Utilisation d'un module de vision stochastique pour localiser un robot mobile.

GECHTER Franck
Equipe MAIA
LORIA UHP NANCY I
BP 239 54506 Vandœuvre
Lès Nancy
gechterf@loria.fr

THOMAS Vincent
Equipe MAIA
LORIA UHP NANCY I
BP 239 54506 Vandœuvre
Lès Nancy
vthomas@loria.fr

CHARPILLET François
Equipe MAIA
LORIA INRIA
BP 239 54506 Vandœuvre
Lès Nancy
charp@loria.fr

Résumé

Le domaine d'application des modèles décisionnels de Markov est vaste et de nombreuses communautés de recherche se sont intéressées à leur application (contrôle de processus, recherche opérationnelle, économie, interprétation de signaux, ...). Ce n'est que récemment, qu'ils se sont progressivement développés dans la communauté de recherche en IA pour la conception d'agents autonomes. Leur succès est lié notamment à leur capacité à modéliser l'incertitude dans laquelle est plongé un agent : incertitudes liées à l'imperfection des capteurs, incertitudes liées à la complexité des modes d'interaction avec l'environnement, les autres agents ou l'utilisateur.

Nous présenterons, ici, une application de ces modèles pour la localisation d'un robot mobile.

I. Introduction.

L'utilisation de la vision en robotique mobile s'impose naturellement de part la quantité phénoménale d'informations pertinentes qu'elle apporte. Cependant, ce type de capteur, bien qu'étant quasiment indispensable pour de telles applications, est assez délicat à maîtriser tout en conservant des temps de réactions raisonnables et sans avoir recours à des processeurs dédiés. En effet, la plupart des approches proposées actuellement fonctionnent sur l'extraction et l'analyse de primitives pertinentes dans les images acquises par le véhicule autonome. Parmi ces types de traitement, nous pouvons citer les algorithmes de vision stéréoscopique qui analysent les éléments d'une scène prise sous différents angles de vue (deux ou trois généralement) pour effectuer une cartographie de l'environnement proche du véhicule ([1], [2], [11] et [12]). Ces algorithmes s'avèrent être très efficaces en environnement structuré, mais ils possèdent deux inconvénients importants. Le premier d'entre eux n'est pas directement lié à l'algorithme utilisé mais est récurrent dans quasiment tous les algorithmes de vision puisqu'il s'agit du temps de calcul. En effet, pour obtenir un traitement suffisamment rapide des images, c'est-à-dire proche du temps réel vidéo, il est nécessaire d'utiliser des processeurs spécialisés (DSP) pour chaque caméra. Le second inconvénient est quant à lui lié à la stéréovision. En effet, celle-ci nécessite un étalonnage préalable des capteurs pour permettre une reconstruction suffisamment fiable de l'environnement. Ces traitements, bien qu'étant correctement maîtrisés et surtout fiables ([4]), se révèlent être contraignants d'autant plus que, dans la majorité des utilisations, les caméras sont susceptibles de se dérégler en cours d'utilisation.

Le but de cet article est de proposer une alternative, peu coûteuse en temps de calcul pour une qualité aussi acceptable que celle obtenue avec des algorithmes classiques, afin de déterminer la position d'un robot mobile dans un environnement structuré et partiellement connu en utilisant une unique caméra en niveaux de gris. L'algorithme développé ici, est dérivé d'une technique de classification éprouvée qui est l'analyse en composantes principales (ACP ou Karhunen-Loeve Transform) ([5], [9], [10] et [13]). Cette technique a déjà été utilisée dans le cadre de la reconnaissance de visage avec ou sans prise en compte de l'influence des conditions de luminosité ([6] et [7]). Cette méthode a de nombreux avantages. En effet, elle permet de délocaliser la majorité des calculs avant l'utilisation sur la plate-forme mobile, ce qui lui permet d'être exploitée sans avoir recours à des processeurs dédiés. Pour obtenir des résultats satisfaisants, il faut disposer au préalable d'une base d'images correctement structurées de l'environnement dans lequel le véhicule doit évoluer. En outre, nous avons associé à cet algorithme de vision une méthode d'évaluation stochastique, basée sur les Processus de Décision Markoviens Partiellement Observables (POMDP), et destinée à fiabiliser les résultats obtenus par l'algorithme de vision seul (cf. résultats expérimentaux).

