

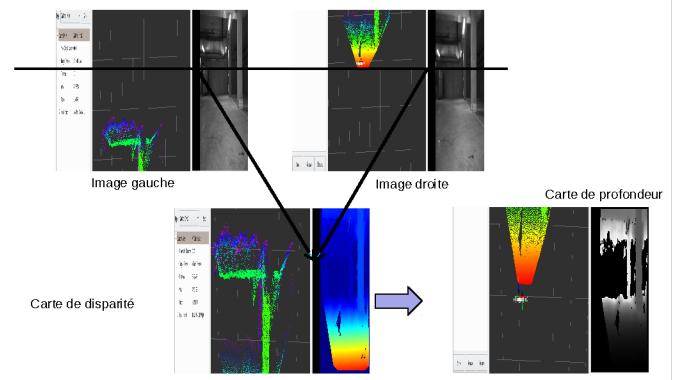
Cartographie 3D de l'environnement pour la navigation

Aurélien Plyer
Julien Moras, Martial Sanfourche, Guy Le Besnerais



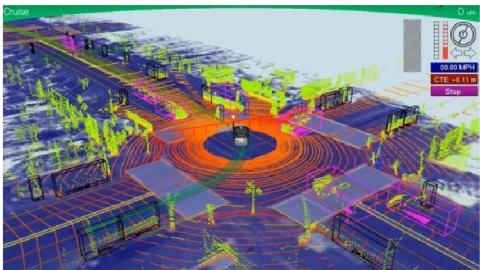
Qu'est ce qu'on a vu en 3D ?

- Construction de carte de profondeur avec un système stéréo



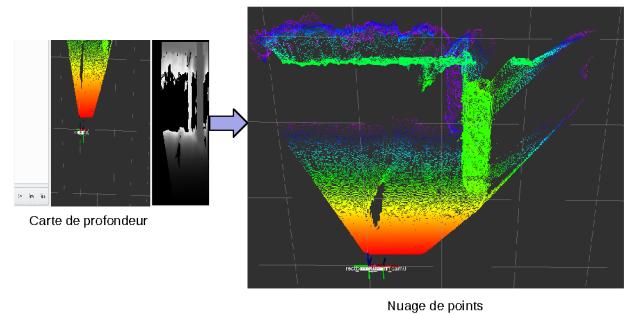
Problème de la navigation

- Nécessité de percevoir l'environnement
 - Eviter les obstacles
 - Trouver son chemin
 - Interagir avec l'environnement
 - Comprendre ce qu'il se passe autour



Qu'est ce qu'on a vu en 3D ?

- Construction d'un nuage de points à partir de la carte de profondeur (stéréo ou capteur RGBD)



Qu'est ce qu'on a vu en 3D ?

- Capteurs stéréo
 - Calcule de disparité
 - Dense



- => carte de profondeur



- Capteur RGBD
 - => Carte de profondeur



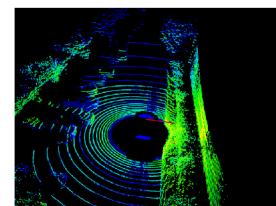
Kinect : résolution 640x480 => 307 200 points
À 30Hz => + de 9 Millions de points / seconde

Autres capteurs 3D

- Lidar
- Camera TOF



- Produisent aussi des nuages de points denses



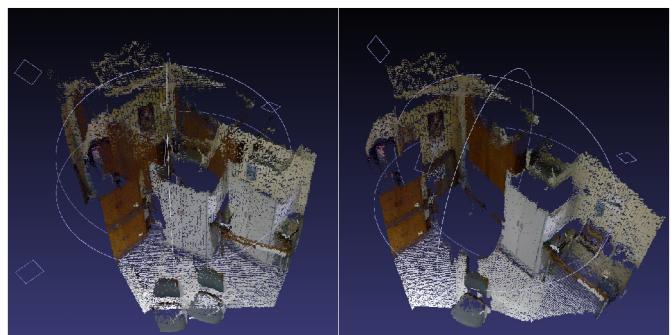
Lidar Velodyne 64 :
1,3 millions de points / seconde

Modélisation de l'environnement à base de nuages de points ?

