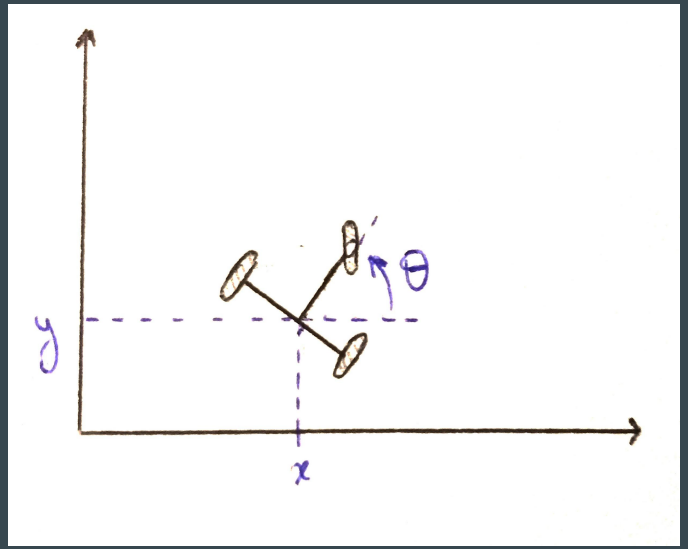
SYMOU
RRT pour un (1, 1)
•••

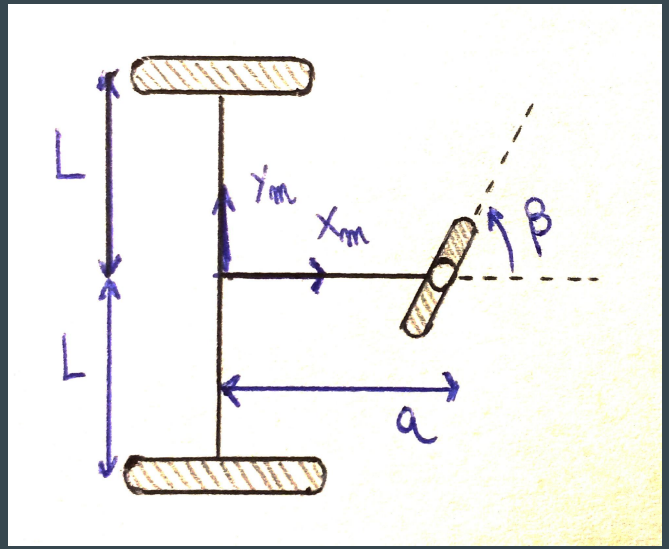
Alexis Dupuis et Corentin Chauvin-Hameau

https://github.com/CorentinChauvin/RRT

Introduction du problème

$$q = \begin{pmatrix} x \\ y \\ \theta \\ \beta \end{pmatrix} \qquad u = \begin{pmatrix} u_m \\ u_s \end{pmatrix} = \begin{pmatrix} {}^{m}\dot{x} \\ \dot{\beta} \end{pmatrix}$$





Première approche: contrôle du robot en (v, bêta point)

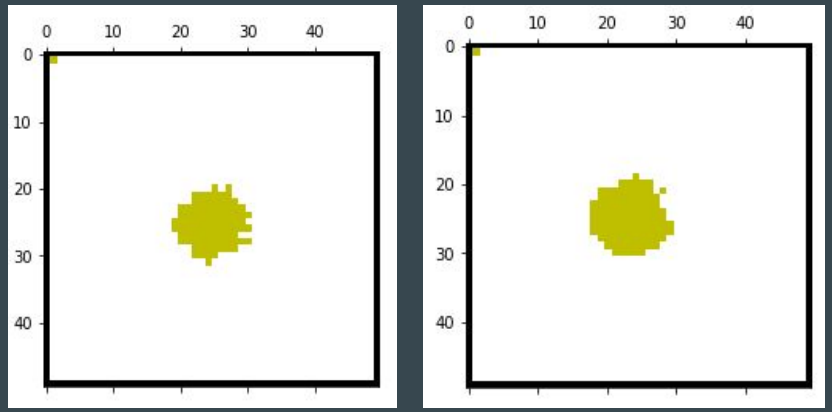
Principe de l'algorithme initial :

- Initialiser deux arbres T_start et T_goal avec les points de départ et d'arrivée
- En alternant les arbres à chaque tirage :
  - Tirer un nœud de l'arbre **aléatoirement**
  - Tirer une commande (v, bêta point) **aléatoire** (dans un certain intervalle)
  - Réaliser la commande pendant une période de temps fixe : on obtient q_new
  - Si pendant la commande, il n'y a pas eu de collision :
    - Ajouter q_rand à l'arbre courant
    - Essayer de merge q_new à l'autre arbre

Contrôle du robot en (v, bêta point) : problèmes rencontrés

L'exploration de la map échoue :

- Agglutination des nœuds
- Des commandes qui font faire des tours au robot sur lui-même (pas beaucoup de trajectoire globalement droite), ou ne le font pas tourner



500 et 5000 tirages

Cercle vicieux : plus les nœuds s'agglutinent, plus on a de chances de créer de nouveaux nœuds autour

Contrôle du robot en (v, bêta point) : problèmes rencontrés

La difficultée rencontrée peut alors se résumer en 3 points :

1- tirer des qrand est crucial, c'est ce qui assure le phénomène d'exploration de la carte.