II. L'Analyse en Composantes Principales appliquée à une base d'image.

L'analyse en composantes principales (ACP) est un algorithme de classification couramment utilisé pour trier des ensembles de données multidimensionnelles homogènes. L'ACP permet, en outre, de rechercher les axes de discrimination maximum d'un nuage de points dans un espace de dimension quelconque. La classification des données s'effectue à partir des projections des points du nuage sur ces axes. La détermination de ceux-ci est réalisée en calculant les vecteurs propres de la matrice de covariance du système de points. Le pouvoir discriminant de chaque axe étant déterminé par sa valeur propre. Plus celle-ci est élevée plus l'axe engendré par le vecteur propre est discriminant.

Pour effectuer une reconnaissance de la position du véhicule mobile dans son environnement, il s'agit de rechercher des images de la base qui possèdent des propriétés radiométriques proches de l'image acquise. L'utilisation de l'ACP permet d'orienter cette recherche uniquement sur des propriétés globales des images contrairement aux algorithmes classiques qui s'attachent davantage à la recherche d'indices significatifs dans les scènes observées.

Pour pouvoir utiliser l'ACP sur des images, il faut tout d'abord considérer qu'une image est en fait un point dans un espace de dimension égale au nombre de pixels qu'elle contient (Les images de la base sont supposées être toutes de la même taille). L'ensemble des images de la base constitue alors un nuage de points dans un espace multidimensionnel. Il suffit d'utiliser l'ACP sur ce nuage de points pour calculer les axes les plus discriminants pour l'ensemble des images.
Cette méthode possède donc quelques avantages considérables sur les méthodes classiques puisqu'elle permet de ne raisonner que sur des propriétés globales des images, ce qui évite l'utilisation de balises dans l'environnement. D'autre part, elle permet de réduire considérablement les calculs pour effectuer la reconnaissance.

III. Mise en pratique.

Nous avons choisi, afin d'optimiser les temps de réponse de la partie embarquée de la reconnaissance de couloirs, de décomposer l'algorithme en deux parties distinctes. La première d'entre elles se charge de la préparation de la base d'images et de la création des vecteurs de poids. Cette partie de l'algorithme s'effectue sur une station fixe. La seconde est uniquement constituée de la partie reconnaissance. Celle-ci s'effectue lors des déplacements du robot au sein de son environnement.

Préparation de la base.

La préparation est l'étape la plus importante étant donné qu'elle conditionne le bon fonctionnement de la partie reconnaissance de l'algorithme.

Dans la suite du développement, les images de la base seront appelées Γi ($i \in [1,M]$ avec M le nombre d'images initialement présentes dans la base). D'autre part, dans tous les calculs de construction de la base, les images sont considérées comme étant des vecteurs colonnes dont les composantes sont les valeurs des différents pixels constituant l'image, ces pixels étant pris ligne par ligne dans chaque image. Une image de la base peut donc être écrite sous la forme :
$\Gamma=(L_1\ L_2\ \dots\ L_n)^{T.}$

Calculs préliminaires.

La première étape de la préparation de la base est le calcul de l'image moyenne notée $\mu = \frac{1}{M} \sum_{i=1}^{M} \Gamma_i$. Le calcul de la moyenne permet à posteriori de comparer les images uniquement à partir de leurs caractéristiques propres.

L'étape suivante consiste à calculer les différences entre les images de la base et l'image moyenne calculée précédemment. Ces écarts par rapport à l'image moyenne sont appelés ϕ_i avec $\phi_i=\Gamma_i - \mu$.

