CESI LINEACT

1. GENERATION DES DATASETS POUR LES ROTATIONS

OBJECTIF

Le but de cette génération de dataset est de simuler une série de rotations spécifiques appliquées à un objet Aruco dans un environnement Gazebo. Ces rotations couvrent une plage d'orientations en roll, pitch, et yaw. En enregistrant les données de pose de chaque orientation, on crée un dataset qui peut sera utilisé pour évaluer la précision de la détection d'un marqueur Aruco et plus précisément les rotations de celui-ci

ÉTAPES DE GENERATION DU DATASET

1. Initialisation de ROS et des Services Gazebo :
    - Le script initialise un nœud ROS avec le nom spawn_box.
    - Un éditeur (Publisher) est configuré pour publier la pose de l'objet Aruco dans un topic nommé /aruco_pose.
    - Le service de Gazebo pour l'ajout d'objets est attendu et activé.
2. Chargement du Modèle Aruco :
    - Le chemin du modèle Aruco est déterminé en utilisant le package tiago_description et le fichier aruco.sdf.
    - Le modèle est chargé sous forme de chaîne XML, prêt à être ajouté dans la scène Gazebo.
3. Définition de la Liste de Rotations :
    - Une liste de rotations (aruco_rota_list) est définie avec des combinaisons spécifiques de rotations en roll, pitch et yaw.
    - Cette liste couvre une large gamme d'angles pour tester différentes orientations, par exemple : roll de 30°, pitch de 45°, ou yaw de -90°.
4. Boucle de Génération des Rotations :
    - Le script itère sur chaque combinaison de rotations dans aruco_rota_list.
    - Pour chaque rotation :
        - Une transformation est obtenue pour aligner le cadre xtion_rgb_optical_frame dans map, assurant ainsi une pose correcte dans l'espace.
        - La rotation est appliquée en combinant des matrices de transformation et en utilisant des quaternions pour représenter la rotation finale.
        - Le modèle Aruco est ajouté à Gazebo avec la pose définie.
        - La pose de l'objet est publiée dans le topic ROS /aruco_pose.
        - Après une courte pause, l'objet est supprimé pour permettre l'insertion de la prochaine orientation.

2. LECTURE ET ENREGISTREMENT DES DONNEES DU DATASET

OBJECTIF

L'objectif de ce script est de lire les données simulées publiées par ROS, de capturer des images et d'enregistrer les informations de pose de l'objet Aruco. En procédant ainsi, on obtient un dataset complet comprenant les images et les coordonnées de pose.

DETAILS DU FONCTIONNEMENT

1. Initialisation des Paramètres et Création de la Classe RecordFromTopic :
    - La classe RecordFromTopic est responsable de la gestion des abonnements aux topics ROS, de la capture d'images et de la sauvegarde des poses dans un fichier CSV.
    - Les paramètres de ligne de commande permettent de configurer le topic d'image, le topic d'informations de la caméra, le topic de pose d'Aruco, le taux de capture d'image (image_rate), le délai avant timeout (timeout_duration), le fichier CSV de sortie pour les poses et le dossier de sauvegarde des images.
2. Initialisation et Préparation des Fichiers de Sortie :
    - Le dossier d'images est nettoyé et recréé pour garantir que les anciennes données ne persistent pas.
    - Le fichier CSV est initialisé avec des en-têtes (time, tx, ty, tz, rx, ry, rz) pour sauvegarder les translations (tx, ty, tz) et rotations (rx, ry, rz) de l'objet Aruco.
3. Récupération des Images et Informations de Pose :
    - Topic d'image : Le script s'abonne au topic d'image, convertit chaque message d'image ROS en format OpenCV via CvBridge, puis enregistre l'image périodiquement (toutes les image_rate images) dans le dossier de sortie.
    - Topic des informations de la caméra : Les informations intrinsèques de la caméra (matrice de calibration K et coefficients de distorsion D) sont enregistrées dans un fichier JSON.
    - Topic de pose d'Aruco : Le script s'abonne également au topic de pose d'Aruco pour capturer les coordonnées de translation et de rotation. Ces informations sont stockées en temps réel dans le fichier CSV.
4. Enregistrement dans le CSV et Gestion du Timeout :
    - Le script enregistre les informations de pose dans le fichier CSV, associant chaque image à sa pose correspondante (positions tx, ty, tz et rotations rx, ry, rz).
    - Un mécanisme de timeout surveille l'arrivée des messages ; si aucun message n'est reçu pendant la durée spécifiée, le script déclenche un arrêt automatique.
5. Exécution et Options du Script :

- Le script peut être exécuté soit directement en enregistrant les messages en temps réel, soit en lecture d'un fichier rosbag contenant des enregistrements ROS.

