Je suis heureux de me présenter devant vous pour présenter mon projet de recherche intitulé "Contrôle d'un bras robotique sous-marin en apprentissage par renforcement".

Le contrôle des robots sous-marins est un domaine en constante évolution qui présente de nombreux défis techniques en raison de l'environnement sous-marin complexe et hostile. L'environnement sous-marin est soumis à des conditions extrêmes telles que la pression élevée, la visibilité réduite et les conditions météorologiques imprévisibles, ce qui rend le contrôle de ces robots extrêmement difficile. En outre, la communication entre les robots sous-marins et les opérateurs est souvent limitée en raison de la distance et de la dégradation des signaux, ce qui rend la tâche de contrôle encore plus compliquée.

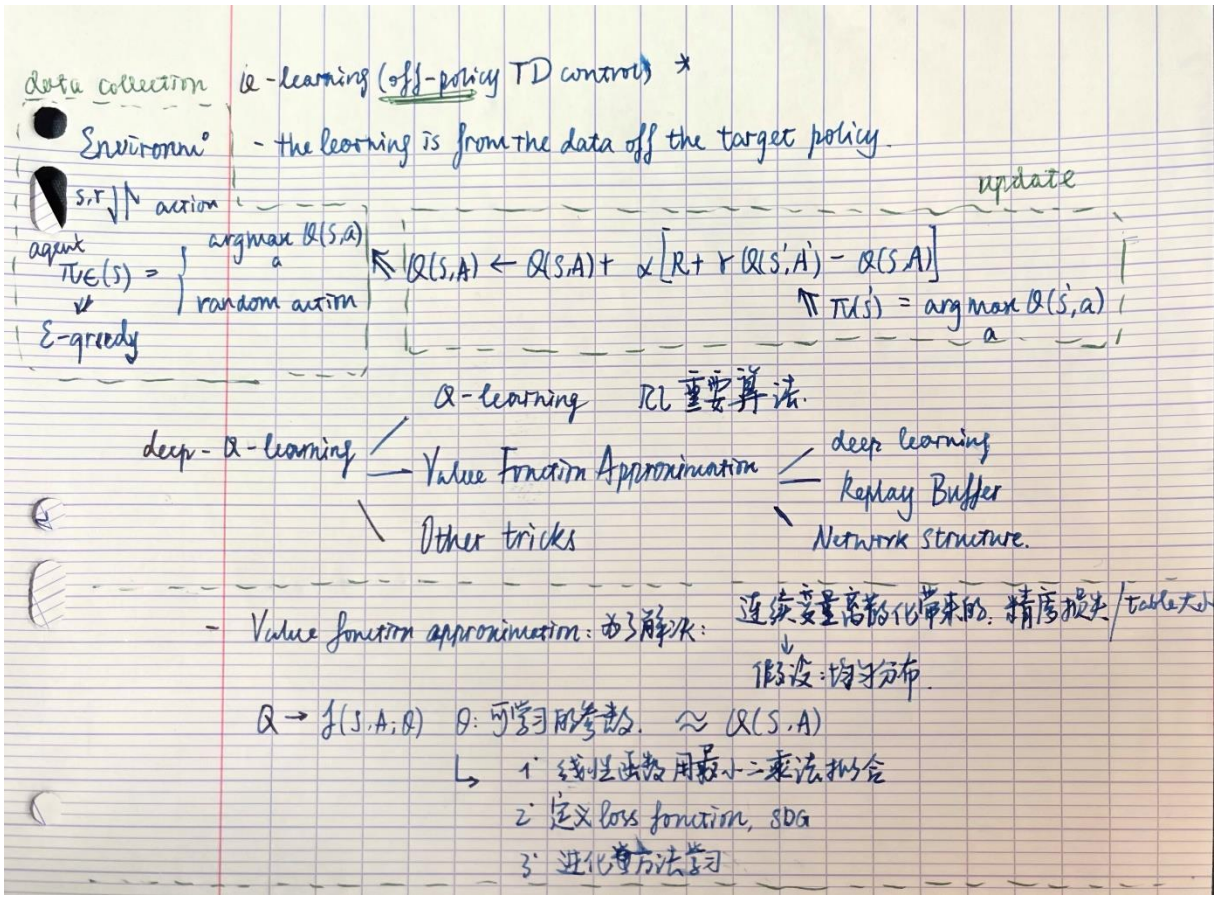
Ces défis techniques peuvent avoir des conséquences significatives sur les performances et la fiabilité des opérations sous-marines. Par exemple, la mauvaise visibilité peut rendre difficile la surveillance des activités du robot, tandis que la pression élevée peut endommager les composants du robot et compromettre son fonctionnement. De plus, la communication limitée peut rendre difficile la transmission de données et de commandes en temps réel entre le robot et les opérateurs, ce qui peut entraver la capacité du robot à accomplir les tâches qui lui sont assignées.