2- réussir à rendre les qrand « attracteurs » est un vrai défi pour le robot (1,1) et réussir à générer des commandes « réalistes » + garder une trajectoire C^1 + s'approcher réellement des qrand est loin d'être un problème simple

3- le calcul d'un q_near est essentiel aussi, il faut que l'extension de l'arbre se fasse prioritairement via sa périphérie

« L'algorithme RRT pour le robot (1,1) convergera rapidement, ou il ne convergera pas du tout ! »

-Nous, Mars 2018

Contrôle du robot en (v, bêta point) : amélioration

Amélioration : implémenter le concept du plus proche voisin

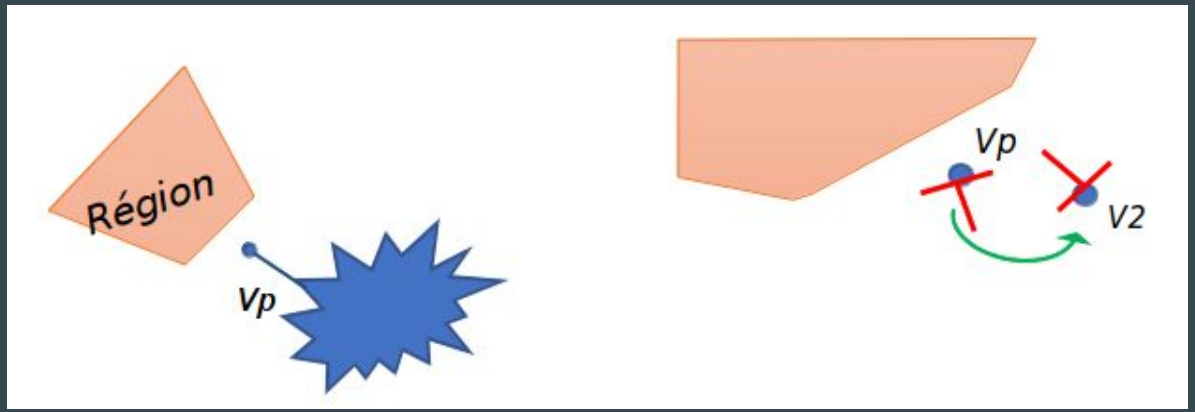
> Plutôt que de tirer q_near aléatoirement, tirer une configuration aléatoire q_rand, puis trouver q_near le plus proche dans l'arbre.

> Qu'est-ce qu'un plus proche voisin ? Choix d'une métrique selon (x,y) ou selon(x, y, thêta, bêta point) ?

Contrôle du robot en (v, bêta point)

Choix de la métrique :

- Distance cartésienne sur (x, y, thêta, bêta point) : efficace pour explorer l'espace (x, y) ?
- Distance cartésienne sur (x, y) : risque de ne pas favoriser les manœuvres qui sont pourtant nécessaires pour changer de direction.



4 approches de résolution :

1- Méthodes d'amélioration du ratio "surface explorée/nombre de noeuds"

Méthode du discriminant

Méthode du plus proche voisin retour

2- Méthodes de contrôle du robot :

Commande en (v, bêta point): amélioration des paramètres

Commande par "sugar-tracking"

4 approches de résolution :

1- Méthodes d'amélioration du ratio "surface explorée/nombre de noeuds"

Méthode du discriminant

Méthode du plus proche voisin retour

2- Méthodes de contrôle du robot :

Commande en (v, bêta point): amélioration des paramètres

Commande par "sugar-tracking"

Méthodes dites “d’amélioration des ratios” - 1/2

Stratégie sous-jacente:

- Lutter contre le problème d’agglutination en ne conservant que des noeuds qui font vraiment grandir l’arbre dans de nouvelles directions.

Méthodes dites “d’amélioration des ratios” - 2/2

surface_couverte = nbr_tirages /(nbr tirages / surface couverte)

temps_calcul = Cte * (nbr_noeuds)^2

= Cte * (nbr nœuds / nbr tirages)^2 * nbr_tirages^2

Donc : **surface/temps** =

$$\frac{1}{(Cte^* (\text{nbr tirages} / \text{surface couverte}) * (\text{nbr nœuds} / \text{nbr tirages})^2 * \text{nbr\_tirages})}$$

