Rapport

Mehdi Mounsif

10 avril 2018

1 Contexte

Dans l'environnement du robot, mes derniers résultats ont montré qu'il était possible de piloter un robot grâce à un modèle entrainé sur des trajectoires simulées. Chaque élement des trajectoires comprend une description de l'état (angles des articulation et vecteur cible → effecteur) ainsi que les vitesses de rotation associées. Ces vitesses de rotations sont issues d'un calcul faisant intervenir l'inverse de la jacobienne de ce robot. En fonction du nombre de degrés de liberté du robot, un réseau de neurones d'une à deux couches permet d'obtenir une erreur faible sur le training set et un comportement satisfaisant en phase de test. On notera néanmoins que, bien que très proche, le comportement du modèle n'est pas exactement le même que celui de l'expert.

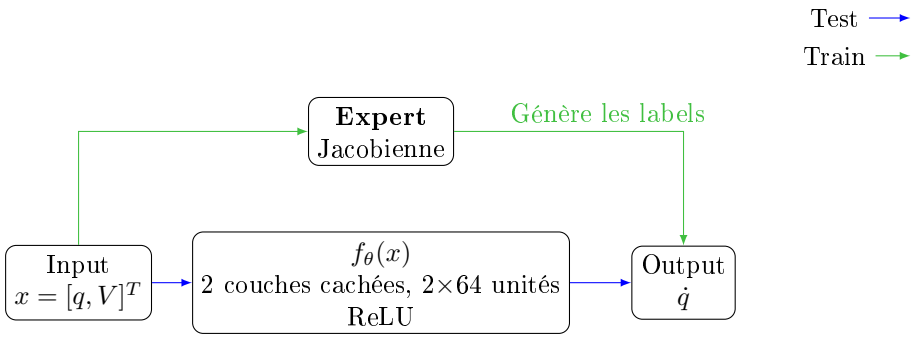

FIGURE 1 – Architecture et configuration. q est le vecteur contenant les valeurs articulaires du robot. V le vecteur entre la cible et l'effecteur

2 Réalisations

Disposant de ces premiers résultats, l'étape suivante consiste à sophistiquer le modèle. Comme noté dans le rapport de la semaine dernière, deux chemins sont possibles :

- Introduire les notions d'incertitudes et concepts associés
- Ajouter des contraintes à l'environnement

Les travaux qui sont décrits ci-après sont issus de réflexions sur la deuxième option.

2.1 Ajouts d'obstacles

Dans un premier temps, j'ai cherché à obtenir un comportement acceptable dans un milieu avec des obstacles. J'ai d'abord utilisé la méthode des champs de potentiels. Bien que non optimale (problèmes des minimas locaux, notamment), l'essence de ces expérimentations était essentiellement d'établir voir s'il était possible d'imiter le comportement de l'expert piloté par ces champs de potentiels.

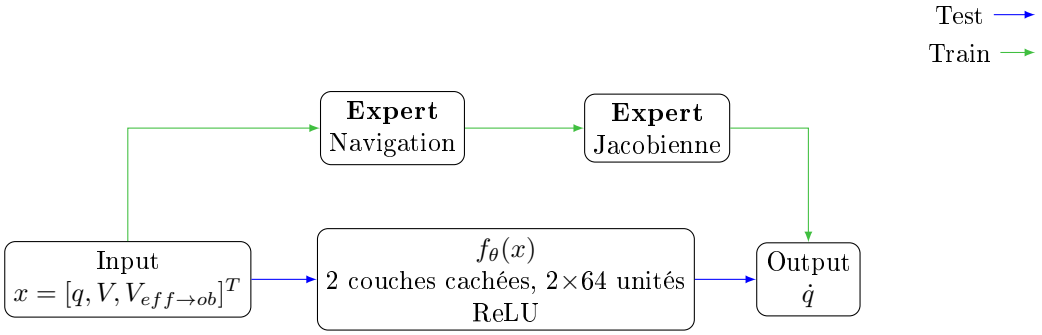

FIGURE 2 – Ajout d'un expert de navigation à l'architecture. Cet expert peut être n'importe quelle méthode de l'état de l'art (Champs de potentiels, $\mathbb{A}^\star$). L'état est augmenté de tous les vecteurs obstacles $\to$ effecteurs

Les résultats de l'incorporation de l'expert de navigation ont été mitigés. Bien que l'erreur sur le training set diminue rapidement, elle n'est pas aussi basse que lors du cas sans obstacles. En phase de test, malgré une dynamique cohérente, on est souvent face à des tressautements dans les zones proches des obstacles. On en conclut que :

- Les minimas locaux ont grandement contribué à l'erreur de training set, puisque l'expert lui-même est erratique dans ces zones
- L'apprentissage sur un dataset de qualité inférieure ne permettra pas d'obtenir des modèles efficaces

En réponse à ces observations, je me suis proposé d'utiliser l'algorithme $\mathbb{A}^\star$ pour piloter l'effecteur du robot. L'utilisation du $\mathbb{A}^\star$ dans cette configuration peut surprendre, puisque l'implémentation que j'ai réalisée calcule un chemin entre l'effecteur du robot et la cible. C'est-à-dire que nulle attention n'est apportée aux auto-collisions ou même aux collisions des autres articulations ou segments du robot avec les obstacles. En voici les justifications :

- L'idée est d'abord de valider si la planification est faisable aux moyen de ces réseaux
- En cas de succès, je sais qu'il est possible d'incorporer des pondérations au $\mathbb{A}^\star$. Il serait donc possible de configurer le calcul de manière à pénaliser les configurations inacceptables.
- D'un point de vue implémentation, c'est un algorithme que je connais et sa mise en place n'a pas été très longue
- Mais il est tout à fait envisageable de changer de méthode de navigation.