A partir de ces écarts, nous construisons la matrice A qui est le tableau dont les vecteurs colonnes sont les vecteurs ϕ_i. Nous avons donc $A=[\phi_1,\phi_2,...,\phi_M]$ qui est de dimension N^2 par M, N étant le nombre de lignes et de colonnes d'une image en supposant bien évidemment que les images de la base sont carrées. Le choix d'utiliser des images carrées a été fait dans un souci de clarté des explications. Le calcul de préparation de la base peut également s'effectuer avec des images non carrées.

Recherche des vecteurs propres et des valeurs propres de la matrice AA^T.

Pour pouvoir calculer les valeurs et les vecteurs propres de la base d'images, il faut, dans un

premier temps, construire une matrice carrée à partir de A calculée précédemment. Pour cela, il suffit de multiplier A par sa transposée. Nous recherchons alors les vecteurs propres et les valeurs propres de la matrice $A.A^T$. Cette matrice est de dimension N^2 par N^2. Etant donné que les images utilisées sont de taille relativement élevée (256 x 256 ou 320 x 240 typiquement), le calcul de vecteurs propres est assez long sur des machines standards. De façon générale, le nombre d'images de la base est très petit devant le nombre de pixels d'une image ($M << N^2$). Dans ce cas, le rang de la matrice $A.A^T$ est au maximum égale à M . Le nombre maximum de sous-espaces propres orthogonaux sera M. Pour réduire les temps de traitement, nous pouvons envisager d'effectuer le calcul des vecteurs propres à partir de la matrice $A^T.A$ qui est une matrice de dimension plus faible (M x M). Il suffit ensuite de retrouver les vecteurs propres de la matrice $A.A^T$ à partir de ceux de la matrice $A^T.A$.

Soit v_i un vecteur propre de la matrice $A^T.A$ attaché à la valeur propre λ_i . Nous pouvons alors écrire $A^TA.vi = \lambda_i.v_i$. Nous obtenons donc, en pré-multipliant cette égalité par la matrice A , $(A.A^T).A.v_i = \lambda_i.A.v_i$. Les vecteurs $A.v_i$ sont donc des vecteurs propres de la matrice $A.A^T$ attachés respectivement aux valeurs propres λ_i. Nous avons donc trouvé toutes les valeurs propres de la matrice $A.A^T$. Il est possible, dans un souci de précision, de conserver toutes les composantes principales pour chacune des images. Cependant, certaines de ces composantes sont inutiles car elles ne contiennent pas d'informations pertinentes. En effet, les vecteurs propres associés à des valeurs propres faibles ne représentent que les variations inter-images dues au bruit d'acquisition. Ces vecteurs ne sont donc pas pertinents dans la discrimination des images de la base. Dans toute la suite du développement, nous noterons M' le nombre de composantes principales conservées.

L'étape finale dans la recherche des vecteurs propres de la matrice de covariance est la construction de ceux-ci à partir des vecteurs propres de AA^T. Ce calcul se fait à l'aide de la formule suivante : $u_l = A.v_l = \Sigma\, v_{lk}\phi_k$ (k variant de 1 à M) où v_{lk} est la k^{ieme} coordonnées du vecteur v_l ,vecteur propre de AA^T attaché à la valeur propres λ_l .

Les vecteurs u_l sont les vecteurs sur lesquels vont être décomposées les ϕ_i.

Décomposition des ϕi sur la base des vecteurs propres : création des vecteurs de poids.

Il s'agit maintenant plus de calculer les vecteurs de poids qui correspondent à la projection sur la base de vecteurs propres de chacune des images de la base. Nous notons Ω_i le vecteur de poids correspondant à l'image ϕ_i. Le calcul des Ω_i se fait très simplement en pré-multipliant ϕ_i par la matrice dont les lignes sont les vecteurs u_l^T (Remarque : Cette matrice est de dimension M' N^2). Ce sont ces vecteurs de poids que nous utilisons pour effectuer des recherches dans la base. La dimension de ceux-ci étant très faible par rapport à la dimension des images initiales, le gain en temps de calcul est considérable.

Reconnaissance d'une image.

La recherche dans la base d'images s'effectue en deux étapes successives.

