

[4.1] Middleware : Voiture autonome

Quentin BRATEAU

24 Avril 2020



2020-04-24

[4.1] Middleware : Voiture autonome

[4.1] Middleware : Voiture autonome

Quentin BRATEAU
24 Avril 2020
 ENSTA
BRETAGNE

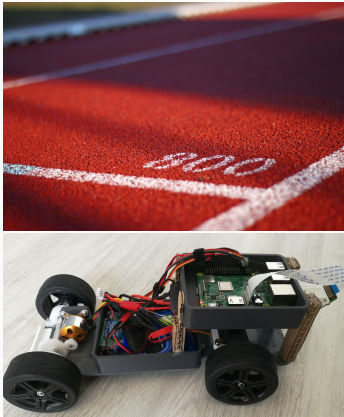


FIGURE 1 – Système actuel

Objectifs du système

- Autonomie
- Rapidité
- Suivi de ligne

Exigences

- Tour de piste
- Vitesse minimum 3 m.s^{-1}

[4.1] Middleware : Voiture autonome

└─ Objectif & Exigences

Objectif & Exigences



Objectifs du système

- Autonomie
- Rapidité
- Suivi de ligne

Exigences

- Tour de piste
- Vitesse minimum 3 m.s^{-1}

Time code : 4 :50 → 4 :20

Objectif automatiser voiture construite Atelier CNC

Réaliser Autonomie tour de piste athlétisme + rapidement possible

Startégie : Suivi d'une ligne de piste

Présenter principales exigences système

Tour de piste peu importe la géométrie de celle-ci

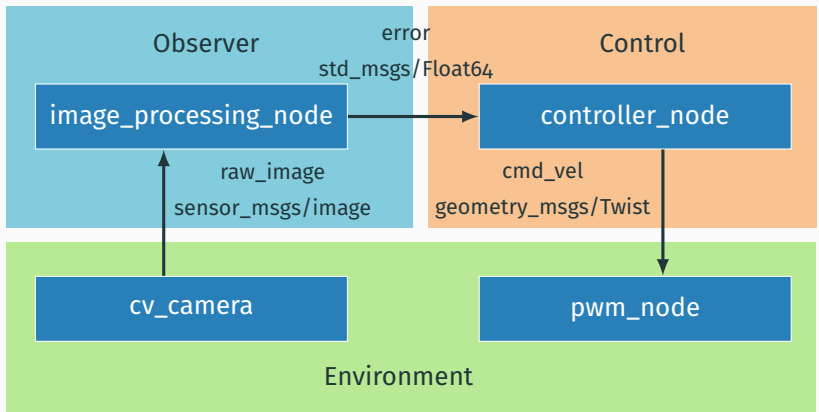
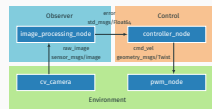


