée
   - Cas du SAM : évaluer les **trajectoires** des amers dans le repère du robot ⇒ dimension trop importante
   - EKF obligatoire : risque d'inconsistance
2. Cartes locales en réinitialisant les incertitudes
   - SAM utilisable ⇒ meilleure consistance
   - Permet de représenter la trajectoire entière du robot *a posteriori*

Quand démarrer une nouvelle carte ? (1/2)

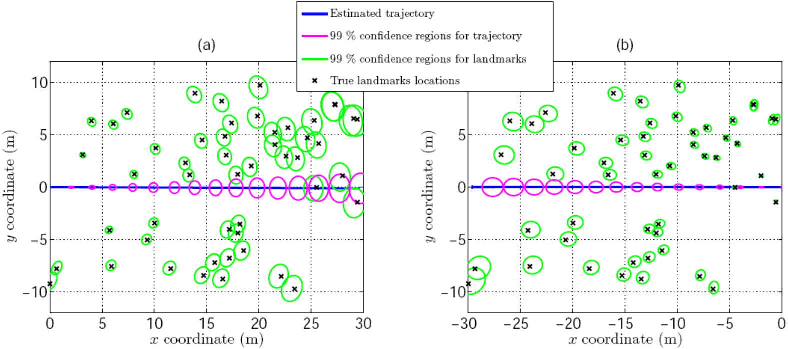
Données à l'instant t :

- Densité de probabilité *a posteriori* : $p\left(\mathbf{x}_{1:t}, \mathbf{m} | \mathbf{x}_0, \mathbf{z}_{0:t}, \mathbf{u}_{1:t}\right)$
- Marginalisation des positions immédiate : $p\left(\mathbf{m} | \mathbf{x}_0, \mathbf{z}_{0:t}, \mathbf{u}_{1:t}\right)$
- $\Rightarrow$ matrice de covariances pour chaque amer i : $\boldsymbol{\Sigma}^i_{|\mathbf{x_0}}$

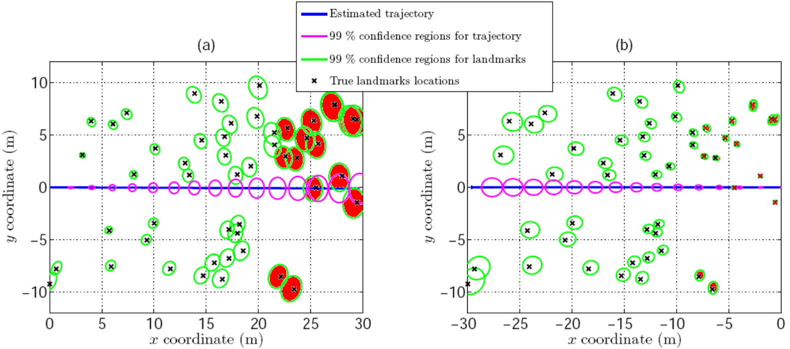
Paramètres déduits en changeant le point de conditionnemment

- Calculer $p\left(\mathbf{m} | \mathbf{x}_t, \mathbf{z}_{0:t}, \mathbf{u}_{1:t}\right)$ à partir de $p\left(\mathbf{x}_{1:t}, \mathbf{m} | \mathbf{x}_0, \mathbf{z}_{0:t}, \mathbf{u}_{1:t}\right)$
- Solution que l'on aurait obtenue en supposant certaine **la dernière position du robot au lieu de la première**
- $\Rightarrow$ matrice de covariances pour chaque amer i : $\boldsymbol{\Sigma}^i_{|\mathbf{x_t}}$

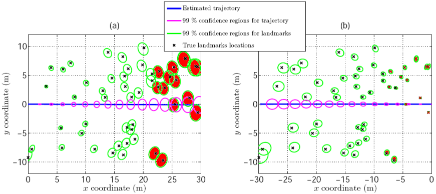
Quand démarrer une nouvelle carte ? (2/2)



Quand démarrer une nouvelle carte ? (2/2)



Quand démarrer une nouvelle carte ? (2/2)



Critère

- Idée : l'incertitude associée aux derniers amers vus à l'instant t doit être beaucoup plus faible lorsque l'on suppose connue la position finale du robot que lorsqu'on suppose connue la position initiale
- Calcul pour chacun des derniers amers de $\sqrt{\dfrac{\det \mathbf{\Sigma}^i_{|\mathbf{x_t}}}{\det \mathbf{\Sigma}^i_{|\mathbf{x_0}}}}$
- Si la moyenne des coefficients est inférieure à un certain seuil : on initialise une nouvelle carte

Gestion des cartes locales (1/3)

Comment créer une nouvelle carte ?

