Configuration d'environnement

***

RosBridgeWeb + noeud AutoCtrl et DockCtrl

Avant connecter le client web:

Pour utiliser normalement les fonctionnalités du client web, il est nécessaire de s'assurer que les fichiers exécutables **dock_bridge_node.py** et **auto_mode_controller.py** existent dans le répertoire home du robot.

Le nœud de contrôle du mode automatique a pour rôle de démarrer et d'arrêter les deux fichiers py liés au mode automatique selon les instructions du client. Par conséquent, il faut également vérifier que **detector.py** et **follow_waypoint.py** existent et disposent des permissions d'exécution. **waypoint.py** est utilisé pour enregistrer les points de navigation dans un fichier txt, plus de détails seront donnés plus tard.

```
ubuntu@turtlebot4:~$ ll
total 236
drwxr-x--- 9 ubuntu ubuntu  4096 Apr 29 16:31 ./
drwxr-xr-x 3 root   root    4096 Oct  4  2024 ../
-rw------- 1 ubuntu ubuntu    56 Apr 16 15:32 .Xauthority
-rw------- 1 ubuntu ubuntu 32375 Apr 29 14:46 .bash_history
-rw-r--r-- 1 ubuntu ubuntu   220 Mar 31  2024 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu  3967 Apr 24 16:01 .bashrc
drwx------ 3 ubuntu ubuntu  4096 Apr 18 13:11 .cache/
-rw------- 1 ubuntu ubuntu    20 Apr 18 11:58 .lesshst
drwxrwxr-x 3 ubuntu ubuntu  4096 Oct  4  2024 .local/
-rw-r--r-- 1 ubuntu ubuntu   807 Mar 31  2024 .profile
drwxrwxr-x 4 ubuntu ubuntu  4096 Oct  4  2024 .ros/
drwx------ 2 ubuntu ubuntu  4096 Apr 16 15:32 .ssh/
-rw-r--r-- 1 ubuntu ubuntu     0 Oct  4  2024 .sudo_as_admin_successful
-rwxrwxrwx 1 ubuntu ubuntu  5816 Apr 29 16:31 auto_mode_controller.py*
-rwxr-xr-x 1 ubuntu ubuntu  2268 Apr 25 15:35 detector.py*
-rwxrwxrwx 1 ubuntu ubuntu  6984 Apr 22 13:11 dock_bridge_node.py*
-rwxr-xr-x 1 ubuntu ubuntu  9113 Apr 25 15:59 follow_waypoint.py*
-rw-rw-r-- 1 ubuntu ubuntu  9108 Apr 18 13:11 frames_2025-04-18_13.11.57.gv
-rw-rw-r-- 1 ubuntu ubuntu 24291 Apr 18 13:11 frames_2025-04-18_13.11.57.pd
f
-rw-rw-r-- 1 ubuntu ubuntu 26152 Apr 29 16:33 image_capture_point_1.jpg
-rw-rw-r-- 1 ubuntu ubuntu 31336 Apr 29 16:33 image_capture_point_2.jpg
-rwxrwxr-x 1 ubuntu ubuntu   236 Apr 25 16:08 mode_auto.sh*
drwxrwxr-x 6 ubuntu ubuntu  4096 Apr 24 14:19 ros2_ws/
-rw-rw-r-- 1 ubuntu ubuntu   862 Apr 18 12:48 save_image.py
drwxrwxr-x 5 ubuntu ubuntu  4096 Feb 11 11:40 turtlebot4_test_results/
drwxrwxr-x 6 ubuntu ubuntu  4096 Jan 31 17:30 turtlebot4_ws/
-rwxrwxr-x 1 ubuntu ubuntu  2144 Apr 25 16:01 waypoint.py*
-rw-rw-r-- 1 ubuntu ubuntu   120 Apr 29 16:29 waypoints.txt
```

Chaque fois que le robot est allumé(5 leds vert && batterie affichage != 0), il est nécessaire d'exécuter les instructions suivantes pour garantir le démarrage des fonctions correspondantes :

Caméra démarrage avant lancer serveur:

lancer camera:

```
ros2 launch turtlebot4_bringup oakd.launch.py
```

```
[INFO] [launch_ros.actions.load_composable_nodes]: Loaded node '/oakd' in container '/oakd_container'
```

