Tests de circulation de véhicules autonomes EZ10 en milieu naturel

Alexis Wilhelm
François Marmoiton

EZ10

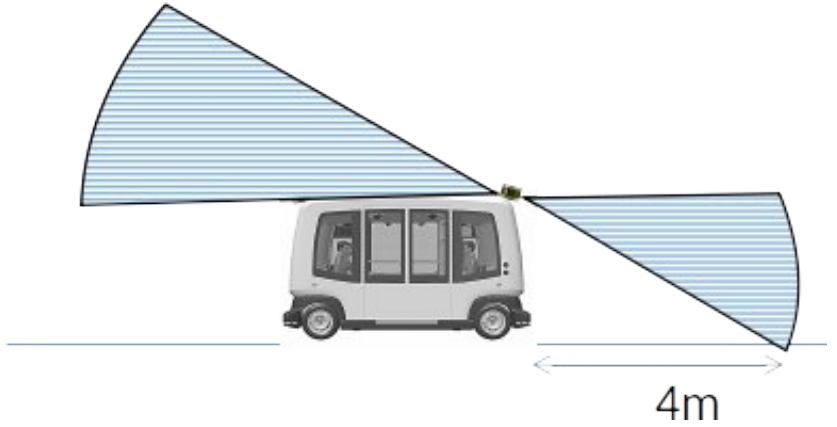
- Navette autonome électrique.
- Llh (m) : 4 x 2 x 2.80
- Poids à vide : 1800 kg
- Véhicule robotisé et instrumenté.
- 6 places assises
- Vitesse max : 40km/h
  25km/h en auto
- Rampe d’accès PMR

Instrumentation

- Capteurs pour le guidage latéral :
  - 2 caméras imperx Bobcat Gige synchronisées (AV+AR)
  - 1 DGNSS proflex 800

- Capteurs pour la sécurité « vulnérables » :
  - 4 Sick LMS151 (Lidar plan aux 4 coins)
  - 2 Hesai PandarXT32 (AV+AR)

- Degrés de liberté à commander :
    - Vitesse
    - Braquage AV
    - Braquage AR



Projet CD03 : Montluçon / Néris les Bains

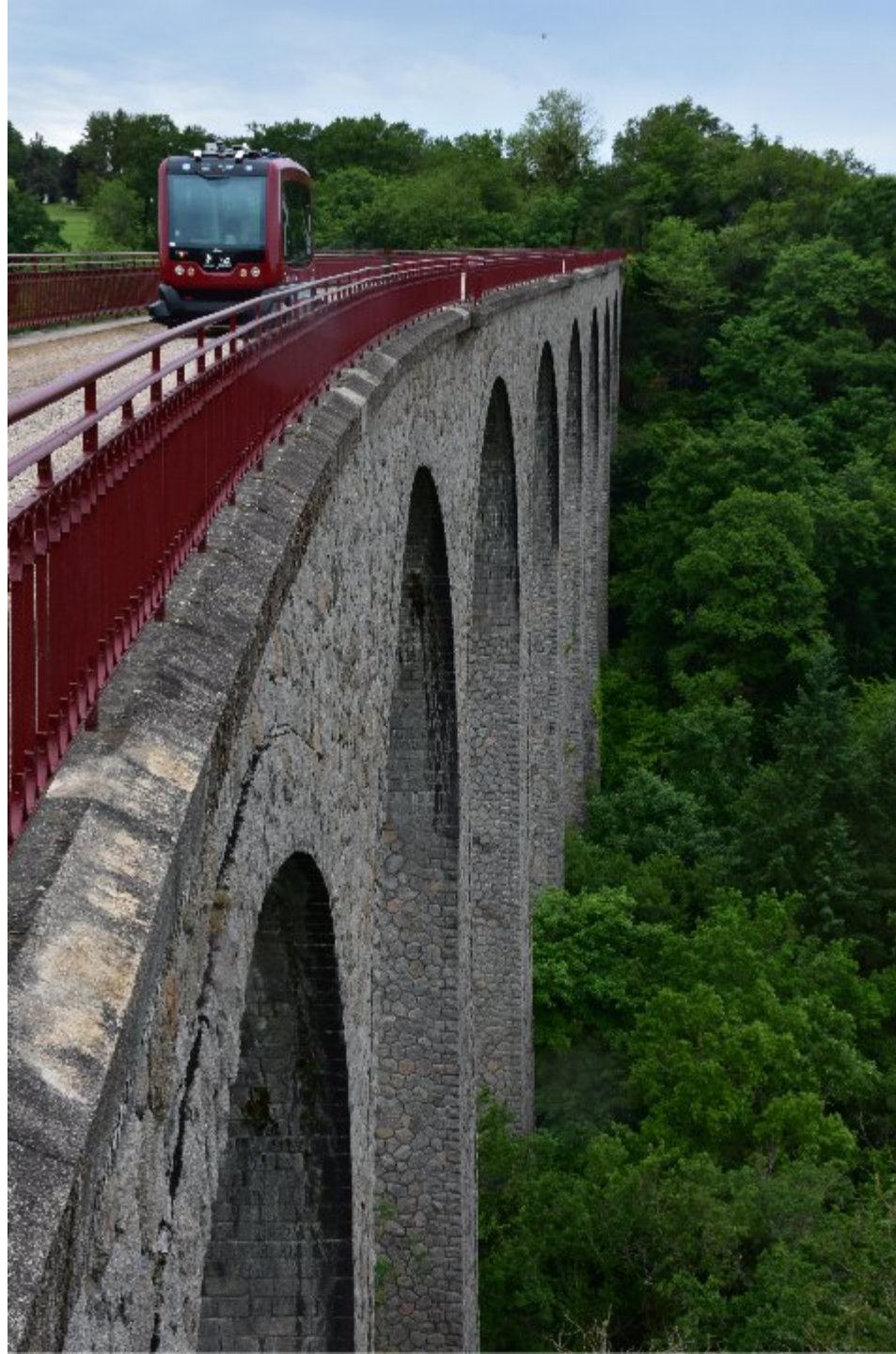
-> Rouler sur une voie ferrée désafectée -
- 6km en voie partagée piéton/cycliste -
- pas de carrefours

3 volets de recherche:

Volet 1 : Tests de circulation d'un véhicule autonome EZ10 en milieu naturel. (Institut Pascal)

Volet 2 : Etude de marché de ce nouveau service de mobilité. (CLERMA)

Volet 3: Etude d'acceptation, acceptabilité, d'expérience utilisateur d'un véhicule autonome est menée par des membres du LAPSCO (Laboratoire de Psychologie Sociale et Cognitive)



Projet CD03 : Montluçon / Néris les Bains

Des niveaux de complexité trés variables pour la localisation du véhicule

* Des éléments structurants.
* De légers masques de la voute céleste.