FIGURE 2 – Architecture logicielle

[4.1] Middleware : Voiture autonome

Architecture logicielle

Architecture logicielle



Time code : 4 :20 → 3 :45

Architecture classique avec ROS

Partie Environnement, Controle, Commande

Acquérir image piste node `cv_camera`

Traitement d'image node `image_processing_node`

Elaboration loi commande `controller_node`

Commande actionneurs `pwm_node`

Bouclage complet pour notre architecture logicielle



FIGURE 3 – Diagramme de l'architecture matérielle

[4.1] Middleware : Voiture autonome

└ Architecture matérielle

Architecture matérielle



Figure 3 – Diagramme de l'architecture matérielle

Time code : 3 :45 → 3 :00

3 Parties : Capteur, Traitement, Actionneurs

Capteur

3 Capteurs à disposition : Ne se sert que caméra

Traitement

Gérée Raspberry Pi 3B+

Actionneurs

Servomoteur pour direction avant

ESC controller moteur arrière donc vitesse voiture

Commande actionneurs gérée par PWM hardware du RPI

Nœud ROS

- OpenCV 4.0
- Driver *cv_camera*
- erreur $e = e_x + e_y$

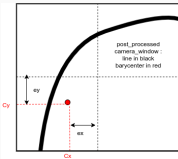


FIGURE 4 – Calcul de l'erreur

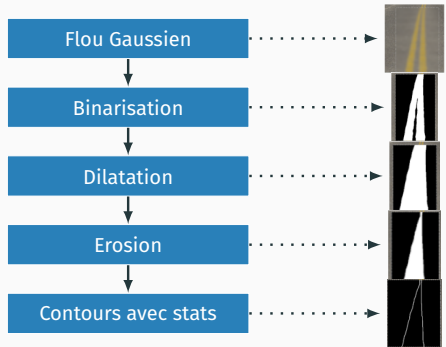


FIGURE 5 – Chaîne de traitement d'images

[4.1] Middleware : Voiture autonome

Traitement d'image

Traitement d'image

- OpenCV 4.0
- Driver *cv_camera*
- erreur $e = e_x + e_y$



FIGURE 4 – Calcul de l'erreur

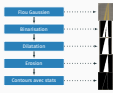


FIGURE 5 – Chaîne de traitement d'images

Time code : 3 :00 → 2 :00

Implémenté node de traitement images

Basé sur lib traitement image Opencv4.0

Récupère image *cv_camera*, node de communauté ROS

Calculer erreur à partir de différence centre image/barycentre ligne

Erreur totale = somme des erreurs suivant 2 axes de l'image

Flou Gaussien Nettoyer image bruit

Binarisation Détacher ligne du fond de l'image

Dilatation/Erosion Refermer ligne + supprimer erreurs binarisation

Contours avec stats Obtenir le barycentre de ligne

Commande actionneurs gérée par PWM hardware du RPI

Avantage ROS

- Réel \leftrightarrow Simulé
- Même topics
- Test des autres nœuds

V-REP

Modélisation en 3 couches

- Inertielle
- Collisions
- Design

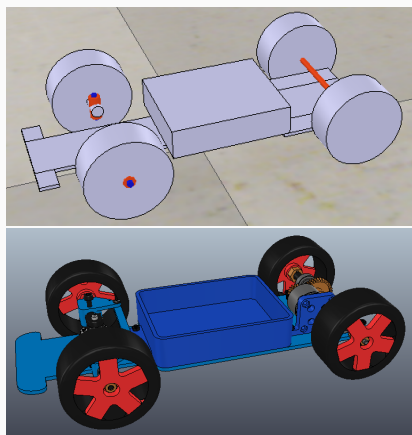


FIGURE 6 – Modélisation dans V-REP

[4.1] Middleware : Voiture autonome

└─ Simulation

Time code : 2 :00 \rightarrow 1 :15

Avantage ROS ...

Utiliser Simulateur V-REP

Interfaceable avec ROS via script en LUA

Modélisation 3 couches :

Partie Calculs : Inertielle + Collision

Modélisée par pavés droits et cylindre

Simplifier grandement calculs \rightarrow Simu + fluide

Masquée affichage par suite

Permet définir comportement voiture

Partie Affichage : Design

Import des fichier CAO sous forme Mesh

Permet affichage esthétique voiture

Simulation

Avantage ROS

- Réel \leftrightarrow Simulé
- Même topics
- Test des autres nœuds

V-REP

Modélisation en 3 couches

- Inertielle
- Collisions
- Design

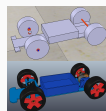


FIGURE 6 – Modélisation dans V-REP

Système Réel

- Boucle caméra fonctionnelle
- Manque d'outils
- Manque de matériel

Système Simulé

- Parfaitement fonctionnel
- Vitesse atteinte 6 m.s^{-1}
- Conditions optimales

Projet

- Architecture ROS sur système réel
- Développement avec GitHub
- Méthode AGILE adaptée à structure ROS

[4.1] Middleware : Voiture autonome

— Résultats & Conclusion

Résultats & Conclusion

Système Réel

- Boucle caméra fonctionnelle
- Manque d'outils
- Manque de matériel

Projet

- Architecture ROS sur système réel
- Développement avec GitHub
- Méthode AGILE adaptée à structure ROS

Système Simulé

- Parfaitement fonctionnel
- Vitesse atteinte 6 m.s^{-1}
- Conditions optimales

Time code : 1 :15 → 0 :00

Système réel

Boucle Caméra fonctionnelle mais pas testée en conditions réelles

Manque outils : Imprimante 3D → support camera

Manque matériel : GPS et Centrale inertielle

Système simulé ...

Projet en général ...

[4.1] Middleware : Voiture autonome

Quentin BRATEAU

24 Avril 2020



[4.1] Middleware : Voiture autonome

2020-04-24

[4.1] Middleware : Voiture autonome