CTRL+C

ou nouveau terminal

compressé le topic qui contient l'image en format JPEG/PNG. :

```
ros2 run image_transport republish raw compressed \
  --ros-args \
  --remap in:=/oakd/rgb/preview/image_raw \
  --remap out/compressed:=/oakd/rgb/preview/image_raw/compressed
```

```
[INFO] [1745320096.432148239] [image_republisher]: The 'out_transport' parameter is set to:
```

CTRL+C

ou nouveau terminal

lancer serveur:

```
ros2 launch rosbridge_server rosbridge_websocket_launch.xml
```

```
[rosbridge_websocket-1] [INFO] [1745320138.946992712] [rosbridge_websocket]: Rosbridge WebSocket server started on port 9090
```

lancer noeuds py:

nouveau terminal

```
./dock_bridge_node.py
```

```
ubuntu@turtlebot4:~$ ./dock_bridge_node.py
[INFO] [1745930372.993699691] [dock_bridge_node]: Dock bridge node started
[INFO] [1745930496.522488667] [dock_bridge_node]: Received dock request
[INFO] [1745930496.529312173] [dock_bridge_node]: Waiting for dock action s
erver...
[INFO] [1745930496.534507177] [dock_bridge_node]: Sending dock goal...
[INFO] [1745930496.551085752] [dock_bridge_node]: Dock goal accepted, waiti
ng for result...
```

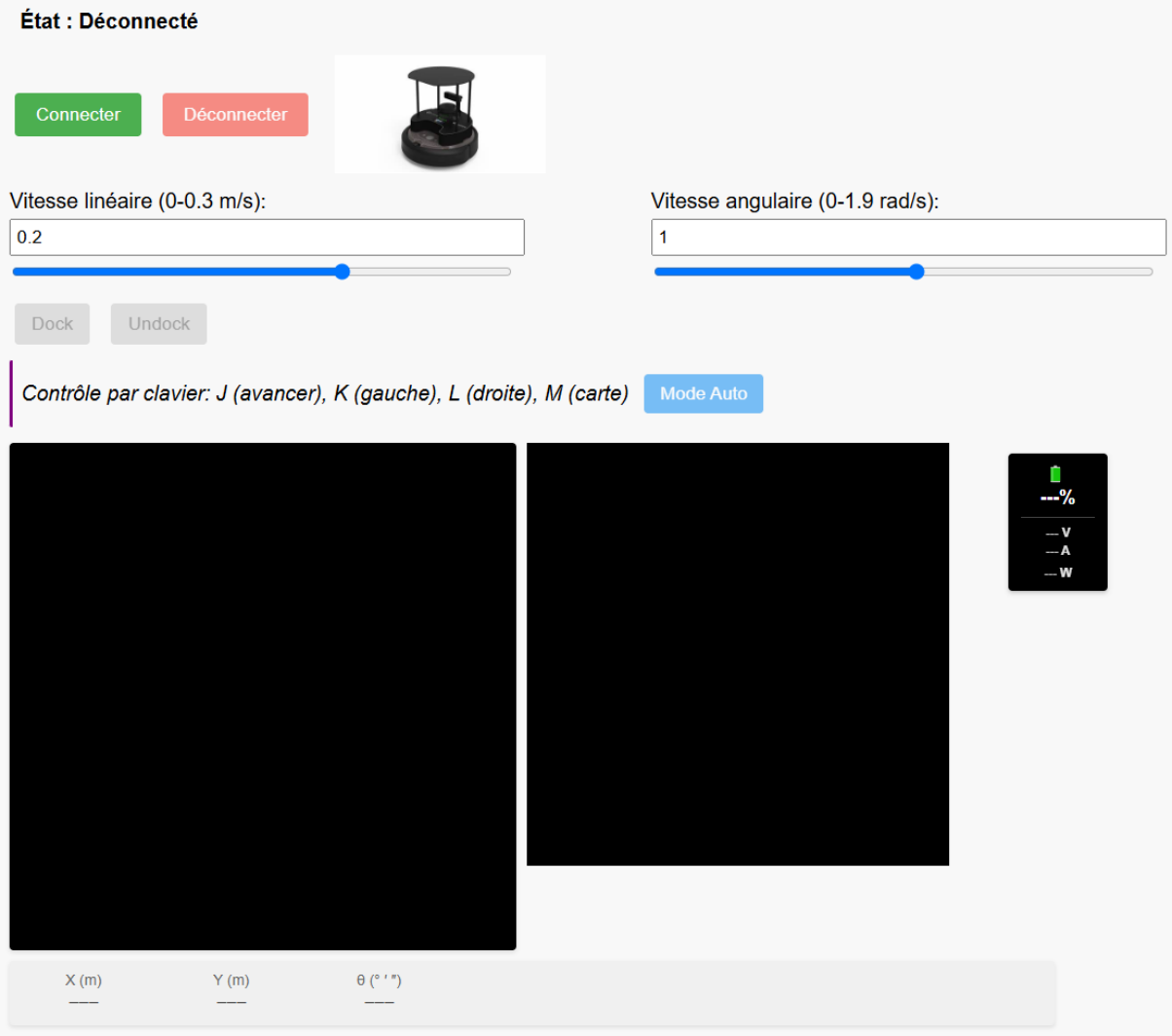
nouveau terminal

```
./auto_mode_controller.py
```

```
ubuntu@turtlebot4:~$ ./auto_mode_controller.py
[INFO] [1745937118.555860520] [mode_auto_node]: 🟢 Contrôleur de mode auto
prêt
[INFO] [1745937123.161710422] [mode_auto_node]: Démarrage du mode autonome.
..
[INFO] [1745937123.221942416] [mode_auto_node]: 🚀 Mode automatique démarré
 avec succès
[INFO] [1745937130.314978441] [mode_auto_node]: Robot détaché du dock penda
nt le mode auto
[INFO] [1745937173.319917471] [mode_auto_node]: Robot revenu au dock, fin d
```