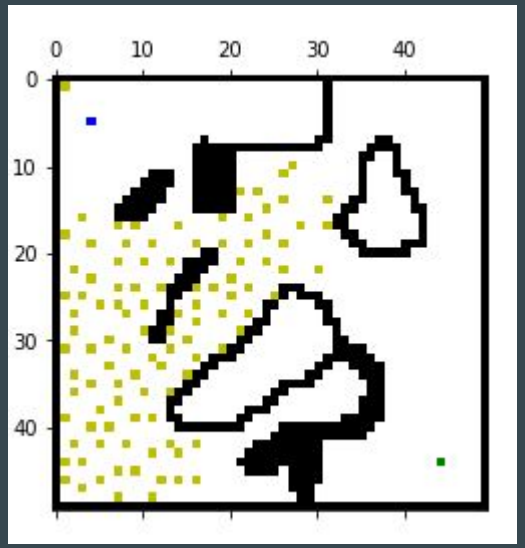
Paramètre d’ajustement

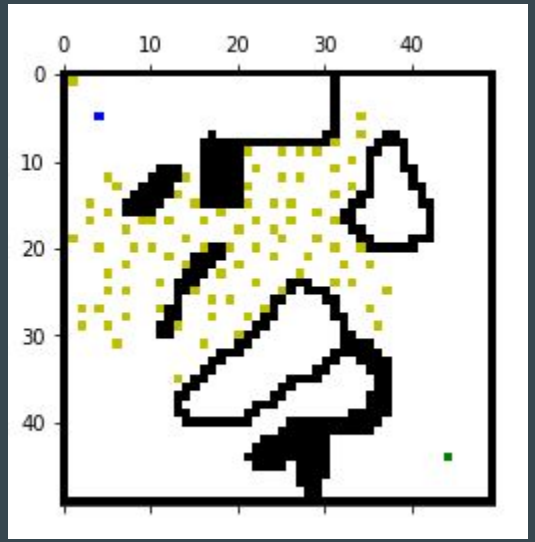
Méthode du “discriminant”

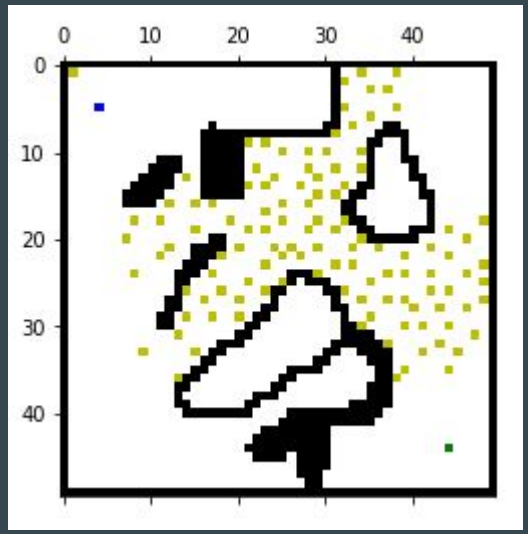
Amélioration : ne pas ajouter le nouveau noeud dans l’arbre si d’autres déjà présents en sont trop proches (à une “distance” inférieure à un discriminant).

> Permet d’éviter l'agglutination

20000 tirages !







4 approches de résolution :

1- Méthodes d'amélioration du ratio "surface explorée/nombre de noeuds"

Méthode du discriminant

Méthode du plus proche voisin retour

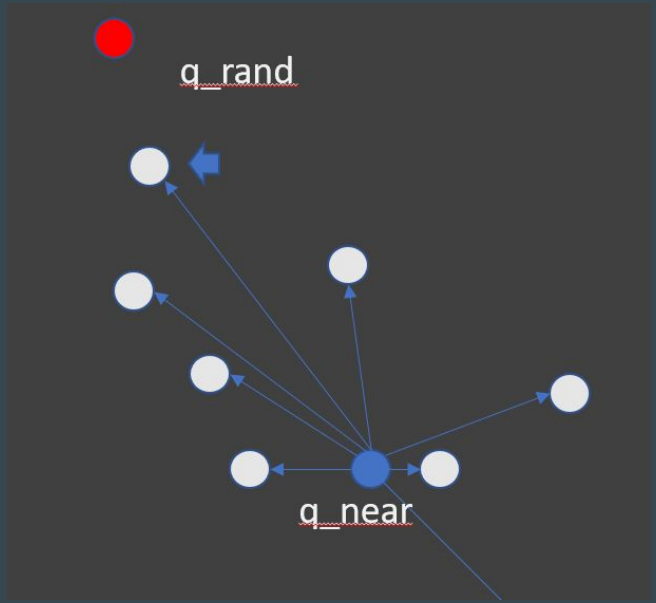
2- Méthodes de contrôle du robot :

Commande en (v, bêta point): amélioration des paramètres

Commande par "sugar-tracking"

Méthode du “plus proche voisin retour”

Amélioration : à chaque “tentative”, tirer plusieurs commandes et garder celle qui rapproche le plus q_near de q_rand.



4 approches de résolution :

1- Méthodes d'amélioration du ratio "surface explorée/nombre de noeuds"

Méthode du discriminant

Méthode du plus proche voisin retour

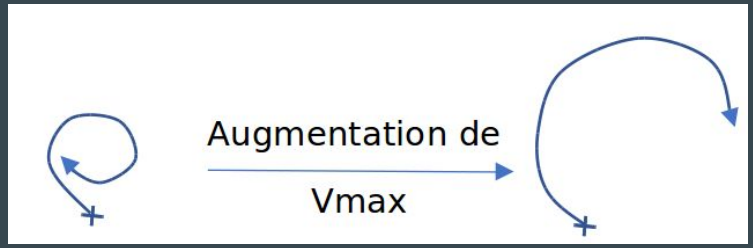
2- Méthodes de contrôle du robot :

Commande en (v, bêta point): amélioration des paramètres

Commande par "sugar-tracking"

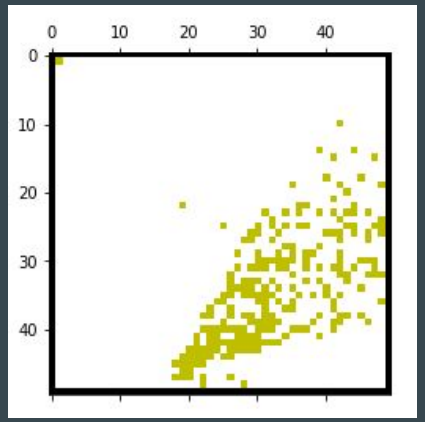
Paramètres importants: Vmax et BETA_POINTmax
Espoir: augmenter la vitesse pour explorer plus efficacement

