ENSEIRB - MATMECA
Filière Robotique

État de l'art : Projet de robotique

travail réalisé par

Clara Dieudonné, Coralie Deplanne, Guillaume Fornes

Modèle dynamique et application sur le robot Reachy de Pollen Robotics

---

Année universitaire 2022 - 2023

Encadrement

Mr	Vincent Padois	Directeur de recherche à l'*INRIA* et encadrant de la partie état de l'art
Mr	Rémi Fabre	Ingénieur Robotique chez *Pollen Robotics* et maître du projet

Résumé

L'objectif de ce projet est d'utiliser la programmation par démonstration, grâce à la téléopération, pour faire reproduire par le robot Reachy des trajectoires complexes inspiré de l'humain de manière autonome, le tout avec un suivi dynamique du mouvement.
Premièrement, des trajectoires seront générées en imitant les mouvements de l'opérateur à partir des données de téléopération de Reachy obtenue à l'aide d'un casque de réalité virtuelle et ses manettes.
Ensuite, nous verrons comment Reachy pourra réaliser ces trajectoires de manière autonome par apprentissage inspiré de la téléopération.
L'objectif suivant est d'être capable de suivre ces trajectoires de manière dynamique, avec un contrôle en couple permis par l'utilisation de *feedback* et *feedforward*, et de programmer en parallèle une fonction de bras zéro gravité pour Reachy.
Enfin, nous étudierons aussi la partie sécurité autour du fonctionnement de Reachy dans divers contextes d'application liés à notre projet.

Mots-clés : *Dynamique, Robot humanoïde, Bras anti-gravité, Téléopération, Suivi de trajectoire.*

Table des matières

Table des figures

Introduction

Le domaine de la robotique est en constante évolution, parallèlement au changement de notre société, aux technologies d'ingénierie qui se modernisent, etc. Ainsi, dans ce contexte de progrès, notamment dans le secteur de l'industrie, les entreprises sont à la recherche de méthodes plus agiles, ce qui se reflète par une demande de robot plus polyvalent. Contrairement aux robots dont l'objectif est de réaliser une tâche répétitive sur une chaîne de production, c'est la capacité d'adaptation de ces derniers qui est sollicitée, dans l'objectif de pouvoir réaliser des tâches plus variées tout en étant plus accessibles pour un plus grand public.

Cette évolution a mené à l'apparition et au développement de la Cobotique et des Cobots, qui sont actuellement au centre de la recherche. En effet, ils sont un réel outil pouvant être la base de la réalisation de nombreuses tâches, de plus avec une meilleure accessibilité de programmation et mise en œuvre. Ils offrent une modularité qui propose des solutions allant du domaine de l'industrie au robot de service au contact du public.

C'est dans ce contexte que l'entreprise *Pollen Robotics* a mis au point *Reachy* (figure 1), un robot humanoïde expressif et open source, possédant 2 bras manipulateur et une base mobile. Conçu pour travailler parmi les humains, avec une plateforme permettant de le prendre en main facilement, Reachy peut trouver sa place dans de nombreux domaines, de la recherche au service divertissant pour des évènements.

FIGURE 1 – Le robot *Reachy* de *Pollen Robotics*

Cependant, l'évolution dans un milieu humain ouvert implique, en plus de la réussite d'une tâche, toute une question centrée sur le social et la manière d'effectuer cette tâche. Dans cet objectif, les robots tendent à imiter les mouvements humains, plus rassurant pour un public non averti. Récemment, l'équipe de *Pollen Robotics* a développé une méthode de contrôle par téléopération à l'aide d'un casque de réalité virtuelle, ce qui permet de faire réaliser à Reachy des mouvements humains pour les étudier.

À partir de tout cela, l'objectif de notre projet est de proposer pour le robot *Reachy* une solution qui, par inspiration des mouvements humains étudiés à partir de la téléopération, permettrait un contrôle dynamique du mouvement d'un bras pour un suivi de trajectoire. Par la suite, cela permettrait la mise en place d'un bras anti-gravité, déplaçable par l'opérateur, avec une résistance juste suffisante à le maintenir dans sa position, ou encore de performer un lancer d'objet sur une cible.

1 Génération de trajectoire

La première étape de la réalisation d'un mouvement est la planification de trajectoire. L'objectif est de générer cette dernière en imitant le mouvement d'un bras humain.

1.1 Programmation par démonstration

L'intelligence et la logique de l'Homme lui permettent de comprendre et de réaliser un certain nombre d'actions, dont la réflexion ayant permis d'amener une solution logique pour lui n'est pas explicable et programmable facilement pour transmettre cette habileté au robot.
La téléopération rend possible la combinaison de la logique du cerveau humain à la capacité d'agir du robot. Ainsi, dans cette approche de **programmation par démonstration**, les connaissances implicites de l'opérateur sont transmises au robot sans avoir à les décrypter dans tous les détails pour les expliquer et transmettre "manuellement".

1.1.1 Téléopération d'un robot

Comme expliqué précédemment, la programmation par démonstration (*PbD*) offre une facilité de programmation qui répond au besoin croissant de robotique modulaire dans des domaines comme l'industrie.

La téléopération est une des méthodes de PbD. Elle est le terme technique qui défini le contrôle à distance d'un robot. Il y a d'un côté un opérateur travaillant sur un dispositif maître qui réalise une tâche au travers d'un dispositif esclave. La téléopération peut être divisée en plusieurs classes, selon le niveau d'autonomie qu'elle propose. On peut trouver des systèmes où l'opérateur donne de simples objectifs que le robot atteint par ses propres moyens, ou bien à l'opposé des systèmes où l'opérateur à un contrôle total sur les mouvements du robot.

Bien qu'étudiée depuis très longtemps pour répondre à des problèmes complexes tel que le contrôle de robots en zone inaccessible comme le proposait Johnsen et al.[4] avec un robot téléopéré sur la lune, la téléopération a subi une accélération dans ses recherches grâce à la modernisation des technologies de communication, et surtout par la popularisation de capteur dit *low-cost*. Par exemple, la *Kinect* de Microsoft, initialement conçu pour communiquer avec la console de jeu *XBox*, est utilisé par Jha et al.[5] pour contrôler le bras industriel *SCORBOT ER-4u*.