1. Calculer $p\left(\mathbf{x}_{t_1+1:t_2}, \tilde{\mathbf{m}} | \mathbf{x}_{t_1}, \mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}\right)$:
   - La nouvelle carte est obtenue indépendemment de l'ancienne carte

Gestion des cartes locales (1/3)

Comment créer une nouvelle carte ?

1. Calculer $p\left(\mathbf{x}_{t_1+1:t_2}, \tilde{\mathbf{m}} | \mathbf{x}_{t_1}, \mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}\right)$:
    - La nouvelle carte est obtenue indépendamment de l'ancienne carte
    - **Les amers présents à la fois dans la carte précédente et dans la nouvelle sont initialisés avec une matrice d'information nulle (ie. covariance infinie)**

Gestion des cartes locales (1/3)

Comment créer une nouvelle carte ?

1. Calculer $p\left(\mathbf{x}_{t_1+1:t_2}, \tilde{\mathbf{m}} | \mathbf{x}_{t_1}, \mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}\right)$:
   - La nouvelle carte est obtenue indépendamment de l'ancienne carte
   - **Les amers présents à la fois dans la carte précédente et dans la nouvelle sont initialisés avec une matrice d'information nulle (ie. covariance infinie)**
2. Est-il possible de prendre en compte les calculs effectués dans la carte précédente ?

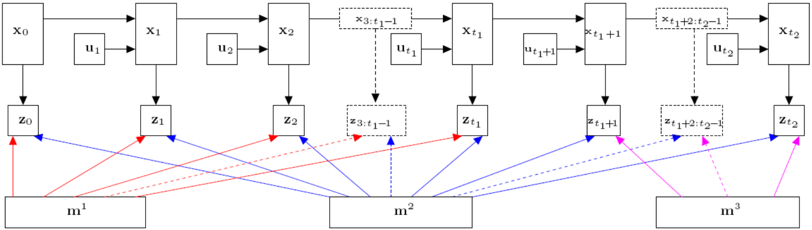
Gestion des cartes locales (1/3)

Comment créer une nouvelle carte ?

1. Calculer $p\left(\mathbf{x}_{t_1+1:t_2}, \tilde{\mathbf{m}} | \mathbf{x}_{t_1}, \mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}\right)$:
   - La nouvelle carte est obtenue indépendamment de l'ancienne carte
   - **Les amers présents à la fois dans la carte précédente et dans la nouvelle sont initialisés avec une matrice d'information nulle (ie. covariance infinie)**
2. Est-il possible de prendre en compte les calculs effectués dans la carte précédente ?

Gestion des cartes locales (2/3)

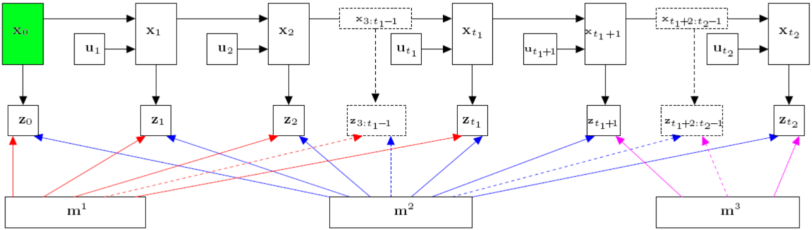
Données :



Gestion des cartes locales (2/3)

Données :

0 : instant initial

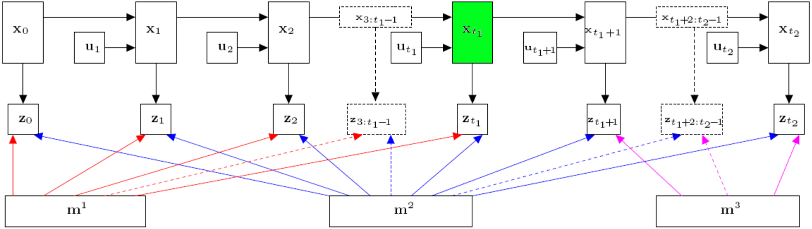


Gestion des cartes locales (2/3)