En plus, dans notre expérience, le bras est un robot souple, et il est construit par des pièces imprimées en 3D et entraîné par deux Servomoteurs, du coup c'est vraiment un robot low-cost et en terme de précision de commande, on peut dire il y a pas de précision quoi... la loi de commande de ce bras est plus compliquée à modéliser dans un réel environnement opérationnel.

Dans ce contexte, notre développer un contrôleur pour un bras robotique sous-marin qui va guider l'extrémité du bras vers une position indiquée par l'opérateur en utilisant des techniques d'apprentissage par renforcement pour surmonter ces défis complexes. En utilisant cette approche, nous espérions former le bras à prendre des décisions optimales en temps réel en fonction de son environnement, sans la nécessité d'une intervention humaine constante. Cela pourrait conduire à une meilleure efficacité et à une plus grande précision dans les opérations sous-marines.

L'apprentissage par renforcement est un type de technique d'apprentissage automatique qui permet à un agent de prendre des décisions en interagissant avec son environnement. L'agent est programmé pour maximiser une récompense en fonction de ses actions, ce qui lui permet d'apprendre au fil du temps comment agir de manière optimale pour atteindre ses objectifs.

Le processus d'apprentissage par renforcement est fondé sur un cycle continu d'observation, de prise de décision et de rétroaction. L'agent observe son environnement, prend une décision en fonction de cette observation, et reçoit une rétroaction sous forme de récompense ou de pénalité en fonction de l'action qu'il a choisie. En utilisant cette information, l'agent peut ajuster sa stratégie pour maximiser la récompense à long terme.



L'apprentissage par renforcement est souvent utilisé dans des domaines tels que les jeux, la robotique et l'intelligence artificielle, car il peut aider les agents à apprendre à prendre des décisions dans des environnements complexes et incertains. De plus, en permettant aux agents d'apprendre par eux-mêmes au fil du temps, cette technique peut également permettre d'éviter le besoin de programmer manuellement de nombreuses règles et stratégies.

A droite on peut voir deux exemples classiques de l'agent qui peut être entraîné à travers des approache de reinforcement learning.

Pour atteindre notre objectif, nous avons effectué une analyse approfondie de la littérature pour évaluer les méthodes de contrôle complex des robots et identifier les opportunités pour l'amélioration. Nous avons choisi de mettre en œuvre un contrôleur basé sur l'algorithme Deep Q-Network, qui est une amélioration du Q-Learning.

Le Q-Learning est un algorithme de renforcement d'apprentissage qui s'appuie sur les théories de la théorie de la décision Markovienne, ça veut dire on considère des systèmes dans lesquels l'état futur dépend uniquement de l'état actuel, et non pas de l'historique complet des états précédents.
L'idée principale de cet algorithme est d'avoir une bonne qualité de Q value par essais et erreurs, où Q value est une fonction de l'état s et l'action a potentiel dans cette situation et en ce moment, il signifie la qualité de choix de cet action à long terme, du coup Q value est une combinaison des reward actuels et futurs. Cette fonction Q peut être actualisée à chaque étape en utilisant la formule suivante :

$Q(s, a) = Q(s, a) + \alpha * (r + \gamma * \max_{a'} Q(s', a') - Q(s, a))$

où :

- α est le taux d'apprentissage
- γ est le facteur de réduction de l'horizon temporel
- r est la récompense immédiate reçue par l'agent en prenant l'action a dans l'état s
- s' est l'état suivant dans lequel l'agent se trouve après avoir pris l'action a
- a' est l'action choisie dans l'état s'
- $\max_{a'} Q(s', a')$ est la valeur maximale de la fonction Q dans l'état s'

L'algorithme continue d'itérer ce processus jusqu'à ce que la fonction Q converge vers une solution optimale qui décrit la stratégie optimale pour l'agent. À chaque étape, l'agent peut choisir l'action qui correspond à la valeur Q maximale dans l'état courant pour maximiser sa récompense future.