FIGURE 3 – Décroissance de l'erreur d'entrainement. 20 epochs, dataset de 1000 trajectoires, architecture du modèle présentée dans les figures précedentes

Pour observer les résultats, voir la vidéo ci-jointe. Si les obstacles sont fixes au cours de l'entrainement, le modèle permet d'obtenir des trajectoires intéressantes. En revanche, face à des obstacles mouvants (c'est-à-dire dont la position est modifié régulièrement), le modèle ne démontre que vaguement des stratégies d'évitement. A mon avis, ceci vient du fait que la représentation de l'état est trop locale, et que la planification requiert une vision globale. Ces considérations sont l'object de la dernière partie.

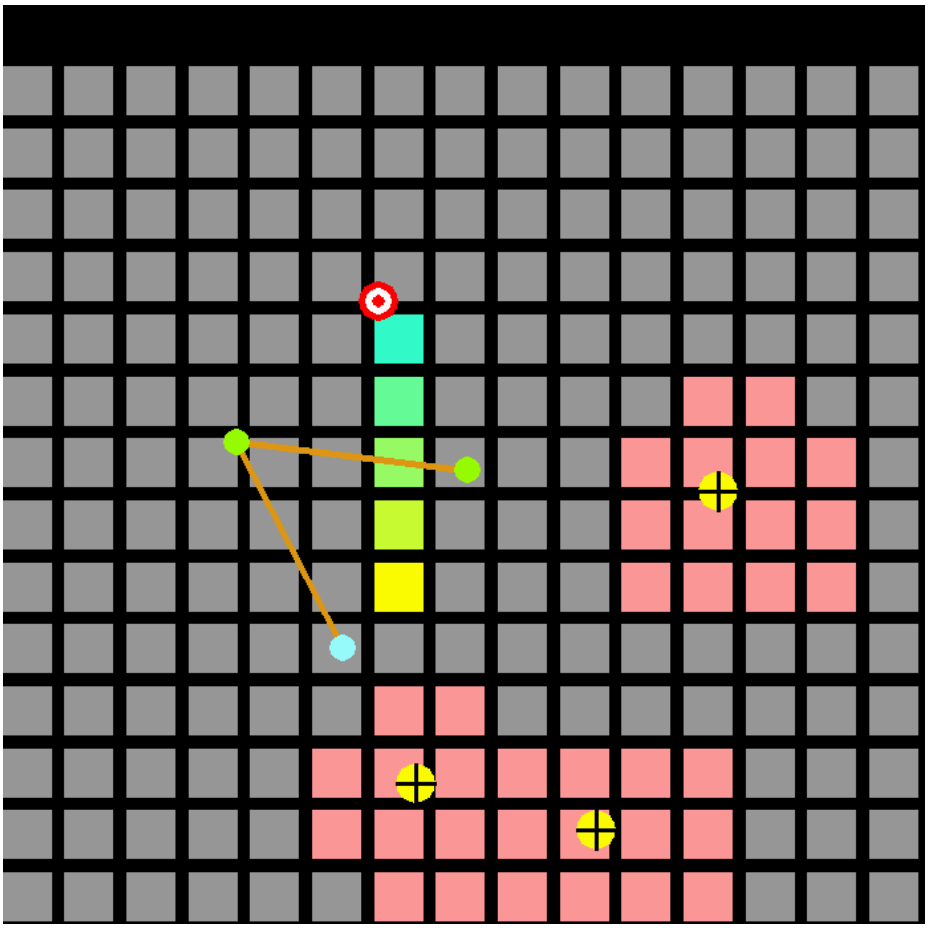

FIGURE 4 – Une configuration avec chemin calculé. En jaune, le point de départ et en bleu, l'objectif. Les zones en rouge sont des marges entourant les obstacles

3 Conclusion

Les travaux des derniers jours ont montré qu'il était possible d'imiter un expert (voir un ensemble d'experts travaillant de concert) grâce à un réseaux de neurones. Deux méthodes différentes ont été mises en oeuvre pour la navigation en milieu encombré. Si l'aspect réactif est très bien approximé, il semblerait que la planification avec un modèle feedforward simple n'est efficace que si l'environnement est figé. Pour un environnement plus complexe dans lequel les obstacles seraient déplacés régulièrement d'autres méthodes doivent être employées. Comme leur implémentation peut demander légèrement plus de temps, je voulais soumettre la pertinence de ces résultats à votre appréciation avant de continuer. Les méthodes applicables sont :

- CNN (réseaux de convolution). L'observation de l'état sous forme d'image permettrait d'avoir une vue globale et donc de planifier un chemin en satisfaisant les contraintes imposées.
- RNN (réseaux récurrents). L'apprentissage des trajectoires sous forme de séquences permettrait également de se détacher des ambiguités grâce à la mémoire inhérente à ces méthodes.