Utilisation du client web:

Télécharger **Robot_SB_client_v2.0.5.7z** et décompresser-le, puis exécuter **main.html** pour lancer le client:



Certains conteneurs de l'interface ont des dimensions fixes. Ainsi, lorsque la taille du navigateur est grande, cela peut entraîner un positionnement ou une l argeur anormale des éléments d'interface comme le contrôle de vitesse (manuel) et l'affichage de l'état de la batterie. Cependant, cela n'affecte pas les fonctionna lités et il suffit de réduire la taille de la page pour revenir à la normale.

L'affichage par défaut montre le nuage de points du lidar. Pour afficher la carte, appuyez sur la touche **M**.

La carte par défaut est celle de la salle blanche. Si vous souhaitez utiliser la carte de la salle 4105, ajoutez *?test* à la fin de l'URL puis appuyez sur Entrée.

Vous pouvez voir les logs d'initialisation correspondants dans la console (F12). Lorsqu'une connexion est établie, les topics disponibles du robot s'affichent dans la console. En cas d'erreur, celle-ci y sera également affichée.

Mode manuel et réglage de la vitesse:

Dans la page, le réglage de la vitesse affecte uniquement la vitesse en mode manuel et n'a aucun lien avec les modes semi-automatique et automatique.

Selon les instructions affichées sur la page, utilisez les touches J, K et L pour contrôler l'avance et la rotation latérale. Si les touches d'avance et de virage sont enfoncées simultanément, le robot appliquera une combinaison de vitesse linéaire et de vitesse angulaire selon des règles prédéfinies, ce qui permet de modifier son orientation tout en avançant.

Le mode manuel a la priorité la plus élevée : appuyer sur J, K ou L interrompt immédiatement le mode semi-automatique ou automatique.

Carte : Calibration et mode semi-automatique

Pour relier le système de coordonnées du robot à celui de la page web (carte), il est nécessaire de conduire manuellement le robot vers trois points non alignés, puis de cliquer sur les positions correspondantes sur la carte (il est recommandé d'utiliser les coins de mur comme points de référence, car ces formes particulières sont facilement identifiables dans le nuage de points du lidar). Une fois le troisième point calibré, un point rouge représentant la position du robot apparaîtra sur la carte et sera mis à jour toutes les 0,5 secondes.

Si un recalibrage est nécessaire, cliquez à nouveau sur le bouton de calibration.

Après la calibration, vous pouvez cliquer sur la carte pour placer un point cible bleu et démarrer le mode semi-automatique. Dans ce mode, si un obstacle est détecté dans un rayon de 90 degrés devant le robot et à moins de 0,5 mètre (distance mesurée par le lidar), le robot s'arrêtera et ne pourra que pivoter pour s'orienter vers la cible. Le point cible bleu se mettra alors à clignoter. Si l'obstacle

disparaît et que le robot n'a pas encore atteint la cible, il reprendra son déplacement vers celle-ci.

Bouton de mode automatique

Le mode automatique ne peut être activé que lorsque le robot est sur sa station de charge. Lorsque le robot a terminé la surveillance de tous les points et retourne à la station, le bouton se réinitialise automatiquement pour permettre un clic et démarrer un nouveau cycle en mode automatique.

