Vers des robots animés : outils et méthodes pour l'intégration d'artistes animateurs dans la conception de robots sociaux expressifs

Etienne Balit

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministériel : 25 mai 2016

Présentée par

Etienne Balit

Thèse dirigée par **Patrick Reignier**
et codirigée par **Dominique Vaufreydaz**

préparée au sein de l'**Inria**, du **Laboratoire d'Informatique de Grenoble** et de l'**École Doctorale de Mathématiques, Sciences et Technologies de l'Information, Informatique**

Vers des robots animés

Outils et méthodes pour l'intégration d'artistes animateurs dans la conception de robots sociaux expressifs

Jury composé de :

M. Pierre De Loor
Professeur à l'École Nationale d'Ingénieurs de Brest, Président
M. Mohamed Chetouani
Professeur à l'Université Pierre et Marie Curie, Rapporteur
M. Rachid Alami
Directeur de Recherche au CNRS, Rapporteur
M. Patrick Reignier
Professeur à Grenoble Institut National Polytechnique, Directeur de thèse
M. Dominique Vaufreydaz
Maître de Conférence à l'Université Grenoble Alpes, Co-Directeur de thèse

Abstract

Once only experimental devices, social robots are beginning to arrive in homes, schools, hospitals and nursing homes. They are becoming companions of our education, health and well-being. These first commercial launches of social robots confirm the importance of the human-robot interaction anticipated by researchers. Indeed, since non-verbal communication forms a major part of communication in humans, we cannot help but interpret the movements and behaviours of robots as the expression of mental states. This effect of the expressiveness of social robots implies that they must be carefully designed to avoid disrupting the interaction.

The fields of video games and animated movies have long been interested in the design of such expressive movements. Many tools and methods have been developed to support the creation of virtual character movements. They have made possible the combination of the latest technological advances with the talents of animation artists who are masters in the art of bringing life into characters. From this observation, the focus of this thesis is the design of tools and methods to integrate animation artists into the design process of social robots. In particular, we investigate the minimal adaptations of the creative process of animation artists made necessary by the specific characteristics of the robotic medium.

The contributions of this thesis are as follows :

— A framework for prototyping animated social robots, based on mapping and synchronizing the physical robot with an animated virtual robot in a 3D character animation tool.

— A second robotic animation tool integrating direct manipulation into the keyframing animation process.

— A methodology for designing parametric animation through the collaboration of an animation artist and a developer, as well as the tools and software interface to support this collaboration.

— The evaluation by animation artists of the robotic animation tool by direct manipulation and of the method of generalization of an animation to new geometric constraints.

Résumé

Autrefois seulement objets de laboratoire, les robots sociaux commencent à arriver dans les foyers, les écoles, les hôpitaux et les maisons de retraite, et deviennent les compagnons de notre éducation, de notre santé et de notre bien-être. Ces premières commercialisations de robots sociaux confirment l'importance de l'interaction homme-robot anticipée par les chercheurs. En effet, la communication non verbale formant une part majeure de la communication chez l'Homme, celui-ci ne peut s'empêcher d'interpréter les mouvements et les comportements des robots comme l'expression d'états mentaux. Cet impact de l'expressivité des robots sociaux implique que leur conception se doit d'être soignée pour ne pas perturber l'interaction.

Les domaines du jeu vidéo et des films d'animation se sont intéressés depuis longtemps à la création de tels mouvements expressifs. De nombreux outils et méthodes ont ainsi été développés afin de faciliter la création de mouvements de personnages virtuels. Ces outils ont permis de combiner les avancées technologiques aux talents d'artistes animateurs passés maîtres dans l'art de donner vie à des personnages. Partant de ce constat, l'objet de cette thèse est la conception d'outils et de méthodes permettant d'intégrer des artistes animateurs dans le processus de conception des robots sociaux. Nous explorons en particulier les adaptations minimales du processus créatif des artistes animateurs rendues nécessaires par les spécificités du medium robotique.

Les contributions de cette thèse sont les suivantes :

— Un framework pour le prototypage de robots sociaux animés, fondé sur la mise en correspondance et la synchronisation du robot physique avec un robot virtuel animé dans un outil d'animation de personnage 3D.

— Un second outil d'animation robotique intégrant la manipulation directe dans le processus d'animation par poses clés.

