



UNIVERSITÉ
LAVAL

GLO-4001/7021

INTRODUCTION À LA ROBOTIQUE MOBILE

Cartes +
Planification

Cartes

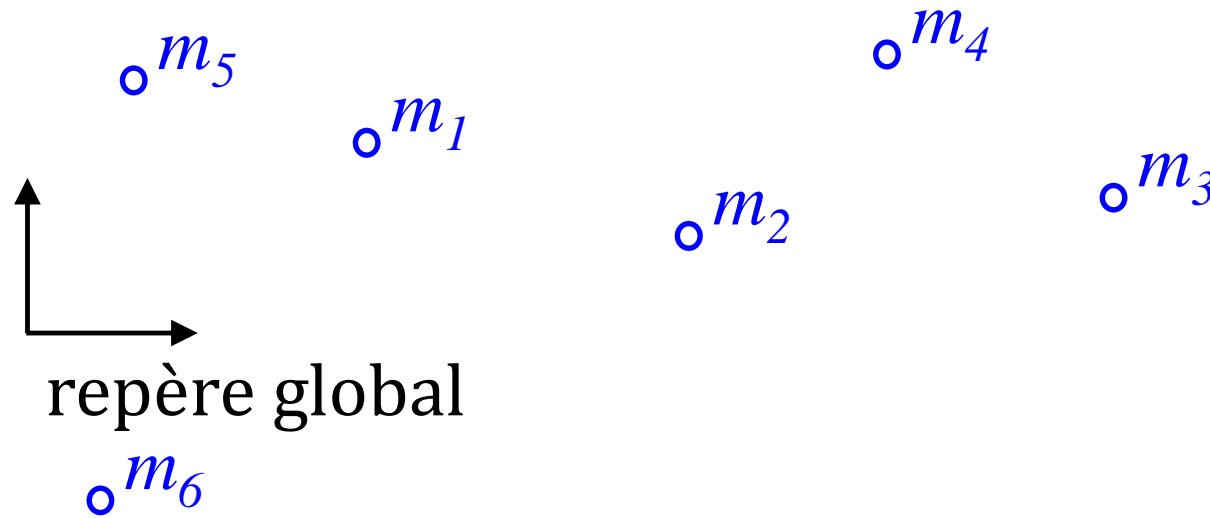
Représentation du monde : carte

- Notre robot accumule de l'information sur le monde : **connaissance**
- Une carte permettra de ^(planifier) raisonner sur le monde
- Plusieurs types possibles de représentation de cette **connaissance**
- Problème fondamental en intelligence artificielle
 - représentation des connaissances
- Compromis entre taille stockage, facilité d'usage, objectifs à accomplir

Types de cartes en robotique mobile

- **Métrique**

- décrire un environnement avec un système de coordonnées absolu
- notion de distance calculable entre tous les points



point de repère $m_i = [x \ y \ signature]^T$

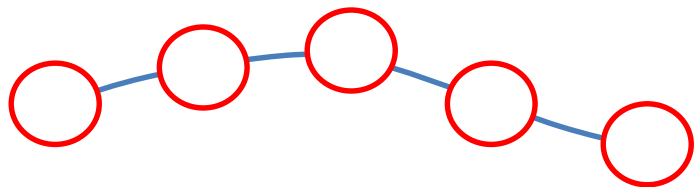
Types de cartes en robotique mobile

- **Métrique**

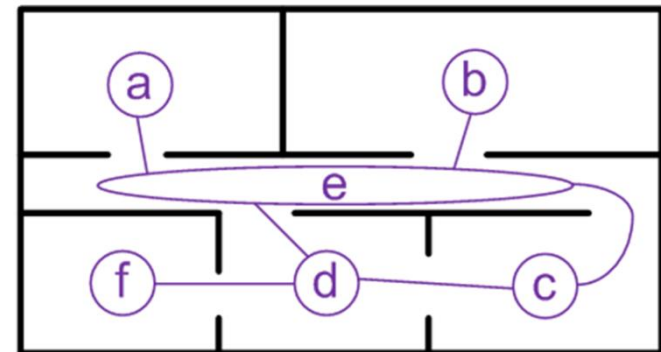
- décrire un environnement avec un système de coordonnées absolu
- notion de distance calculable entre tous les points

- **Topologique**

- graphe (nœud=endroit, arêtes=connexion)
- évacue beaucoup d'information (~~distance~~)
- représentation des relations locales entre les endroits



ou



Types de cartes en robotique mobile

- **Métrique**

- décrire un environnement avec un système de coordonnées absolu
- notion de distance calculable entre tous les points

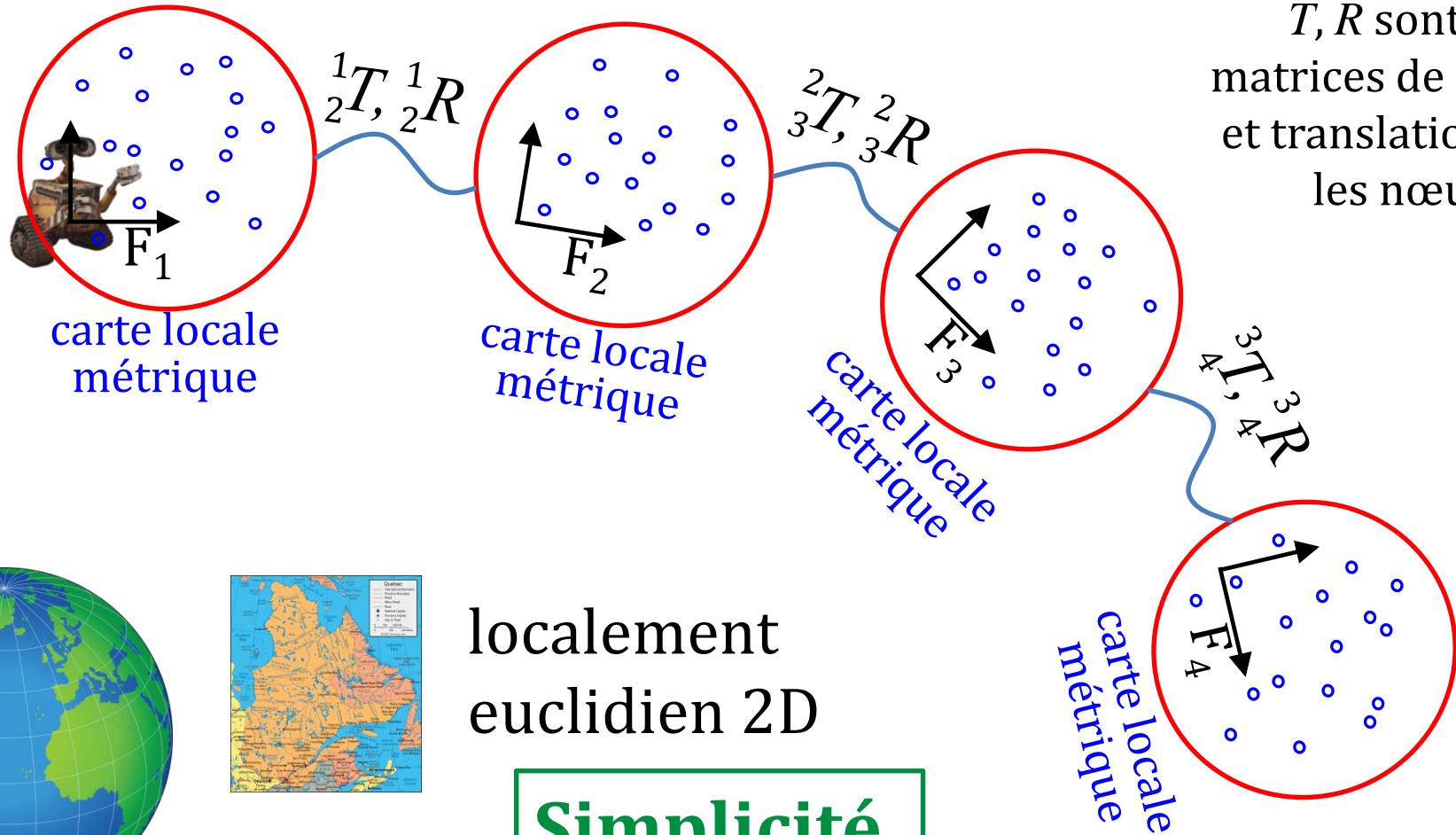
- **Topologique**

- graphe (nœud=endroit, arêtes=connexion)
- évacue beaucoup d'information (~~distance~~)
- représentation des relations locales entre les endroits

- **Topométrique**

- graphe, carte métrique à chaque nœud
- arêtes peuvent contenir une notion de distance + orientation relatives

Exemple carte topométrique



T, R sont des matrices de rotation et translation entre les nœuds



(manifold dans 3D)



localement euclidien 2D

Simplicité

Local Metrical and Global Topological Maps in the Hybrid Spatial Semantic Hierarchy, B. Kuipers, et al., *ICRA* 2004. 11

Caractéristiques des cartes (2)