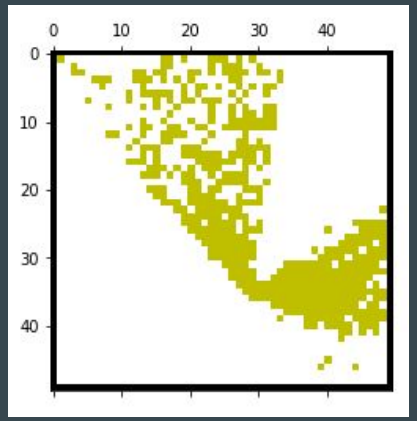
Contrôle du robot en (v, bêta point)

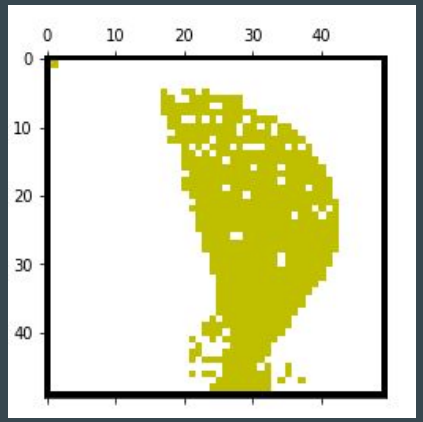


De nombreux problèmes :

- Agglutination des nœuds
- Des commandes qui font faire des tours au robot sur lui-même (pas beaucoup de trajectoires globalement droites), ou ne le font pas tourner





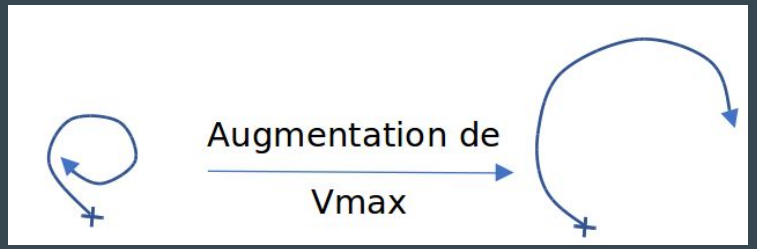


Bêta_point_max faible et Vmax fort

500, 2000, 5000 tirages

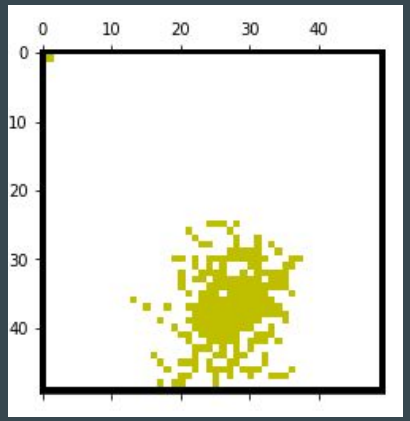
> mauvaise couverture

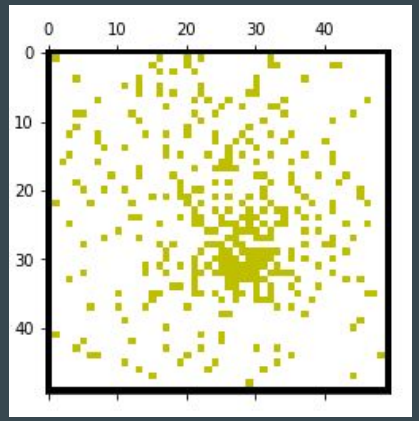
Contrôle du robot en (v, bêta point)

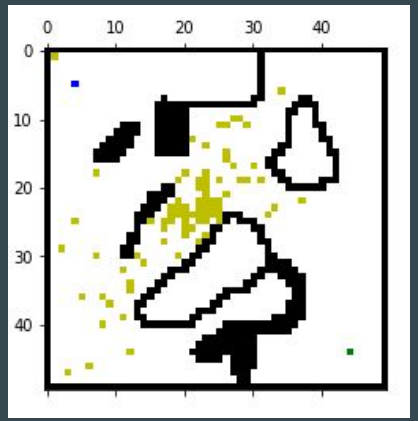


De nombreux problèmes :

- Agglutination des nœuds
- Des commandes qui font faire des tours au robot sur lui-même (pas beaucoup de trajectoires globalement droites), ou ne le font pas tourner







Bêta_point_max fort et Vmax faible puis fort

> Pas efficace en présence d'obstacle

4 approches de résolution :

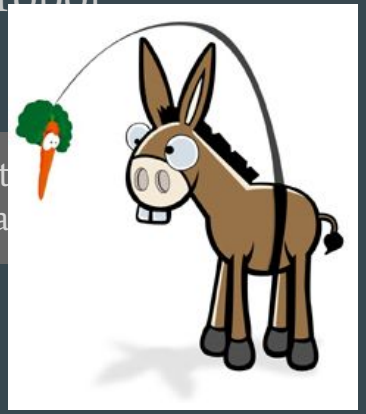
1- Méthodes d'amélioration du ratio "surface explorée/nombre de noeuds"

Méthode du discriminant

Méthode du plus proche voisin retour

2- Méthodes de contrôle du robot :

