Planification probabiliste de trajectoires

Théo Zimmermann

Février 2013

Résumé

Dans ce rapport de lecture, on résume l'article [2]. On présente le problème traité, la méthode appliquée et des résultats qu'ils soient de nature théorique ou expérimentale. On note au passage quelles sont les difficultés soulevées, on signale des perspectives intéressantes, qu'elles soient ou non proposées par les auteurs. Enfin, ce rapport contient une critique de la preuve du théorème de convergence bien qu'il est probable que le résultat soit vrai pour autant.

Table des matières

1 Introduction à la planification de trajectoires

1.1 Présentation du problème

Le probème traité dans [2] est celui de la *planification de trajectoire* (*kinodynamic planning* en anglais). Il s'agit en fait, comme d'habitude dans le domaine de la planification de mouvement, de prévoir comment mouvoir un robot dans l'espace par des calculs préliminaires à l'action. Le robot évoluant dans le monde réel, ou un monde réaliste dans le cas d'une simulation, il est soumis à des *contraintes* dues aux lois de la physique.

L'approche traditionnelle consiste à s'abstraire de ces contraintes supplémentaires, calculer un chemin sans collision, puis demander au robot de le suivre le mieux possible. C'est-à-dire qu'une fois en mouvement, le robot doit approximer un chemin donné tout en respectant des contraintes propres sur son équilibre, sa vitesse, etc. Ainsi, on repousse à la phase d'action une partie des calculs.

De plus, une telle approche peut même parfois échouer. L'exemple suivant n'est pas tiré de l'article. Même en s'abstraignant des lois de la physique, de simples limitations sur les capacités du robot suffisent, à l'aide de considérations purement cinématiques, à constater l'échec, dans le cas général, de l'approche traditionnelle. Ici, on considère que les vitesses du robot en marche et en rotation sont comprises entre des valeurs minimales et maximales. Le robot peut bien-sûr s'arrêter, mais on suppose qu'il doit être avancer pour effectuer un tour en même temps.

Exemple 1. *Un robot se déplace à une vitesse minimale v et pivote à une vitesse maximale ω. Pour réaliser un virage en épingle, il aura besoin de la longueur $D = \frac{v\pi}{\omega}$.*

Ainsi, le robot de l'exemple a besoin d'un certain espace pour pouvoir effectuer son virage.

Ces différentes raisons justifient qu'on s'intéresse ici à la manière de planifier une trajectoire réaliste.

Définition 1 (Planifier une trajectoire)**.** *Déterminer comment contrôler le robot pour le conduire d'une position et une vitesse initiale $(p_i, \vec{v}_i)$ dans une position et une vitesse finale $(p_f, \vec{v}_f)$ tout en obéissant à un modèle dynamique et en évitant les obstacles.*

Pour résoudre ce problème, l'article propose un algorithme basé sur les *Rapidly-exploring Random Trees* (RRT) (une présentation résumée et panoramique en est donnée dans [1]). Cet algorithme permettra à la fois de traiter le cas de systèmes holonomes et non holonomes, ce qui est naturel car les contraintes provenant du modèle physique sont elles-même souvent non holonomes. Par contre, la méthode proposée ne cherche pas à fournir une trajectoire optimale, proche de l'optimum ou même homotopiquement équivalente à l'optimum (continûment déformable en une trajectoire optimale). Cette question est seulement esquissée dans les perspectives.

1.2 Différentes approches

Tout d'abord, vu la complexité du problème, qui provient notamment du grand nombre de dimensions de l'espace à explorer, une approche complète n'est pas envisageable. Pour illustrer ce propos, les auteurs remarquent que le problème de planification de trajectoire est au moins aussi complexe que le problème du déménageur de piano généralisé, qui est lui-même PSpace-dur [3].

Il faut donc s'orienter vers une méthode probabiliste. Les auteurs analysent les deux méthodes les plus utilisées et expliquent pour chacune pourquoi elle est mal adaptée au problème.

Le reproche qui est fait à la méthode de descente du gradient est sa trop grande dépendance vis à vis du choix de la fonction de potentiel. Elle ne permet pas de donner une méthode générale, facilement applicable et efficace pour la planification de trajectoire, ce qui est le but des auteurs.