DGNSS : +
Vision : ++

* Peu d'éléments structurants.
* Pas de masque de la voute céleste.

DGNSS : ++
Vision : +

* Pas d'éléments structurants.
* De légers masques de la voute céleste.

DGNSS : +
Vision : -

* Pas d'éléments structurants.
* Voute céleste masquée.

DGNSS : -
Vision : --

INTERNATIONAL RESEARCH CENTRE ON INNOVATIVE TRANSPORTATION AND PRODUCTION SYSTEMS
Université Clermont Auvergne

ALLIER BOURBONNAIS
Le Département

cnrs

INSTITUT PASCAL

Solution de localisation DGNSS

(c) L. Malaterre

Guidage : suivi de chemin

1) Connaitre les coordonnées du chemin à suivre (UTM)

Guidage : suivi de chemin

1) Connaitre les coordonnées du chemin à suivre
2) calculer le positionnement (position, orientation) du véhicule dans cet espace.
-> Localisation

Guidage : suivi de chemin

1) Connaitre les coordonnées du chemin à suivre
2) calculer le positionnement (position, orientation) du véhicule dans cet espace.
-> Localisation
3) calculer les erreurs de positionnement par rapport au chemin (position, orientation)

Guidage : suivi de chemin

1) Connaitre les coordonnées du chemin à suivre
2) calculer le positionnement (position, orientation) du véhicule dans cet espace.
-> Localisation
3) calculer les erreurs de positionnement par rapport au chemin (position, orientation)
4) Calculer les braquages (consignes) à l'aide d'une loi de commande
-> Contrôle

Guidage : suivi de chemin

1) Connaitre les coordonnées du chemin à suivre
2) calculer le positionnement (position, orientation) du véhicule dans cet espace.
-> Localisation
3) calculer les erreurs de positionnement par rapport au chemin (position, orientation)
4) Calculer les braquages (consignes) à l'aide d'une loi de commande
-> Contrôle
5) Détecter les obstacles puis ralentir ou arrêter le véhicule si nécessaire

Localisation : DGNSS et vision

- GNSS différentiel

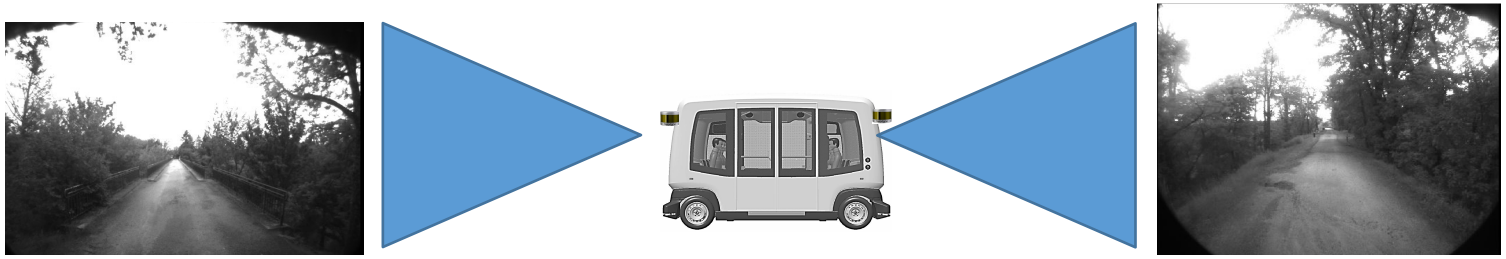
Mise en oeuvre d'un système pour accéder à des corrections différentielles distantes.

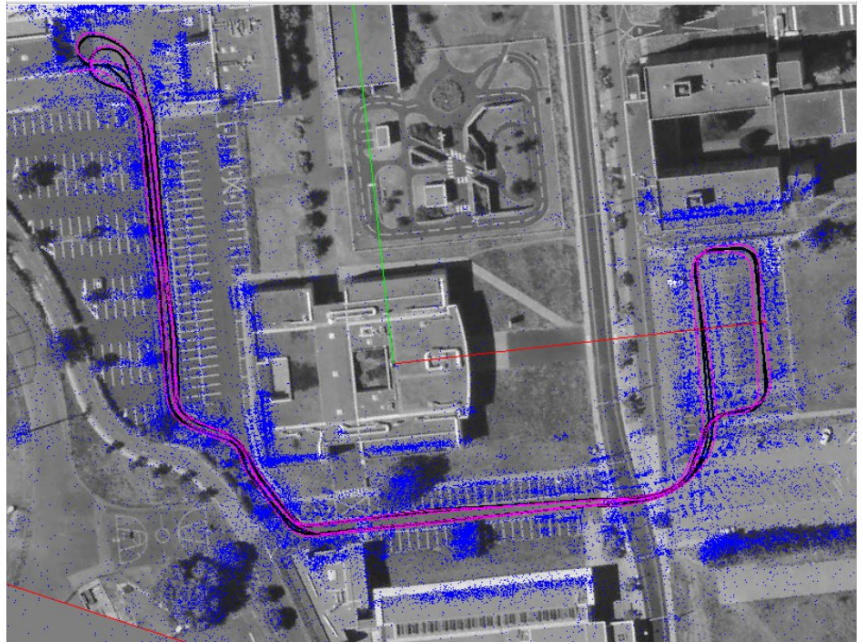
Obtention d'un système GNSS à précision centimétrique sur le site de Néris / Montluçon.

- SLAM visuel

Utilisation des caméras pour la localisation après avoir préalablement construit une cartographie du site.