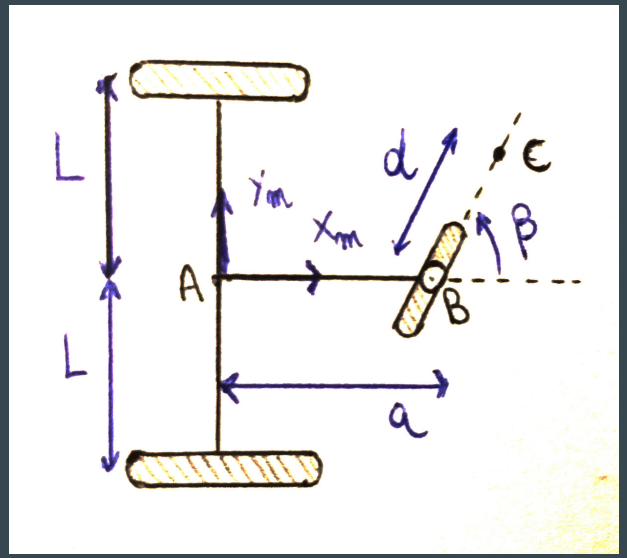
Commande en (v, bêt
amélioration des para

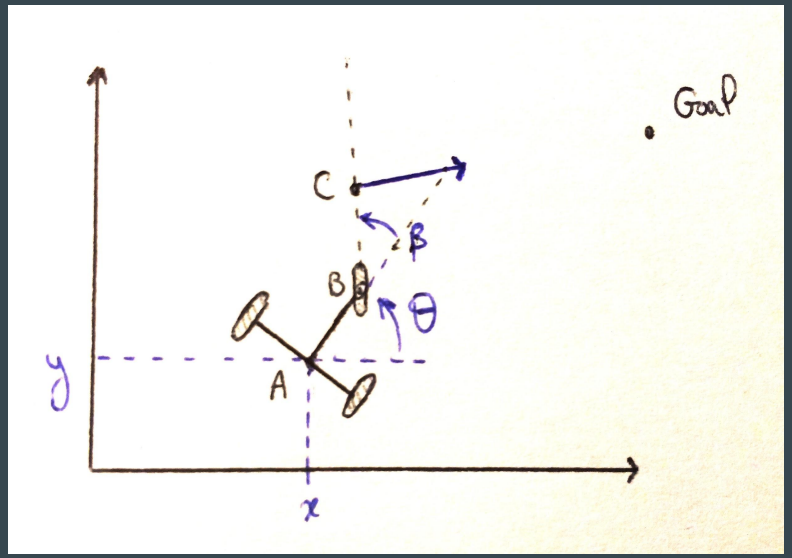


Commande par "sugar-tracking"

Seconde approche: “sugar tracking”

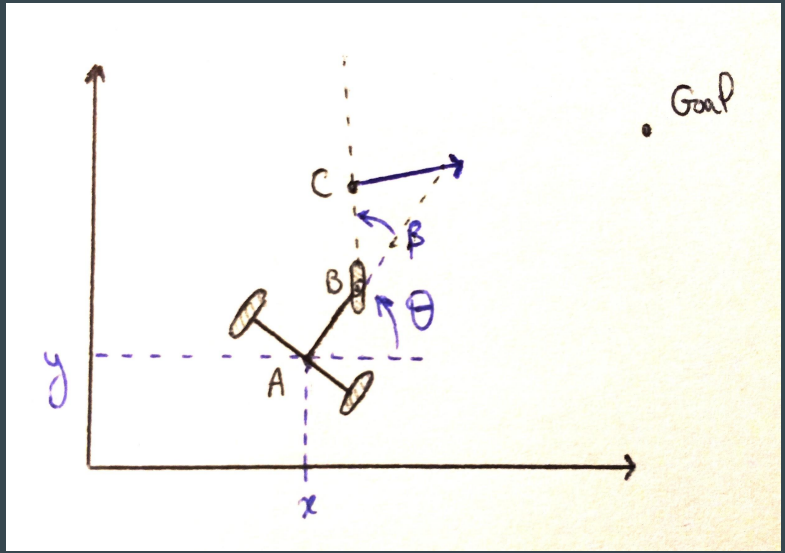
But : générer une commande pour amener un point en avant du robot vers q_rand





Seconde approche: “sugar tracking”

But : générer une commande pour amener un point en avant du robot vers q_rand



$$\vec{v_{C/0}} = \begin{pmatrix} \dot{x} - a\,\dot{\theta}\sin\theta - d\,(\dot{\theta}+\dot{\beta})\sin(\theta+\beta) \\ \dot{y} + a\,\dot{\theta}\cos\theta + d\,(\dot{\theta}+\dot{\beta})\cos(\theta+\beta) \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \end{pmatrix}$$

D'où on déduit $\dot{\theta}$ et $\dot{\beta}$

Seconde approche: “sugar tracking”

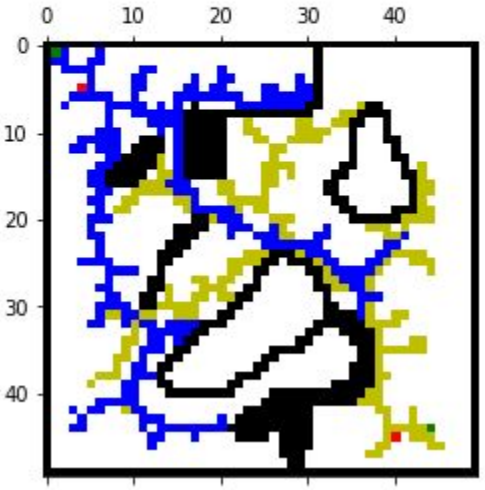
Résultats :