3. DETECTION DES TAGS ARUCO ET ENREGISTREMENT DES POSES

OBJECTIF

L'objectif de cette étape est d'identifier les tags Aruco dans les images, de déterminer leur pose (position et rotation) dans le repère de la caméra et d'enregistrer ces informations dans un fichier CSV. Ces données permettent de comprendre l'alignement du tag par rapport à la caméra et facilitent l'évaluation des performances des algorithmes de détection.

DETAILS DU FONCTIONNEMENT

1. Définition des Paramètres et Initialisation de la Classe de Détection :
    - J'utilise TagPoseProvider pour gérer la détection des tags dans les images. Cette classe utilise un fichier de calibration pour configurer les paramètres intrinsèques de la caméra.
    - Les paramètres sont définis via des arguments de ligne de commande, incluant le dossier d'images, le fichier de calibration de la caméra, l'identifiant du tag à détecter, et le nom du fichier CSV pour sauvegarder les poses.
2. Configuration des Repères et Transformation :
    - Le script configure les repères du gripper (end effector du robot) et du tag pour calculer la transformation entre eux. Cette transformation est définie par la fonction transformation_matrix, qui génère une matrice de rotation entre les deux repères.
    - Les points et rotations sont ensuite transformés à l'aide des matrices de transformation calculées, afin de représenter la pose du tag dans le repère du gripper.
3. Détection et Enregistrement des Poses :
    - Pour chaque image dans le dossier spécifié, le script utilise la méthode detect_marker pour détecter les tags Aruco.
    - Si le tag spécifié est détecté, le script calcule sa pose .
    - La pose est ensuite sauvegardée dans le fichier CSV, avec les informations de translation (tx, ty, tz) et de rotation (rx, ry, rz).
4. Affichage des Axes de Repère et Sauvegarde des Images :
    - Une représentation visuelle de la pose du tag est ajoutée à chaque image, avec les axes X, Y, Z du tag projetés dans l'image ainsi que les corners détéctés, facilitant ainsi la vérification visuelle de la pose estimée.
    - Les images où un tag est détécté sont sauvegardées dans un sous-dossier dédié.
5. Enregistrement des Données dans le Fichier CSV :
    - Le fichier CSV généré contient les poses de chaque tag détecté dans chaque image. Ces données comprennent la translation et la rotation du tag par rapport à la caméra,l'id du tag ainsi que le nom de l'images correpondante.