- Les capteurs sont capables de fournir des nuages de points denses avec une cadence importante

- L'idée est de les utiliser pour reconstruire l'environnement en 3D
- Problème de recalage !
 - Les nuages de points sont mesurés dans le repère du capteur qui se déplace
 - Ces nuages de points ne seront pas correctement alignés
 - => à terme, un énorme ensemble de points désordonnés

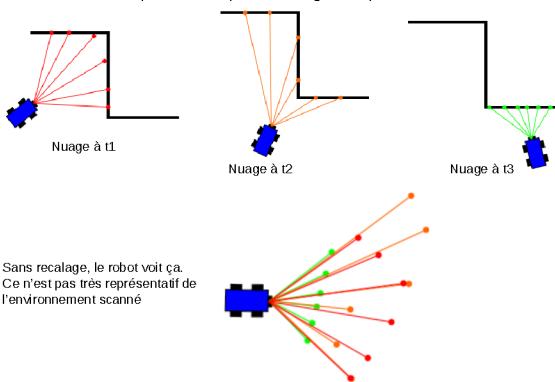
Recalage des nuages de points



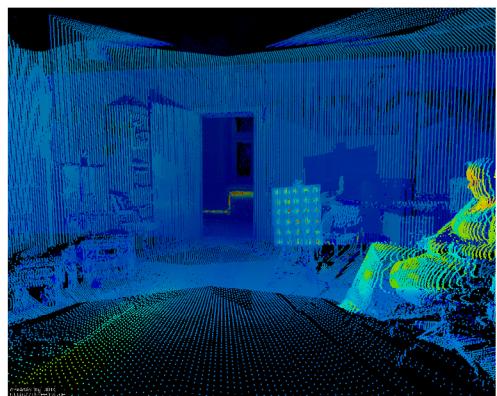
Modélisation de l'environnement à base de nuages de points ?

Exemple :

- Un robot se déplace et acquiert 3 nuages de points de l'environnement



Nuages de points recalés



Comment déterminer les transformations ?

Très gros nuages de points recalés



Modèle 3D d'environnement à base de nuages de points

•Avantages

- Création de carte simple

- L'affichage et la manipulation sont simple

•Inconvénients

•Taille en mémoire très importante

- On a vu que la Kinect est capable de fournir près de 9 millions de points/s
- Et même si on n'enregistre qu'à 10Hz (3Mpt/sec) et uniquement les coordonnées (x,y,z), on remplit 1Go en environ 35s.

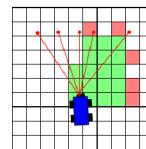
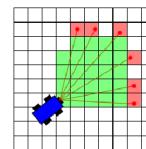
•Pas de structure (voisinage, connectivité, etc)

- Pas adapté à la décision ou à la planification de trajectoire

Construction d'une grille d'occupation binaire

•Pour plusieurs nuages de points

- Il faut prendre en compte la position du capteur pour les cases libres
 - On ne peut pas calculer la grille d'un seul coup avec tous les points, il faut ajouter les nuages de points un par un.



•Grace à ces grilles on peut :

- diminuer l'espace mémoire
- classifier tout l'espace comme libre ou occupé
- Calculer rapidement des chemins (A*)

Planification de trajectoires

•Une solution simple : les voxels

- On découpe une portion de l'espace en petits cubes (eg. minecraft)
- Chaque voxel (petit cube) a un état binaire :
 - 1 / occupé et
 - 0 / libre

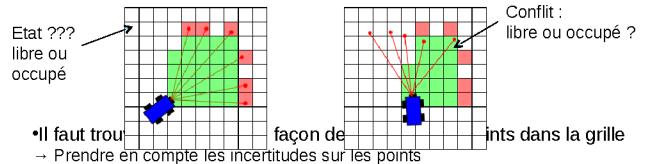
•Principe simple de remplissage des voxels

- si on a un point du nuage dans un voxel, c'est qu'il est occupé
- les voxels intermédiaires sur le rayon de triangulation entre le capteur et le point sont considérés comme libres