- Géoréférencement des cartes





Localisation : DGNSS et vision

- GNSS différentiel

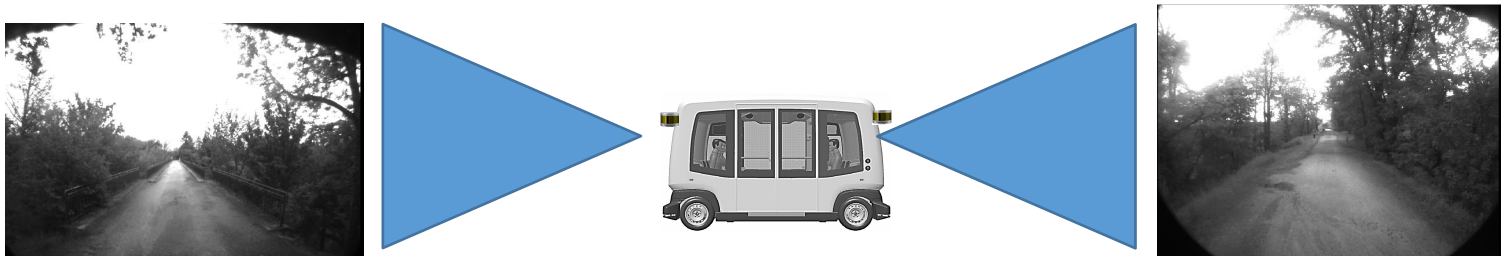
Mise en oeuvre d'un système pour accéder à des corrections différentielles distantes.

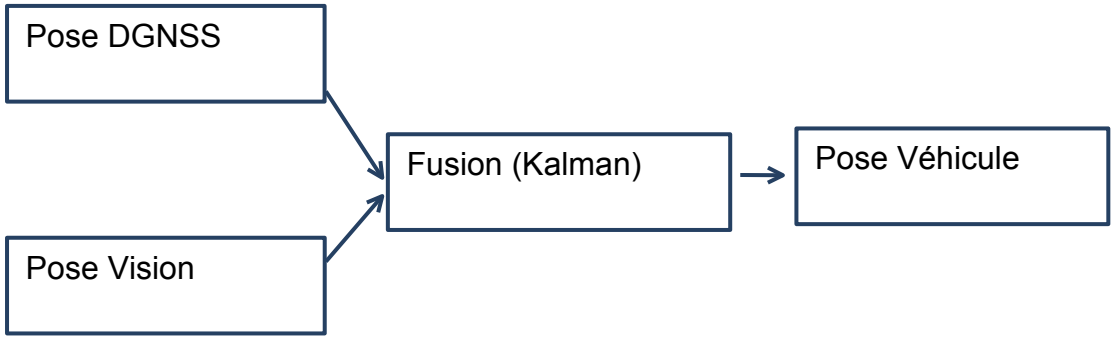
Obtention d'un système GNSS à précision centimétrique sur le site de Néris / Montluçon.

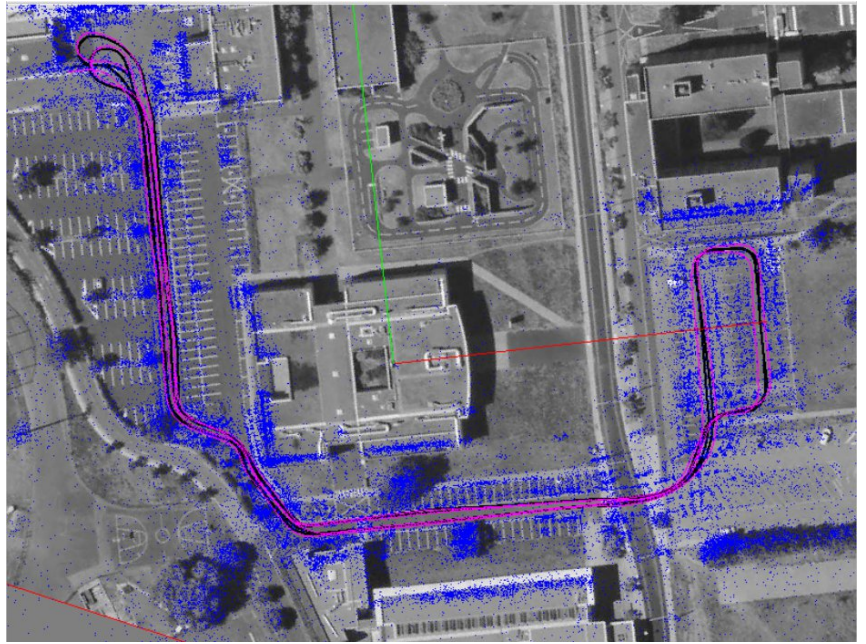
- SLAM visuel

Utilisation des caméras pour la localisation après avoir préalablement construit une cartographie du site.

- Géoréférencement des cartes







Détection d'obstacles

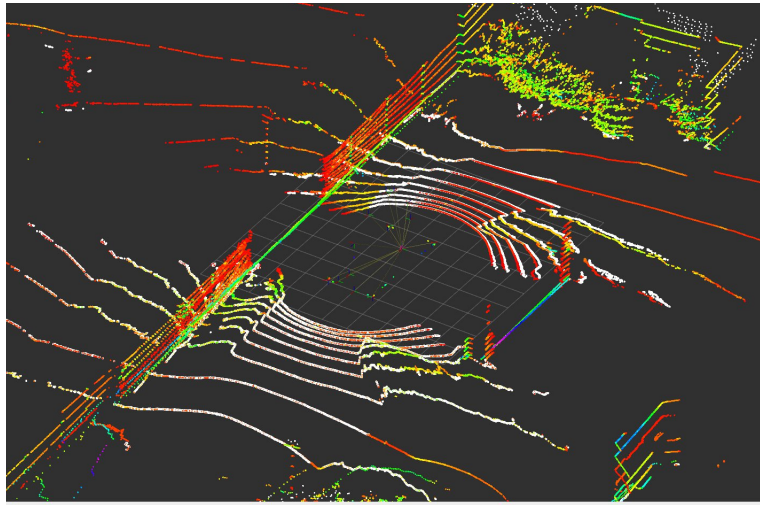
On considère comme obstacles :

- Les éléments au dessus du plan du sol
- Les éléments sur la trajectoire

Utilisation des LIDAR multinappes AV+AR pour détecter le plan du sol.