- L’arbre s’étend vraiment vers les q_rand tirés, et ce avec des Vmax raisonnables
- Beaucoup moins de nœuds tirés
- Des résultats toujours un peu aléatoires, surtout pour sortir d’obstacles étroits

Seconde approche: “sugar tracking”

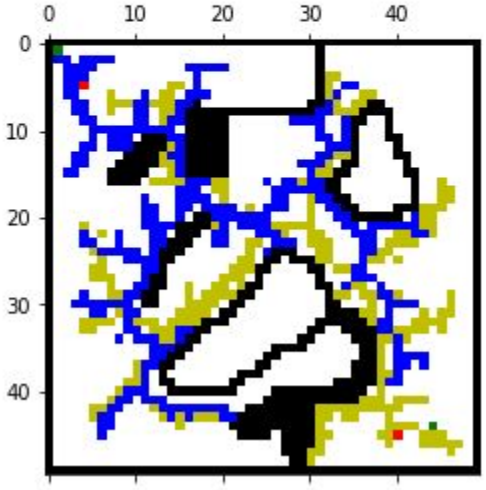
Nombre de merging = 459

Temps d'execution = 5.515349



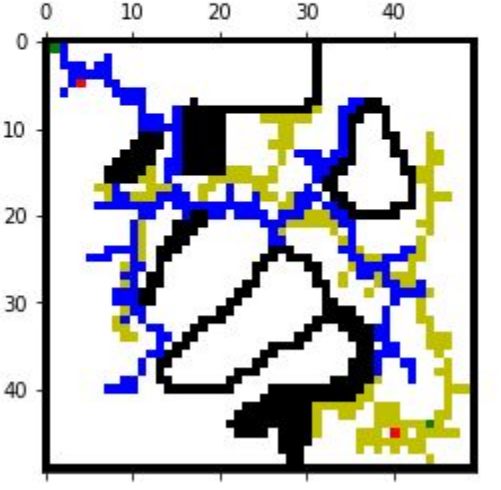
Nombre de merging = 1183

Temps d'execution = 7.554845



Nombre de merging = 403

Temps d'execution = 3.805297



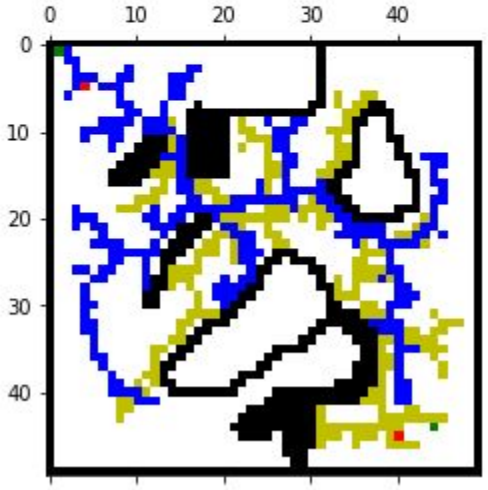
Pour 4000 tentatives

Seconde approche: “sugar tracking”