Vers les grilles d'occupation

•Problèmes

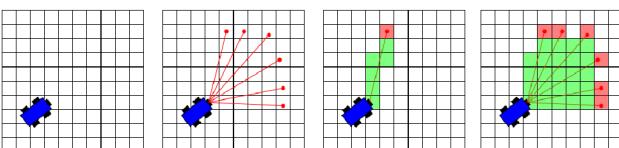
- Comment classifier les voxels pas détectés (libre, occupé, inconnu ?)
- Si on remplit simplement la grille on risque d'avoir des conflits quand on ajoute un nouveau nuage de points
 - A cause du bruits de mesures et des éventuelles erreurs de recalage
 - Les points risquent de tomber dans des voxels libres et des voxels occupés risquent d'être traversés par des rayons.



Construction d'une grille d'occupation binaire

•Principe simple de remplissage des voxels

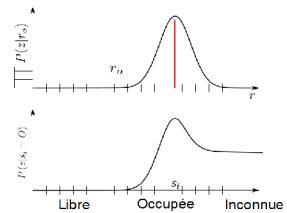
- si on a un point du nuage dans un voxel, c'est qu'il est occupé
- les voxels intermédiaires sur le rayon de triangulation entre le capteur et le point sont considérés comme libres



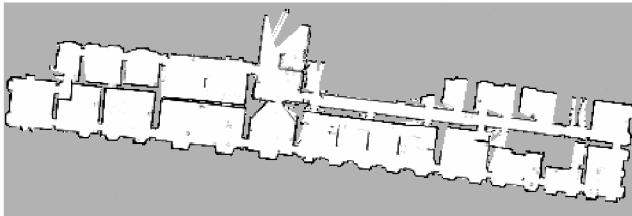
Prise en compte des incertitudes

•De façon intuitive, on remarque bien que :

- les cellules situées avant une mesure sont plutôt libres puisque le faisceau a pu les traverser
- Les cellules proches de la mesure sont plutôt occupées
- Les cellules situées derrière la mesure sont plutôt inconnues car elles sont masquées par l'obstacle



Exemple de grille d'occupation 2D



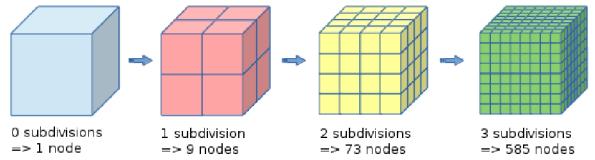
La probabilité d'occupation est représentée par l'échelle de gris
 Blanc : $P(s_i = O) = 0$ et en noir $P(s_i = O) = 1$
 Gris : $P(s_i = O) = P(s_i = E) = 0.5$

Grille d'occupation

- Avantages
 - Affichage simple (images ou voxels)
 - Permet la fusion et le filtrage des données
 - Permet de représenter l'espace libre (nécessaire pour la navigation)
- Inconvénients
 - Relation probabiliste entre les mesures et l'occupation pas simple à utiliser
 - Coûteux en mémoire (mais moins que les nuages de points)
 - Mise à jour calculatoire
 - Un environnement est à 90% fait d'espace vide

QuadTree et OcTree

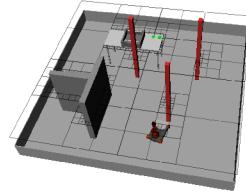
- Structure en arbre où chaque nœud a :
 - QuadTree => 4 nœuds descendants
 - OcTree => 8 nœuds descendants



QuadTree et OcTree

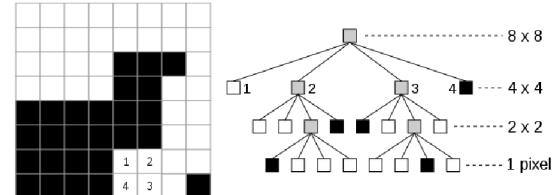
- Problème de la résolution de la grille
 - Résolution trop faible : pas assez de détails dans les zones fines
 - Résolution trop importante : Coût de mise à jour et utilisation mémoire
- Une solution serait de faire une grille multi-résolution
 - Des zones très résolues pour avoir des détails sur les obstacles
 - Des zones faiblement résolues pour les zones vides

=> Les quadTree (2D) et OcTree (3D)