Il existe par ailleurs différentes méthodes de téléopération. Premièrement, l'opérateur peut commander le robot avec comme interface un outil tel qu'un clavier, une souris, un joystick, etc[6]. Le résultat n'est pas le plus optimal, car l'action n'est pas intuitive et l'opérateur doit être entraîné avant.

Une seconde solution est de réaliser la téléopération en manipulant une réplique à distance du robot. L'opérateur peut ainsi contrôler chacune des articulations indépendamment. Cependant, cette solution reste complexe à mettre en place, et il est compliqué et lent pour l'opérateur de manipuler autant d'articulations en même temps.
Enfin, la dernière solution regroupe toutes les méthodes de contrôle basées sur le mouvement de l'homme. Ces mouvements peuvent être récupérés par différents capteurs, comme une caméra[7], des capteurs sur le corps de l'opérateur[8], ou encore grâce à un casque et des manettes pour de la réalité virtuelle.
Au niveau de notre projet, l'équipe de *Pollen Robotics* a mis au point un contrôle du robot *Reachy* à l'aide d'un casque de réalité virtuelle (figure 2).

FIGURE 2 – Téléopération du robot *Reachy* avec un casque de réalité virtuelle

1.1.2 Récupération des données

L'opérateur de la téléopération possède un casque de réalité virtuelle et deux manettes, afin retourner une information sur la position de ses deux mains et l'orientation de la tête.

Pour envoyer cette information coté opérateur, c'est le *steamVR package* qui est utilisé comme interface, et permet d'obtenir la position courante des manettes et du casque dans le repère de *Unity* pour la transmettre.
Le côté robot agit comme un serveur, qui réceptionne les informations de position des 3 éléments, puis effectue une résolution par la cinématique inverse pour positionner correctement les effecteurs et orienter la tête de *Reachy*.
Le framework *gRPC* est utilisé comme interface de communication entre Unity et le robot. Il gère l'envoi du message des positions sous la forme de deux matrices pour la position et l'orientation des manettes, un quaternion pour l'orientation de *orbita*, l'actionneur du cou de Reachy qui oriente sa

tête. Un second message contient la position cible de l'effecteur, et de manière ponctuelle un troisième message gère la compliance, le couple et la vitesse des moteurs. D'un autre côté, le client Unity envoie une requête pour recevoir l'image de la caméra du robot.

1.1.3 *Mapping* du mouvement humain en un chemin pour le robot

La problématique de cette section est la génération d'une trajectoire naturelle pour Reachy à partir des données récupérées par la téléopération. Cependant, différents sous problèmes entrent en jeu.

Dimension des bras : Le bras de l'opérateur n'a pas les mêmes dimensions que celui de Reachy, segment par segment, et de plus les différents opérateurs non pas non plus la même longueur de bras entre eux. Il faut donc faire un choix entre garder le rapport d'amplitude d'un mouvement, i.e. le bras du robot est tendu quand celui de l'est, ou bien conserver la trajectoire exacte du mouvement et ses dimensions. Dans l'idée de pouvoir lancer un objet selon la même trajectoire que l'aurait fait l'opérateur, nous choisissons la deuxième option.

Multiplicité des solutions : La téléopération avec un casque de réalité virtuelle et deux manettes renvoie comme jeu de données la position demandée pour l'effecteur. Cependant, le bras de *Reachy* possède 7 degrés de liberté, donc pour un point donné atteignable qui n'est pas un cas limite, il existe une infinité de solutions possibles.
Étant un problème connu et important dans le domaine de la robotique, de nombreuses solutions sont proposées, basées sur différents critères et/ou caractéristiques. Il existe par exemple des critères qui essaient de quantifier l'inconfort "humain" pour une position d'articulation donnée[9], ou d'autres critères plus simples de minimisation du *jerk*, de l'énergie potentielle ou encore utilisant des méthodes basées sur les graphes pour trouver la meilleure trajectoire.
J. Zhao et al.[10] propose une solution combinant plusieurs de ces critères. Un ressort virtuel à raideur variable est proposé pour représenter l'énergie potentielle des muscles, un algorithme de *"Gradient Projection Method based Rapidly-exploring Random Tree"* (GPM-RRT) s'ajoute pour la planification de la prise d'objet, en synthétisant la minimisation du total de l'énergie potentielle et l'inconfort des articulations.

1.2 Apprentissage inspiré de la téléopération

Reproduire une trajectoire à partir d'un set de coordonnées relative au temps pour l'emplacement cible de l'effecteur est une chose, mais la program-

mation par démonstration tente d'aller plus loin en proposant des solutions ou le robot est capable d'apprendre à partir d'exemples fournis, afin d'optimiser une tâche. Plusieurs outils ont donc été mis au point pour parvenir à ce résultat.

Premièrement, pour répondre au problème de la quantité de données à traiter que représente de nombreux exemples de démonstrations fournis à un robot, la recherche s'intéresse aux DMP, Dynamic Movement Primitives, décrites par S. Schaal et al.[11]. Les DMP représentent une formulation mathématique des primitives de moteurs non linéaire, qui permettent par une subdivision intéressante du mouvement de commander de manière adouci des moteurs pour créer une trajectoire plus proche de l'homme. Elles sont donc une forme de modélisation de mouvements complexes qui sont particulièrement utilisés dans la planification de trajectoire à partir d'un set d'exemple.

A. Hewitt et al.[12] proposent de combiner les DMP au modèle de mélange gaussien, une méthode statistique utilisée ici pour la modélisation de fonctions non-linéaires, notamment pour l'expression de la probabilité de position d'une articulation, qui aboutissent à la génération de trajectoires optimisées pour un robot Kuka apprises d'une série de démonstrations.

J. Aleotti[13] propose lui une solution basée sur le modèle caché de Markov et les *Non-Uniform Rational B-Spline*, NURBS, pour réaliser un apprentissage par un bras robotique à partir d'une clusterisation des tâches de manipulations, avec une démonstration réalisée environnement de réalité virtuelle.

2 Suivi de trajectoire dynamique

Une fois que la trajectoire a été générée, il faut permettre au robot d'effectuer cette trajectoire tout en respectant la dynamique du mouvement.

2.1 Feedback et Feedforward

Pour pouvoir suivre la trajectoire souhaitée, il faut, à chaque instant, connaitre de combien et dans quelle direction le robot doit se déplacer et surtout quelle est l'erreur entre la position réelle du bras et celle souhaitée. Les deux principales méthodes utilisées sont le *feedfoward* et le *feedback*. Le *feedfoward* concerne la partie théorique de la dynamique du mouvement alors que le *feedback* s'occupe de la compensation de l'erreur.