Moyens utilisés :

- Lidar du véhicules
- Système de détection du plan du sol
- Trajectoire à suivre

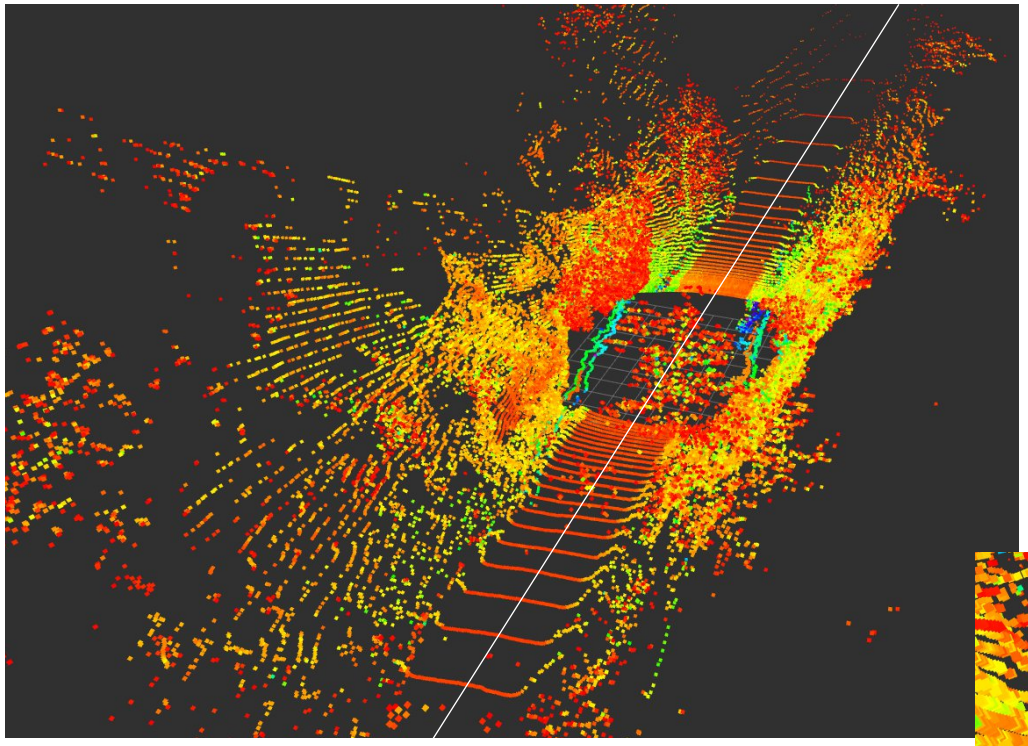


Les points blancs sont sur le sol.

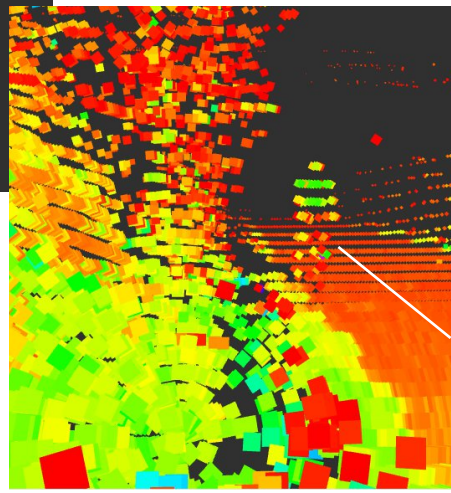
Détection d’obstacles

Scène

Obtention de données à 360° autour du véhicule à partir des 6 capteurs LIDARS.



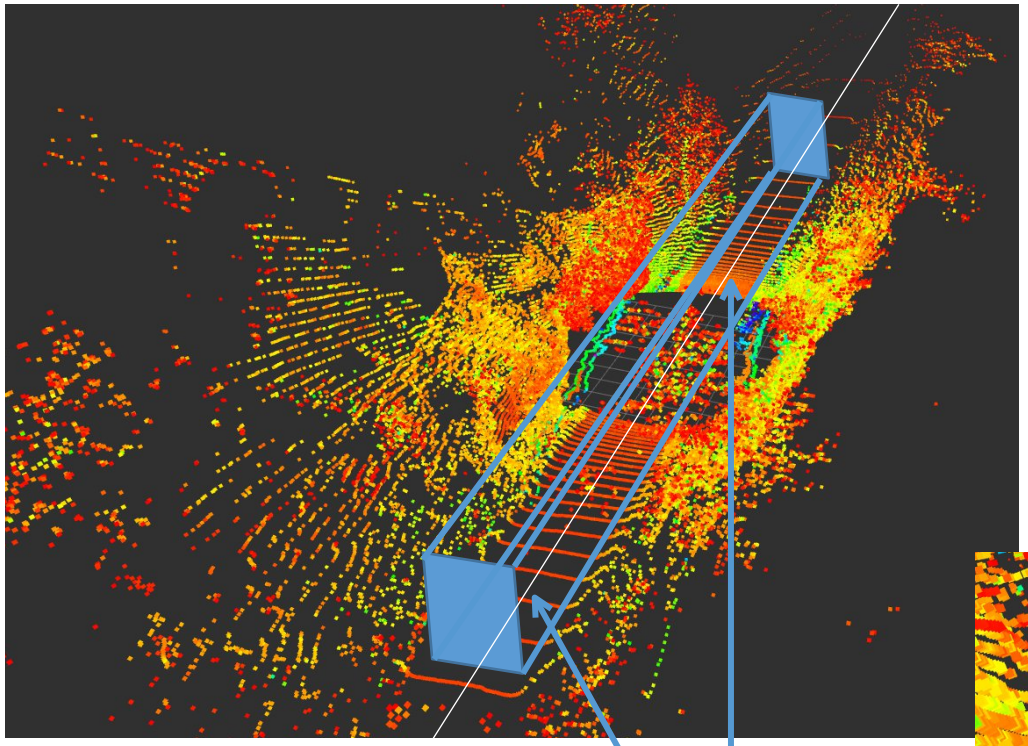
Chemin à suivre



Détection d’obstacles

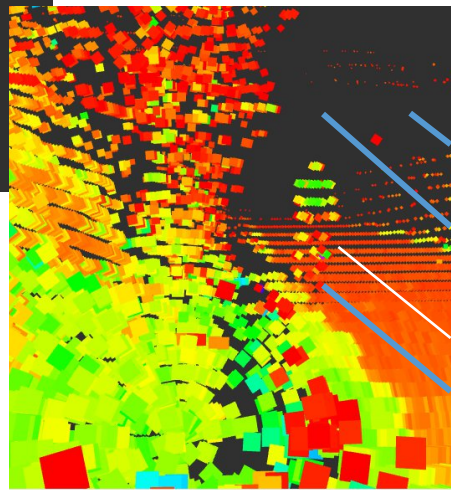
Scène

Obtention de données à 360° autour du véhicule à partir des 6 capteurs LIDARS.