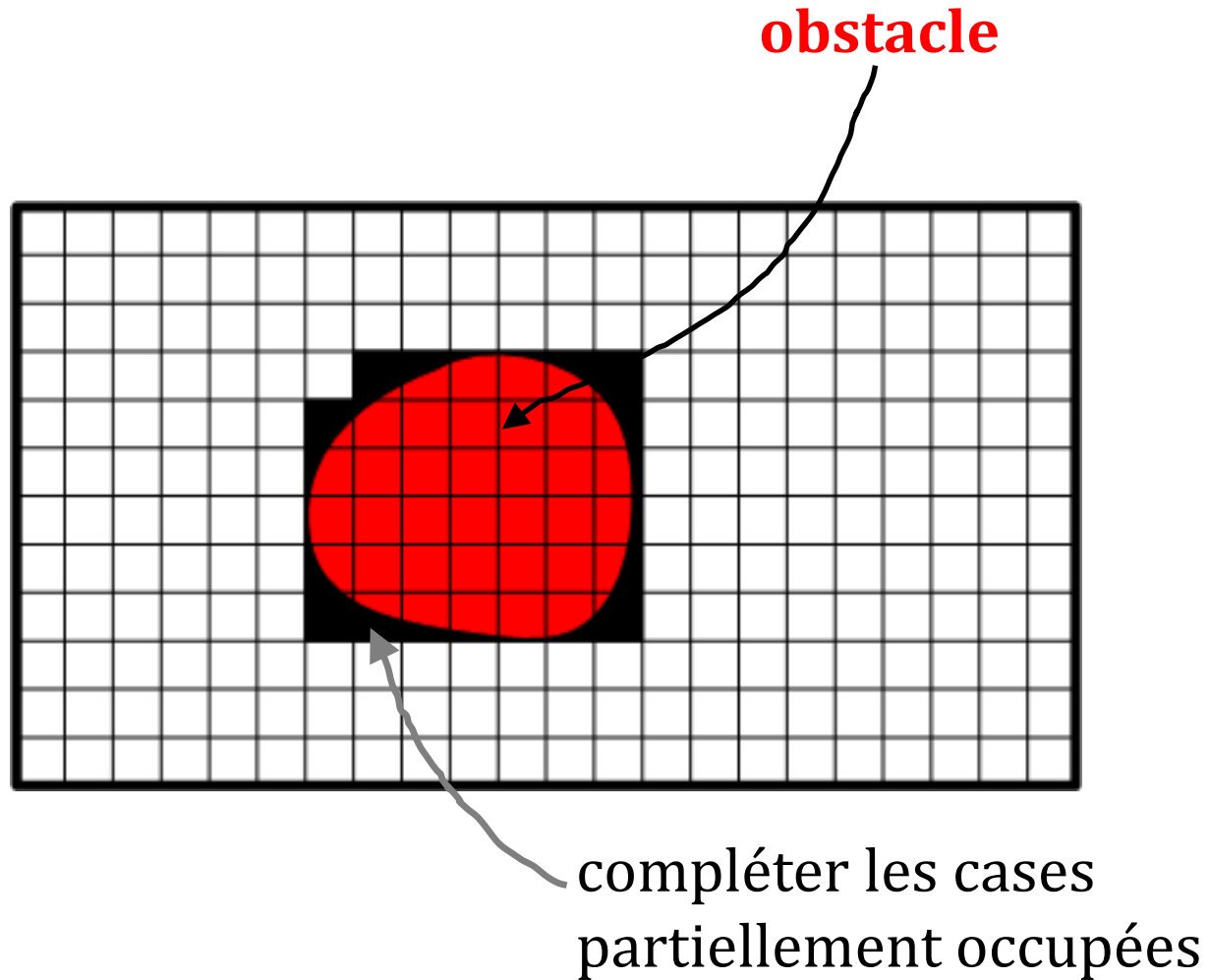
- **Continue**

- positions sont des réels
 - une lampe à $(-1.45534, +5.899485)$
 - impact laser sur mur à $(+2.323, +1.234)$

- **Discrète**

- discrétiser le monde en case de taille fixe
- valeur binaire : 0= libre 1= obstacle
- valeur continue
 - $[0, 1]$ (probabilité obstacle)
 - $[0, \infty]$ (cote obstacle)
 - $[-\infty, \infty]$ $\log(\text{cote})$
- *e.g.* grilles d'occupation (*occupancy grids*)

Cartes : grillage uniforme

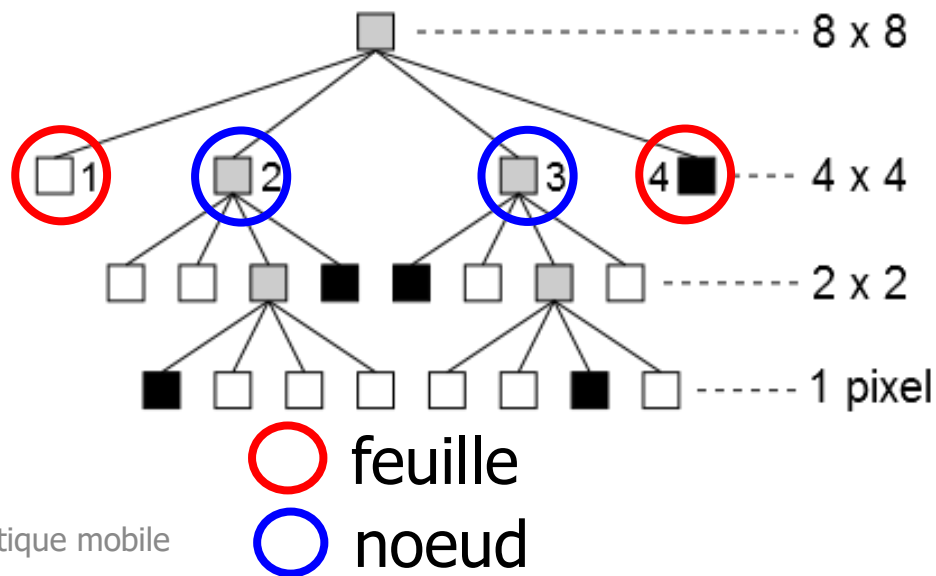


Cartes : grille uniforme

- Décide de l'intervalle échantillonnage, *e.g.* 1 *cm*
 - 2D : pièce de 10 *m* x 10 *m*
 - 1000 x 1000 = 1 000 000 « pixels »
 - 3D : pièce de 10 *m* x 10 *m* x 3 *m*
 - 1000 x 1000 x 300 = 300 000 000 « voxels »
- (-) Espace de stockage
- (-) Erreur de discrétisation
- (+) Temps d'accès rapide

Cartes : échantillonnage Quadtree

- Structure de données en arbre
- Compression de l'information
- Par contre, structure de l'arbre change beaucoup pour petits changements d'image



Quadtree

feuille

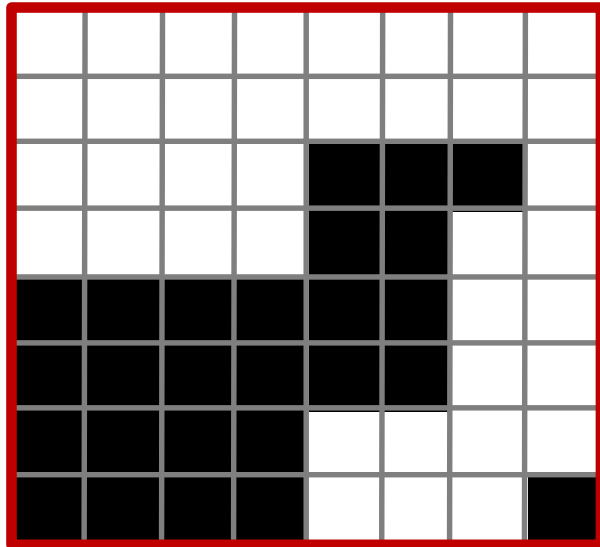
○ vide

nœud

● en partie plein

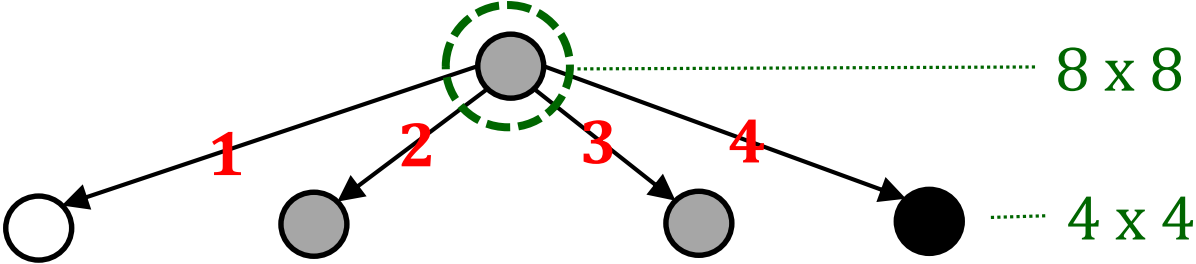
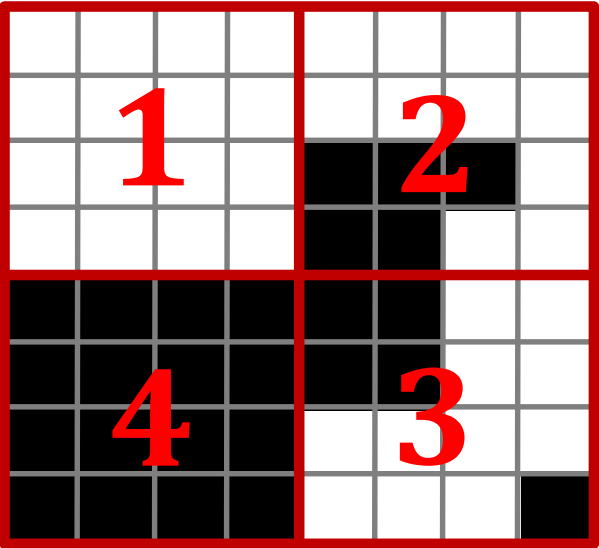
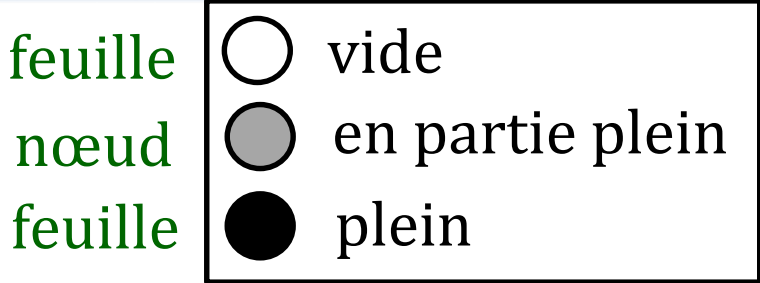
feuille