4. FUSION DES POSES DU TAG ET DU GRIPPER

OBJECTIF

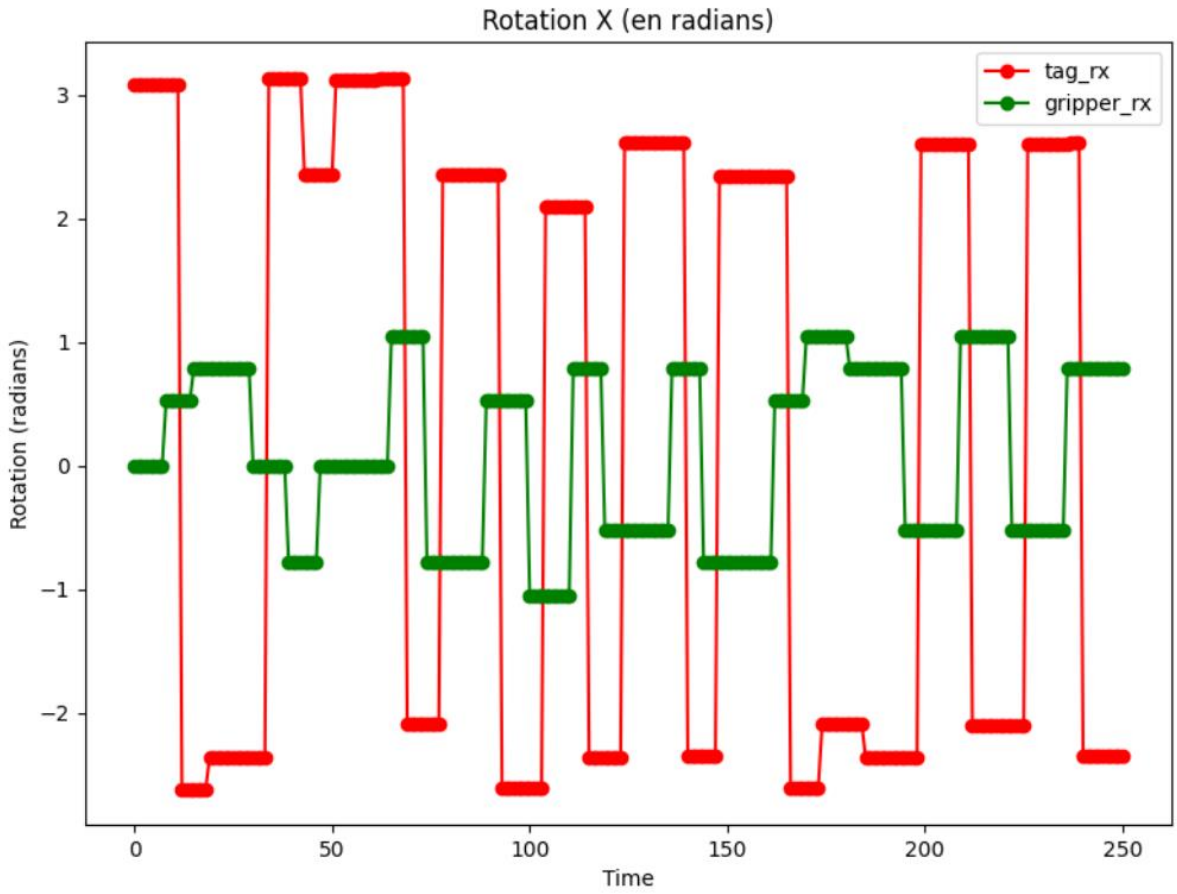
Le but de cette étape est d'associer chaque enregistrement de pose du tag (identifié par une image spécifique) à la pose correspondante du gripper, en générant un fichier de sortie qui réuni les données de ces deux fichiers.

DETAILS DU FONCTIONNEMENT

1. Vérification des Formats de Fichiers :
    - Avant de fusionner les fichiers, la méthode isFormatGood vérifie que chaque fichier CSV contient les champs requis pour les poses du gripper et du tag.
    - Chaque fichier doit avoir des champs spécifiques pour être accepté, garantissant ainsi que les données sont complètes et prêtes pour le merge.
2. Lecture des Fichiers CSV de Pose :
    - La méthode read_pose_csv lit les données de pose du gripper à partir du premier fichier, qui contient des informations sur les translations (tx, ty, tz) et les rotations (rx, ry, rz) du gripper pour chaque instant time.
    - La méthode read_image_csv lit les données de pose du tag Aruco à partir du second fichier. Chaque ligne contient l'identifiant du tag, les translations, et les rotations en lien avec une image spécifique.
3. Association des Poses du Tag et du Gripper :
    - La méthode associate_poses prend les données de pose du tag et les associe aux poses du gripper en se basant sur le numéro d'image (extrait du nom de fichier de chaque image).
    - Pour chaque pose du tag, elle recherche la pose du gripper correspondante basée sur l'ordre des images, permettant ainsi de lier les informations de pose à une même scène ou moment.
    - Si un numéro d'image ne trouve pas de correspondance, un message d'avertissement est affiché.
4. Sauvegarde des Données Fusionnées dans un Fichier CSV :
    - La méthode save_to_csv enregistre toutes les données fusionnées dans un fichier CSV spécifié.
    - Chaque ligne du fichier de sortie contient les informations de pose du tag (tag_tx, tag_ty, tag_tz, tag_rx, tag_ry, tag_rz) et celles du gripper (gripper_tx, gripper_ty, gripper_tz, gripper_rx, gripper_ry, gripper_rz), associées à une même image et le timestamp associé.
5. Accès aux Poses Spécifiques :
    - Les méthodes get_certain_tag_pose et get_certain_gripper_pose permettent d'accéder facilement aux poses du tag ou du gripper pour une image donnée, facilitant ainsi les consultations ponctuelles ou les vérifications sur des images spécifiques.