Nombre de merging = 304

Temps d'execution = 5.178273

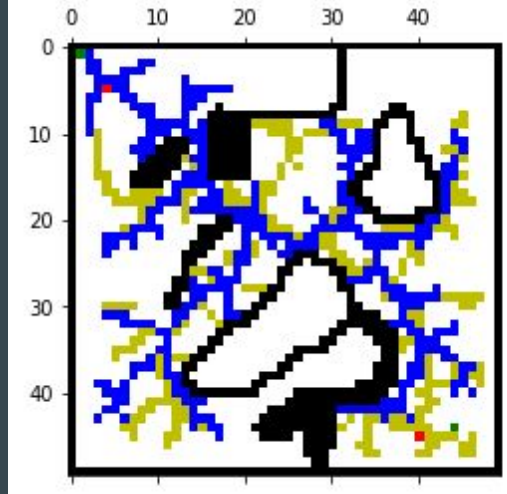
Nbr refus : 897



Nombre de merging = 694

Temps d'execution = 6.964396

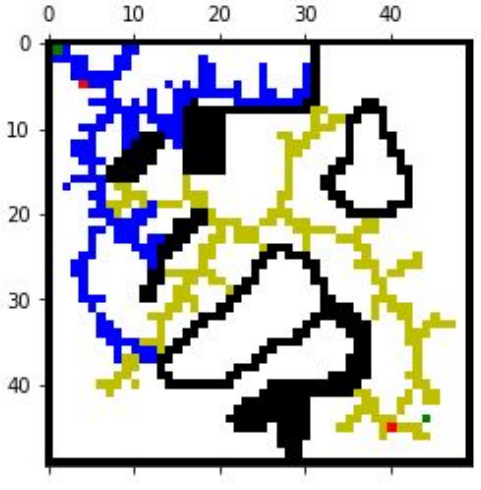
Nbr refus : 1033



Nombre de merging = 35

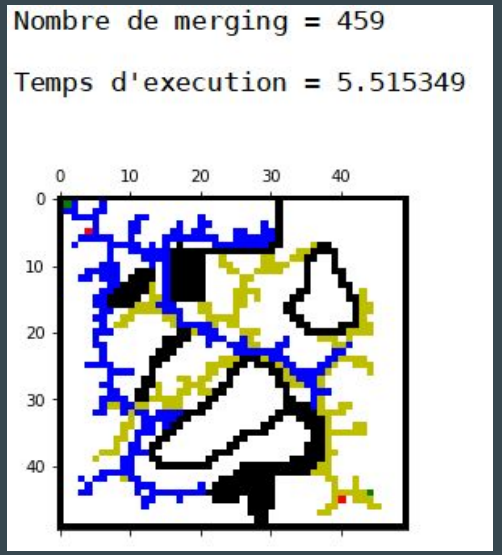
Temps d'execution = 3.471565

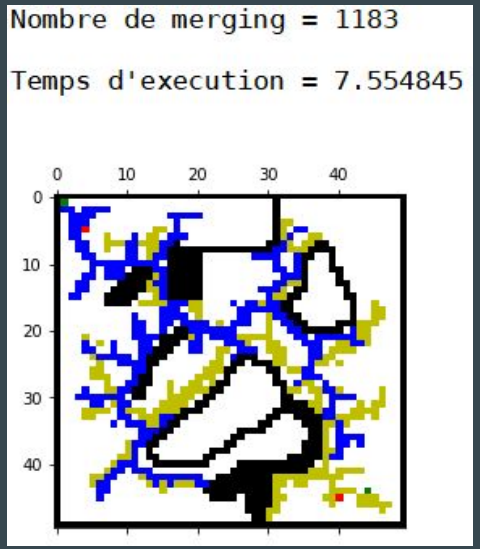
Nbr refus : 880

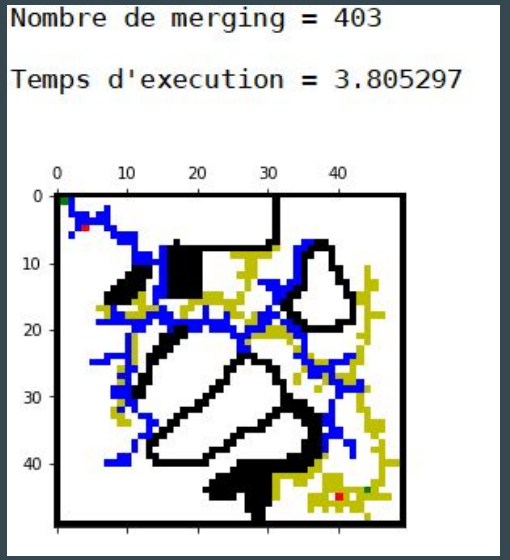


Avec méthode du discriminant : 24% plus rapide en moyenne

Comparaison des temps de calcul avec et sans méthode des ratios







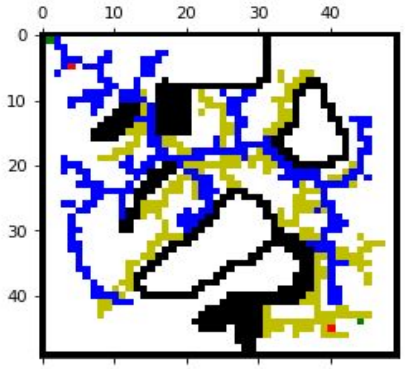
Sans

Comparaison des temps de calcul avec et sans méthode des ratios

Nombre de merging = 304

Temps d'execution = 5.178273

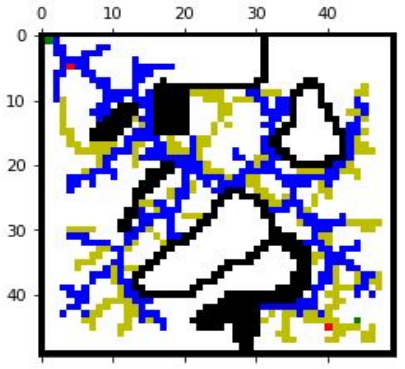
Nbr refus : 897