Dans un premier temps, il s'agit de détecter si l'image acquise par le robot est bien une image analogue à celles de la base. Pour cela, nous comparons cette image à l'image moyenne de la base. Si la différence entre ces deux images est suffisamment faible, nous considérons que l'image acquise est bien une image de couloir. La comparaison est réalisée en fonction d'un seuil déterminé comme étant le module du plus large écart entre les images de la base et l'image moyenne : $\max_i \| \Gamma_i - \mu \|^2$.

Dans un second temps, il faut rechercher la ou les image(s) de la base qui ressemble(nt) le plus à l'image acquise. Il s'agit ici de construire le vecteur de poids de l'image acquise et de le comparer à celui de chacune des images de la base. Cette comparaison s'effectue en calculant la norme des différences entre les vecteurs de poids. Nous classons, tout d'abord, ces normes. Puis nous les comparons à un seuil étroitement lié à la nature de la base. Nous ne conservons ensuite que les normes inférieures à ce seuil.

Nous pouvons également rechercher les images les plus proches en introduisant une notion d'incertitude. En effet, nous pouvons remarquer ici qu'il est assez aisé de probabiliser les résultats de proximité en fonction des normes obtenues. Nous définissons ainsi la probabilité de proximité entre l'image acquise Φ et l'image Φ_i de la base par la relation suivante : $P_i = \dfrac{1/\|\Omega - \Omega i\|}{\sum_i 1/\|\Omega - \Omega i\|}$. En procédant de cette manière, le seuil devient une probabilité limite de ressemblance qui est beaucoup plus simple à régler.

Il est important de noter que tous ces calculs s'effectuent à partir des vecteurs de poids et non à partir des images initiales ce qui réduit considérablement le coût en temps-machine.

Utilisation de la notion de voisinage pour limiter la recherche dans la base.

Pour obtenir une base la plus représentative possible de l'environnement dans lequel doit évoluer le robot, nous avons choisi de mailler ce dernier de manière analogue à ce qui est fait en planification. Ce maillage correspond au découpage de l'environnement en carrés d'environ un mètre de coté. A chacun de ces carrés, nous associons quatre images correspondantes aux quatre orientations possibles (Nord, Sud, Est et Ouest). Ce maillage, bien qu'étant relativement grossier, nous donne une représentation suffisamment complète de l'environnement. Cependant, pour un couloir d'une vingtaine de mètres le nombre d'images dans la base est déjà important (environ 80 images). Pour cette raison, nous avons choisi d'effectuer une recherche sélective pour effectuer la reconnaissance.

Ainsi, pour optimiser la recherche dans la base d'images, il faut limiter au maximum les comparaisons entre les vecteurs de poids. Dans ce but, nous avons choisi d'associer à chaque élément de la base un voisinage constitué de cinq images. Ces images correspondent aux acquisitions les plus probables lors de l'étape suivante du déplacement. Nous considérons que, au cours d'un déplacement normal, le robot n'effectue que des translations vers l'avant et des rotations de $\pi/2$ ou $-\pi/2$ étant donné que les translations vers l'arrière ne sont utilisées qu'en cas de nécessité impérieuse (Choc contre un mur ou évitement d'obstacle). Néanmoins, il est tout à fait envisageable de modifier les voisinages pour certaines positions particulières de l'environnement. Une fois les voisinages constitués, la reconnaissance s'effectue en priorité parmi ces cinq images. Si les résultats de cette recherche s'avèrent infructueux, c'est-à-dire si le voisinage envisagé ne contient pas d'images suffisamment proches de l'image acquise, il suffit d'étendre la recherche au reste de la base.

Le principal avantage de cette méthode est de limiter considérablement les temps de recherche puisque l'utilisation des voisinages permet, dans la majorité des cas, d'anticiper sur les acquisitions futures. Le principal problème réside dans le fait qu'il faut correctement étiqueter la base pour pouvoir construire sans trop de peine les voisinages. Néanmoins, nous pouvons envisager de réaliser la structuration de la base en voisinages en effectuant un apprentissage au cours des déplacements du robot dans son environnement. Dans ce cas, la construction des voisinages ne demande aucune intervention extérieure et permet en outre une mise à jour en fonction des évolutions de l'environnement.