Pour pouvoir faire un contrôle avec du *feedback* et donc pouvoir comparer la valeur attendue avec la valeur réelle, il est nécessaire d'avoir des capteurs afin de mesurer les données réelles au cours du temps. Les capteurs ne mesurent pas réellement en temps réel, ils ont chacun une fréquence

d'échantillonnage différente. Cependant, cette fréquence étant la plupart du temps comprise entre cent et mille hertz, il est possible de considérer que la récupération des données se fait en temps réel. Par la suite, nous considérerons les capteurs comme parfaits.

L'asservissement d'un robot a pour objectif de faire tendre l'erreur vers zéro. Dans le cas d'un modèle à une seule articulation, l'erreur dynamique de l'articulation est de la forme suivante :

$$\boldsymbol{\theta_e(t) = \theta_d(y) - \theta(t)} \tag{1}$$

Même s'il serait souhaitable que l'erreur soit corrigée parfaitement sans aucun délais, cela n'est pas réalisable dans la réalité. Dans les conditions réelles, la correction de l'erreur se décompose en deux parties, la réponse en **régime transitoire** et la réponse en **régime permanent**. L'erreur en régime permanent ($\boldsymbol{e_{ss}}$) est décrite par l'écart entre la valeur réelle et la valeur souhaitée lorsque le temps tend vers l'infini. L'erreur en régime transitoire quant à elle est définie par deux caractéristiques, le dépassement décrit par la formule suivante :

$$\boldsymbol{depassement = \| \frac{\theta_{e,min} - e_{ss}}{\theta_e(0) - e_{ss}} \| 100\%} \tag{2}$$

et le temps de réponse ($\mathbf{T}$) à n% qui correspond au temps que la réponse met pour que la valeur absolue de l'erreur ne dépasse plus n% de la commande souhaitée, c'est-à-dire :

$$\boldsymbol{\|\theta_e(t) - e_{ss}\| \leq \frac{n}{100}(\theta_e(0) - e_{ss}) \ \forall t \geq T} \tag{3}$$

Les différentes caractéristiques sont représentées sur la figure 3.

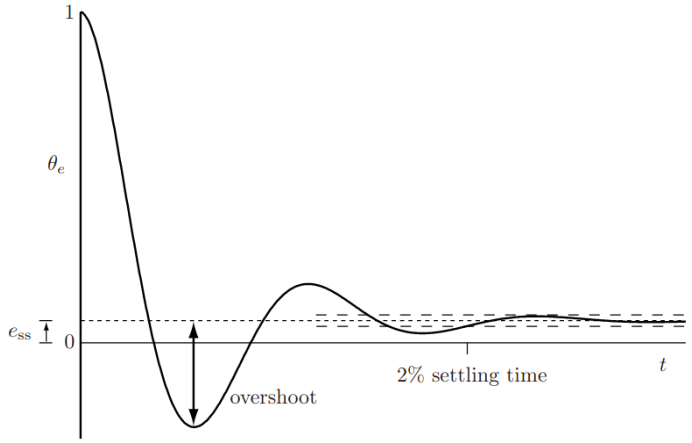

FIGURE 3 – Exemple de réponse à une erreur [1]

Le *feedfoward control* quant à lui, utilise le modèle dynamique pour générer le couple nécessaire sans avoir à attendre le retour concernant l'erreur. Il permet en plus de prendre en compte certains paramètres, tel que l'inertie, qui ne sont pas pris en compte par le *feedback control*. Le modèle dynamique permet donc d'améliorer les performances.

Le *feedback control* appliqué au Reachy étant effectué en bas niveau, il consiste à modifier les paramètres du contrôleur PID. Les différents coefficients de ce dernier doivent être choisis consciencieusement pour un contrôle précis. Pour cela, il est important de connaître l'impact des différents coefficients. Lynch Kevin M. et Park Frank C. [1] expliquent que le coefficient proportionnel influe sur la réduction de l'erreur en position, le coefficient dérivé influe sur la réduction de l'erreur en vitesse et le coefficient intégrateur quant à lui influe sur la réduction de l'erreur en régime permanent. Quant à Freddy Mudry [2], il mentionne les différentes possibilités afin de déterminer ces différents coefficients.

La première méthode est celle de la synthèse par compensation des pôles. Elle consiste à identifier les coefficients en identifiant les termes de la fonction de transfert du système en boucle fermée à la forme classique du second ordre. Pour cela, il faut compenser les pôles les plus lents puis identifier le gain afin que la réponse soit optimale par rapport à la consigne qu'on lui a envoyée.

La deuxième est celle de Ziegler et Nichols, qui contient elle-même deux méthodes, une avec la réponse indicielle et l'autre avec le point critique. Pour celle avec la réponse indicielle, il suffit d'enregistrer cette dernière sans régulateur pour ensuite tracer la tangente à la courbe à son point d'inflexion. La tangente permettra de calculer sa pente **p**, le retard **L** correspondant à son point d'intersection avec l'abscisse et le gain $\boldsymbol{K_0 = e_{ss}/E}$ avec E la valeur de la consigne. Les coefficients sont ensuite obtenus à l'aide du tableau figure 4. Cependant, il ne faut pas hésiter à diminuer le coefficient K_p car la valeur obtenue engendre souvent un dépassement d'environ 20%.

Type	K_p	T_i	T_d
P	$1/(pLK_0) = 1/(aK_0)$		
PI	$0.9/(pLK_0) = 0.9/(aK_0)$	$3L$	
PID	$1.2/(pLK_0) = 1.2/(aK_0)$	$2L$	$0.5L$

FIGURE 4 – Calcul des valeurs des coefficients à partir de la réponse indicielle [2]

Pour la méthode du point critique, il faut faire osciller le système en augmentant le gain de l'unique régulateur de la boucle qui est un régulateur proportionnel. On obtient donc $\boldsymbol{K_{CR}}$ le gain critique et $\boldsymbol{T_{CR}}$ la période d'oscillation. Il faut ensuite s'aider du tableau figure 5 pour avoir la valeur des

différents coefficients. Là aussi, il peut être nécessaire de réduire le coefficient K_p pour éviter le dépassement.