Quant à la méthode des feuilles de route probabilistes, tout d'abord elle se révèle trop peu efficace en grande dimension car elle nécessite en pratique de calculer un nombre de noeuds trop important. Par ailleurs, elle est bien adaptée à des requêtes multiples car elle intègre une phase de calcul préliminaire mais l'objectif de l'article est de répondre efficacement à une requête unique.

1.3 Représentation du problème

1.3.1 L'espace des états

Pour représenter le problème, les auteurs s'inspirent largement de ce qui se fait déjà en planification de mouvement. C'est pourquoi, on définit l'espace des états $\mathcal{X}$, qui est pensé sur le modèle de l'espace des configurations pour un problème de simple calcul de chemin.

Définition 2. *L'espace des états $\mathcal{X}$ est l'ensemble des combinaisons position-vitesse $x = (q, \dot{q})$. On pourrait théoriquement prendre en compte aussi des dérivées d'ordre supérieur.*

Par rapport à l'espace des configurations, celui-ci a une dimension plus grande (deux fois plus grande si on considère que q appartient à l'espace des configurations). Comme dans le cas du problème de planification de mouvement, il s'agira de déterminer un chemin admissible dans cet espace $\mathcal{X}$, c'est-à-dire respectant un certain nombre de contraintes.

1.3.2 Contraintes

On considère deux types de contraintes. D'une part les contraintes dynamiques et locales provenant des lois de la physique, d'autre part les contraintes statiques et globales de non-collision avec les obstacles (on ne considère pas d'obstacle en mouvement).

Les contraintes provenant du modèle physique sont des *contraintes différentielles* car elles s'expriment généralement en fonction de la position, de la vitesse et de l'accélération et des contraintes non holonomes (lois de conservation). En prenant en compte l'ensemble de ses contraintes, on se retrouve normalement avec un problème sous-contraint. On complète donc

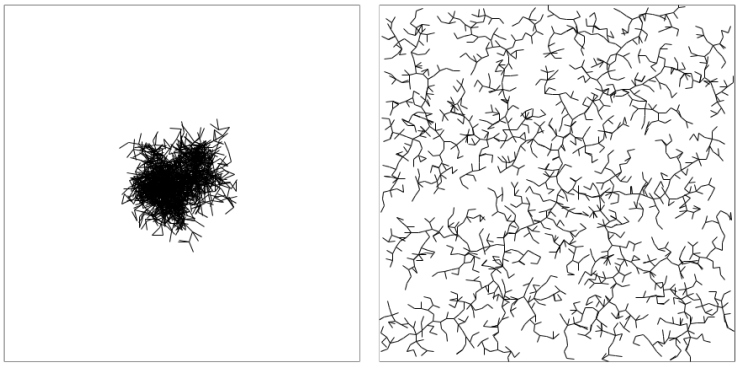

FIG. 1 – Exploration naïve contre exploration rapide. Chaque arbre compte 2000 noeuds

avec des contraintes supplémentaires qui expriment l'impact des différents contrôles $u \in U$ sur le mouvement.

Alors, on va essayer d'écrire l'ensemble des équations paramétrant le mouvement du robot en fonction uniquement de $x \in \mathcal{X}$, $u \in U$ et $\dot{x}$ sous la forme

$$\dot{x} = f(x, u)$$

L'accélération se retrouve alors cachée dans $\dot{x}$.

On intègre ensuite numériquement cette équation afin de pouvoir construire la trajectoire de manière *incrémentale*. Si on ne peut pas trouver une telle expression, l'important pour utiliser l'algorithme est en tout cas de parvenir à une expression de la forme

$$x(t + \Delta t) \approx g(x(t), \Delta t, u)$$

Les obstacles sont traités de manière traditionnelle. On définit $\mathcal{X}_{obst} \subseteq \mathcal{X}$ comme le sous-ensemble de l'espace des états dans lequel le robot est en collision avec l'un des obstacles. Une subtilité spécifique au problème est que l'on peut définir aussi $\mathcal{X}_{ric} \subseteq \mathcal{X}$, $\mathcal{X}_{ric} \supseteq \mathcal{X}_{obst}$ comme l'ensemble des positions conduisant inévitablement à une collision (en tenant compte de la vitesse et de contraintes différentielles de type inertie). Alors, la trajectoire doit nécessairement être recherchée dans le sous-ensemble $\mathcal{X}_{free}$ où

$$\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{ric}$$

2 Approche adoptée

2.1 L'algorithme RRT

La méthode adoptée est basée sur la construction d'arbres RRT [1]. Cet algorithme permet d'explorer au plus vite tout l'espace accessible, contrairement à une approche naïve de construction d'arbre aléatoire (voir la différence Figure 1). Rappelons le principe de cet algorithme.