● plein



..... 8 x 8

Quadtree



Quadtree

feuille

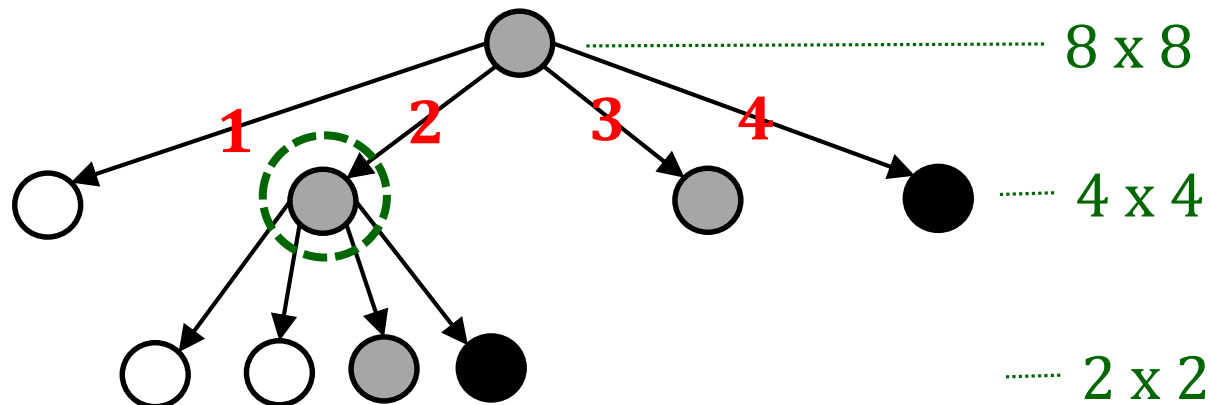
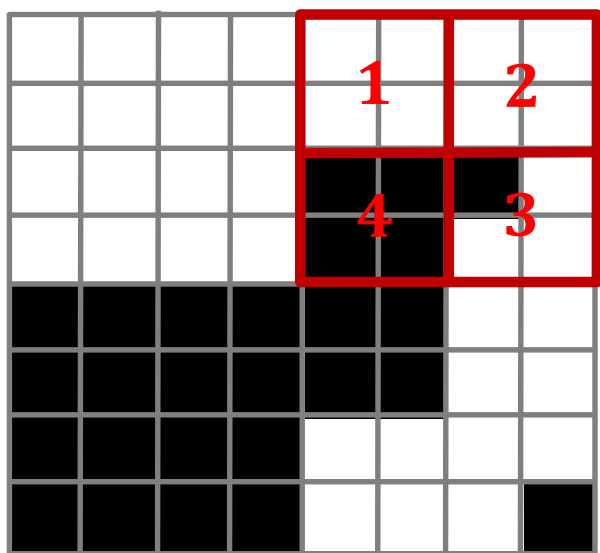
○ vide

nœud

● en partie plein

feuille

● plein



Quadtree

feuille

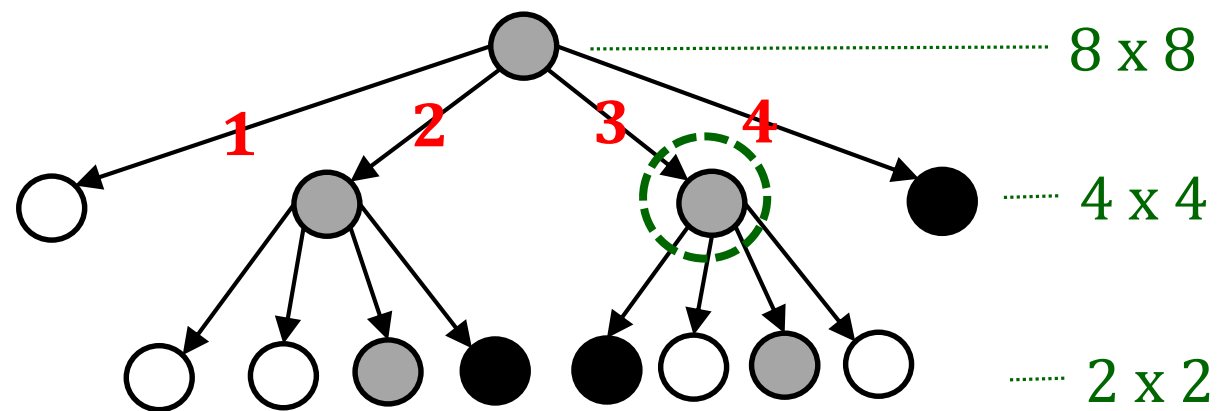
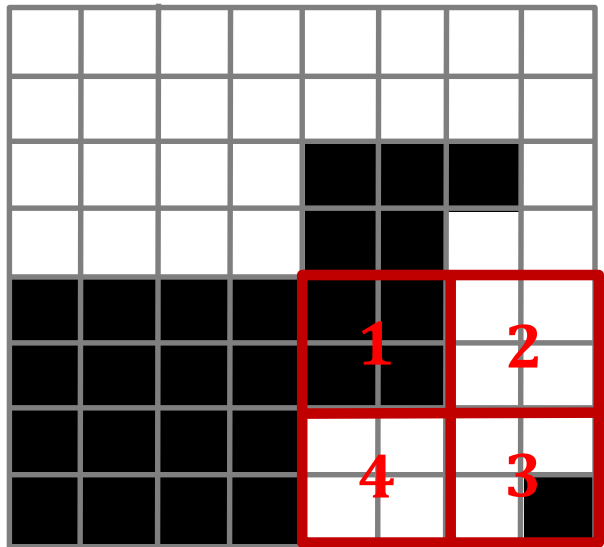
○ vide

nœud

● en partie plein

feuille

● plein



Quadtree

feuille

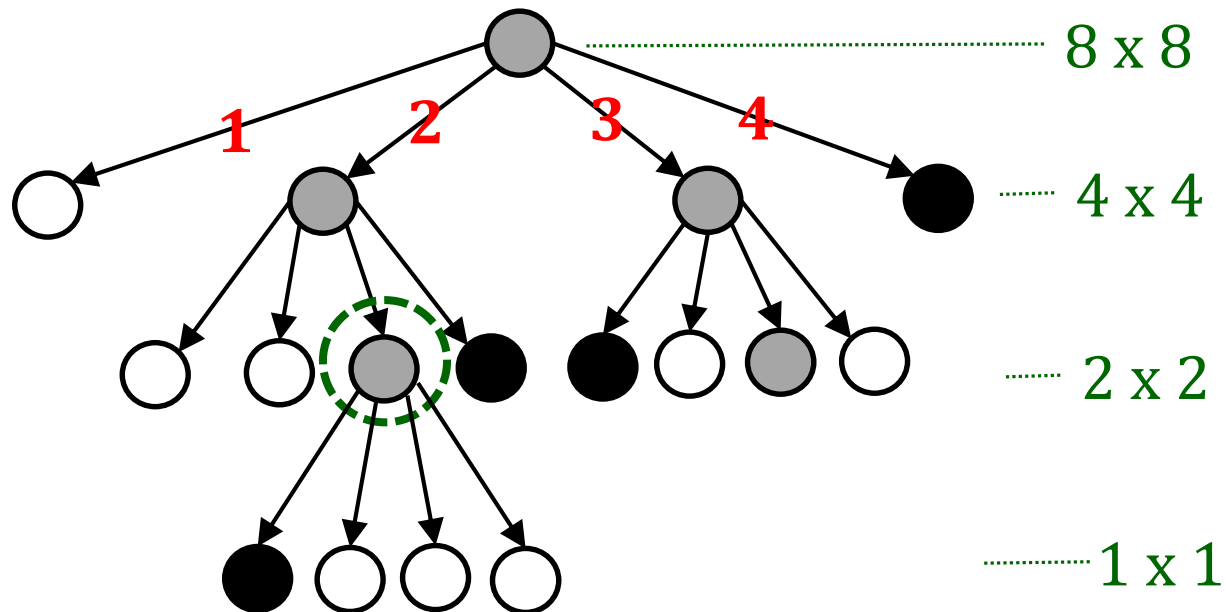
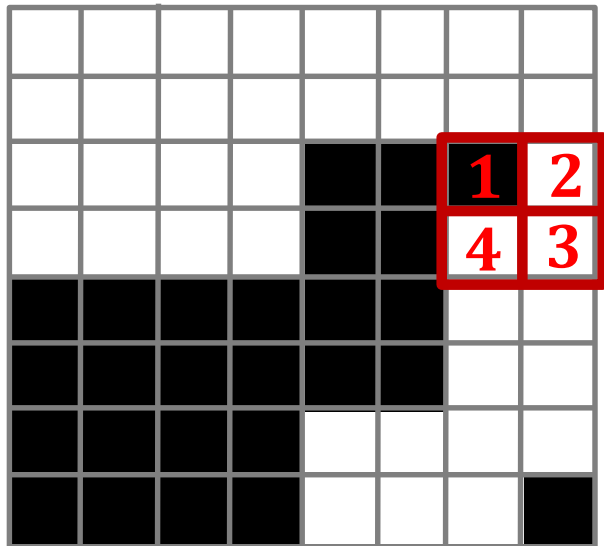
○ vide