Type	K_p	T_i	T_d
P	$0.5K_{cr}$		
PI	$0.4K_{cr}$	$0.8T_{cr}$	
PID	$0.6K_{cr}$	$0.5T_{cr}$	$0.125T_{cr}$

FIGURE 5 – Calcul des valeurs des coefficients à partir du point critique [2]

La troisième méthode est celle de Åström et Hägglund qui se décompose tout comme celle de Ziegler et Nichols en deux sous méthodes, une en utilisant la réponse indicielle et l'autre en utilisant le point critique.

La méthode de la réponse indicielle suite le même principe que celle de Ziegler et Nichols sauf qu'elle prend en compte un paramètre supplémentaire, le temps apparent **T**. Le temps apparent est le temps pour que la réponse atteigne 63% de sa valeur asymptotique moins le retard. Ces coefficients permettent de calculer $\boldsymbol{K_n = K_0 \frac{L}{T}}$ et $\boldsymbol{\tau = \frac{L}{L+T}}$ qui permettront ensuite de déterminer a_0, a_1, a_2 servant à calculer K_p, T_i, T_d à l'aide d'une fonction de la forme :

$$\boldsymbol{f(\tau) = a_0.exp(a_1\tau + a_2\tau^2)} \tag{4}$$

La méthode avec le point critique ressemble aussi à celle de Ziegler et Nichols avec comme différence une variable supplémentaire $\boldsymbol{K_0}$ correspondant au gain statique. Comme pour la méthode avec la réponse indicielle, une fonction de la forme de l'équation 4 permettra de déterminer a_0, a_1, a_2 servant à calculer K_p, T_i, T_d.

2.2 Contrôle du couple pour l'asservissement de la position

Afin de pouvoir contrôler les mouvements du bras pour effectuer une trajectoire ou compenser des forces extérieures, il est possible d'asservir le robot en position en lui envoyant une commande en couple comme cela est expliqué dans *Modern Robotics* [1].

Comme son nom l'indique, l'entrée qui sera prise en compte est la valeur du couple des articulations. Dans un premier temps, pour l'étude de ce contrôle, seulement une articulation sera considérée. La dynamique pour une articulation s'écrit sous la forme

$$\boldsymbol{\tau = M\ddot{\theta} + mgrcos(\theta) + b\dot{\theta}} \tag{5}$$

avec $\boldsymbol{\tau}$ un couple moteur, **M** la matrice inertielle, $\boldsymbol{\theta}$ l'angle du moteur, **m** la masse du lien, **r** la distance entre l'articulation et le centre de gravité du

lien et **g** la constante de gravité terrestre. L'ensemble de ces paramètres et de ces forces est représenté sur la figure 6.

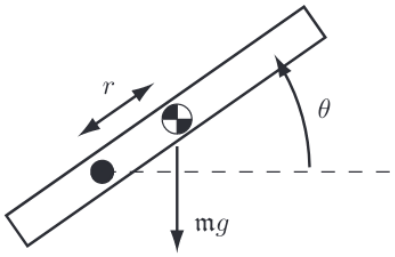

FIGURE 6 – Représentation des couples exercés sur une seule articulation [1]

Comme présenté dans la partie précédente, il y a deux types de contrôle, le *feedforward control* et le *feedback control*. Ces méthodes permettent de corriger l'erreur et suivre la trajectoire avec chacune ses avantages et inconvénients.

La formule du couple *feedforward* est la suivante :

$$\boldsymbol{\tau} = \tilde{\boldsymbol{M}}\ddot{\boldsymbol{\theta}}_{\boldsymbol{d}}(\boldsymbol{t}) + \tilde{\boldsymbol{h}}(\dot{\boldsymbol{\theta}}_{\boldsymbol{d}}(\boldsymbol{t}), \boldsymbol{\theta}(\boldsymbol{t})) \tag{6}$$

avec $\boldsymbol{\tau}$ un couple moteur, $\tilde{\boldsymbol{M}}$ la matrice inertielle calculée, $\boldsymbol{\theta_d}$ l'angle désiré et $\tilde{\boldsymbol{h}}$ une fonction des couples extérieurs à compenser. Si le modèle dynamique était exact, ce couple correspondrait à la commande à envoyer pour que le robot atteigne son objectif. Cependant, il est impossible d'avoir un modèle dynamique parfait, il est donc nécessaire de l'utiliser en parallèle du *feedback control*.

Le *feedback control* consiste, pour le robot Reachy, en un contrôle proportionnel, dérivatif et intégratif qui dépend de l'erreur mesurée. Le couple à donner est donc de la forme :

$$\boldsymbol{\tau} = \boldsymbol{K_p}\boldsymbol{\theta_e} + \boldsymbol{K_i}\int \boldsymbol{\theta_e}(\boldsymbol{t})\boldsymbol{dt} + \boldsymbol{K_d}\dot{\boldsymbol{\theta}}_{\boldsymbol{e}} \tag{7}$$

avec $\boldsymbol{K_i}, \boldsymbol{K_p}, \boldsymbol{K_d}$ des gains positifs et $\boldsymbol{\theta_e}$ l'angle de l'erreur.

La meilleure solution reste la combinaison des deux méthodes de contrôle pour prendre en compte les erreurs au cours du temps tout en améliorant les performances grâce au modèle dynamique comme le montre la figure 7.

La valeur du couple associé à la combinaison des deux types de contrôle est donnée par l'équation :

$$\boldsymbol{\tau} = \tilde{\boldsymbol{M}}(\ddot{\boldsymbol{\theta}}_{\boldsymbol{d}} + \boldsymbol{K_p}\boldsymbol{\theta_e} + \boldsymbol{K_i}\int \boldsymbol{\theta_e}(\boldsymbol{t})\boldsymbol{dt} + \boldsymbol{K_d}\dot{\boldsymbol{\theta}}_{\boldsymbol{e}}) + \tilde{\boldsymbol{h}}(\dot{\boldsymbol{\theta}}_{\boldsymbol{d}}(\boldsymbol{t}), \boldsymbol{\theta}(\boldsymbol{t}) \tag{8}$$

Ce type de contrôle du couple est représenté par le schéma bloc de la figure 8.