1. Partir d'un arbre initial enraciné en x_{init} et ne comportant aucune branche.
2. Choisir un état x_{rand} au hasard dans l'espace entier à explorer ($\mathcal{X}_{free}$ dans notre cas).
3. Sélectionner son plus proche voisin x_{near} dans l'arbre.
4. Etendre l'arbre par un nouvel état x_{new} raccordé à x_{near}, le plus proche possible de x_{rand}.
   C'est-à-dire dans notre cas, sélectionner au mieux u_{new} de manière à minimiser
   $$d(x_{new}, x_{rand})$$
   où
   $$x_{new} = x(t_{near} + \Delta t) = g(x_{near}, \Delta t, u_{new})$$
5. Recommencer à l'étape 2.

2.2 Adaptation au problème traité

2.2.1 Les complications spécifiques

Cet algorithme fait appel à des procédures spécialisées : choisir un état x_{rand} au hasard (on prendra toujours une distribution uniforme ; l'article ne se pose pas la question de l'impact de ce choix et des autres solutions envisageables), calcul du plus proche voisin x_{near}, calcul d'un noeud x_{new} permettant de se rapprocher au mieux de la cible x_{rand}. Il nécessite aussi de fixer des paramètres, la métrique en particulier.

Les choix des paramètres et des algorithmes utilisés pour ces procédures spécialisées sont importants. Ils peuvent avoir un fort impact sur la complexité globale.

Par exemple, le calcul du plus proche voisin ne peut pas être effectué par une méthode complète car il serait trop complexe à cause du nombre élevé de dimensions de l'espace des états. Il faut recourir à des approximations.

De même, il est absolument inenvisageable de calculer la métrique idéale, qui serait une fonction estimant le coût qu'il y a à aller d'un point à un autre. Calculer cette métrique serait au moins aussi compliqué que de résoudre le problème initial de manière complète. Pourtant, l'impact du choix de la métrique peut être énorme. Il faudra donc faire une estimation grossière de cette métrique idéale si on veut gagner en performance. Cependant, ce choix nécessaire nous conduit à avoir le même genre de problème qui était soulevé par la méthode du gradient avec le choix de la fonction de potentiel.

Enfin, on notera un problème lorsqu'on applique cette méthode : comme on doit rester dans $\mathcal{X}_{free}$, il est plus difficile d'emprunter les passages étroits.

2.2.2 Une amélioration empirique

En pratique, on observe une légère amélioration des performances lorsque l'exploration de l'espace des états a lieu à l'aide de **deux arbres** enracinés l'un en x_{init} et l'autre en x_{goal}.

On procède alors de la manière suivante : on étend alternativement les deux arbres et pour chacun on alterne entre une étape d'exploration usuelle (vers x_{rand} aléatoire) et une étape de rapprochement (vers le dernier noeud x'_{new} ajouté dans l'autre arbre). On s'arrête lorsqu'on a construit un chemin, en reliant les deux arbres (exactement ou approximativement).

Il est naturel de constater une augmentation des performances. En effet, l'exploration n'est plus entièrement aléatoire mais, au contraire, elle est dirigée vers une cible choisie une fois sur deux. Cela est particulièrement intéressant lorsque les deux arbres ont particulièrement grandi pour être "visibles" ou "presque visibles" l'un de l'autre. Bien que ça ne soit pas proposé dans l'article, il serait certainement intéressant de raffiner cette idée en rajoutant aléatoirement quelques buts intermédiaires et faire grossir autant de nouveaux arbres. Cela reviendrait à combiner l'exploration efficace des RRT avec la capacités des feuilles de route probabilistes à trouver des passages inattendus.