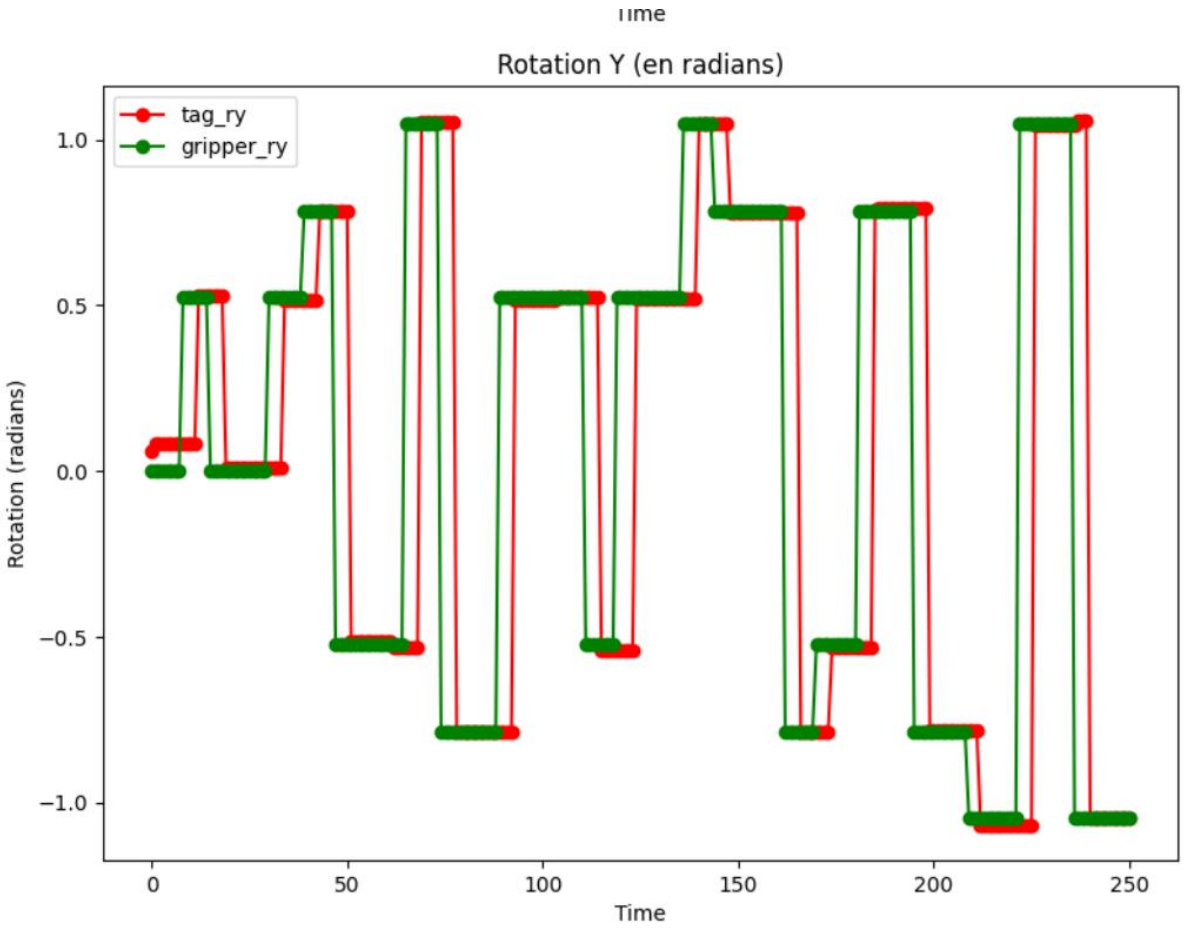
5. ANALYSE DES RESULTATS ET CONCLUSION

ROTATION X (EN RADIANS)