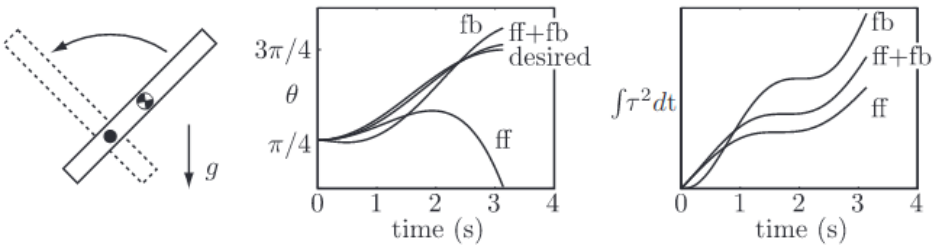

FIGURE 7 – Comparaison entre le feedforward control, le feedback control et la combinaison des deux [1]

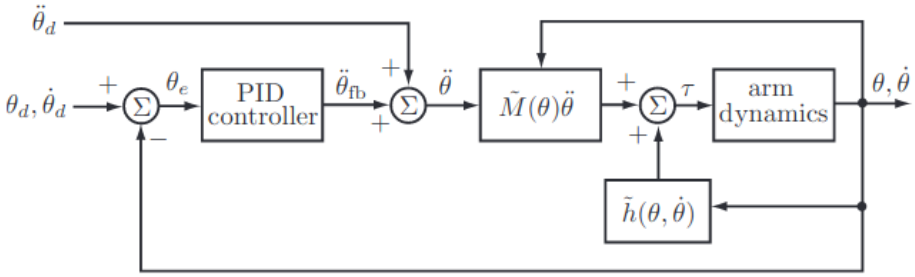

FIGURE 8 – Schéma bloc représentant la combinaison du feedforward control et du feedback control [1]

Le bras du robot Reachy aillant plusieurs articulations en série, la généralisation des différents contrôles pour de multiples contrôleurs est indispensable. Pour cela, le calcul reste le même sauf que $\theta, \theta_d, \theta_e$ vont être des vecteurs de taille n et K_i, K_p, K_d des matrices avec l'ensemble des coefficients positifs.

Finalement, pour pouvoir appliquer le contrôle du couple avec Reachy, il nous reste à savoir comment il est possible de mesurer le couple sur ses différents moteurs. Il est expliqué dans Dynaban, an Open-Source Alternative Firmware for Dynamixel Servo-Motors [14] que les articulations de Reachy sont réalisées à l'aide de moteur MX-64 qui sont équipés de capteur de courant. L'intensité du courant étant proportionnel au couple, il semblerait qu'il soit possible de mesurer l'intensité du courant pour obtenir le couple pour l'asservissement. Cependant, d'après le firmware des moteurs, cela n'est pas possible à cause d'une résistance branchée en série du moteur qui engendre du bruit sur la mesure de l'intensité. Cette mesure est d'autant plus difficile, car elle dépend d'un bus de communication dont les caractéristiques sont peu connues. Il est tout de même possible d'obtenir le couple des moteurs, pour cela, il faut mesurer le signal de modulation de largeur d'impulsion (PWM) ainsi que la vitesse actuelle du moteur. Pour finalement relier la tension $\mathbf{u}$, les couples $\boldsymbol{\tau}$ et la vitesse de rotation $\boldsymbol{\omega}$ avec la formule suivante :

$$\boldsymbol{u(t+dt) = k_e \omega(t) + \frac{R}{k_e}(\tau_0(t) + \tau_a(t) + \tau_f(t))} \tag{9}$$

avec $\boldsymbol{R}, \boldsymbol{K_e}$ les constantes du moteur à courant continu.

2.3 Bras zéro gravité

Un bras zéro gravité a pour but de compenser les forces émises par la gravité terrestre. Cela signifie que le bras est mis dans une position initiale et si aucune force n'est exercée sur lui, il ne se déplacera pas. L'utilisateur peut donc utiliser et déplacer le bras du robot plus facilement, car il n'a pas besoin de compenser la gravité qui est déjà gérée par le robot. Le déplacement du bras robot par l'utilisateur permet la réalisation d'actions très variées sans rendre la programmation du robot difficile, les mouvements de l'action étant réfléchie et effectuée par l'utilisateur directement.

Il existe de nombreuses méthodes pour effectuer un bras zéro gravité. Arakelian Vigen [15] décrit quelques-unes de ces méthodes. Une méthode qui est utilisée depuis longtemps est l'utilisation d'un contrepoids pour compenser la force de gravité. Il y a eu ensuite une autre solution utilisant un ressort, qui a pour avantage que la force exercée par le ressort dépend de la déformation de ce dernier, rendant la compensation plus modulable au mouvement. Dans certaines situations, l'utilisation d'un système auxiliaire peut être utile pour compenser les forces en se rajoutant en parallèle sur le système déjà existant. Ces systèmes auxiliaires utilisent principalement des systèmes pneumatiques ou hydrauliques pour effectuer la compensation.

Ces dispositifs n'étant pas disponibles sur Reachy, il n'est donc pas possible d'utiliser ces méthodes. Cependant, une autre méthode est décrite dans Sensorless Friction-Compensated Passive Lead-Through Programming for Industrial Robots [16]. Cette méthode consiste en un contrôle moteur à l'aide d'un asservissement par contrôle du couple. Pour pouvoir effectuer ce contrôle, il est nécessaire de désactiver les servo-contrôleurs de bas niveau dans les articulations et d'envoyer comme consigne de contrôle, le couple correspondant à la compensation de la gravité, des frottements moteurs et éventuellement d'un objet qu'il porterait.

3 Normes de sécurité

Il est nécessaire pour un robot en contact avec les humains de respecter des normes de sécurité. Pour cela, il faut définir les dangers liés au robot et mettre en place des solutions pour réduire les risques.

3.1 Dangers inhérents au robot

Le robot peut représenter un risque pour l'homme sur différents aspects.

3.1.1 Arrêt d'urgence du robot

L'accès direct à une fonction d'arrêt d'urgence du robot est une chose essentielle lorsque l'on parle de collaboration humain-robot.

Dans la plupart des robots traditionnels, lorsque l'on coupe l'alimentation du robot, les moteurs se coupent également et donc le robot se relâchent. Cela peut entrainer des risques pour les personnes aux alentours. Par exemple dans le cas d'un bras robot lourd ou portant un objet lourd qui tomberait rapidement en cas de relâchement des moteurs.