3 Résultats et conclusion

3.1 Analyse théorique de l'algorithme

Deux types de résultat d'analyse de l'algorithme sont présentés et ils permettent de valider la pertinence de la méthode d'un point de vue théorique. Ce sont des résultats de convergence et de vitesse de convergence. Pour des raisons de simplicité, l'algorithme qui est analysé est celui n'utilisant qu'un seul arbre RRT.

Le théorème suivant est un théorème de convergence.

Théorème 1. *S'il existe une solution (une suite de contrôles* $(u_1, u_2, \cdots, u_k)$ *conduisant à une trajectoire admissible menant de* x_{init} *à* x_{goal}*), alors la probabilité que l'algorithme trouve une solution tend vers 1 lorsque le nombre d'étapes tend vers l'infini.*

3.1.1 Preuve du théorème 1

Les véritables hypothèses du théorème 1 sont en fait beaucoup plus restrictives. Une de ces hypothèses suppose que le choix des paramètres de contrôle u à un instant t se fasse dans un ensemble fini U. On aurait pu s'attendre au contraire à ce qu'il ait lieu dans un ensemble continu mais on peut aussi comprendre que cet ensemble, continu en théorie, puisse en fait être discrétisé à cause de la limite de précision des contrôles du robot ou encore qu'il soit réellement de nature finie (par exemple, une seule vitesse d'avancement ou de rotation est autorisée).

Par ailleurs, pour des raisons de simplicité, on suppose que le choix du contrôle à appliquer a lieu de manière aléatoire alors qu'en vrai on cherche à minimiser la distance entre x_{new} et x_{rand} et que c'est ce dernier qui est sélectionné de manière aléatoire. Les deux visions ne sont pas tout à fait équivalentes puisque le choix de x_{near} dépendait déjà de x_{rand} et que par conséquent, on ne peut pas supposer que les variables aléatoires X_{near} et U_{new} soient indépendantes. De plus, cela est implicite dans l'article

mais il faut comprendre que tous les paramètres de contrôles de U ont une chance non nulle (toujours supérieure à une certaine constante $a > 0$) d'être sélectionnés.

Si on peut faire un reproche à cette preuve, il est davantage dans la nature de l'argument apporté que dans le nombre impressionnant d'hypothèses. En effet, la preuve cherche à montrer que la trajectoire définie par la suite de contrôles $(u_1, u_2, \cdots, u_k)$ a une probabilité tendant vers 1 d'être découverte.

Pour cela, on montre que chaque étape a une chance tendant vers 1 de finir par être réalisée et on raisonne ensuite par une induction tout ce qu'il y a de plus naturelle.

Supposons qu'on ait construit la trajectoire de $x_0 = x_{init}$ jusqu'à x_i. Passer à l'étape x_{i+1} suppose seulement que deux facteurs soient réunis. Le premier est que le choix de x_{near} tombe précisément sur x_i et le second est que le choix de u_{new} tombe précisément sur u_{i+1}.

On sait que tous les paramètres de contrôles ont une probabilité supérieure à la constante $a > 0$ d'être sélectionnés. C'est en particulier vrai pour u_{i+1}.

Quant à x_i, sa région associée, dans le diagramme de Voronoï construit à partir des noeuds du RRT dans son état actuel, est non vide et on peut donc considérer $c_1 > 0$ tel que $\mu(Vor(x_i)) > c_1$, où μ est la mesure de probabilité utilisée pour faire le tirage aléatoire de x_{rand} (la mesure de Lebesgue pour faire simple). Alors x_i a la probabilité $\frac{\mu(Vor(x_i))}{\mu(\mathcal{X}_{free})} > \frac{c_1}{\mu(\mathcal{X}_{free})} > 0$ d'être sélectionné. L'erreur consiste alors à croire que tant que x_i ne sera pas sélectionné, il aura encore une chance supérieure à $\frac{c_1}{\mu(\mathcal{X}_{free})}$ d'être sélectionné au coup d'après. Si c'était vrai, on pourrait effectivement conclure. Ce n'est pas le cas, car à chaque étape, l'algorithme fait grossir le RRT en rajoutant de nouveaux noeuds. Par conséquent, le diagramme de Voronoï n'est plus le même et la région associée à x_i est rétrécie.