IV. Diffusion de la position la plus probable en utilisant les Processus de Décision Markoviens.

Devant les limitations montrées par la méthode des voisinages expliquée dans le paragraphe précédent, nous avons développé un algorithme alternatif utilisant les Processus de Markov (MDP). Cette approche permet d'évaluer la position la plus probable du robot en prenant en compte la nature des déplacements du robot. Alors que dans la méthode des voisinages tous les états ont la même probabilité, la méthode à base de MDP diffuse les différentes probabilités parmi toutes les positions de l'environnement et partant d'une position de départ et en utilisant les probabilités de transition associés à chaque action déjà exécutée par le robot. L'algorithme utilisé se rapproche de celui développé par Simon et Koenig [8]. L'espace d'évolution du robot est décomposé en états qui correspondent chacun à une position et à une orientation. Ce modèle fournit des estimations de la position du robot appelés *«Belief States »* notés B(t,s). Ceux-ci représentent la probabilité, à chaque pas de temps t, de se trouver dans l'état s. Ils sont calculés en prenant en compte les actions et les observations effectuées.

Un Processus de Décision Markovien (MDP) est un automate probabiliste : un état de départ et une action donnés peuvent aboutir à plusieurs états différents selon une distribution de probabilité. Un MDP est donc définit de la façon suivante :

- S, un ensemble fini d'états (Dans notre cas correspond à une localisation du robot (position et orientation).
- A, un ensemble fini d'actions (Dans notre cas : avance, tourne à droite et tourne à gauche).
- T[s'| s,a], une distribution de probabilité.

Le résultat d'une action est déterminé par la matrice T qui donne la probabilité qu'a le système de se trouver dans l'état ***s'*** sachant que la position de départ était ***s*** et que l'action ***a*** a été effectuée. Dans la formulation MDP le système connaît toujours sont état courant. Malheureusement, cette formulation ne convient à notre problème étant donné que nous cherchons à localiser le robot dans son environnement, nous ne pouvons donc pas connaître sa position à chaque instant. Nous devons donc utiliser une formulation plus générale : les POMDP (Processus de Décision Markoviens Partiellement Observables). Dans ce cas la position actuelle du robot n'est pas connue mais nous disposons d'une nouvelle information sous la forme d'une observation représentative de la position courante. Un POMDP est donc défini de la façon suivante :

- Un MDP (un ensemble d'état S, un ensemble d'actions A et une distribution de probabilité T)
- O, un ensemble d'observation.
- P[o|s], une distribution représentant la probabilité d'effectuer l'observation o en se trouvant dans l'état s.

Ce modèle fournit des estimations de la position du robot sous la forme de *belief states* qui sont calculés en prenant en compte l'état précédent, l'action effectuée et l'observation réalisée selon la loi suivante :

$$B(t,s)=K\times(\sum_{s'\in S}T(s,s',a)\times B(t-1,s'))\times P(o,s)$$

Equation 1

Cette loi peut être décomposée en deux parties :

- La première partie, ($\sum_{s'\in S}T(s',s,a)\times B(t-1,s')$), représente l'évolution des *belief states* en prenant en compte les actions effectuées par le véhicule. Elle consiste en l'estimation de la probabilité de suivre une transition entre un état inconnu et l'état ***s***. La probabilité de suivre la transition de ***s'*** vers ***s*** est égale à la probabilité d'être dans l'état ***s'*** au temps ***t-1*** multipliée par la probabilité d'effectuer la transition vers ***s*** sachant que le système se trouve dans la position ***s'***. Il s'agit ensuite d'effectuer la somme sous l'ensemble des états s' possibles.
- La seconde partie $P(o,s)$ est une mise à jour du *belief state* en utilisant l'observation réalisée.