Chemin à suivre

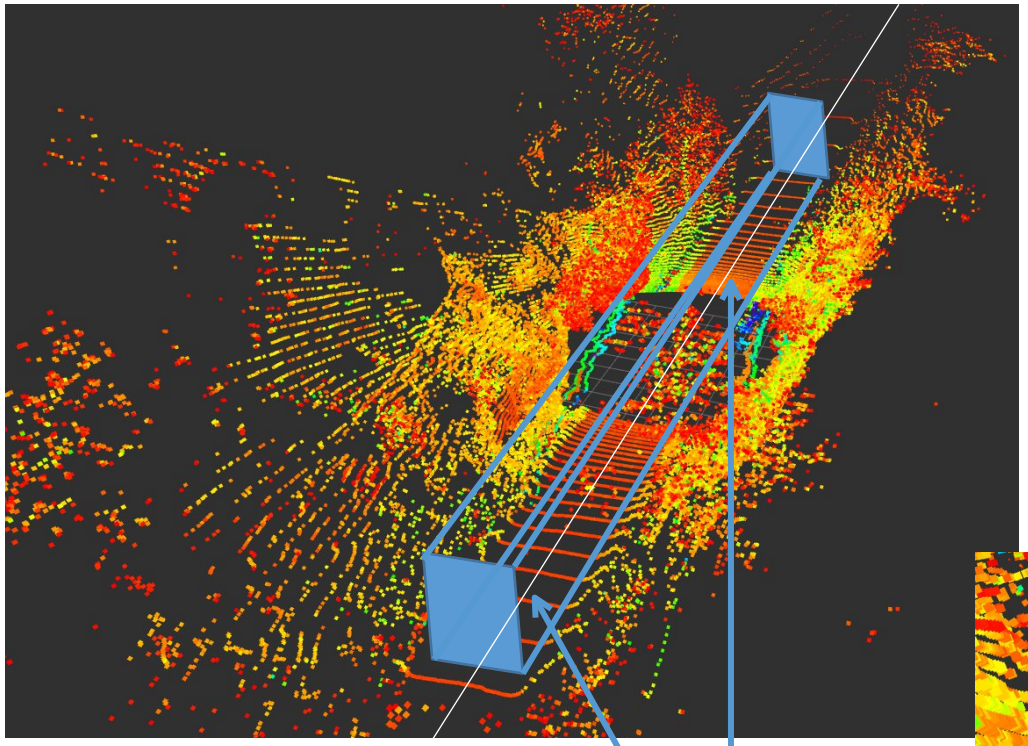
Zone où tout impact est considéré comme un obstacle



Détection d'obstacles

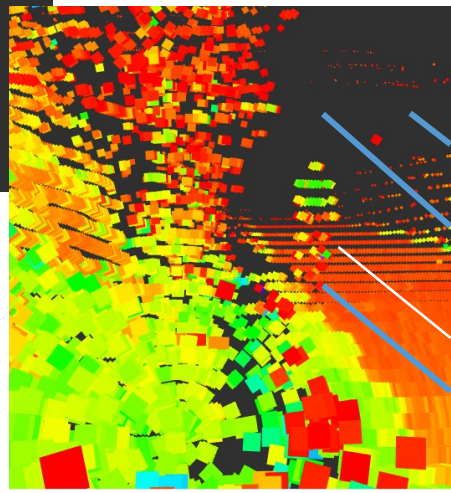
Scène

Obtention de données à 360° autour du véhicule à partir des 6 capteurs LIDARS.



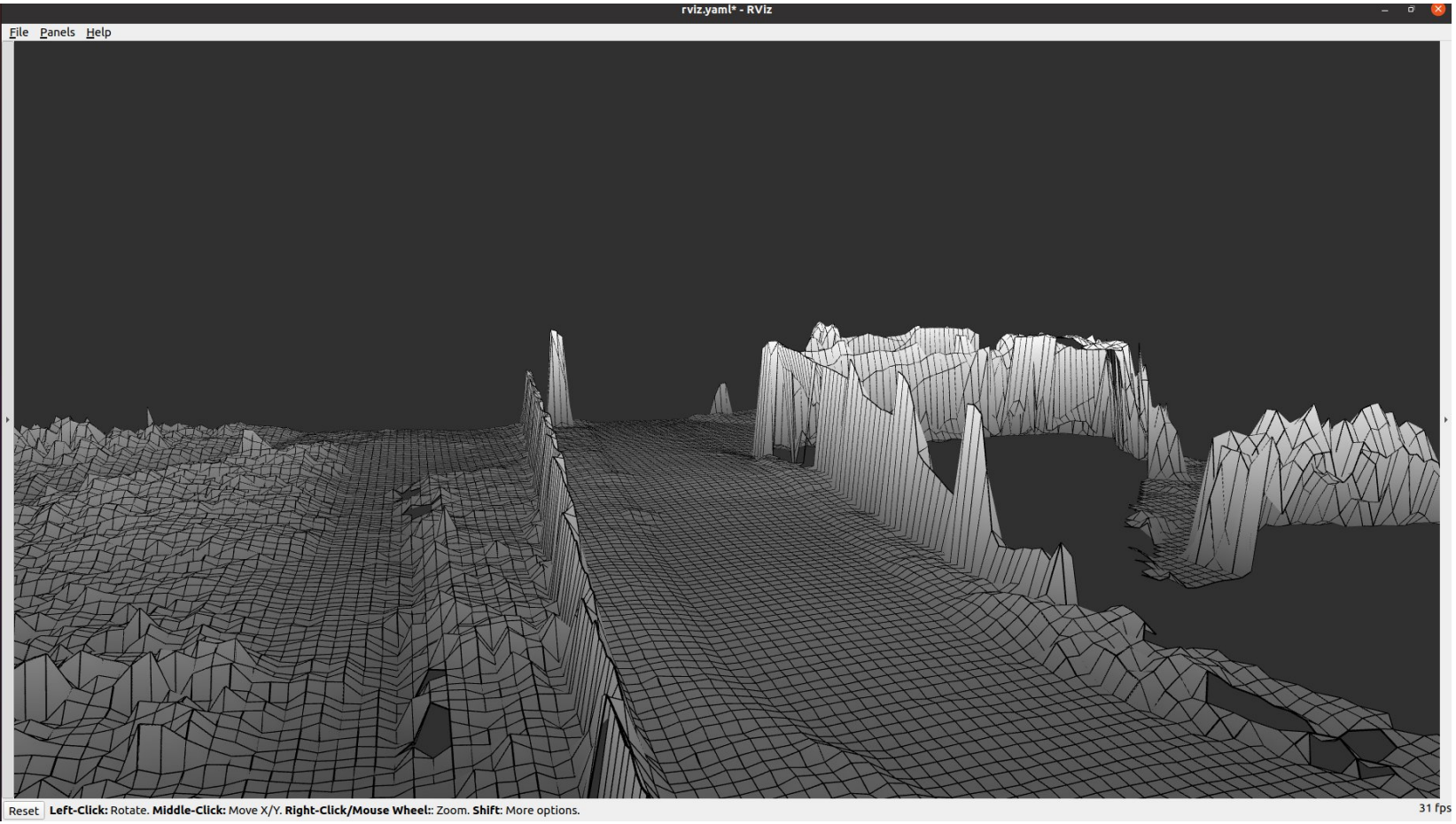
Chemin à suivre

Zone où tout impact est considéré comme un obstacle



==> Régulation de la vitesse en fonction de la distance à obstacle
==> Détecter les objets près du sol reste difficile —-> Test des gridmaps.