Pendant le fonctionnement en mode automatique, le mode semi-automatique n'est pas disponible. Le mode automatique peut être arrêté en cliquant manuellement sur le bouton d'arrêt ou en appuyant sur l'une des touches J, K ou L.

(**Évitez de déplacer la station de charge après l'allumage du robot, car elle sert de point (0, 0).**)

Mode Auto (IA “rouge”, waypoints)

Configuration et utilisation du système de détection de lumière rouge avec envoi d’alerte par email

Afin de répondre aux exigences fonctionnelles du cahier des charges imposant une pause du robot pour la capture et l’analyse d’image dans un environnement tel qu’une salle blanche, nous avons mis en place un système de détection automatique de lumière rouge couplé à un mécanisme d’alerte par email. Ce système repose sur l’abonnement du robot à un flux vidéo publié par une caméra RGB connectée au TurtleBot4, l’analyse de cette image en temps réel, et l’envoi d’un email en cas de détection significative de pixels rouges. L’ensemble de cette chaîne a été développé sous ROS 2 Jazzy avec des scripts Python s’exécutant dans un environnement Ubuntu 22.04/24.04.

La première étape de la configuration consiste à s’assurer que le système dispose de tous les prérequis logiciels. Il est indispensable que ROS 2 Jazzy soit installé et fonctionnel, que le topic /oakd/rgb/preview/image_raw soit accessible (ou un autre topic image en fonction du matériel), et que les dépendances Python suivantes soient installées : rclpy, cv_bridge, opencv-python, ainsi que smtplib et email pour la gestion des courriers électroniques. L’installation peut se faire via les commandes suivantes :

```
sudo apt install python3-opencv python3-pip
pip install opencv-python
```

Concernant l’envoi d’email, l’intégration avec Gmail nécessite une configuration préalable sur le compte. Il faut activer la vérification en deux étapes, puis générer un mot de passe d’application depuis la section “Sécurité” de l’interface Gmail. Ce mot de passe, à usage unique, remplace le mot de passe classique pour les connexions SMTP sécurisées. Dans notre cas, nous avons utilisé un compte dédié turtlebotsalleblanche@gmail.com, avec un mot de passe d’application du type yqik xquz axlq gcyt, utilisé dans le script Python pour établir une connexion sécurisée via le serveur smtp.gmail.com sur le port 465 (SSL).

Le script Python implémenté est structuré autour d’un nœud ROS RedLightEmailDetector qui s’abonne au topic image et traite la première image reçue. Une fois l’image capturée et convertie en format OpenCV, elle est convertie de l’espace de couleurs BGR vers HSV. Cette conversion permet une détection plus robuste de la teinte rouge. Deux plages sont définies pour couvrir les deux extrémités du spectre HSV correspondant au rouge (environ 0–10° et 170–180°). Des masques binaires sont

ensuite appliqués pour isoler les pixels rouges. Le ratio de pixels rouges par rap port au total est évalué : s' il dépasse 1 %, le système considère que de la lumiè re rouge est présente.

Dans ce cas, une alerte est déclenchée : un email est préparé avec un objet et un corps de message, et l' image capturée est ajoutée en pièce jointe au format JPEG. L' envoi est effectué via smtplib.SMTP_SSL, en se connectant au serveur de Gmail. Si aucun rouge n' est détecté, le programme se termine sans action supplémentaire.

Voici un extrait du code utilisé pour envoyer l' email :