— Une méthodologie pour la création d'animation paramétrique au travers de la collaboration d'un artiste animateur et d'un développeur, ainsi que les outils et interface logiciel facilitant cette collaboration.

— L'évaluation par des artistes animateurs de l'outil d'animation de robot par manipulation directe et de la méthode de généralisation d'une animation à des nouvelles contraintes géométriques.

Remerciements

Je souhaite tout d'abord remercier Mohamed Chetouani et Rachid Alami de l'honneur qu'ils me font en ayant accepté d'être rapporteurs, ainsi qu'à Pierre De Loor d'avoir accepté de faire partie du jury en tant qu'examinateur.

Je tiens en particulier à remercier mes directeurs de thèse, Patrick Reignier et Dominique Vaufreydaz, pour tout leur soutien. Je leur suis reconnaissant pour leur disponibilité et leurs encouragements aux cours de ces années de thèse.

Je souhaite remercier la direction de l'Ecole Ariès de Meylan qui m'a accueilli afin de mener les tests utilisateurs ainsi que les étudiants qui ont accepté d'y participer. Je tiens également à remercier Nadine Mandran pour toute son aide dans la préparation de cette évaluation.

Merci également à l'équipe d'Amiqual4Home avec qui j'ai appris la fabrication. Merci en particulier à Stan et Nicolas pour leurs précieux conseils et pour avoir fait vivre ce fablab si utile pour nos recherches. Je veux également remercier l'équipe du projet Poppy de l'Inria Bordeaux pour la conception de ce robot avec lequel j'ai passé tant de temps.

Je veux remercier Lukas, Maxime, Marvin, JAD, Jérôme, Amr, Laurence, Pierre, Lucas, Varun, Eva, Fabien, Emeric et tous les collègues de l'Inria avec qui j'ai partagé de très bons moments et de longues discussions.

Cette thèse n'aurait pas été possible sans le soutien indéfectible de ma famille et de mes amis. Merci à mes parents et à mes frères pour avoir toujours été là pour moi dans les moments difficiles. Merci également à Julio, Romain, Jem, Johan, Kevin, Axel et Julien pour leur amitié si précieuse. Enfin, merci Nachwa pour chaque moment passé à mes côtés et pour tous ses encouragements.

Table des matières

Liste des figures

Liste des tableaux

Introduction

Le grand public a vu apparaître ces dernières années un nouveau type de robots dans les salons de nouvelles technologies et dans les médias : les robots sociaux. A la différence des robots industriels qui ont révolutionné l'industrie depuis une soixantaine d'années, ces nouveaux robots ont pour objectif d'aider les utilisateurs dans leur vie quotidienne.

L'éducation et la santé font partie des principaux domaines d'applications envisagés pour ce nouveau type de robots. On peut citer parmi les cas d'usages le fait de motiver un enfant dans son apprentissage d'une nouvelle langue (Westlund et al., 2015), des mathématiques (Brown and Howard, 2014) ou encore du jeu d'échecs (Leite et al., 2011). Les robots sociaux pourraient également aider les personnes âgées à améliorer leur qualité de vie et rester autonomes plus longtemps en favorisant le maintien d'une activité sociale et cognitive. Ils sont également utilisés pour assister les équipes de soins dans les hôpitaux (Jeong et al., 2017) ou les maisons de retraite (Badeig et al., 2016).

Ces différents usages nécessitent de créer des interactions sociales engageantes entre les robots et les utilisateurs. La communauté scientifique de l'Interaction Homme-Robot s'intéresse depuis longtemps à ce sujet d'étude, recherchant la meilleure façon de concevoir les robots sociaux et leurs interactions en fonction de la tâche à effectuer.

Alors que les robots prennent déjà des formes très variées, du robot aspirateur Roomba (Forlizzi and DiSalvo, 2006) au bras robot d'assistance Kinova (Campeau-Lecours et al., 2017) en passant par le robot Jibo [1], les communautés de l'Interaction Homme Machine et d'Interaction Homme Robot ont présenté ces dernières années de nombreux meubles robotisés, tels qu'une lampe (Hoffman et al., 2008) et un repose-pied (Sirkin

1. https://www.jibo.com - dernière visite : 01/11/2019

et al., 2015). Cette recherche de la meilleure forme pour un robot social est notamment motivée par la *Vallée de l'étrange*, une expression proposée initialement par Mori (1970) pour décrire le fait que les personnes perçoivent négativement un robot lorsque celui-ci ressemble de façon imparfaite à un humain.