Le facteur K est un facteur de normalisation destiné à s'assurer que la sommes des probabilités vaut 1.

Première approche : pseudo POMDP

Nous pouvons utiliser les POMDP selon deux approches différentes. La première consistent en l'utilisation des MDP en tant qu'une aide passive à la localisation. Dans ce cas un pseudo POMDP est utilisé pour évaluer les positions possibles du robot dans l'ensemble de l'environnement en prenant en compte la position initiale et les transitions successives. Le pseudo POMDP peut être considérer comme étant un POMDP dont les probabilités d'observation sont toutes égales à 1. (cf. équation 2). Cela signifie donc que les *belief states* ne sont pas mis à jour avec les probabilités d'observation.

$$B(t,s)=K\times\sum_{s'\in S}T(s,s',a)\times B(t-1,s')$$

Equation 2

Le robot part d'une position parfaitement connue : le *belief state* vaut 0 partout sauf en un état correspondant à la position de départ. Le système évolue ensuite étape par étape tout en considérant que la vraie position du robot se trouve parmi les états qui possède le plus grand *belief state*. Les résultats obtenu par cette méthode sont très encourageant au début de l'exécution. Après un certain nombre d'étapes, les résultats sont moins probant puisque les probabilités tendent à devenir uniformes. Dans ce cas, la localisation ne plus être effectué que par l'ACP comme si elle était utilisée seule.

Seconde approche : POMDP

Dans cette seconde approche le POMDP est utilisé pour améliorer les performance du système. Cette méthode est sensiblement identique à celle développé précédemment. Cependant, nous avons ajouté une rétroaction en prenant en compte l'observation réalisée par le module de vision. L'élément clef de cette approche est l'utilisation de l'information fournie par l'analyse en composantes principales pour mettre à jour les *belief states* (cf. Equation 3). L'information fournie par l'ACP est une évaluation de distance entre l'image acquise est une image donnée de la base de départ effectuée dans l'espace de projection. Pour effectuer la mise à jour, nous prenons en compte le fait que plus la distance entre les images est petite plus la probabilité de se trouver dans l'état correspondant doit être grande. Comme dans la méthode précédente la recherche dans la base est réduite en utilisant un seuil défini empiriquement.

$$Bt(s)=K\times\left(\sum_{s'\in S}T(s,s',a)\times B_{t-1}(s')\right)\times\frac{1}{d(u,s)}$$

Equation 3

Les résultats obtenus par cette méthode sont très prometteurs. Si l'image reconnue par le module de vision correspond au *belief state* le plus élevé celui-ci devient encore plus prédominant puisque la distance entre les images est très faible. Par opposition, si l'image reconnue n'est pas la bonne, le *belief state* ne sera pas profondément modifié. Cette méthode évite tous les problèmes de téléportation caractéristiques de la localisation par ACP. Cette méthode est également très robuste :

même si la position de départ est inconnue le robot arrive à retrouver sa vraie position en très peu d'étapes. Ainsi la reconnaissance n'est plus nécessaire à toutes les étapes : il suffit de pouvoir évaluer si le robot est perdu et de déclencher une reconnaissance visuelle uniquement dans ce cas. Dans la situation contraire le POMDP est suffisamment robuste pour permettre une navigation précise.

v. Résultats expérimentaux.

L'environnement de test

Pour obtenir une base de départ la plus représentative possible de l'environnement, nous avons découpé ce dernier en petites parcelles d'environ un mètre carré. Chacune de celles-ci est en outre composée de huit images correspondants au orientations suivantes : N, N-O, O, S-O, S, S-E, E et N-E.

L'environnement est une surface d'environ 18 m² possédant des conditions d'éclairage non contrôlables (cf. figure 1). La zone de test possède une très grande surface vitrée les conditions d'éclairage sont donc fortement conditionnées par la lumière du jour. Nous pouvons considérer la zone de test comme étant un environnement semi-ouvert dans le sens ou il possède une structuration propre à un environnement fermé tout en ayant les propriétés lumineuses d'un environnement extérieur. De plus, les murs et le sols sont très réflectifs ce qui rend difficile l'élaboration d'un algorithme de vision fiable.