Une majorité des bras robot collaboratifs actuels utilisent des freins activés par défaut lorsqu'ils sont éteints. Il faut alimenter le robot en courant pour désactiver ces freins. Ils sont réactivés en cas d'arrêt d'urgence du robot ou même lorsque le robot ressent un couple d'opposition trop important.
Pour empêcher le cas ou un humain serait bloquer par le robot après un arrêt d'urgence, il est quand même possible, en exerçant un peu de force, de déplacer le bras pour se libérer. Cette solution est onéreuse et demande de repenser tout le système de fonctionnement du cobot.

Une solution possible pour le Reachy serait d'implémenter le bras antigravité comme mode de fonctionnement par défaut ou en cas d'arrêt d'urgence, ce qui permettrait donc de bloquer le bras dans sa position en cas de problème, mais qu'il soit facilement déplaçable.

3.1.2 Robot collaboratif fixe

Les robots collaboratifs sont par définition en interaction directe avec les humains. Hommes et cobots travaillent dans un espace partagé ou à proximité les uns des autres.
Cela entraine des risques pour les hommes et donc des normes doivent être mises en place pour les protéger.

Les normes ISO 10218-1/2 et la spécification technique ISO/TS 15066 :2016 ont défini quatre formes de fonctionnement pour les robots collaboratifs comme on peut le voir figure 9 :

- ***Safety-rated monitored stop*** **(SMS)** : Le robot s'arrête lorsque l'opérateur entre dans la zone de travail du robot.
- ***Hand guiding*** **(HG)** : Ce mode permet à l'opérateur de déplacer manuellement le cobot lorsqu'il est à côté. Le cobot peut fonctionner à pleine puissance si personne n'est proche.
- ***Speed and separation monitoring*** **(SSM)** : Le robot fonctionne à pleine vitesse lorsque le robot est seul dans la zone de travail et diminue sa vitesse proportionnellement à la distance qui le sépare de l'opérateur lorsque celui-ci est présent dans la zone.
- ***Power and force limiting*** **(PFL)** : Ces cobots ont une puissance

et une force limitées et contiennent de nombreux capteurs qui leur permettent de détecter des forces de résistances anormales pour empêcher toutes collisions.

FIGURE 9 – Mode de collaboration des robots [3]

Dans le cadre de notre projet, le robot Reachy utilise des servomoteurs Dynamixsel RX-64 qui possède une vitesse maximale sans charge de 78rpm et un couple de décrochage de 7.3N. Il est donc assez simple pour un homme de bloquer le déplacement du robot à la main. Dans le cas d'un mouvement brusque ou inattendu du robot ne possédant aucun objet dans la main, les conséquences d'une collision avec l'opérateur seraient sans grande gravité. On peut donc considérer que le robot fonctionne en PFL.
Pour réduire les risques de chocs avec l'utilisateur, on peut définir la zone maximale d'action du robot (zone en bleu figure 10) et de créer un marquage au sol(en vert figure 10) pour la mettre en évidence. Cela permettrait à l'opérateur d'être plus attentif dans la zone et donc de limiter au minimum des collisions imprévues.

Dans le cas ou le robot Reachy possède un objet pouvant blesser un humain dans sa main, le fonctionnement du Reachy doit être modifié.

Une solution serait d'applique le fonctionnement SSM. Pour cela, il faut définir plusieurs zones, en fonction de la zone dans laquelle l'opérateur se trouve, le robot adapte sa vitesse de fonctionnement pour limiter les risques de collision. Ces zones pourraient être définies comme dans la figure 11

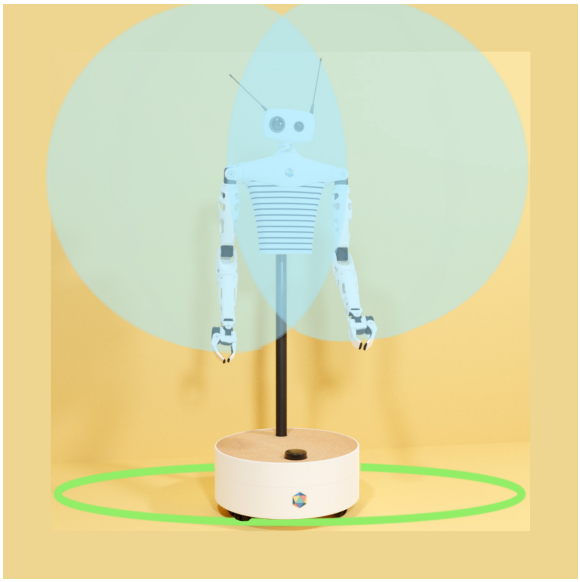

FIGURE 10 – Zone d'action du robot Reachy

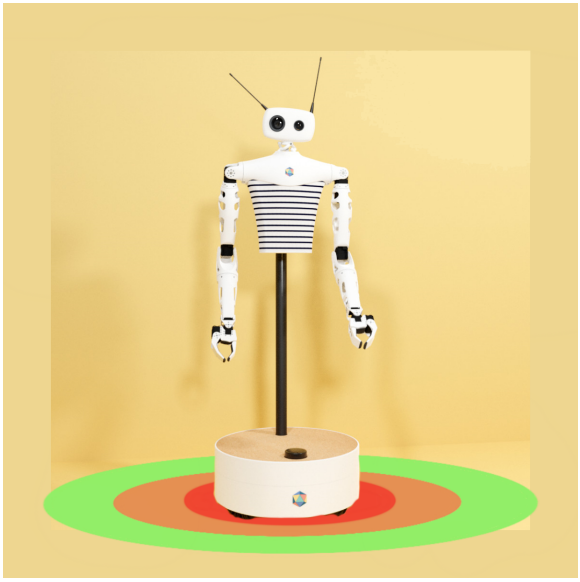

FIGURE 11 – Zone de danger du robot Reachy

Pour mettre en place cette solution, il faut que le robot soit en capacité de détecter la présence de l'opérateur et de calculer la distance avec celui-ci.

L'article [3] propose une solution pour détecter la distance qui sépare un robot d'un humain. Pour cela, elle utilise des kinects et des capteurs laser. Les données des kinects sont utilisées pour calculer la distance. Un premier traitement de l'image vient retirer le robot de l'image, puis un algorithme vient calculer la distance aux obstacles à partir des images transmises.