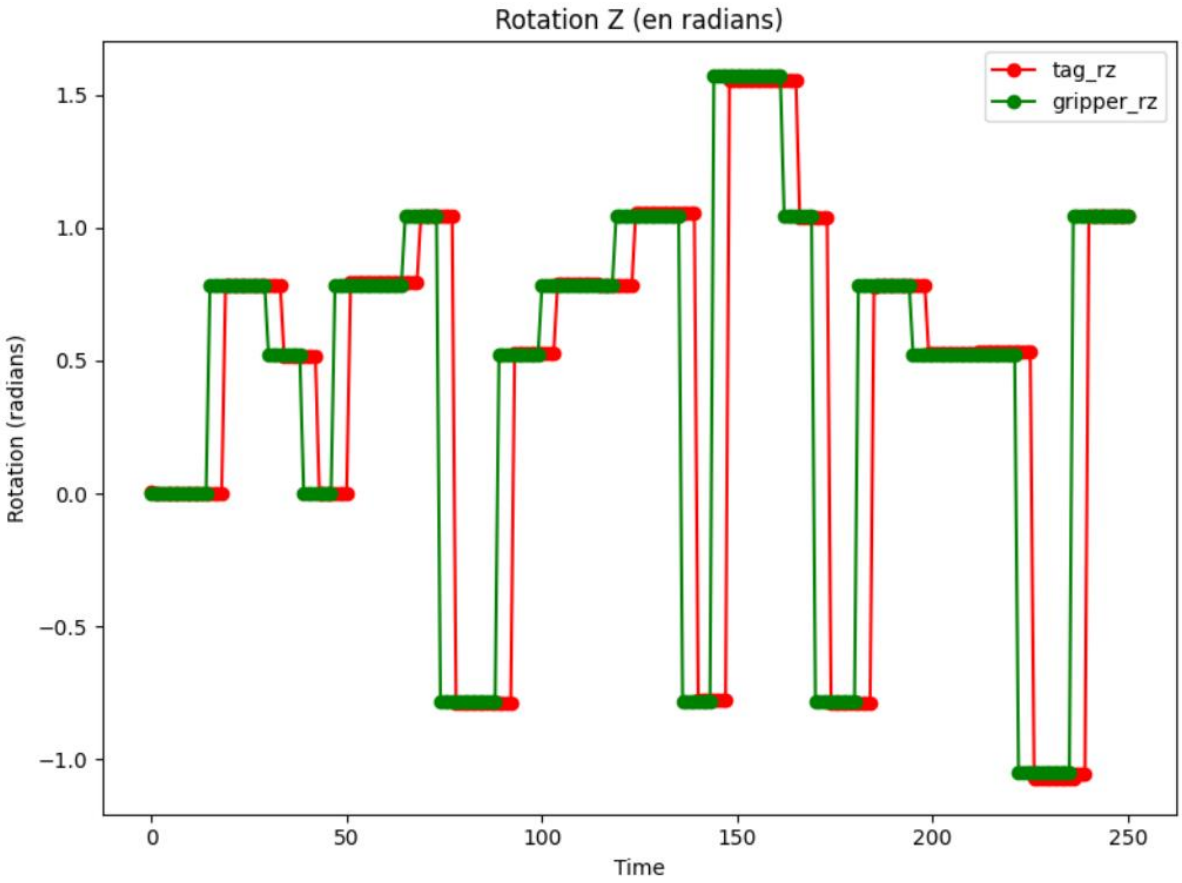
- Observation : La rotation autour de l'axe X montre des différences importantes entre le tag (en rouge) et le gripper (en vert). Les courbes fluctuent de manière distincte et semblent désynchronisées.
- Conclusion : Ce graphique met en évidence un problème de synchronisation entre le tag et le gripper. En plus de cela on constate un décalage de valeur lié au fait que la rotation initiale du tag est à 180° au lieu d'être alignée sur l'axe X. Pour corriger cela, un offset sera appliqué pour ramener la rotation initiale à 0° sur l'axe X. Cela devrait stabiliser l'angle solide, qui oscille actuellement principalement entre 180° et -180°. Cette correction devrait permettre d'obtenir des valeurs plus cohérentes et d'éliminer les divergences actuelles dans le suivi de la rotation autour de l'axe X.

ROTATION Y (EN RADIANS)