Données :

0 : instant initial

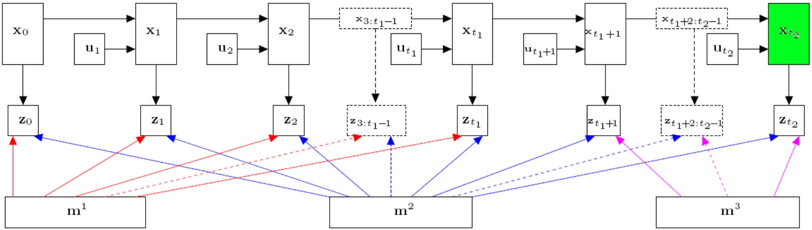
t_1 : fin de la première carte



Gestion des cartes locales (2/3)

Données :

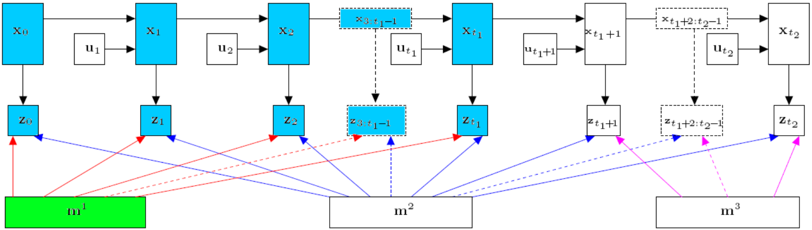
- 0 : instant initial
- t_1 : fin de la première carte
- t_2 : fin de la seconde carte



Gestion des cartes locales (2/3)

Données :

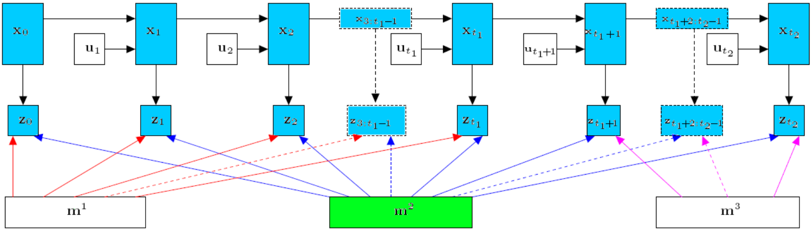
- 0 : instant initial
- t_1 : fin de la première carte
- t_2 : fin de la seconde carte
- $\mathbf{m}^1$: amers uniquement visibles dans la première carte



Gestion des cartes locales (2/3)

Données :

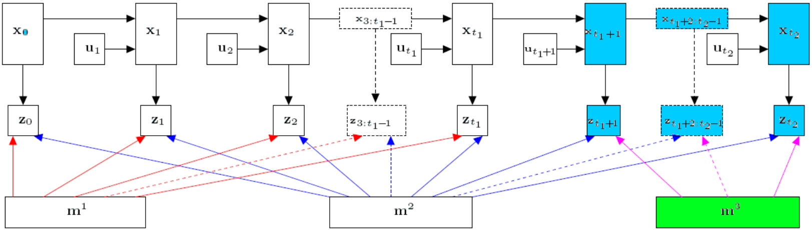
- 0 : instant initial
- t_1 : fin de la première carte
- t_2 : fin de la seconde carte
- $\mathbf{m}^1$: amers uniquement visibles dans la première carte
- $\mathbf{m}^2$: amers communs aux deux cartes



Gestion des cartes locales (2/3)

Données :

- 0 : instant initial
- t_1 : fin de la première carte
- t_2 : fin de la seconde carte
- $\mathbf{m}^1$: amers uniquement visibles dans la première carte
- $\mathbf{m}^2$: amers communs aux deux cartes
- $\mathbf{m}^3$: amers uniquement visibles dans la seconde carte



Gestion des cartes locales (2/3)

Données :

0 : instant initial

t_1 : fin de la première carte

t_2 : fin de la seconde carte

$\mathbf{m}^1$: amers uniquement visibles dans la première carte

$\mathbf{m}^2$: amers communs aux deux cartes

$\mathbf{m}^3$: amers uniquement visibles dans la seconde carte

Problème :

Est-il possible de calculer la densité associées aux positions du robot entre $t_1 + 1$ et t_2 ainsi que les amers $\mathbf{m}^2$ et $\mathbf{m}^3$ en prenant en compte l'**intégralité des mesures** ?