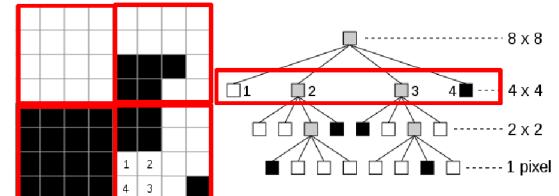
QuadTree/OcTree et grilles d'occupation

- Grille d'occupation
 - Une probabilité par case
- Avec des Quad-Tree / OcTree
 - Chaque nœud en fin de branche à une probabilité d'occupation



QuadTree/OcTree et grilles d'occupation

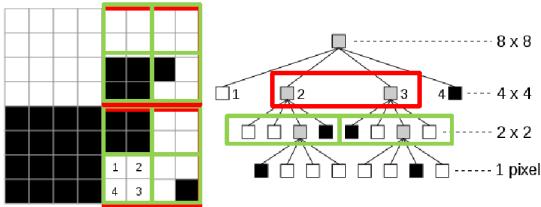
- Grille d'occupation
 - Une probabilité par case
- Avec des Quad-Tree / OcTree
 - Chaque feuille a une probabilité d'occupation



QuadTree/OcTree et grilles d'occupation

- Grille occupation
 - Une probabilité par case

- Avec des Quad-Tree / OcTree
 - Chaque feuille a une probabilité d'occupation



Octomap : Apports et limites

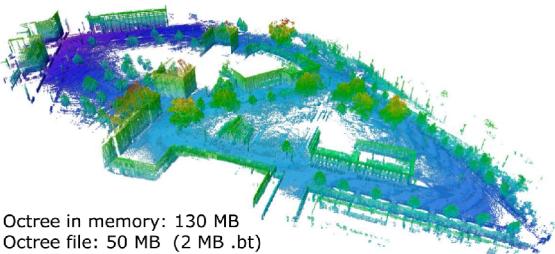
• Par rapport aux grilles d'occupation classiques

- Structure en OcTree
 - Permet de réduire fortement l'espace mémoire pour les grandes zones libres ou occupées
 - Utilisation des log-Odds
 - Réduit le nombre de calculs pour faire une mise à jour (plusieurs multiplications et divisions) => une addition

• Limites

- Visuellement : pas très fidèle avec l'environnement
- Compromis : Performance/Qualité
 - Dans un environnement réel, il faut une résolution assez fine pour avoir une reconstruction assez fine
 - Dans ces cas la les performances sont quand même limitées
- Prise en compte des incertitudes du capteurs
 - Octomap utilise un modèle de capteur parfait
 - Pas de prise en compte des incertitudes de pose
 - => Carte approximative

Octomap : solution d'octree probabiliste



Octree in memory: 130 MB
Octree file: 50 MB (2 MB .bt)
3D Grid: 649 MB

Application : robots autonomes ONERA

• Robots équipés d'un système de caméras stéréo

- algorithme de localisation eVO
- calcul de disparité dense
- cartographie Octomap



Octomap : Résultats

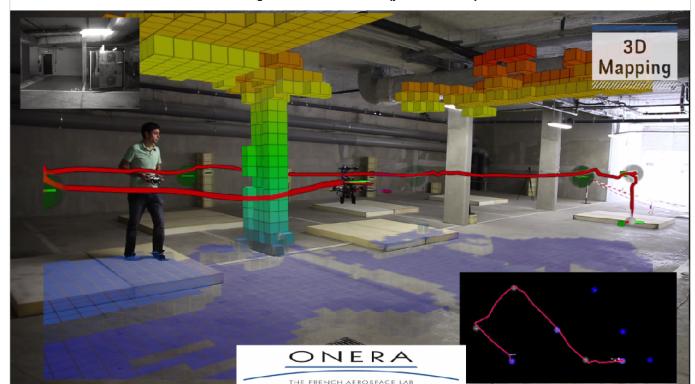
▪ RGBD freiburg1_360 (8 x 7 x 5 m³, 2 cm resolution)