Nombre de merging = 694

Temps d'execution = 6.964396

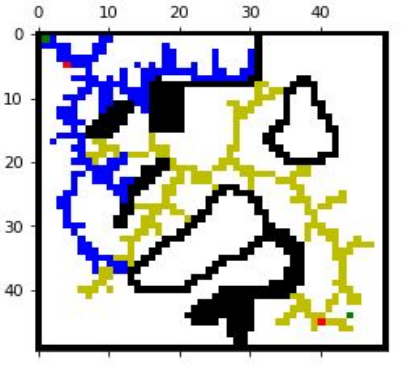
Nbr refus : 1033



Nombre de merging = 35

Temps d'execution = 3.471565

Nbr refus : 880



Ave

c

Comparaison des temps de calcul avec et sans méthode des ratios

En chiffres : la moyenne, sur 20 tests de 2000 tentatives chacun

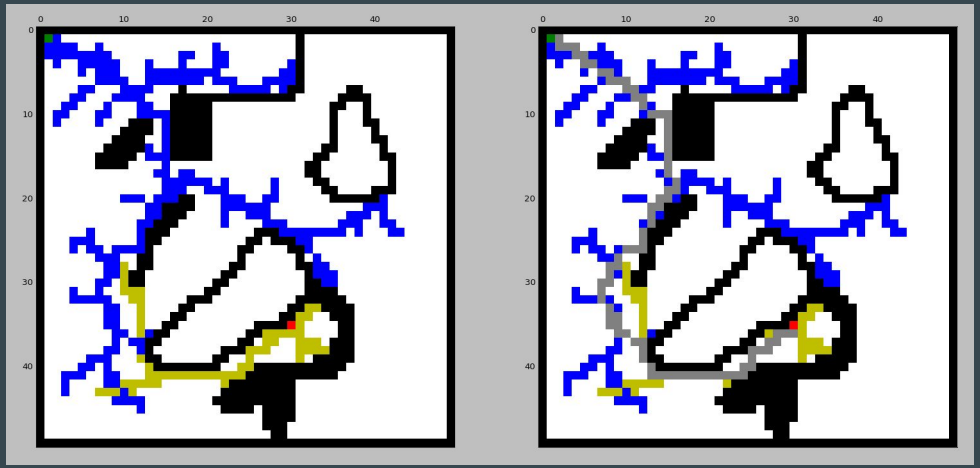
Temps d'exécution moyen:

Sans : 1,48 secondes

Avec 1,12 secondes

Soit un gain de 24%

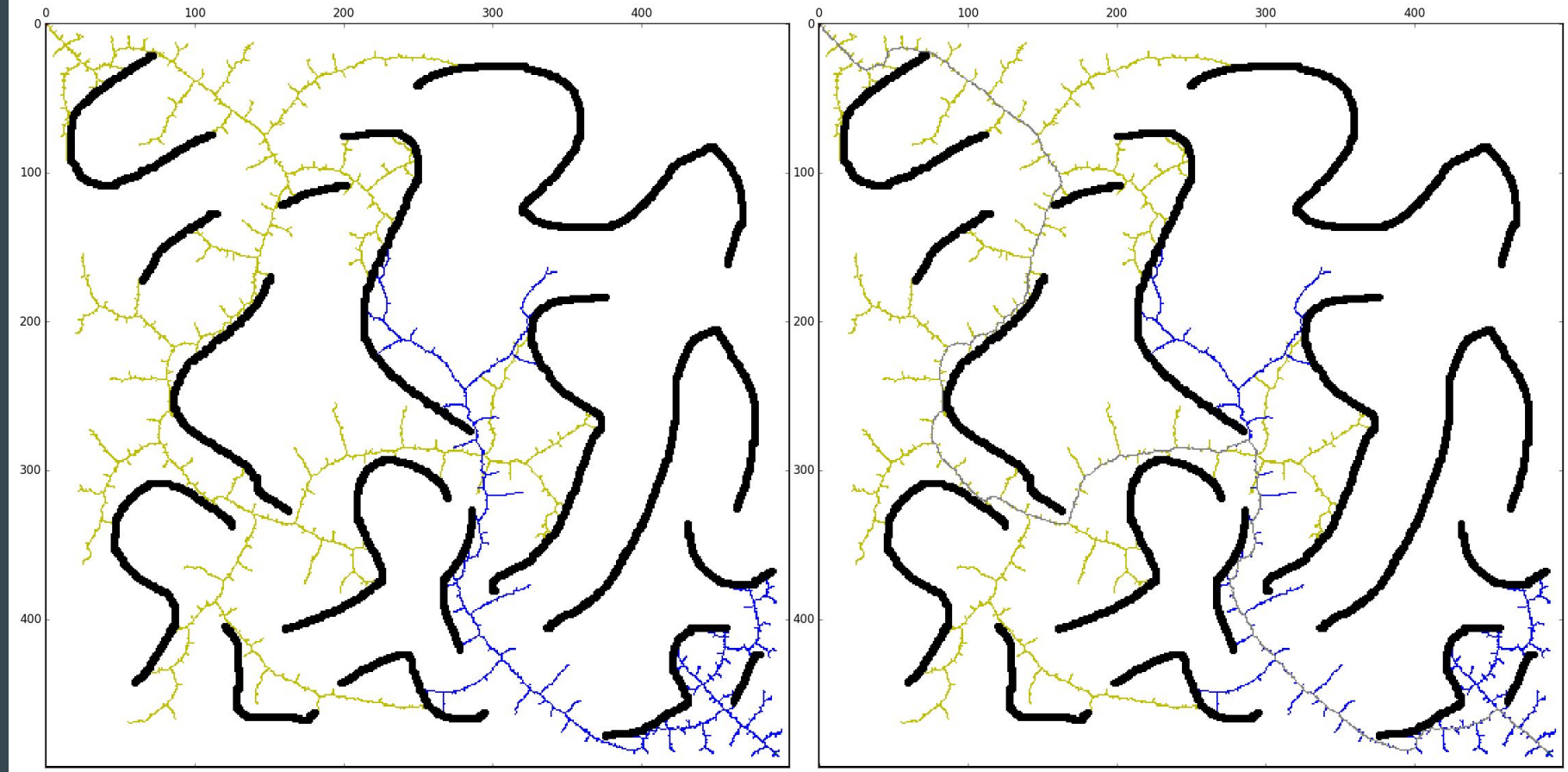
En bonus, le A*



En bonus, le A*

```
106388
106389
Number of merges : 20
Discriminant refusals : 521
Entering a_star...
Success of a_star
RRT execution time : 539.04186
A* execution time : 377.011572
```

Pour une image de 500x500


0
100
200
300
400
0
100
200
300
400
0
100
200
300
400
0
100
200
300
400