nœud

● en partie plein

feuille

● plein



Quadtree

feuille

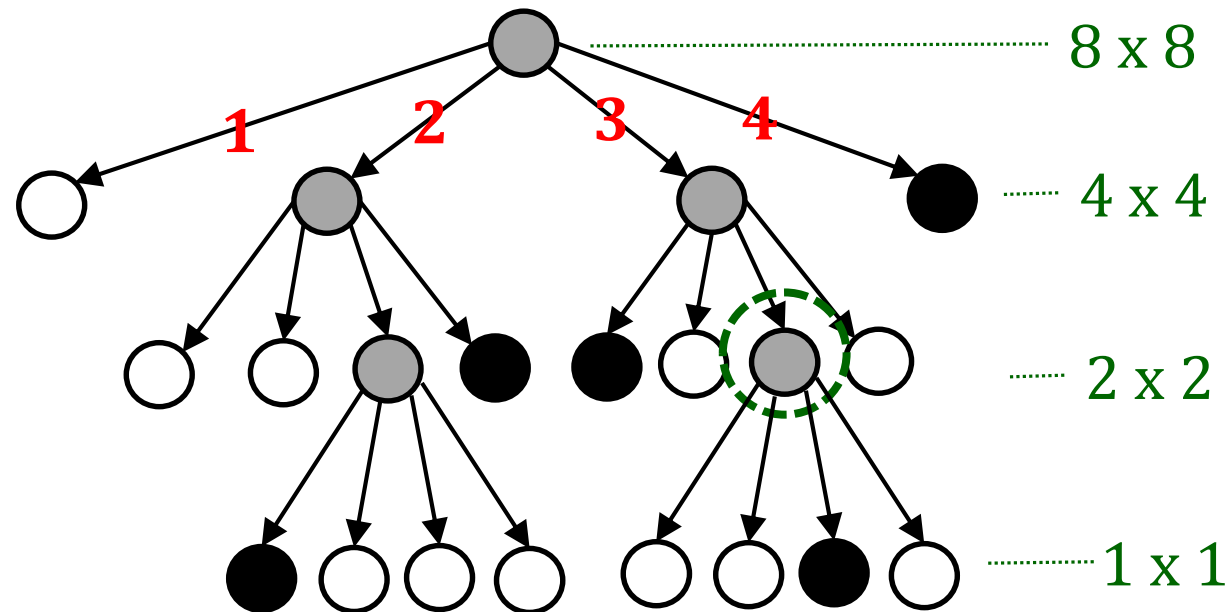
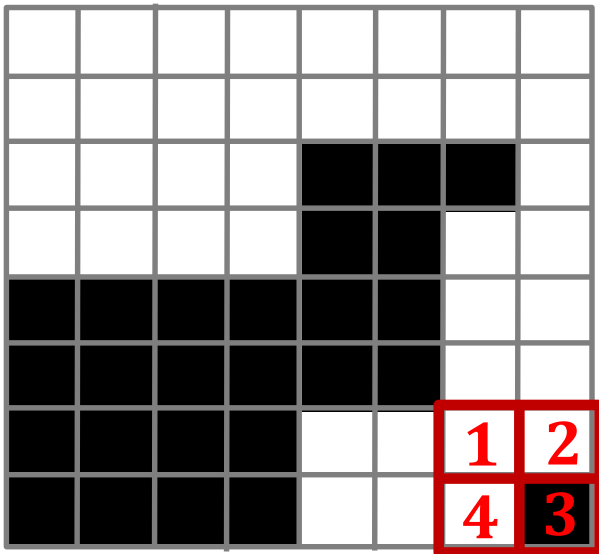
○ vide

nœud

● en partie plein

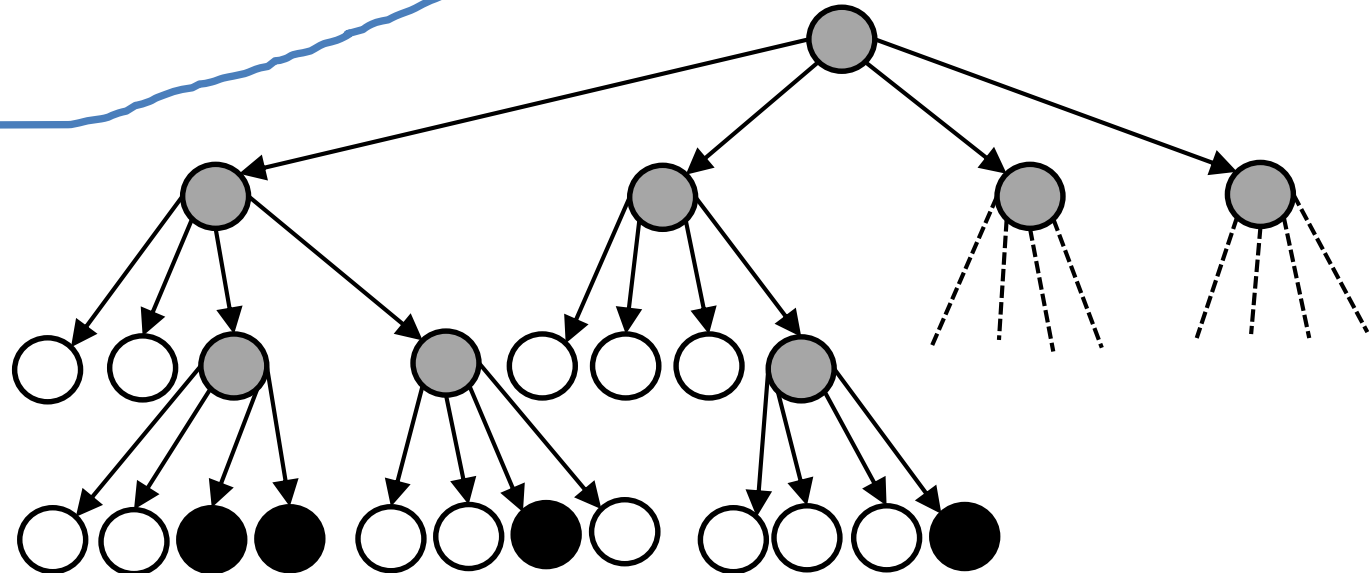
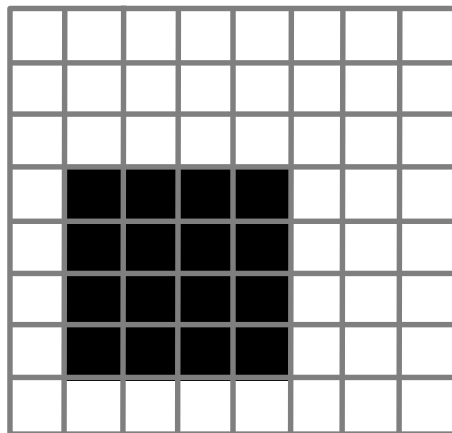
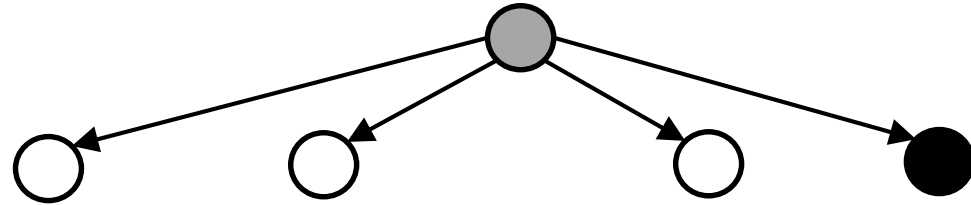
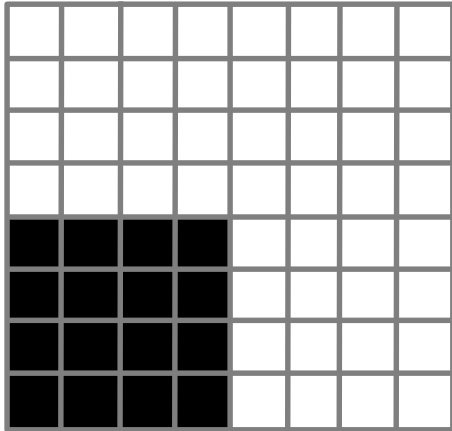
feuille