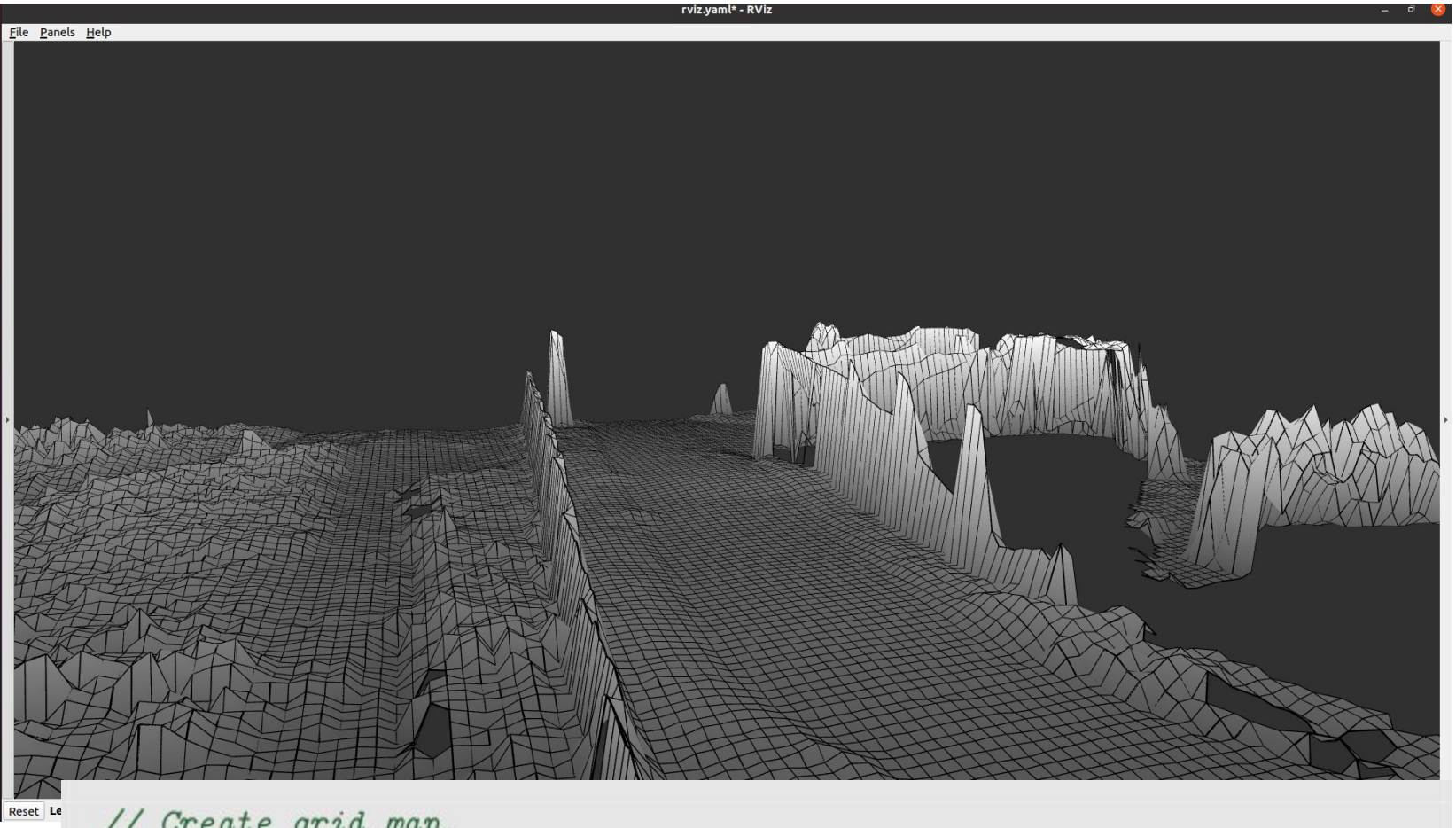
Gridmap : une bibliothèque pour construire des cartes locales d'élévation



Carte d'élévation : tableau à 2 dimensions contenant une valeur d'élévation (float)

P. Fankhauser and M. Hutter, **"A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation"**, in Robot Operating System (ROS) – The Complete Reference (Volume 1), A. Koubaa (Ed.), Springer, 2016.
(ETH Zurich (Autonomous Systems Lab & Robotic Systems Lab).

Gridmap : une bibliothèque pour construire des cartes locales d'élévation

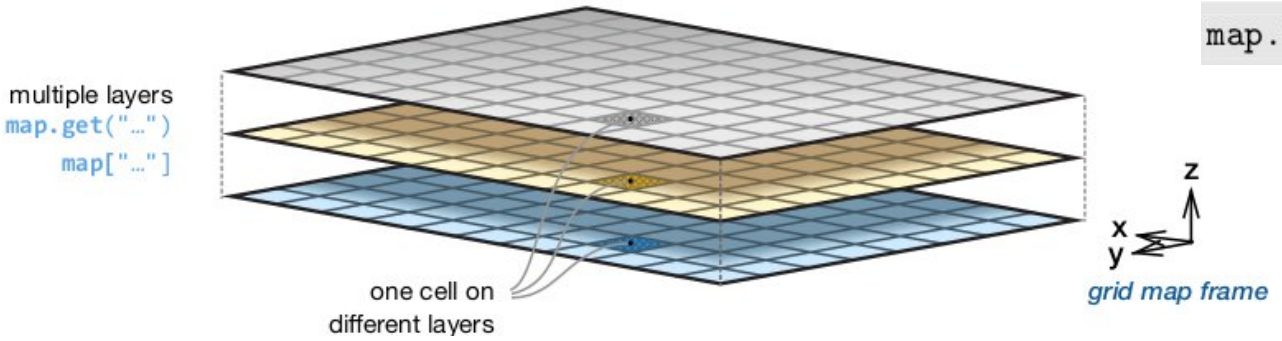



```
// Create grid map.
GridMap map({"elevation"});
map.setFrameId("map");
map.setGeometry(Length(1.2, 2.0), 0.03);
ROS_INFO("Created map with size %f x %f m (%i x %i cells).",
  map.getLength().x(), map.getLength().y(),
  map.getSize()(0), map.getSize()(1));
```