Le Deep Q-Network, utilise des algorithmes de réseaux de neurones profonds pour évaluer la fonction de valeur Q(s, a) avec une apprpche d'approximation. Cela permet d'améliorer la performance de l'apprentissage et il présente plusieurs avantages par rapport au Q-Learning traditionnel :

C'est évident que dans le Q-Learning traditionnel, les états doivent souvent être discrétisés, ce qui peut entraîner une perte d'informations et certain bias de données surtout quand les états réels continus ne sont pas distribués uniquement quoi.

$$\theta_{k+1} = \theta_k - \alpha \Delta_\theta E_{s' \ P(s'|s,a)}[(Q_\theta(s,a) - target(s'))^2]|_{\theta=\theta_k}$$

où θ est le parametres du réseau de neurone
Au lieu de créer un tableau avec les lignes et colonnes de nombres finis, les réseaux de neurones profonds peuvent traiter de grands ensembles de données, ce qui peut être utile dans des environnements complexes avec beaucoup d'états et d'actions possibles.

Dans nos expériences, nous avons conçu et réalisé un bras robotique sous-marin très flexible en utilisant des ressorts et des pièces imprimées en 3D. Ici, notre bras robotique ne peut se déplacer que dans un plan 2D pour le moment, et nous supposons que la méthode d'apprentissage par renforcement profond pour contrôler le mouvement 2D du bras robotique peut être appliquée au contrôle du mouvement 3D une fois que les résultats du sujet auront mûri.

Nous avons marqué une section du bras robotique avec un cure-dent vert, dont la position peut être capturée par la caméra.
A l'aide de opencv de python, la détection des couleurs etait accomplie en convertissant d'abord l'image en espace de couleur HSV au lieu de RGB. Cela permet de mieux cibler une gamme de couleurs particulière en sélectionnant un intervalle de valeurs de teinte, de saturation et de luminosité. Après, un masque peut être créé en utilisant une fonction dans cv2.inRange qui retourne une image binaire avec uniquement les pixels qui se trouvent dans l'intervalle sélectionné. Puis on a appliqué l'algorithme de RANSAC (Random Sample Consensus) dans scikit-learn pour effectuer la détection des lignes droites dans la partie verte.
En faisant entrer les valeurs l'angle de rotation des servomoteurs, nous pouvons obtenir toutes les positions potentiellement atteignables du sommet de ce bras robotique, comme présenté dans la figure à gauche.

Q(s, a | θ), avec
s = (position actuelle, position cible), où la position actuelle est la valeur de retour de fonction de détection de position à chaque instant, et la position cible est une valeur aléatoire parmi toutes les positions possibles.
a = (input de moteur1, input de moteur2), qui peut prendre les valeurs de -10 degrés, 0 et 10 degrés.

Le processus d'apprentissage peut être décrit comme cet algorithme presenté ici :
Nous utilisons ici un jeu d'expérience, qui s'apelle experience replay memory, où nous stockons les expériences de l'agent à chaque pas de temps dans un ensemble de données D. Pour initialisation, le réseau de neurones profond est initialisé avec des poids aléatoires.

À chaque étape, l'algorithme choisit une action a en utilisant une stratégie d'exploration-exploitation, telle que la stratégie d'échantillonnage aléatoire ε-greedy.
Après avoir pris une action, l'algorithme observe la récompense obtenue r et l'état suivant s'. Et on les tous garde dans le jeu de donnés d'expérience D.
Puis, nous tirons des échantillons aléatoires et les utilisons pour mettre à jour le réseau de neurone Q.