$\Longrightarrow p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^2, \mathbf{m}^3 | \mathbf{x}_{t_1}, \mathbf{z}_{0:t_2}, \mathbf{u}_{1:t_2}\right)$

Gestion des cartes locales (2/3)

Théorème :

$$p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^2, \mathbf{m}^3 | \mathbf{x}_{t_1}, \mathbf{z}_{0:t_2}, \mathbf{u}_{0:t_2}\right) \propto$$
$$p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^2, \mathbf{m}^3 | \mathbf{x}_{t_1}, \mathbf{z}_{t_1+1:t_2} \mathbf{u}_{t_1+1:t_2}\right) \times p\left(\mathbf{m}^2 | \mathbf{z}_{0:t_1} \mathbf{u}_{0:t_1}, \mathbf{x}_{t_1}\right)$$

Interprétation

1er facteur : s'obtient en appliquant un algorithme de SAM classique n'utilisant que les mesures et commandes de la seconde carte

2ème facteur : se déduit de la première carte

Multiplication : vecteur et matrice d'observation se déduisent par addition des paramètres déduits des deux facteurs

Plan

Plan

4 Le capteur omnidirectionnel
- Présentation générale
- Avantages et inconvénients
- Adaptations spécifiques

Présentation générale

Caméra omnidirectionnelle

- Il s'agit d'une caméra qui filme un miroir.
- La forme du miroir est une quadrique de révolution classique (ellipsoïde, paraboloïde, hyperbololoïde)

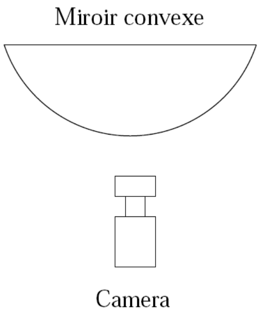


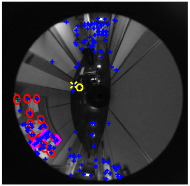
Figure 1: Capteur catadioptrique

Figure 2: Capteur à miroir parabolique

Avantages et inconvénients

Avantages

- Permet une vision totale de la scène à 360deg
- Permet de suivre les points même en cas de très forte rotation



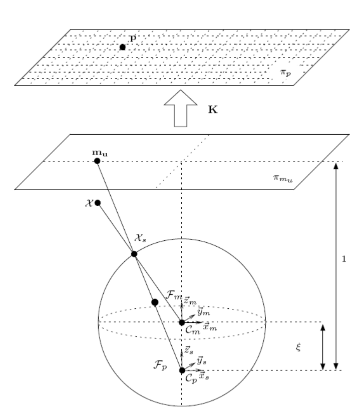
Inconvénients

- Les équations de projection classiques de la vision ne sont plus applicables
- Résolution non uniforme
- Système mécanique sensible aux vibrations

Adaptations spécifiques (1/3)

Modèle de Barreto

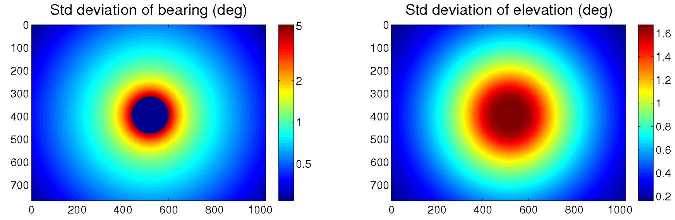
- Toute caméra omnidirectionnelle peut être identifiée à un modèle de projection sphérique vue depuis une caméra classique
- Implique de transformer les distances focales classiques
- Procédure spécifique de calibration



Adaptations spécifiques (2/3)

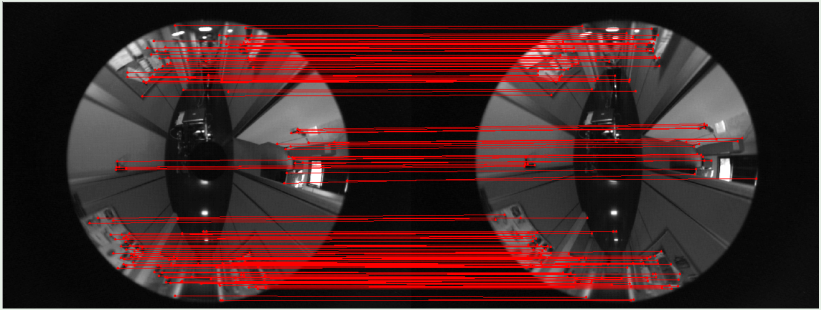
Covariances des mesures

- Calcul des covariances de l'azimuth et de l'élévation en fonction du point dans l'image
- Elle augmente lorsqu'on s'approche du centre de l'image