Figure 1: Some examples of the pictures used for the space model

Comparaison des méthodes

Pour effectuer les tests, nous avons réalisé un programme de simulation qui prend en compte les contraintes liées à l'utilisation d'un vrai robot. En particulier, chaque action est effectuée en utilisant une modélisation des défauts et des aléa du système de propulsion.

La base structurée représentant l'environnement est composée de 144 images. La base de test est également composée de 144 images mes prises avec des positions légèrement décalées par rapport à la base précédente et des conditions d'éclairage différentes.

Nous avons effectué les tests en réalisant la même séquence d'action pour les deux méthodes exposée dans le document: l'utilisation des voisinages et la méthode à base de processus de Markov. Nous avons dénombré le nombre d'échecs dans l'estimation de la position du robot et le nombre de re-calibrages réussis. La figure 2 montre le taux de mauvaise estimation obtenu par les méthodes. La méthode étiqueté *"POMDP à deux caméras"* est en fait la même méthode que celle exposée dans la partie IV mais en utilisant une seconde observation fournie par une caméra placée perpendiculairement à la première. La mise à jour des probabilités s'effectue en multipliant l'équation des *belief states* par l'inverse de la somme des distances fournies par les deux caméras.

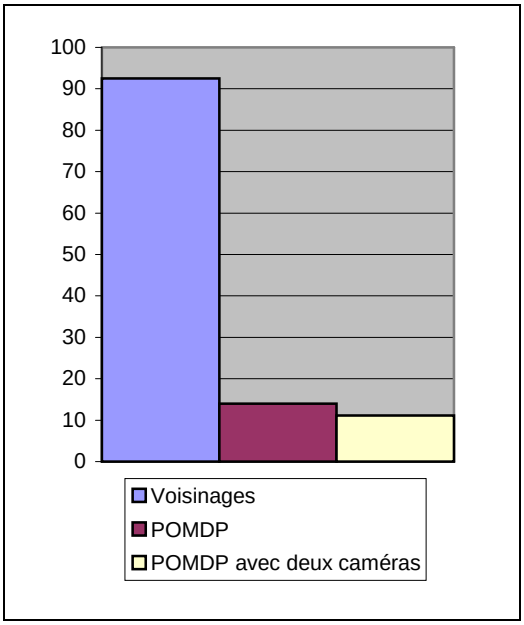

Figure 2: taux de mauvaises estimations.

Ces premiers résultats montrent que la méthodes à base de processus de Markov possède un taux très bas qui est en outre accentué par l'utilisation d'un second capteur. La figure 3 représente une évaluation de l'efficacité des trois méthodes. L'efficacité est calculée en effectuant une comparaison entre le taux de mauvaises estimations et le nombre de re-calibrage réussis. De plus, pour permettre une meilleure évaluation des performances les résultats ont été forcé à 100% pour la méthode POMDP à une seule caméra.

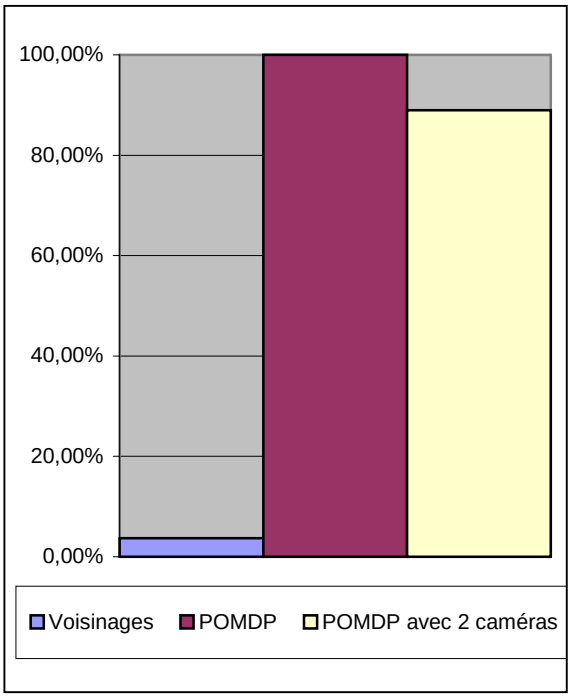

Figure 3 : Efficacité des différentes méthodes.