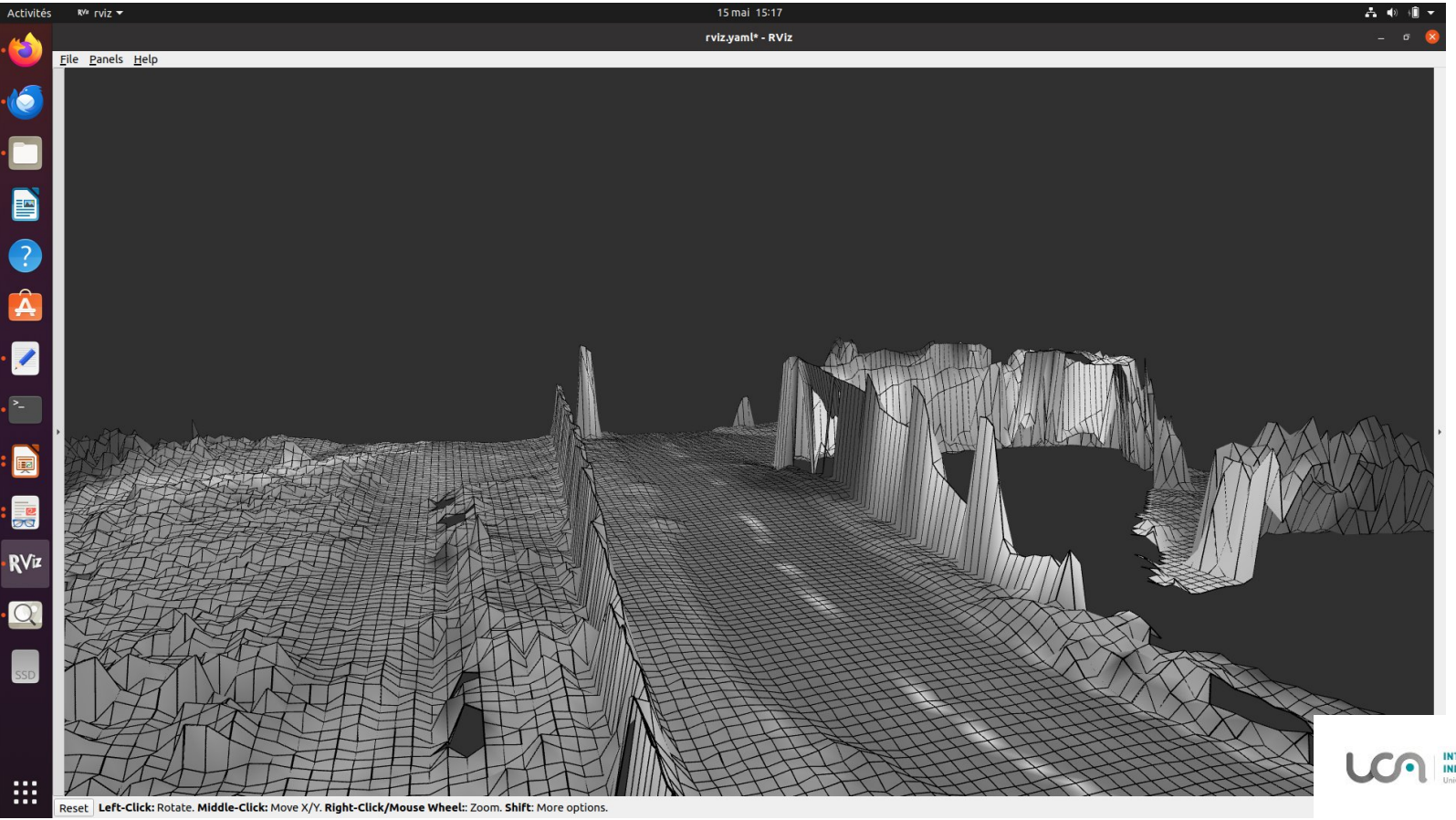
Carte d'élévation : tableau à 2 dimensions contenant une valeur d'élévation (float)

Technologie de mise en oeuvre : Eigen

Gridmap : une bibliothèque pour construire des cartes locales d'élévation




```
map.add("noise", Matrix::Random(map.getSize()(0), map.getSize()(1)));
```