Adaptations spécifiques (3/3)

Mise en correspondance entre deux images omni



Plan

Plan

5 Résultats

- Première sequence : couloir de l'INRIA
- Seconde séquence : environnement d'intérieur avec de fortes variations lumineuses

Première sequence : couloir de l'INRIA

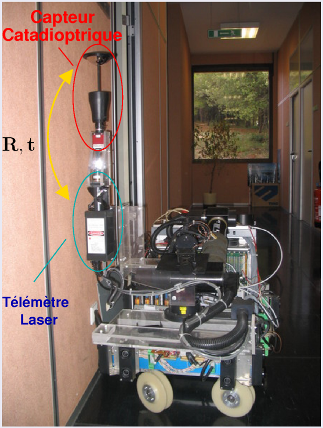
Paramètres

Lieu : couloir du bâtiment Borel de l'INRIA Sophia Antipolis

Plateforme : robot ANIS (développé en interne à l'INRIA)

Résultat : application de l'algorithme avec cartes locales

Lire la vidéo videoBorel.mpeg



Seconde séquence : environnement d'intérieur avec de fortes variations lumineuses (1/2)

Paramètres

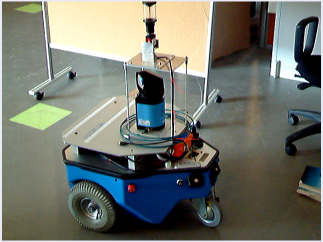
Lieu : halle robotique de l'INRIA

Plateforme : nouvelle plateforme Hannibal. Principale différence avec Anis : présence de pneumatiques (odométrie dégradée)

Difficulté : passage devant une baie vitrée très lumineuse

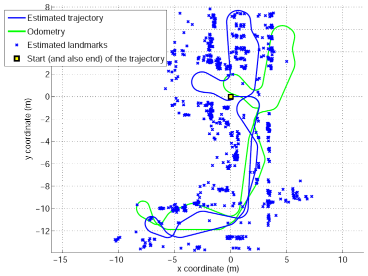
Résultat : application de l'algorithme avec cartes locales

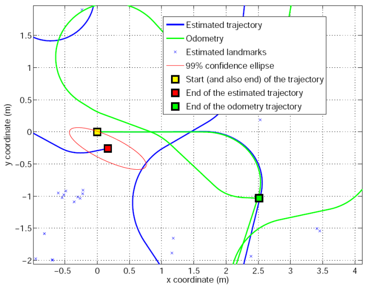
Lire la vidéo videoKahn.mpeg



Seconde séquence : environnement d'intérieur avec de fortes variations lumineuses (2/2)

Qualité du résultat :





Plan

Plan

6. Conclusion et perspectives
   - Conclusion
   - Perspectives

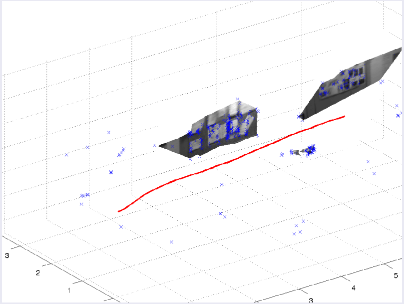
Conclusion

- Critère pour démarrer une nouvelle carte ne nécessitant pas de valeur absolue.
- Algorithme de création de nouvelle carte garantissant l'utilisation de toutes les informations disponibles
- Résultats expérimentaux convaincants :
  - Cartes cohérentes (angles droits respectés pour les murs, largeur des couloirs constante)
  - L'odométrie permet de fixer le facteur d'échelle, même lorsqu'elle est mauvaise

Perspectives (1/2)

Projection de textures

- Détection automatique de plan
- Reprojection des textures en fusionnant les différentes vues



Perspectives (2/2)

Estimation de l'espace libre

- Calculer la triangulation de Delaunay associée au nuage de points
- “ Creuser ” l'espace libre en fonction des rayons associés aux observations