● plein



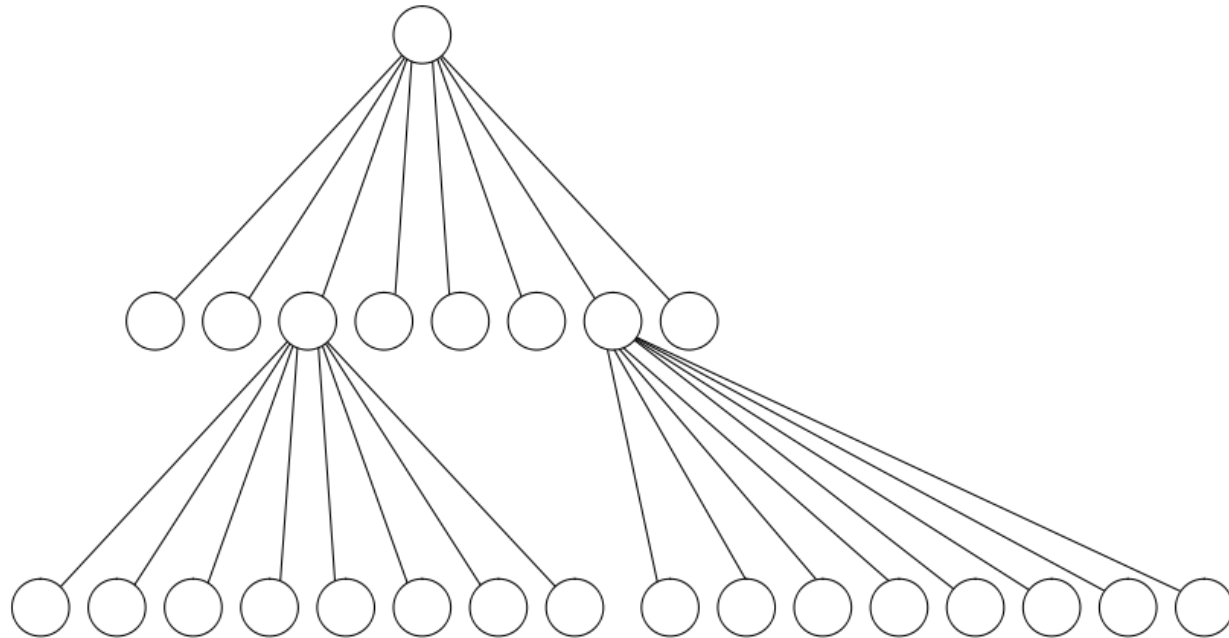
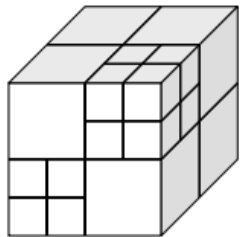
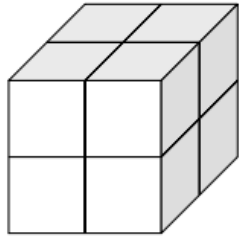
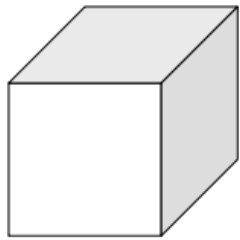
Quadtree : instabilité

- Petit changement à l'entrée \neq petit changement à la sortie



Octree

- Pour les représentations en 3D

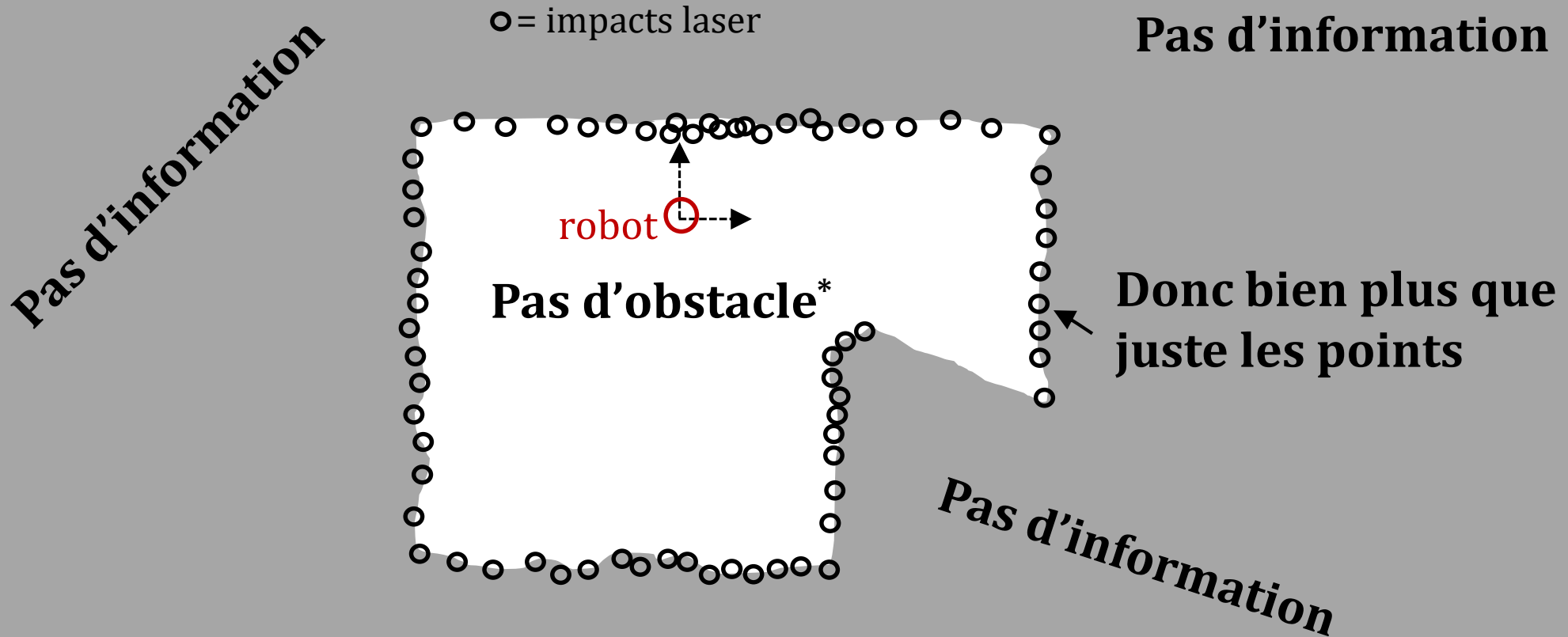


tiré de wikipédia

Construction d'une carte de type grille d'occupation (*occupancy grid*)




Grille d'occupation : intuition

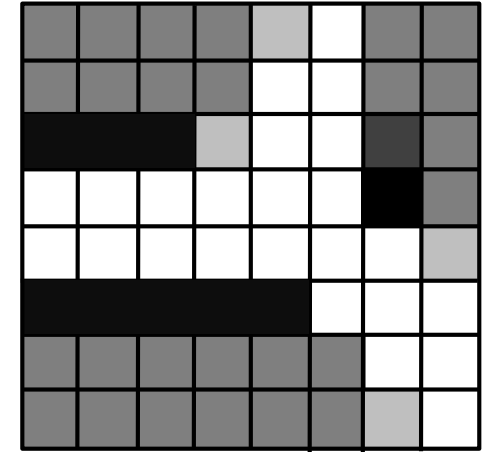
- Quelles informations vous avez, avec ces mesures laser?



* En supposant que les obstacles soient plus larges que l'espace entre les faisceaux

Gérer l'information : grille d'occupation

- Carte du monde divisée en carrés de tailles égales ($10 \times 10 \text{ cm}$)
- Chaque case de la grille encode la probabilité qu'un obstacle y soit présent :
 - $p(c)=0$  si libre (certain)
 - $p(c)=0.5$  si on ne connaît rien
 - $p(c)=1.0$  si obstacle (certain)
- On connaît parfaitement la pose x_t du robot¹
- Au début, toutes les cases sont à $p(c)=0.5$ (environnement inconnu)
- Mesure z_t du capteur indique présence/absence d'obstacle, en probabilité, via l'inverse de la fonction du capteur



¹Par exemple, on a fait le SLAM, puis on génère la grille d'occupation. Elle sera utile pour planifier des mouvements, alors que la carte de SLAM n'est pas faite pour ça.