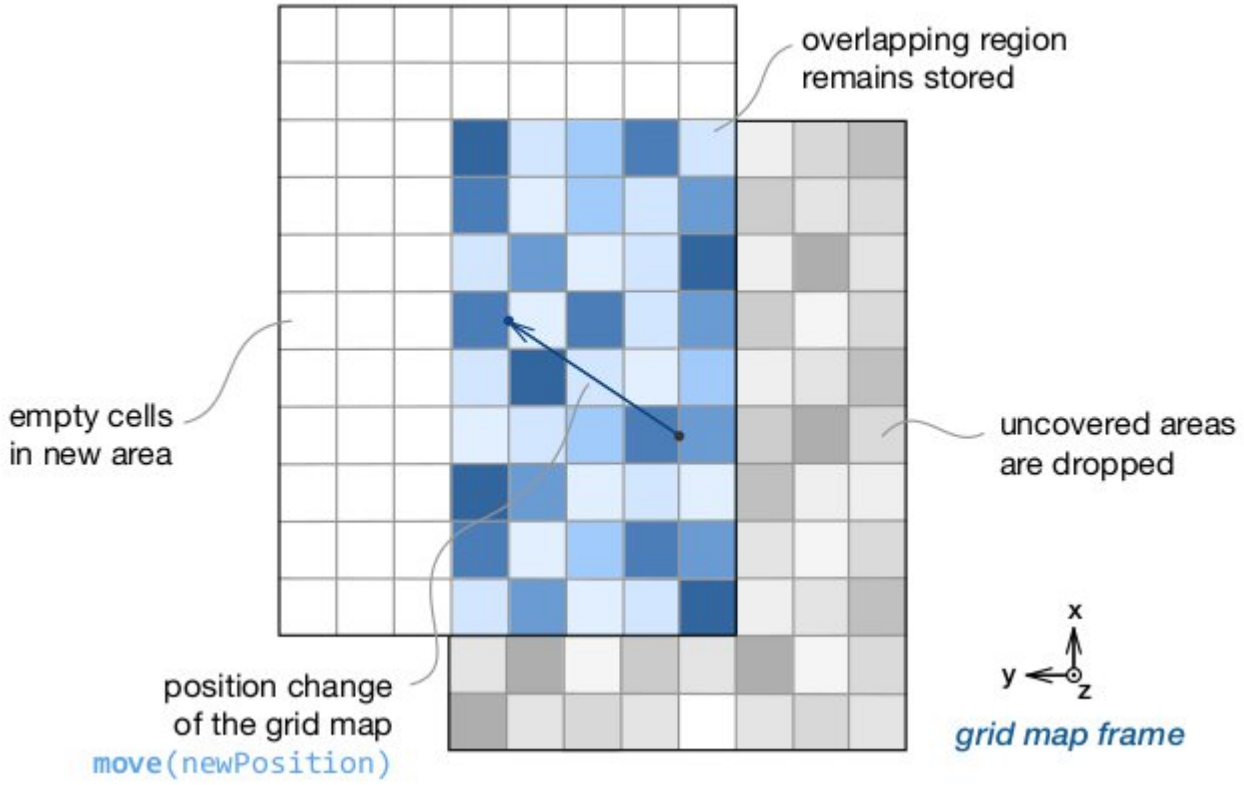


Il est possible d'ajouter des couches (layer) à la carte locale.

Avec les plugin RViz, on peut choisir les layers pour l'élévation, pour la couleur, …

—> Ici stockage de la couleur

Gridmap : une bibliothèque pour construire des cartes locales d'élévation



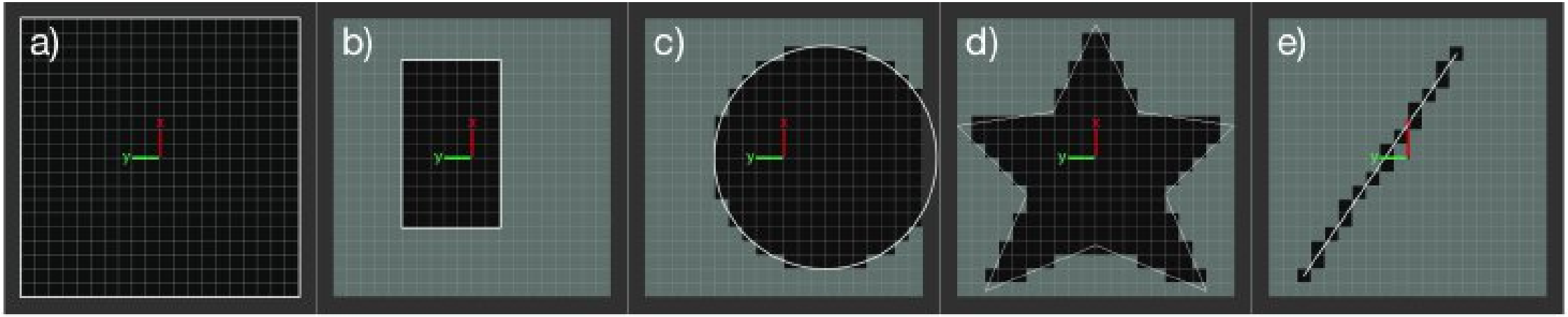
Une commande efficace et trés pratique :

```
void move(const grid_map::Position& position)
```

Il devient facile de déplacer la carte à partir de l'odométrie du robot.

Gridmap : une bibliothèque pour construire des cartes locales d'élévation

Des itérateurs personnalisables …



GridMapIterator SubmapIterator CircleIterator PolygonIterator LineIterator

+ ellipse et spirale …

```
grid_map::Matrix& data = map["layer"];
for (GridMapIterator iterator(map); !iterator.isPastEnd(); ++iterator) {
    const Index index(*iterator);
    cout << "The value at index " << index.transpose() << " is " << data(index(0), index(1)) <<
}
```

Gridmap : une bibliothèque pour construire des cartes locales d'élévation

```yaml
grid_map_filters1:

  # Fill holes in the map with inpainting.
  - name: elevation_inpaint
    type: gridMapCv/InpaintFilter
    params:
      input_layer: elevation
      output_layer: elevation_inpainted
      radius: 1.15


  # Reduce noise with a radial blurring filter.
  - name: el_mean_in_radius2
    type: gridMapFilters/MeanInRadiusFilter
    params:
      input_layer: elevation_inpainted
      output_layer: elevation_filter
      radius: 0.6


  # Test vehicle.
  - name: el_mean_in_radius
    type: gridMapFilters/MeanInRadiusFilter
    params:
      input_layer: elevation_filter
      output_layer: elevation_smooth
      radius: 1.25
```

Des filtres pour traiter les données …

Au format de la FilterChain de ROS

Conclusion

Navigation en milieu rural :

- Ces tests montrent les limites des technologies de type SLAM pour rouler dans un environnement peu structuré et dynamique.
- La relative faible portée des LIDARs fait que dans ce type d'environnement, le SLAM LIDAR peut être en difficulté également.
- Faiblesses du système de détection d'obstacle. Questionnements autour des LIDARs par rapport aux problèmes liés à la poussière et à la vapeur d'eau (brouillard, évaporation)

Gridmap :

- Une bibliothèque facile à prendre en main.
- Il est possible d'atteindre de bons résultats même avec peu de filtrage sur les données.
- Il est possible d'obtenir des performances temps réel.
- Le système de filtre est rapidement gourmand en CPU. Il n'est pas parallélisable directement même si cela semble théoriquement possible.

Pour aller plus loin en sortant (un peu) du format proposé

- pour des performances accrues, possibilité d'accéder directement à chaque couche (Eigen::MatrixXf)
- la plupart des opérations qu'on effectue sur les grilles sont facilement parallélisables avec OpenMP ou TBB (mais l'implémentation fournie dans les filtres n'est pas parallèle)
- Format float32 limitant (pas de float64, int, uint...).
- Pas de fonctions pour coder un RGB8 dans un float32 —> assez pénible à gérer.