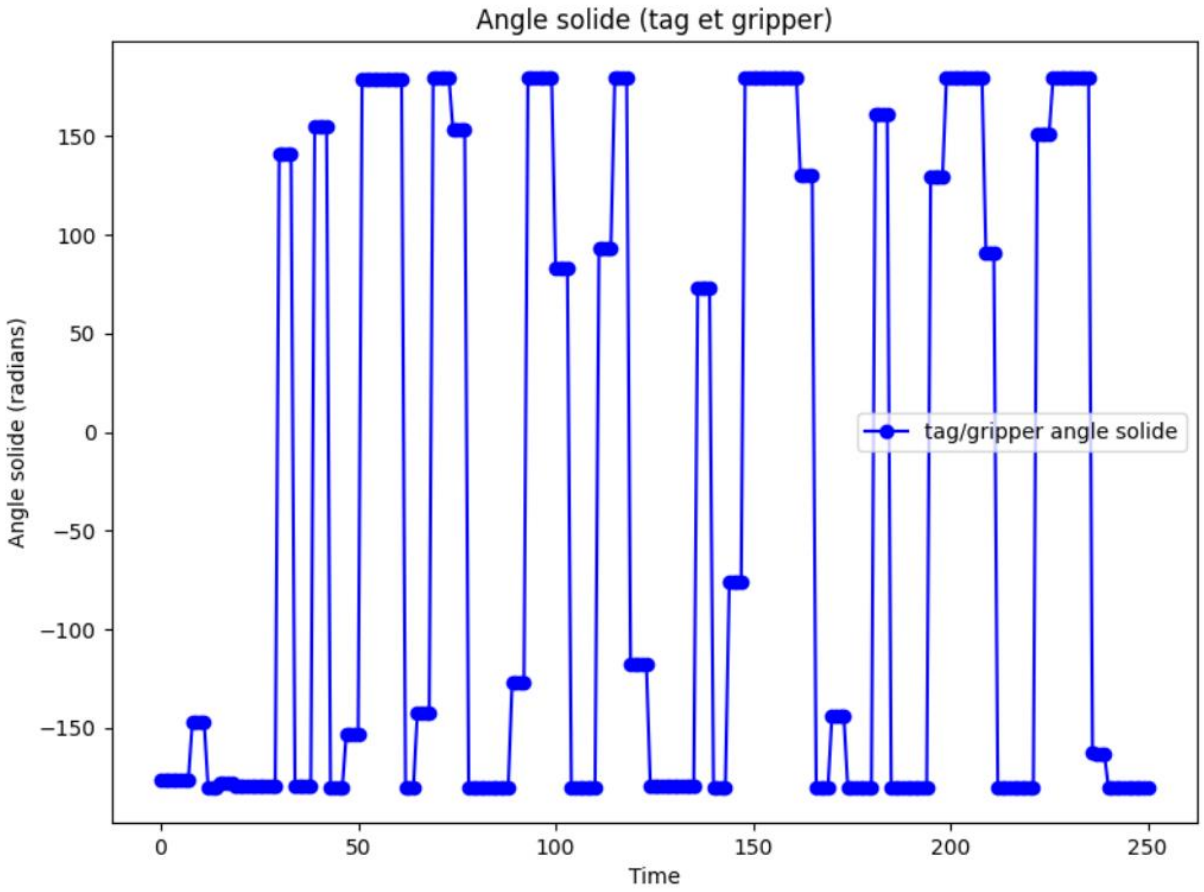
- Observation : La rotation autour de l'axe Y présentes des variations mineures entre le tag et le gripper.

ROTATION Z (EN RADIANS)



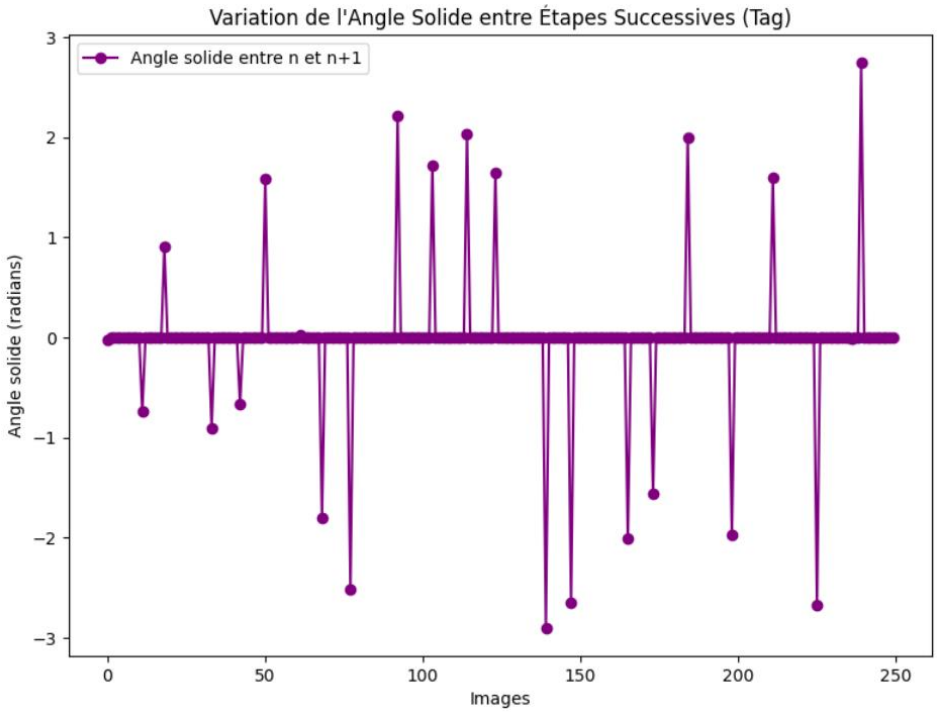
- Observation : La rotation autour de l'axe Z présente des variations mineures entre le tag et le gripper, similaire au comportement observé pour l'axe Y.