3.1.2 Vitesse de convergence

Le résultat suivant est un résultat purement théorique car il dépend d'une notion (les *suites d'attractions*) aussi complexe à calculer que ce serait de résoudre le problème initial de manière complète. Cependant, il reste intéressant car il permet de "voir" que les passages étroits ralentissent la trouvaille d'une solution.

Théorème 2. *Si une suite d'attraction $(\mathcal{A}_0, \mathcal{A}_1, \cdots, \mathcal{A}_k)$ existe, alors l'espérance du nombre d'étapes nécessaires pour trouver une solution est inférieure à $\frac{k}{p}$ où*

$$p = \min_{i=1..k} \left\{ \frac{\mu(\mathcal{A}_i)}{\mu(\mathcal{X}_{free})} \right\}$$

Définition 3. *Une suite d'attraction est une suite de sous-ensembles $(\mathcal{A}_0, \mathcal{A}_1, \cdots, \mathcal{A}_k)$ de l'espace libre $\mathcal{X}_{free}$ tel que $\mathcal{A}_0 = \{x_{init}\}$, $\mathcal{A}_k = \mathcal{X}_{goal}$ (l'ensemble des points suffisamment proches de l'objectif pour que ce dernier soit considéré comme atteint) et pour chaque $i = 1..k$, il existe $\mathcal{B}_i \subseteq \mathcal{X}_{free}$ contenant $\mathcal{A}_{i-1}$ et $\mathcal{A}_i$ et vérifiant :*

1. *Tous les points de $\mathcal{A}_{i-1}$ sont plus proches de tous les points de $\mathcal{A}_i$ que de n'importe quel point extérieur à $\mathcal{B}_i$.*

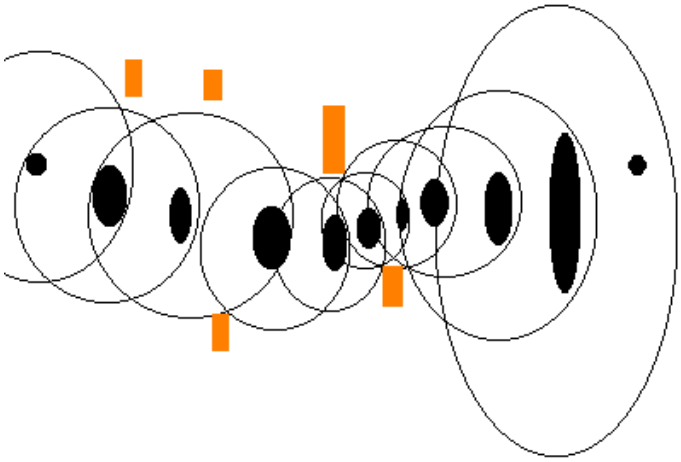

FIG. 2 – Suite d'attraction contruite à la main pour un exemple dans le plan

2. *De tout point de $\mathcal{B}_i$, il existe une suite de contrôles conduisant à une trajectoire admissible menant à un point de $\mathcal{A}_i$.*

Plus cette suite d'attraction est courte et plus les ensembles $\mathcal{A}_i$ sont grands, plus on réduit l'espérance du nombre d'étapes pour trouver une solution. Cependant, il est difficile même de manière visuelle, de trouver la meilleure suite d'attraction pour un problème donné. En effet, agrandir les $\mathcal{A}_i$ a tendance à rallonger la suite. On présente en Figure 2 un exemple de suite d'attraction. Les obstacles sont les rectangles oranges, les $\mathcal{A}_i$ les ovales pleins et les $\mathcal{B}_i$ les cercles.

3.2 Résultats expérimentaux

3.2.1 Des paramètres variables...

Des tests ont été effectués pour des simulations d'aéroglisseurs et de vaisseaux spatiaux. Ces deux objets ont en commun d'être des corps 3D rigides. Dans tous les tests, on choisit pour métrique la distance euclidienne. Les auteurs remarquent qu'il serait intéressant d'en essayer d'autres et de mesurer l'impact de ce choix. A chaque fois, on tient compte des lois de la physique Newtonnienne (l'inertie en particulier) mais pas de la gravité.