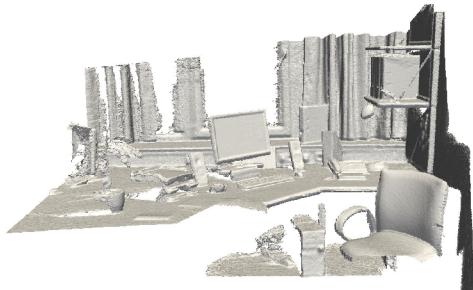
Application : robots autonomes ONERA

• Robots équipés d'un système de caméras stéréo

- Démonstration de navigation autonome (juillet 2015)

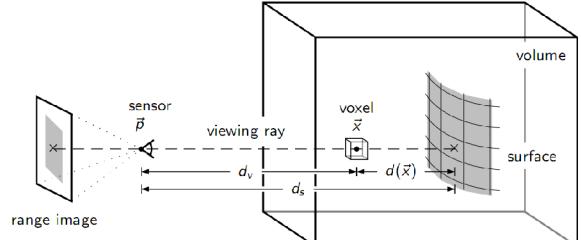


Et les maillages ???



Fonction de distance signée (SDF)

- Une approche basée sur une grille de voxel mais qui ne contient pas l'occupation mais la distance à l'obstacle le plus proche
- La distance est signée (+) devant la surface (-) derrière => passage à 0 sur la surface



Maillages 3D

- Les maillages sont des représentations très utilisées en infographie

Avantages

- Représentation très compacte en terme de mémoire
- Rendu simple et pris en charge par les cartes graphiques
- Rendu de très bonne qualité

Problèmes

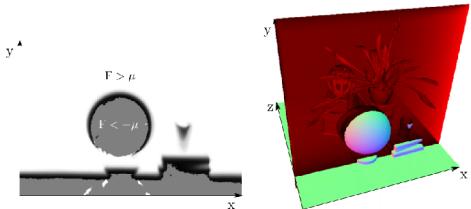
- Pas de représentation de l'espace libre
- Comment les construire à partir de nuages de points ??

TSDF

- En pratique, il ne sert à rien de calculer la SDF sur tout le volume, si on est assez loin de toutes surfaces, alors une grande valeur arbitraire suffit, c'est pour cela que les approches tronquent la SDF à partir d'une certaine distance => TSDF

- Optimisation de temps de calcul

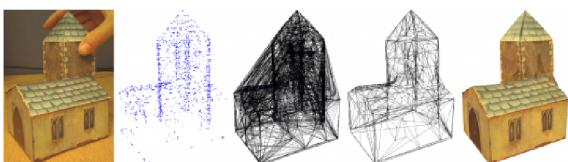
- La figure ci-dessous présente une coupe du cube de TSDF



Construction de maillage à partir d'un nuage de point

• On peut utiliser des algorithmes de triangulation qui permettent de regrouper les points suivant des triangles pour former des facettes.

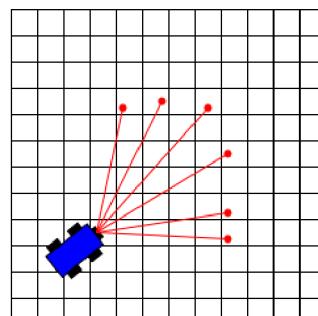
• Par exemple, la triangulation de Delaunay est très utilisée car elle maximise le plus petit angle de l'ensemble des angles des triangles, évitant ainsi les triangles « allongés ».



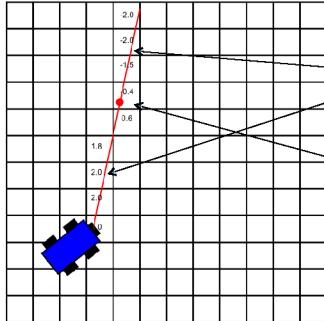
Exemple de reconstruction d'un modèle 3D à partir de la triangulation d'un nuage de points

Construction de TSDF avec des nuages de points

- Chaque case contient une valeur de distance et un poids
- A chaque nouveau nuage de point, la grille de TSDF est mise à jour



Construction de TSDF avec des nuages de points



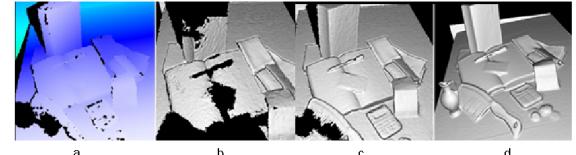
La distance est tronquée à partir d'une certaine valeur
La distance est signée de sorte que le point détecté (i.e. la surface) se trouve au passage à 0