La vitesse du robot s'adapte ensuite proportionnellement à la distance qui le sépare de l'homme. L'ordinateur qui effectue les calculs est également capable de savoir lorsque les calculs de distance sont dysfonctionnels ou que les caméras ne renvoie pas d'images. Dans ce cas, les capteurs laser, permettant de détecter ou non la présence d'un humain dans la zone de travail, sont utilisés et le robot s'arrête simplement lorsque ces capteurs détectent quelque chose.

Il est également possible d'utiliser un lidar pour scanner les alentours du robot et ainsi pouvoir récupérer les distances aux objets environnants.

Dans le cadre du projet, l'objectif est que le robot Reachy reproduise un mouvement dynamique produit par un humain. Un bon moyen de vérifier la véracité du mouvement serait en effectuant un lancé précis d'un objet.
Dans ce cas, il faut pouvoir définir une zone de travail plus grande, correspondante à la zone dans laquelle le robot à la capacité de lancer l'objet.
Il est également possible de mettre en place un système sonore qui permettrait d'avertir toutes personnes se trouvant aux alentours pour éviter tout choc de l'objet lancé avec un humain.

3.1.3 Chute du robot

Le robot Reachy possède une base roulante, il est donc possible que le robot chute lors de son utilisation.
Tout d'abord, en cas d'obstacles sur sa route, cela peut entrainer un déséquilibre et donc une chute du robot.
De plus, une accélération ou freinage brusque, peut également représenter un risque de chute.
Enfin, une pente peut être une difficulté pour les robots mobiles. Pollen Robotics a montré que le robot Reachy peut monter une pente de 17° maximum sans entrainer la chute du robot.

Le robot pèse environ 40kg. Sa chute peut donc à la fois causer des dégâts matériels, mais aussi mettre en danger les personnes qui sont proches du robot.

3.2 Dangers pour le robot

Il existe aussi de nombreuses situations dans laquelle le robot peut être endommagé.

La structure mécanique du corps du robot Reachy est entièrement imprimé en 3D. Le corps est donc assez fragile et il y a un risque de casser certaines pièces pendant l'utilisation en faisant un mouvement trop brusque ou s'il y a une collision entre deux parties de son corps.
Les chutes du robot, comme vu partie 3.1.3, peuvent également entraîner

d'importants dégâts matériels.

Sur un robot, il existe aussi des problèmes liés à la partie électrique.
Sur le robot Reachy, toute la partie électrique est visible ou caché par un t-shirt. Il serait donc possible d'endommager le système électrique en cas de collision avec le robot.
Il y a aussi un risque de court-circuit qui peut entrainer un départ de feu pouvant causer des dommages au circuit et au robot en général. Cela peut même, dans le pire des cas, mener à un incendie.
Les moteurs sont reliés entre eux par des câbles à l'extérieur du robot. Il est donc possible que les câbles s'accrochent à un objet, ce qui pourrait détériorer les câbles ou même entrainer la chute du robot.
Enfin, il y a un risque de surchauffe des moteurs. Reachy possède des moteurs Dynamixsels qui ont des capteurs de température dont ce risque est faible, mais en cas de surchauffe trop rapide, il serait quand même possible d'endommager les moteurs. Il est possible de réduire la température maximale des capteurs pour diminuer ces risques, mais cela revient donc à diminuer la puissance maximale du robot.

3.3 Dangers liés à la téléopération

La téléopération d'un robot permet de contrôler à distance le robot, ce qui apporte de nombreux avantages, mais également de nouveaux risques.

3.3.1 Problèmes de transmission de données

La téléopération utilise un transfert de données souvent permis par une connexion internet. La téléopération d'un robot n'est pas un problème tant que le robot peut communiquer correctement avec l'opérateur et faire ce qu'il veut qu'il fasse. Une mauvaise connexion ou une perte de connexion détériorerait la transmission des données au robot. Cela pourrait avoir pour conséquence que le robot ne puisse pas répondre aux demandes malgré la commande à distance ou la réalisation d'actions inattendues dues à un mauvais transfert de données. Il est donc important de s'assurer de la qualité de la connexion avant l'utilisation du robot.

3.3.2 Hacking

L'utilisation de la téléopération implique la nécessité d'un transfert de données entre l'opérateur et le robot. Généralement, ces données sont transmises par internet, mais cela peut laisser place à des piratages informatiques.

D'après l'article [17], on peut classifier les attaques potentielles en trois catégories :

— Attaque de modification d'intentions : l'objectif est de modifier la commande qui a été envoyé par l'opérateur. Le pirate peut par exemple

modifier un paquet envoyé pour changer la position finale de la main du robot.
— Attaque de manipulation d'intentions : cela consiste à modifier la réponse du robot à l'opérateur pour lui faire penser que ses actions ont été effectuées alors que ce n'est pas le cas.
— Attaque par détournement : Le pirate prend le contrôle du robot pendant une durée plus ou moins longues. Pendant cette période les ordres de l'opérateur sont totalement ignorés

Il est également possible de catégoriser les profils d'attaque :
— observateur : le pirate observe les données transmises et peut les modifier.
— intermédiaire *(man-in-the-middle)* : le pirate est l'intermédiaire entre les deux points de transmission.

Pour contrer les attaques d'observation, il est possible de chiffrer les données transmises de bout en bout.
Les attaques d'intermédiaires sont les plus dangereuses, car elles sont généralement plus dures à détecter et il est plus difficile de s'en protéger.

Dans ce projet, Pollen Robotics utilise le framework gRPC pour gérer la transmission des données. gRPC est un framework de *Remote Procedure Call* développé par Google. Il se base sur les protocoles HTTP pour la transmission de paquets. Pour limiter au mieux les attaques, il est possible pour les développeurs qui utilisent ce framework de sécurisé les canaux de transmission et d'ajouter un système d'authentification pour la transmission de données.

3.3.3 Problème de visions

Lors de la téléopération du robot Reachy, l'opérateur voit à travers le casque ce que voit le robot. Si le robot ne peut pas regarder dans toutes les directions, alors l'opérateur ne pourra pas avoir les informations de tout l'environnement qui entour le robot. Dans le cadre d'une utilisation d'un robot sur le terrain, la présence d'angle mort peut être particulièrement dangereux lorsqu'il travaille, car cela l'empêche de détecter d'éventuels dangers sur le terrain.