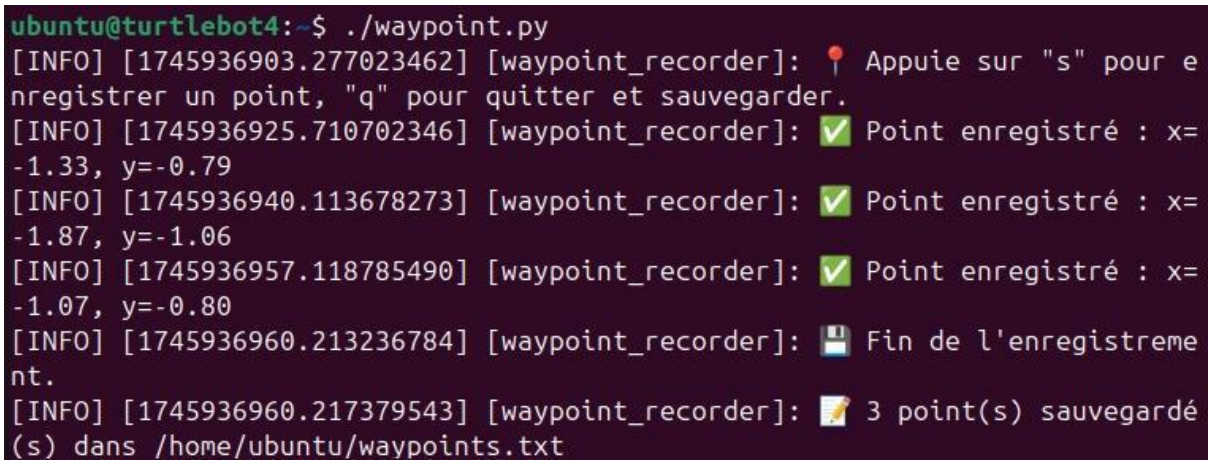
```
msg = EmailMessage()
msg.set_content("🔴 Lumière rouge détectée dans l'image.")
msg['Subject'] = 'Alerte : couleur rouge détectée'
msg['From'] = "turtlebotsalleblanche@gmail.com"
msg['To'] = "yaraahmadsaid@gmail.com"
with open("image_capture.jpg", "rb") as f:
    img_data = f.read()
    msg.add_attachment(img_data, maintype='image', subtype='jpeg', filename='image
_capture.jpg')

with smtplib.SMTP_SSL('smtp.gmail.com', 465) as server:
    server.login("turtlebotsalleblanche@gmail.com", "mot_de_passe_application")
    server.send_message(msg)
```

Si une lumière rouge est détectée, l' email est automatiquement envoyé avec la cap ture en pièce jointe. Si ce n' est pas le cas, le programme s' arrête après avoir enregistré l' image.

En cas de problème de connexion SMTP ou de non-réception de l' image, plusieurs vé rifications peuvent être effectuées : la validité du mot de passe d' application, la connectivité réseau, la disponibilité du topic image, et la validité de l' adre sse email destinataire.

En résumé, ce module autonome de surveillance visuelle par analyse d' image s' int ègre parfaitement au comportement du robot dans un environnement contrôlé, et assu re une réactivité immédiate en cas d' anomalie visuelle détectée, répondant ainsi avec précision aux exigences du cahier des charges.

Navigation automatique du robot



Ce guide vous permettra d'utiliser le robot pour suivre un trajet défini automatiquement, capturer des images à certaines positions et retourner à sa station de charge.

Avant de commencer, assurez-vous que :

- vous êtes bien connecté au robot.
- Les scripts suivants sont disponibles dans un répertoire : waypoint.py (pour l'enregistrement) et follow_waypoint.py (pour la navigation automatique)

1. Enregistrement des points de passage

Lancez la téléopération manuelle :

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard --ros-args -p stamped:=true
```

Avant de lancer le script d'enregistrement, dans un autre terminal, faites un Undock pour faire sortir le robot de sa station de charge:

ros2 action send_goal /undock irobot_create_msgs/action/Undock "{}" --feedback

Ensuite, placez vous dans le répertoire contenant les fichiers avec la commande cd puis lancez le script d'enregistrement :

python3 waypoint.py

Pendant que vous pilotez le robot(i= avancer, l= tourner à droite, j= tourner à gauche) ,appuyez sur la touche s de votre clavier pour enregistrer un point et appuyez sur q pour arrêter l'enregistrement.

Les points sont enregistrés dans le fichier texte waypoints.txt. que vous trouverez dans le même répertoire.

PS:
Pour choisir quels points enregistrer : essayez par exemple de prendre des points aux intersections, aux points de contrôle (où une analyse de photo doit être faite).

2. Lancement de la navigation autonome

Une fois les points enregistrés, ouvrez le script follow_waypoint.py et modifiez le code de la méthode move_to_waypoints() à la ligne:

```
if idx in [0, 1] and idx not in self.checked_points:
```

Ici 2 captures sont prises et analysées au 1er et au 2e point enregistré correspondant respectivement aux indices 0 et 1 de la liste complète des points. Changez les indices 0, 1 de [0,1] selon les positions sélectionnées pour les détections de lumière rouge.

Après que vous vous êtes placé dans le bon répertoire lancez le script de navigation :

python3 follow_waypoint.py

Le robot :

- Se détache automatiquement de la station de charge (undock).
- Suit les points enregistrés dans waypoints.txt.
- Capture et analyse une photo aux points sélectionnés (modifiable dans le code).
- Retourne à sa station de charge (dock) à la fin du trajet.