Kinect Fusion

Méthode qui utilise une Kinect

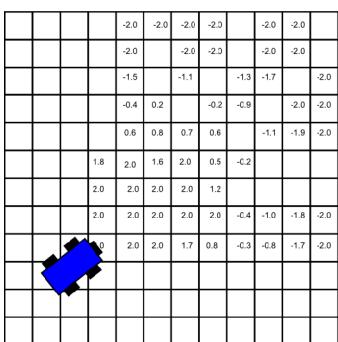
- Réalise un SLAM dense pour recaler les points + ICP pour l'affiner
- Construit une TSDF pour modéliser l'environnement

Méthode calculatoire mais qui fonctionne en temps réel grâce au calcul GPU (limite => mémoire sur le GPU)



Exemple d'une scène modélisant un bureau
Fig. a : Image de profondeur
Fig. b,c,d : La surface produite par la fusion TSDF au cours du temps

Construction de TSDF avec des nuages de points



$$D_t(i) = \frac{W_{t-1}(i).D_{t-1}(i) + w(i).d(i)}{W_{t-1}(i) + w(i)}$$

$$W_t(i) = W_{t-1}(i) + w(i)$$

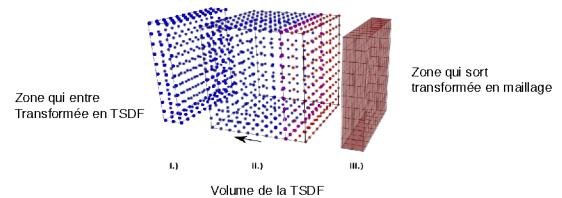
w(i) est le poids

Kintinious

L'approche Kinect fusion est limitée par le volume de la TSDF

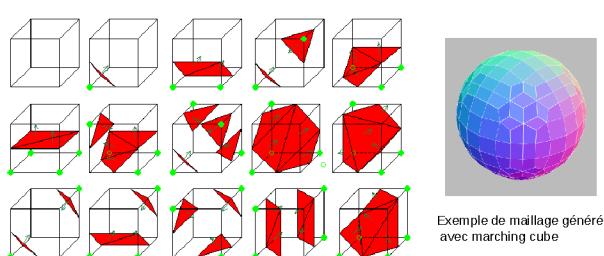
Kintinious est une extension de cette méthode

- Conserver la TSDF pour une petite zone dans laquelle on fait la mise à jour et transformer le reste de la carte en maillage
- Elle utilise l'algorithme marching cube pour construire un maillage sur les zones qui sortent de la TSDF
- Elle réintègre dans la TSDF le maillage sur les zones qui rentrent à nouveau dans le champ de vue

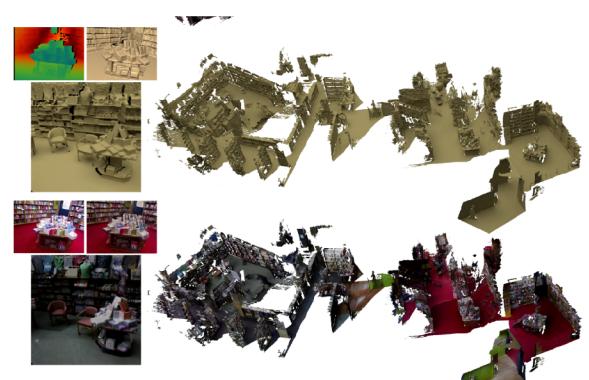


Extraction de surface : Marching cube

- On considère un champ scalaire 3D, par exemple une SDF
- Pour chaque cube formé par 8 points adjacents dans la grille, on compare leurs signes et leurs valeurs
 - En fonction du signe de chacun des coins, on crée des facettes suivant la table ci-dessous



Exemple de maillage généré avec marching cube



Et les objets mobiles ???