Les paramètres qui varient sont la forme de l'objet, la distribution (aléatoire ou déterministe) et la forme des obstacles, la dimension de l'espace. Ces différents paramètres affectent la forme et la dimension de $\mathcal{X}_{free}$.

Les contrôles disponibles sur le mouvement varient aussi d'un test à l'autre. Ainsi, on pourra parfois avancer selon certaines directions uniquement, tourner selon certains axes ou dans un certain sens (contrainte non holonome). Ces paramètres affectent aussi la dimension de l'espace des états $\mathcal{X}$.

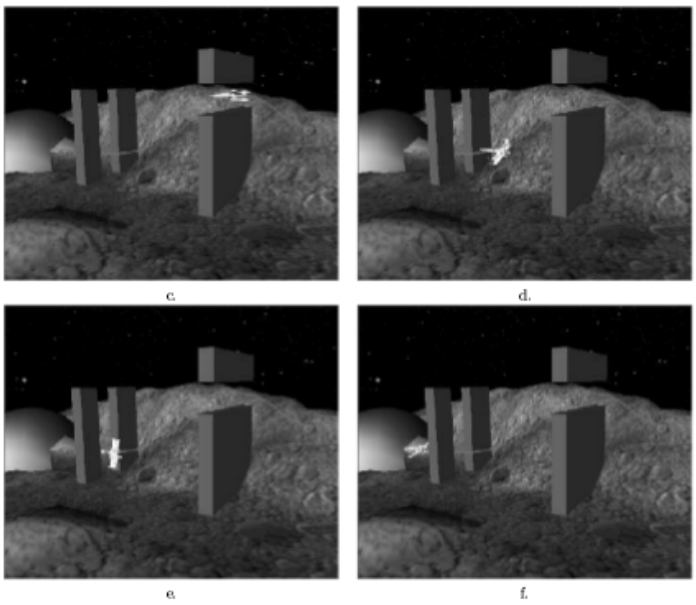

FIG. 3 – Un joli résultat

Dans les différents tests, la dimension de $\mathcal{X}$ variera ainsi de 4 à 12.

3.2.2 ...pour des résultats variables

On observe, selon les tests, une forte variation de la taille moyenne des RRT construits pour parvenir à une solution (de 400 à 23800 noeuds). De même, on observe une forte variation du temps de calcul (entre quelques secondes et quelques dizaines de minutes). Il n'y a pas eu d'analyse de faite pour mesurer la corrélation entre les résultats (temps de calcul, taille des RRT) et les différents paramètres.

3.3 Conclusion

Dans cet article est défini un problème de type nouveau, la planification de trajectoire. Ce problème, appartenant au domaine de la planification de mouvement en robotique, est pertinent au sens qu'il traite d'un sujet plus réaliste que la simple planification d'un chemin admissible pour un robot.

Etant donnée la complexité du problème, on est conduit à développer une approche probabiliste, spécifique et efficace. Pour cela, l'algorithme RRT, méthode générale pour découvrir un chemin entre deux points, a pu être appliqué sans que les adaptations nécessaires soient très importantes.

Comme la méthode est par nature incrémentale, il serait intéressant d'essayer d'adapter certaines procédures auxquelles on fait appel pour tenir compte de cet aspect et gagner ainsi en complexité (calcul des plus

proches voisins, détections des collisions, etc).

Enfin, l'article ne se donnait pas pour objectif de trouver une solution optimale. Cependant, puisqu'il permet de trouver une solution, il est naturel de s'interroger sur les manières d'améliorer cette dernière. D'abord, on peut envisager de lisser certaines courbes, couper là où c'est évident. Mais pour tenter d'améliorer une solution au point de changer peut-être sa classe d'homotopie, des perturbations plus importantes pourraient être introduites, avant de relancer partiellement l'algorithme.

Références

[1] Julien Guitton. Cell-rrt : décomposer l'environnement pour mieux planifier.

[2] Steven M LaValle and James J Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5) :378–400, 2001.

[3] John H Reif. Complexity of the mover's problem and generalizations extended abstract. In *Proceedings of the 20th Annual IEEE Conference on Foundations of Computer Science*, pages 421–427, 1979.