Création de la carte d'occupation m

- L'on cherche carte m composée de cases c_i contenant un obstacle (notez l'absence des commandes $u_{1:t}$)

$$p(m \mid x_{1:t}, z_{1:t}) \text{ où } m = \{c_1, \dots, c_N\}$$

- En considérant les cases comme indépendantes¹, on peut simplifier le problème en factorisant selon c_i : $p(c_1, c_2 \mid A) = p(c_1 \mid A)p(c_2 \mid A)$

$$p(m \mid x_{1:t}, z_{1:t}) = \prod_{i=1}^N p(c_i \mid x_{1:t}, z_{1:t})$$

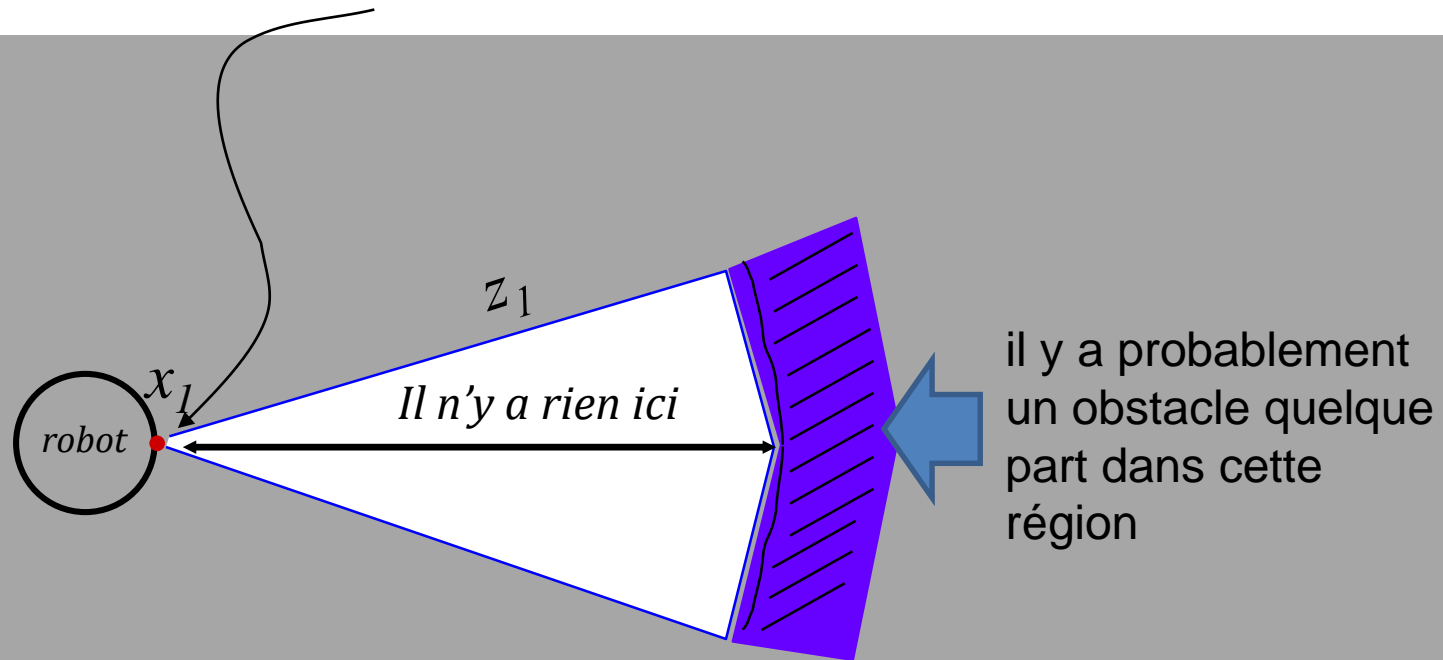
¹ce qui n'est pas réel, car il y a une forte corrélation spatiale...⁴³

Création de la carte d'occupation m

- Comme les cases c_i sont indépendantes, incorporer une nouvelle donnée $\{x_t, z_t\}$ consistera à mettre à jour de façon individuelle les cases c_i affectées
- Simplifie/accélère grandement le problème...
- ... mais au prix d'une certaine perte d'exactitude (corrélacion spatiale non-respectée)
- Quelles sont les cases c_i affectées?
 - dépend (du modèle inverse) du capteur




Grille d'occupation avec sonar

Si un sonar indique obstacle à 4 mètres?

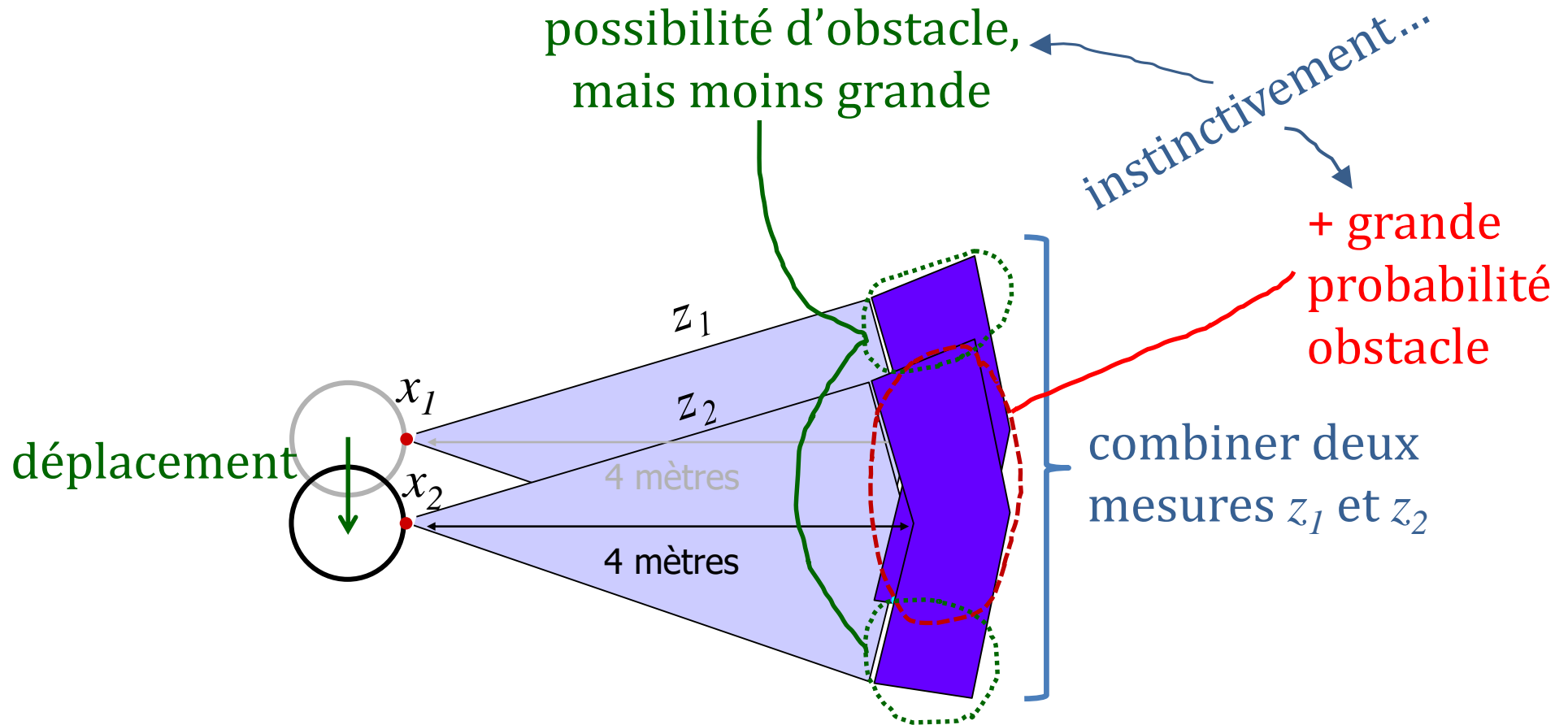


Rappel : sonar émet en forme de cône

Carte locale

-  vide
-  pas d'information
-  obstacle

Grille d'occupation : accumulation d'évidences



Comment combiner ces informations de manière fondée et efficace?

Combinaison d'évidences sonar

- Combiner les **cotes**: $\text{cote}(p) = \frac{p}{(1-p)} = \frac{p}{\bar{p}}$ p ne s'est pas produit

- Si $p=75\%$ chance de gagner, la **cote** sera :

$$\frac{0.75}{(1-0.75)} = \frac{0.75}{0.25} = 3$$

- Chacune des cases c_i contiendra la **cote** (ou $\log(\text{cote})$) qui encode la probabilité de présence d'un obstacle.
- Pourquoi les **cotes**? pour faciliter les calculs¹...

Algorithme grille d'occupation

```
occupancy_grid_mapping( $\{C_{i,t-1}\}$ ,  $x_t$ ,  $z_t$ )
```

```
  for all cells  $m_i$  in map do
```

```
    if  $m_i$  is in perceptual field of  $z_t$  then
```

```
       $C_{i,t} = C_{i,t-1} \times$  cote_sensor( $m_i$ ,  $x_t$ ,  $z_t$ )
```

```
    else
```

```
       $C_{i,t} = C_{i,t-1}$ 
```

```
    endif
```

```
  endfor
```