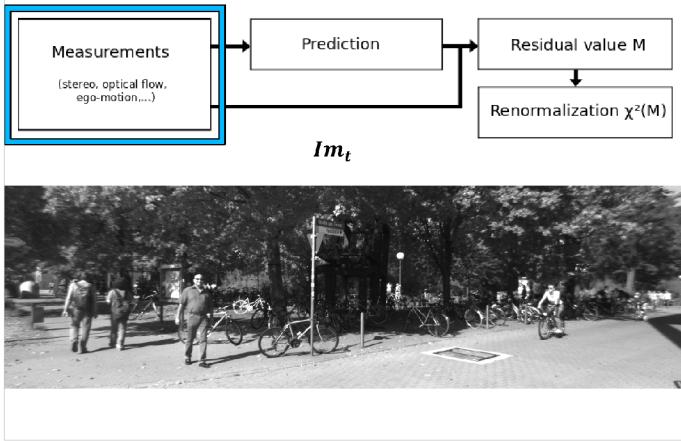
- La détection des objets mobiles est une fonction importante pour la navigation de véhicules autonome.
 - Les méthodes de perception considèrent en général une scène fixe
- Il faut utiliser des approches dédiées à ce problème
- Une solution est de faire des traitements de recalage en considérant le monde statique et de détecter des erreurs importantes
 - Zones occupées qui deviennent libres ou des zones libres qui deviennent occupées
 - Erreurs de recalage
 - Erreurs dans l'image prédictive

Résultats



Résultat M. Derome et al., ONERA/DTIM , 2014

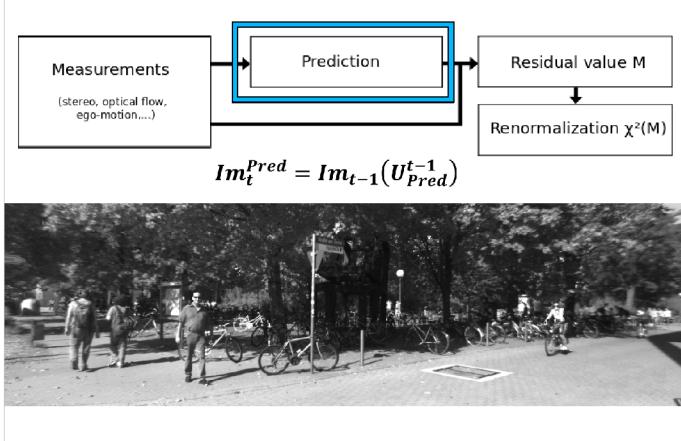
Détection des objets mobiles



Conclusion sur la perception pour la navigation

- Cartographie pour la navigation initialement 2D pour la robotique terrestre
 - Grille d'occupations
- La problématique 3D est arrivée récemment
 - Arrivée de capteurs 3D embarquables
 - Augmentation de la puissance de calcul disponible
- Approches spécifiques qui commencent à se développer
 - Grilles d'occupations sont de moins en moins utilisées
 - Au profit des méthodes à base de SDF

Détection des objets mobiles



- Méthodes à base de maillages ne sont pas très utilisées pour la navigation mais elles sont très utilisées pour faire de la reconstruction 3D fine (pas de contraintes liées à la navigation)
- Méthodes pour gérer les objets mobiles
 - Détection pour ne pas les intégrer dans la carte
 - Pour l'instant pas vraiment de bonne représentation de ces objets
- Reconstruction sémantique
 - Reconnaitre les objets
 - Compréhension de scène
 - Couplage avec des techniques d'apprentissage (Deep learning, etc ...)

Tendances

Références

- A. Elfes, Using occupancy grids for mobile robots perception and navigation, *Computer*, 22, 6, 1989
- A. Hornung and al., OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees, in *Autonomous Robots*, 2013
- S. Izquierdo and al. KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera. In Proc. ACM Symp. User Interface Software and Technology, 559–568, 2011
- H. Roth and al., Moving Volume KinectFusion, British Machine Vision Conference (BMVC), September, 2012
- W. Lorensen and al., Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics* 21, 4, 163–169, 1987
- J. Chen and al., Scalable real-time volumetric surface reconstruction. *ACM Transactions on Graphics (TOG)* 32, 4, 113, 2013