En plus, pendant le training de network, on a introduit un autre réseau cible Q^ du même modèle que Q, et pour chaque C étapes, on va copier les paramètres du réseau Q pour obtenir un nouveau réseau cible Q ^ qui est plus précis. Et on utilise Q^ pour générer des valeurs cibles pour la mise à jour de Q. De cette façon, les valeurs cibles sont maintenues constantes dans certaine durée de temps et ne se déplacent pas lorsque le réseau Q est mis à jour, ce qui améliore la stabilité de la convergence de l'algorithme.
Et finalement, ce processus est répété jusqu'à ce que la fonction Q converge vers la valeur optimale.

Figures de évolution de la fonction objectif avec epoch de training,
Avec loss fonction définie comme mean square loss de différence entre la valeur Q chapeau de réseau de cible et la valeur Q de réseau actuel.
En fait, grâce à la convergence de algorithme de Q learning, on sait bien que la valeur de Q associé à chaque situation de l'état et l'action tend vers une valeur optimale, du coup ces deux figures signifie bien la convergence de notre propre réseau de neurone de Deep Q network.

Implementer cet algorithme ce dqn en notre bras robotique
Laisse l'agent prendre decision de l'action de moteur à chaque étape, afin d'atteindre la position cible donnée au début de tâche.
Vidéo d'exemple, position accessible par le bras, qui est
C'est pas une structure mécanique assez stable et c'est pas un système qui peut effectuer notre commande précisément.
Pas à pas
En fait, nous pouvons constater que la récompense n'est pas toujours décroissante, et ce parce que notre fonction Q prend en compte non seulement la récompense actuelle, mais aussi l'impact de la décision actuelle sur la situation future.

Voici une autre figure de l'évolution de la récompense dans le temps où le bras a également trouvé le point cible. Et ici, dans cette figure, nous avons échantillonné au hasard 20 positions cibles et laissé le réseau contrôler le bras du robot pour qu'il se déplace vers la position spécifiée. Nous pouvons observer que sous le contrôle du réseau d'apprentissage par renforcement, la récompense diminue dans tous les cas au cours de l'exécution. Et dans un grand nombre de cas, l'algorithme a fini par accomplir la tâche de contrôle.
De plus, on laisse l'algorithme chercher tous les points cibles possibles qu'on a obtenus au début de entraînement, et si on définit une récompense finale de moins de 15 pixels comme une réussite de la tâche, alors notre modèle actuel peut atteindre une précision de 82,8%.

On a développé un contrôleur en 2D basé sur deep q learning de l'apprentissage par renforcement pour un bras robotique souple qui va guider l'extrémité du bras vers une position indiquée par l'opérateur au début de chaque essaie.

Notre modèle actuel peut atteindre une précision de 82,8%.

Cette méthode de contrôle, après l'entraînement du réseau de neurone, peut réaliser spontanément les objectifs de la tâche sans nécessiter d'intervention humaine, et le temps de calcul entre chaque décision de contrôle, est très court, ce qui en fait une solution efficace et effective pour contrôler les bras robotiques sous-marins dans des situations complexes.

Pour l'instant, notre recherche rencontre encore les problèmes et les limites suivants.
Détection imprécise de la position. Dans chaque frame de la vidéo opencv, le toothpick vert utilisé pour marquer la position n'est pas une ligne droite, mais plutôt un cylindre, et il y a toujours une erreur de quelques pixels dans l'identification de la pointe du toothpick.

En outre, le contrôle du moteur n'est pas précis, et cette stucture de mécanique n'est pas assez stable, par exemple, pour la même entrée d'angle de rotation du moteur, le résultat de l'algorithme de détection de position n'est pas nécessairement le même à chaque fois, et après chaque exécution de la commande du moteur, le bras du robot subit une oscillation considérable due à l'inertie, ce qui peut encore augmenter l'erreur du module opencv.

Enfin, on doit aussi appliquer l'algorithme actuel d'apprentissage par renforcement pour les mouvements en 2D au contrôle du bras de mouvement en 3D dans le futur, on doit aussi introduire une deuxième caméra pour finalement atteindre nos objectifs de contrôle.

Voila c'est tout ce que je vais dire pendant cette presentation, vous pouvez poser des questions sur des points pas clairs et que j'ai pas bien détaillé merci pour votre attention.