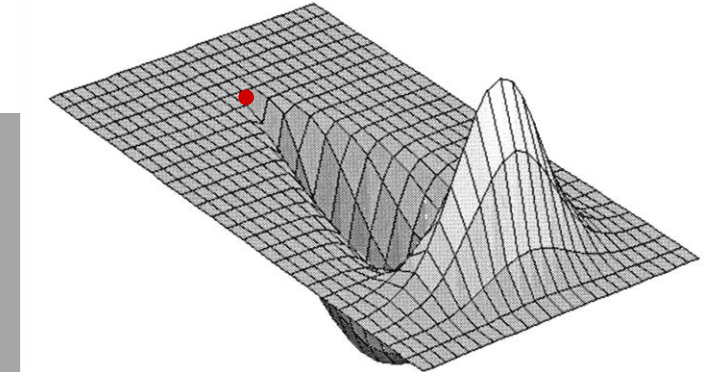
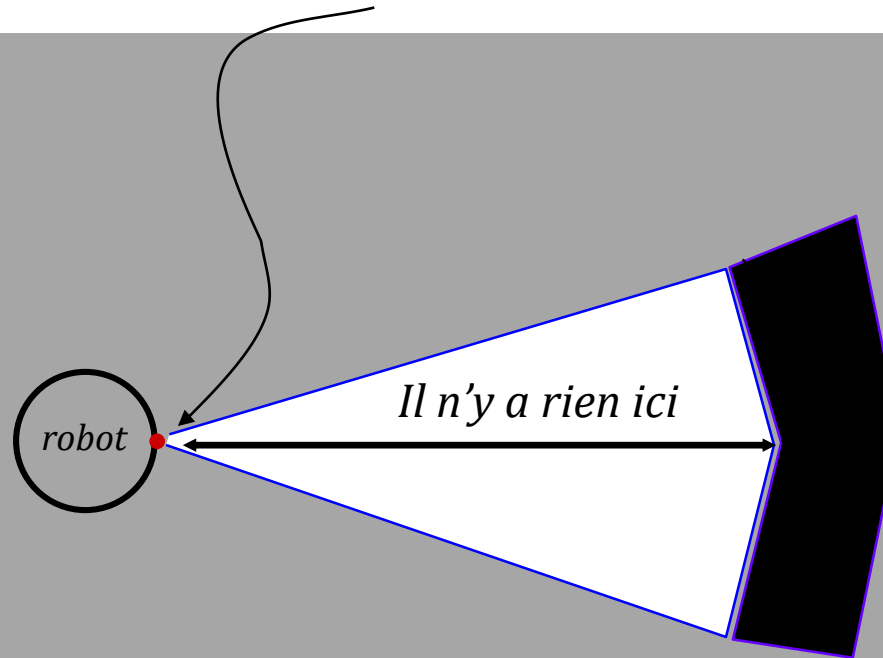
```
  return  $\{C_{i,t}\}$ 
```

Inverse du modèle du capteur

Carte de départ est le prior $C_{i,0} = \frac{p(\text{libre})}{p(\text{occupé})}$ (souvent 1)

Modèle inverse du capteur




Si un sonar indique obstacle à 4 mètres?



il y a probablement un obstacle quelque part dans cette région

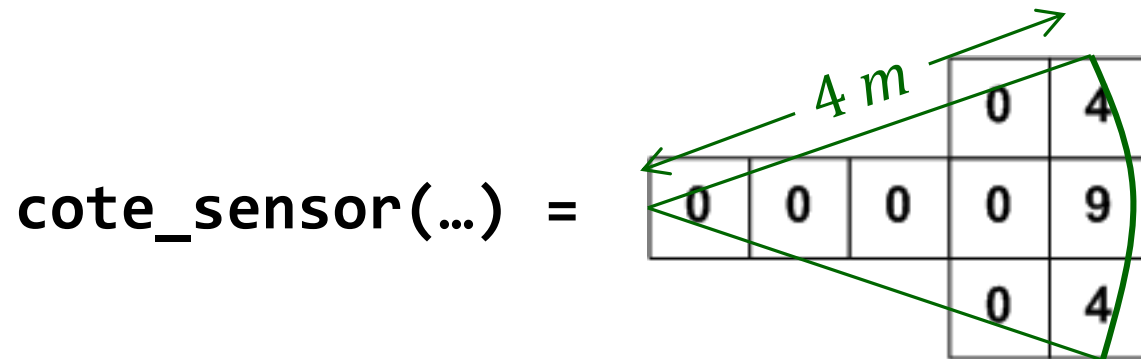
Rappel : sonar émet en forme de cône

Cote du modèle inverse

-  cote < 1
-  cote = 1 (pas d'info)
-  cote > 1

Exemple

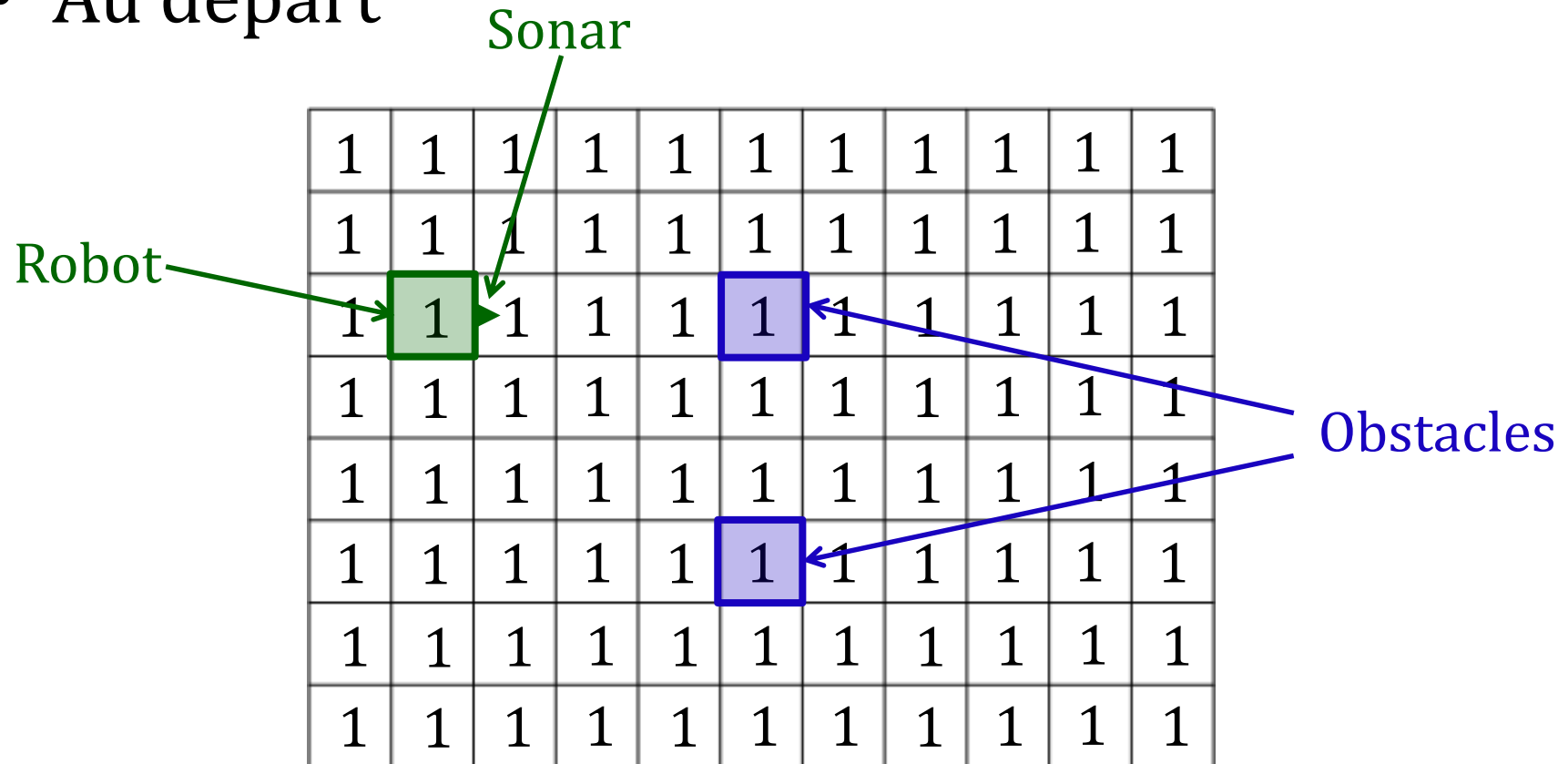
Fonction du capteur sonar pour $z = 4$ m



À l'intérieur du
cône de sonar

Exemple

- Au départ



Exemple

- Première mesure $Z_1 = 4 m$

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

`cote_sensor(z=4)`

			0	4
0	0	0	0	9
			0	4

`cote_sensor(z=3)`

			0	4
0	0	0	9	
			0	4

Exemple

- Première mesure $Z_1 = 4 m$

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	4	1	1	1	1	1
1	0	0	0	0	9	1	1	1	1	1
1	1	1	1	0	4	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

`cote_sensor(z=4)`

			0	4
0	0	0	0	9
			0	4

`cote_sensor(z=3)`

			0	4
0	0		0	9
			0	4

Exemple

- Déplacement

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	4	1	1	1	1	1
1	0	0	0	0	9	1	1	1	1	1
1	1	1	1	0	4	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

`cote_sensor(z=4)`

			0	4
0	0	0	0	9
			0	4

`cote_sensor(z=3)`

			0	4
0	0	0	9	
			0	4

Exemple

- Mesure $Z_2 = 4 m$

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	4	1	1	1	1	1
1	0	0	0	0	9	1	1	1	1	1
1	1	1	1	0	4	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

`cote_sensor(z=4)`

			0	4
0	0	0	0	9
			0	4

`cote_sensor(z=3)`

			0	4
0	0	0	9	
			0	4

Exemple

- Mesure $Z_2 = 4 m$

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	4	1	1	1	1	1
1	0	0	0	0	36	1	1	1	1	1
1	0	0	0	0	36	1	1	1	1	1
1	1	1	1	0	4	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

cote_sensor(z=4)

			0	4
0	0	0	0	9
			0	4

cote_sensor(z=3)

			0	4
0	0	0	9	
			0	4

Exemple

- Déplacement

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	4	1	1	1	1	1
1	0	0	0	0	36	1	1	1	1	1
1	0	0	0	0	36	1	1	1	1	1
1	1	1	1	0	4	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

`cote_sensor(z=4)`

			0	4
0	0	0	0	9
			0	4

`cote_sensor(z=3)`

			0	4
0	0	0	9	
			0	4

Exemple

- Mesure $Z_3 = 4 m$

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	4	1	1	1	1	1
1	0	0	0	0	36	1	1	1	1	1
1	0	0	0	0	36	1	1	1	1	1
1	1	1	1	0	4	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

cote_sensor(z=4)

			0	4
0	0	0	0	9
			0	4

cote_sensor(z=3)

			0	4
0	0		0	9
			0	4

Exemple

- Mesure $Z_3 = 4 m$

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	4	1	1	1	1	1
1	0	0	0	0	36	1	1	1	1	1
1	0	0	0	0	144	1	1	1	1	1
1	0	0	0	0	36	1	1	1	1	1
1	1	1	1	0	4	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

cote_sensor(z=4)

			0	4
0	0	0	0	9
			0	4

cote_sensor(z=3)

			0	4
0	0	0	9	
			0	4

Exemple

- Long déplacement...