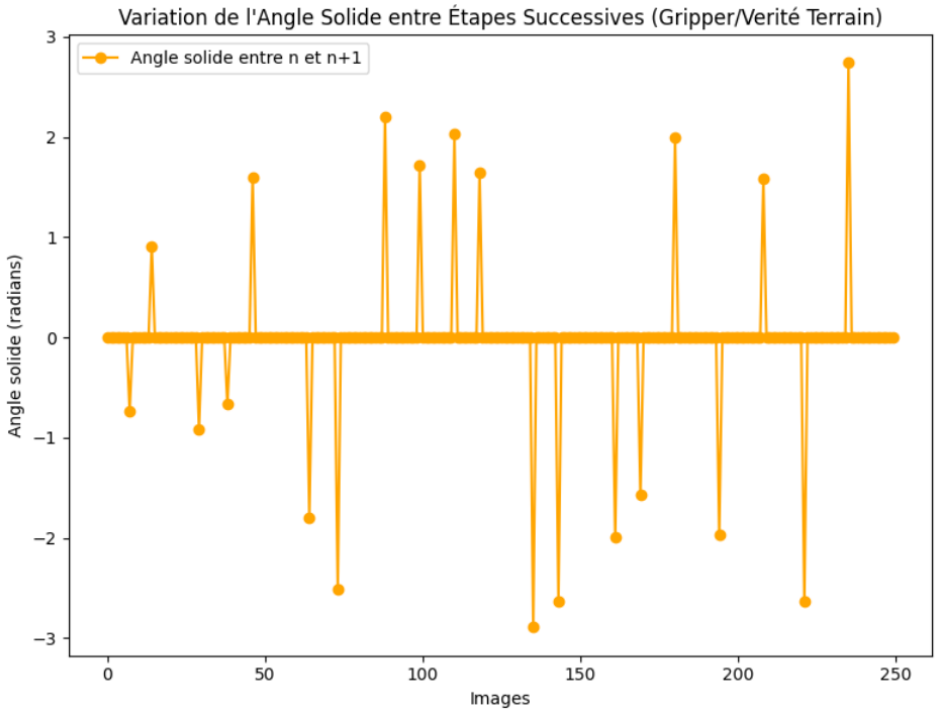
ANGLE SOLIDE (TAG ET GRIPPER)



- Observation : L'angle solide (erreur angulaire) entre le tag et le gripper montre des variations significatives sur toute la durée de la capture, oscillant principalement entre 180° et -180°.
- Conclusion : Les fluctuations de l'angle solide confirment une désynchronisation continue entre le tag et le gripper, causée en grande partie par la mauvaise orientation initiale du tag sur l'axe X. En appliquant un offset pour aligner la rotation initiale du tag à 0° sur l'axe X, l'angle solide devrait être stabilisé et mieux synchronisé avec le gripper, ce qui permettrait de corriger cette erreur angulaire.

VARIATIONS DE L'ANGLE SOLIDE ENTRE ÉTAPES SUCCESSIVES POUR LE TAG ET LE GRIPPER





Les deux graphiques ci-dessus montrent les variations de l'angle solide entre chaque étape successive pour le tag et le gripper (vérité terrain) respectivement. On observe que les

changements brusques dans les angles solides correspondent aux changements de rotation imposés dans Gazebo. Ces transitions, qui se traduisent par des sauts importants dans les valeurs des angles, sont bien visibles dans les deux graphiques.

Bien que les changements de rotation soient globalement identiques entre le tag et le gripper, un léger décalage de quelques images est présent, indiquant un déphasage dans la synchronisation