Pour réduire ces risques, il est possible d'ajouter certains capteurs ou autres éléments permettant de réduire au minimum ses angles morts. Une base rotative permettrait au robot de voir ce qui se trouve derrière lui. Il est également possible d'ajouter des capteurs tel qu'un lidar qui permettrait de connaitre en temps réel son environnement.

Conclusion & perspectives

La paramétrisation de la trajectoire d'un bras humain est un problème très complexe. En effet, non seulement bouger le notre bras pour atteindre un objectif en évitant des obstacles est un acte logique pour nous, et de plus le mouvement humain peut contenir un grand nombre d'autres informations liées aux intentions de l'opérateur, en fonction de son objectif et des risques par exemple.

Ainsi, la programmation par démonstration apparait comme étant la solution idéale pour répondre à ce problème. Toutes ces informations sont donc transmises indirectement, sans avoir à les quantifier ni à les caractériser.

C'est dans ce cadre que l'entreprise *Pollen Robotics*, au travers de M. Rémi FABRE, a proposé ce projet. Ce dernier consiste donc à se servir de la téléopération, déjà mise au point à l'aide d'un casque de réalité virtuelle, comme base de programmation par démonstration pour faire suivre dynamiquement des trajectoires humaines de manière autonome à leur robot Reachy par apprentissage.

Nous avons donc, durant ce travail d'état de l'art, effectuer des recherches sur les différentes étapes et méthodes permettant d'arriver à ces objectifs.

Dans un premier temps, nous nous sommes concentrés sur la génération d'une trajectoire issue de la programmation par démonstration, ainsi que sur la manière de faire reproduire une trajectoire par apprentissage au robot Reachy.

Ensuite, nous avons travaillé sur le problème du suivi dynamique de trajectoire, grâce à des méthodes utilisant du *feedforward* et du *feedback*, pour réaliser un contrôle en couple des articulations. Cela nous a aussi permis de nous renseigner sur les technologies de bras anti-gravité, qui découle de ce contrôle.

Enfin, nous nous sommes concentrés sur les risques et les normes de sécurité à prendre en compte lors de la réalisation d'un tel projet, au niveau des différents aspects que contient la mise en application, autour du robot, de l'humain, de la téléopération...

En conclusion, ce document présente des solutions, qui permettent de résoudre certains problèmes liés à la réalisation de mouvements dynamiques par un robot, sur lesquelles nous pourrons nous baser pour réaliser notre propre solution.

Références

[1] Park Frank C. Lynch Kevin M. *Modern robotics : mechanics, planning, and control*. Cambridge University Press, 2017.

[2] Mudry F. Ajustage des paramètres d'un régulateur pid. *Ecole d'ingenieurs du canton de Vaud*, pages 8–17, 2002.

[3] E. Magrini ; F. Ferraguti ; A. Jacopo Ronga ; F. Pini ; A. De Luca ; F. Leali. Human-robot coexistence and interaction in open industrial cells. *Robotics and Computer-Integrated Manufacturing*, 61 :101846, 2020.

[4] Edwin G Johnsen. Man, teleoperators, and robots : An optimum team for space exploration. *Journal of Spacecraft and Rockets*, 9(7) :554–556, 1972.

[5] Abhishek Jha, Shital S Chiddarwar, Veer Alakshendra, and Mayur V Andulkar. Kinematics-based approach for robot programming via human arm motion. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 39(7) :2659–2675, 2017.

[6] Nikolaos Mavridis, Georgios Pierris, Paolo Gallina, Nikolaos Moustakas, and Alexandros Astaras. Subjective difficulty and indicators of performance of joystick-based robot arm teleoperation with auditory feedback. In *2015 International Conference on Advanced Robotics (ICAR)*, pages 91–98. IEEE, 2015.

[7] Sagar Shirwalkar, Alok Singh, Kamal Sharma, and Namita Singh. Telemanipulation of an industrial robotic arm using gesture recognition with kinect. In *2013 International Conference on Control, Automation, Robotics and Embedded Systems (CARE)*, pages 1–6. IEEE, 2013.

[8] Sungman Park, Yeongtae Jung, and Joonbum Bae. A tele-operation interface with a motion capture system and a haptic glove. In *2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 544–549. IEEE, 2016.

[9] Eui S Jung, Jaeho Choe, and Sung H Kim. Psychophysical cost function of joint movement for arm reach posture prediction. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 38, pages 636–640. SAGE Publications Sage CA : Los Angeles, CA, 1994.

[10] Jing Zhao, Biyun Xie, and Chunyu Song. Generating human-like movements for robotic arms. *Mechanism and Machine Theory*, 81 :107–128, 2014.

[11] Stefan Schaal, Jan Peters, Jun Nakanishi, and Auke Ijspeert. Control, planning, learning, and imitation with dynamic movement primitives. In *Workshop on Bilateral Paradigms on Humans and Humanoids : IEEE International Conference on Intelligent Robots and Systems (IROS 2003)*, pages 1–21, 2003.

[12] Alexander Hewitt, Chenguang Yang, Yong Li, and Rongxin Cui. Dmp and gmr based teaching by demonstration for a kuka lbr robot. In *2017*

23rd International Conference on Automation and Computing (ICAC), pages 1–6. IEEE, 2017.

[13] Jacopo Aleotti and Stefano Caselli. Robust trajectory learning and approximation for robot programming by demonstration. *Robotics and Autonomous Systems*, 54(5) :409–413, 2006.

[14] Fabre Rémi ; Rouxel Quentin ; Passault Grégoire ; N'Guyen Steve ; Ly Olivier. *Dynaban, an Open-Source Alternative Firmware for Dynamixel Servo-Motors*. Springer International Publishing, 2017.

[15] Arakelian Vigen. Gravity compensation in robotics. *Advanced Robotics*, 30 :1–18, 2015.

[16] Stolt Andreas ; Bagge Carlson Fredrik ; Ghazaei Mahdi ; Lundberg Ivan ; Robertsson Anders ; Johansson Rol. Sensorless friction-compensated passive lead-through programming for industrial robots. *nternational Conference on Intelligent Robots and Systems (IROS)*, pages 3530–3537, 2015.

[17] Tamara Bonaci ; Jeffrey Herron ; Tariq Yusuf ; Junjie Yan† ; Tadayoshi Kohno ; Howard Jay Chizeck. To make a robot secure : An experimental analysis of cyber security threats against teleoperated surgical robots. 2015.