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	4	1	1	1	1	1
1	0	0	0	0	36	1	1	1	1	1
1	0	0	0	0	144	1	1	1	1	1
1	0	0	0	0	36	1	1	1	1	1
1	1	1	1	0	4	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

`cote_sensor(z=4)`

			0	4
0	0	0	0	9
			0	4

`cote_sensor(z=3)`

			0	4
0	0	0	9	
			0	4

Exemple

- Mesure $Z_4=3m$

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	4	1	1	1	1	1
1	0	0	0	0	36	1	1	1	1	1
1	0	0	0	0	144	1	1	1	1	1
1	0	0	0	0	36	1	1	1	1	1
1	1	1	1	0	4	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

`cote_sensor(z=4)`

			0	4
0	0	0	0	9
			0	4

`cote_sensor(z=3)`

			0	4
0	0	0	9	
			0	4

Exemple

- Mesure $Z_4=3m$

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	4	1	1	1	1	1
1	0	0	0	0	144	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1
1	0	0	0	0	36	1	1	1	1	1
1	1	1	1	0	4	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

`cote_sensor(z=4)`

			0	4
0	0	0	0	9
			0	4

`cote_sensor(z=3)`

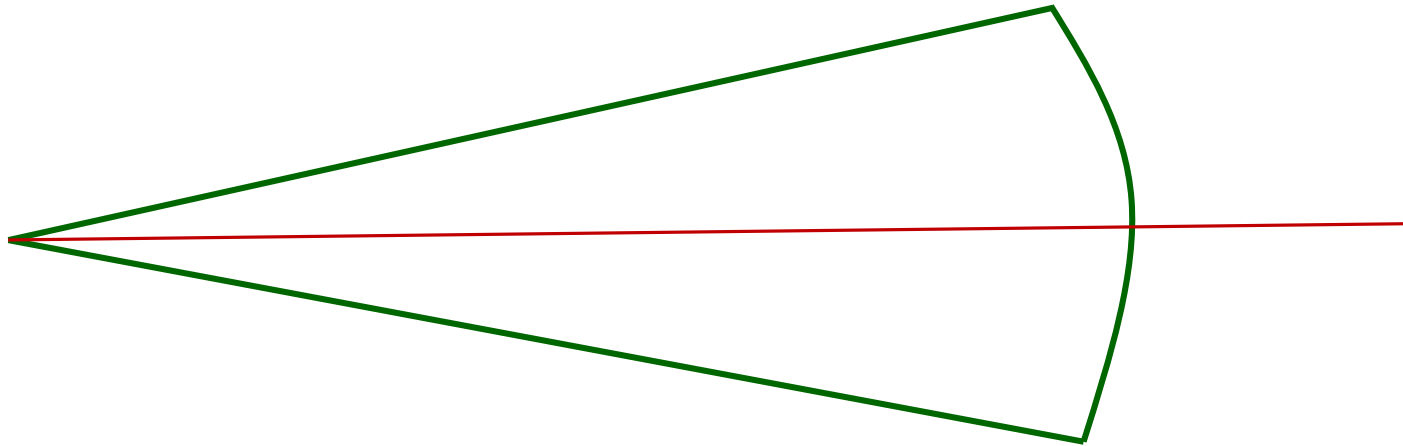
			0	4
0	0	0	9	
			0	4

Notes sur l'exemple précédent

- Mauvaise idée d'avoir des cotes de 0 pour le capteur. Préférable d'avoir des valeurs faibles mais non nulles pour tenir compte des erreurs possibles
- Importance de la trajectoire dans la construction
- Peut utiliser le log des cotes :
 - passe de multiplications à des additions
 - va aller de $-\infty$ à $+\infty$ (0 = aucune connaissance)
 - meilleure stabilité numérique

Avec laser 2D?

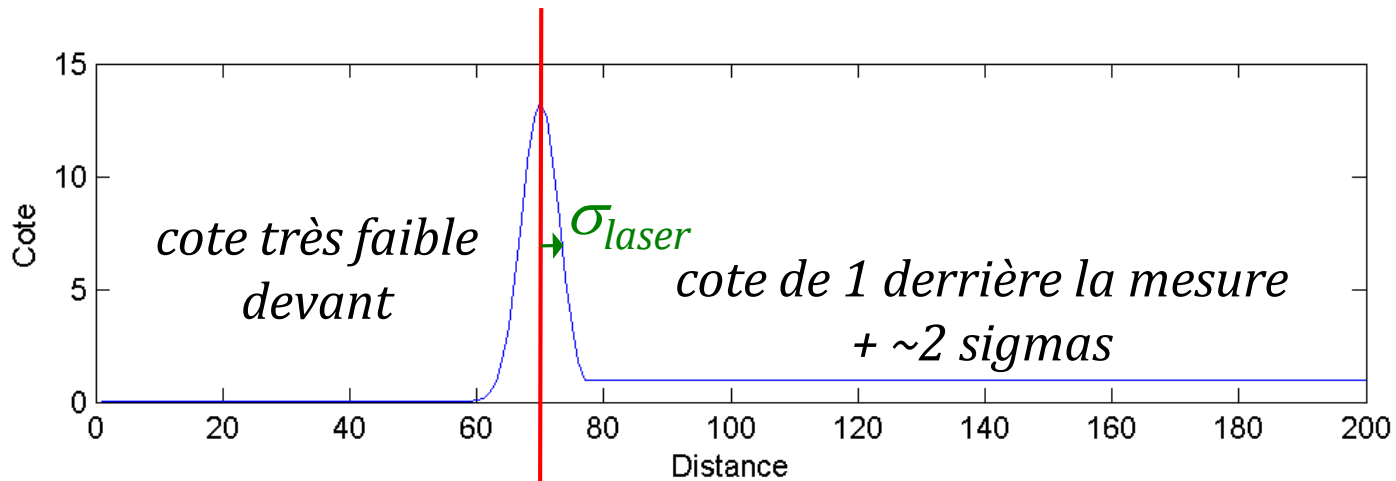
- Un peu plus compliqué, car le « cône » du **laser** est beaucoup plus étroit...



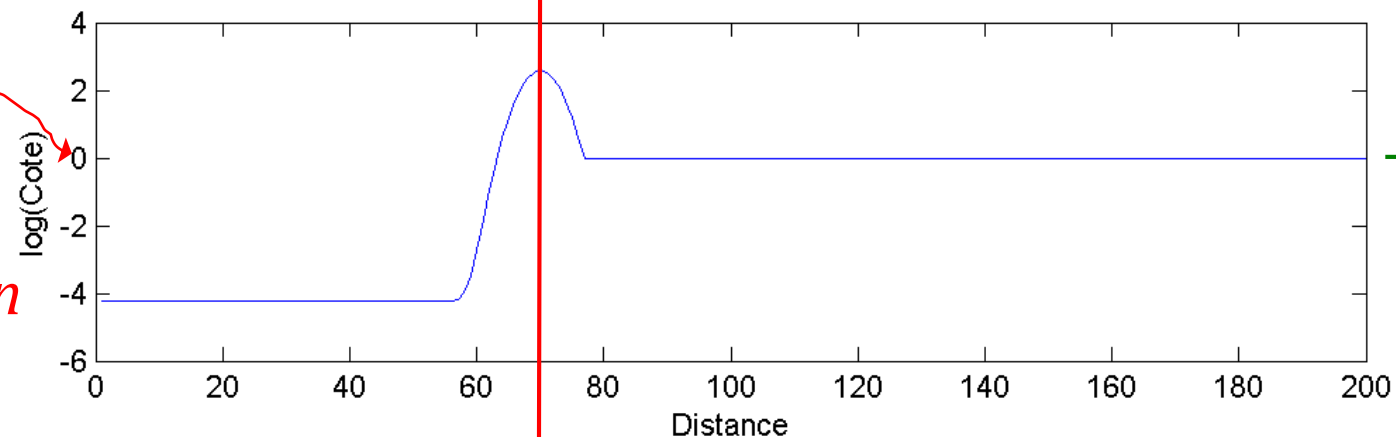
- Il ne mettra pas à jour beaucoup de cases...

Exemple matlab GridMap . zip

- Pour les cotes du capteur, j'ai utilisé un modèle approximatif



*log(1)=0
indique
absence
d'information*



occupé

libre

Exemple matlab GridMap . zip