VI. Conclusion.

Nous avons donc réussi à développer un algorithme de reconnaissance de la position d'un robot mobile dans un environnement structuré dynamique en utilisant uniquement un seul dispositif d'acquisition d'images. Les avantages majeurs de cet algorithme sont, d'une part, son faible coût en ressources tant du point de vue logiciel que du point de vue matériel et, d'autre part, son efficacité. Le module de reconnaissance embarqué est caractérisé par un temps de calcul très faible en comparaison avec les algorithmes couramment utilisés. Le fait de n'utiliser qu'une seule caméra pour évaluer la position du robot permet d'envisager une utilisation en parallèle avec d'autres algorithmes de vision. En outre, le faible coût de ce module lui permet d'être utilisé sur des plate-formes mobiles de faible puissance. L'ajout d'une seconde caméra améliore encore la précision du module de reconnaissance de position.

Les résultats étant très prometteurs, nous envisageons de développer davantage cette méthode notamment au niveau de l'apprentissage de la base et de l'utilisation d'autres types de capteurs.

Des tests plus approfondis sont envisagés pour vérifier les performances de cette localisation, notamment en situation réelle et dans un environnement d'extérieur.

VII. Bibliographie.

[1] D. Murray et J. Little, *Stereo vision based mapping and navigation for mobile robots*, Computer Science Dept University of British Columbia Vancouver 1996.

[2] D. Murray et J. Little, *Using real-time stereovision for mobile robot navigation*, Computer Science Dept University of British Columbia Vancouver 1996.

[3] HARTLEY Richard I. et STURM Peter, *Triangulation*, Computer Vision and Image Understanding Vol.68 No 2 November pp 146-147, 1997

[4] **TSAI R. Y.,** *An Efficient an Accurate Camera Calibration Technique for 3D Machine Vision*, IEEE J. of Robotics and Automation, RA-3 : 364 –374 1987

[5] M.J. O'Connel, *Search Program for Significant Variables*, Comp. Phys. Comm. 8 (1974) 49.

[6] **M. A. Turk and A. P. Pentland,** *Face recognition using eigenfaces*, In Proc. IEEE Conf. on Computer Vision and Pattern Recognition, pages 586-591, 1991.

[7] **P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman,** *Eigenfaces vs. fisherfaces: Recognition using class specific linear projection*, IEEE Trans. Pattern Analysis and Machine Intelligence, 19(7):711-720, 1997.

[8] R. SIMMONS and S. KOENIG, *Probabilistic Robot Navigation in Partially Observable Environments*, International Joint Conference on Artificial Intelligence, Montreal Canada, August 1995.

[9] R. SIM and G. DUDEK, *Learning and Evaluating visual Features for Pose Estimation*, CVPR Workshop on Perception for Mobile Agents, Ft. Collins, CO, June 1999.

[10] R. SIM and G. DUDEK, *Learning Visual Landmarks for Pose Estimation*, ICRA 1999

[11] Clark F. OLSON, *Subpixel Localization and Uncertainty Estimation Using Occupancy Grids***,** Proceedings of the IEEE International Conference on Robotics and Automation Detroit May 1999.

[12] Sharon L. LAUBACH, Clark F. OLSON, Joel W. BURDICK, and S. HAYATI *Long Range Navigation for Mars Rovers Using Sensor-Based Path Planning and Visual Localisation*, In Proceedings of the 5th International Symposium on Artificial Intelligence, Robotics and Automation in Space, 1999.

[13] F. POURRAZ and J. L. CROWLEY, *Continuity Properties of the Appearance Manifold for Mobile Robot Position Estimation*, Workshop on Perception for Mobile Agents, CVPRí99 June 1999.