Une composante essentielle de l'interaction entre un utilisateur et un robot est la communication non verbale de ce dernier. En effet, la communication non verbale forme une part majeure de la communication chez l'Homme (Vinciarelli et al., 2009). Les mouvements des personnes que nous croisons ne sont pas de simples déplacements de corps dans l'espace. Ils deviennent par notre interprétation l'expression d'états mentaux, d'émotions ou encore d'intentions. Ces compétences apprises pour faciliter nos interactions sociales sont également à l'œuvre lorsque nous observons des robots bouger. Comme face à un humain, nous ne pouvons nous empêcher d'interpréter la façon de bouger des robots comme l'expression d'états mentaux. Nous leur attribuons ainsi des compétences et des attributs particuliers. Cette propension à interpréter les mouvements des robots par anthropomorphisme implique que l'expressivité de ces mouvements doit être prise en compte dès leur conception.

On retrouve ce besoin de concevoir les mouvements d'un personnage pour les jeux-vidéos et les films d'animation. Deux approches coexistent dans ces domaines. La première est de capturer les mouvements d'acteurs afin de les reproduire sur les personnages modélisés en 3D (motion capture). La seconde est de faire concevoir les mouvements par des artistes animateurs, formés à la création de mouvements expressifs. Cette seconde approche est particulièrement intéressante car elle a l'avantage de permettre une grande liberté quant à la forme du personnage et au style de ses mouvements.

De nombreux films présentent ainsi des robots d'une grande expressivité et de formes très diverses. Les films Wall-E, Big Hero ou encore Star Wars 7 sont des exemples récents de films dans lesquels des robots expressifs font partie intégrante de l'histoire. Qu'est-ce qui distingue ces robots de fiction des robots réels d'aujourd'hui ? Bien que l'histoire et les situations dans lesquelles ces robots sont plongés influencent notre interprétation de leurs comportements, l'expressivité dégagée par ces robots est saisissante, même en les isolant de la trame narrative. L'expressivité de ces robots de fiction tient en grande partie à la créativité et au savoir-faire des artistes qui ont conçu leurs formes et leurs comportements. Les animateurs appellent cette expressivité

"l'Illusion de Vie", selon l'expression consacrée créée par Frank et Ollie afin de décrire l'objectif ultime de leur travail (Thomas et al., 1995).

Des outils et des méthodes ont été développés dans le domaine des jeux vidéos et des films d'animation pour permettre la collaboration des artistes animateurs responsables de la création des mouvements des personnages et des développeurs chargés de la création des moteurs de rendus d'images de synthèse ou de la logique du jeu.

Notre hypothèse est qu'une même synergie entre artistes et développeurs serait bénéfique au domaine de la robotique sociale en permettant de créer des robots plus expressifs et in fine des interactions plus engageantes pour les utilisateurs. Une telle synergie nécessite de concevoir les outils et les méthodes facilitant cette collaboration. L'objectif de cette thèse est de concevoir ces outils et méthodes qui permettront d'intégrer les artistes animateurs dans le processus de conception des robots sociaux.

Plan de thèse

Dans le premier chapitre, nous explorons les apports théoriques de différentes disciplines sur l'étude du mouvement expressif. Nous présentons ensuite les différentes méthodes de création et de génération d'animation développées dans les domaines des films d'animation et des jeux vidéos.

Dans le deuxième chapitre, nous décrivons un framework de prototypage de robots expressifs animés développé à partir d'un logiciel d'animation 3D. Nous développons deux exemples de réalisations permises par ce système : le prototypage d'une tête robotique expressive et l'animation d'un robot humanoïde.

Dans le troisième chapitre, nous décrivons un outil d'animation robotique développé au cours de cette thèse basé sur l'utilisation du robot comme interface tangible. Nous présentons également des principes pour le retour d'informations sur les contraintes physiques du robot au travers de l'interface graphique.

Dans le quatrième chapitre, nous présentons nos travaux sur la création d'animations paramétriques. Nous présentons une méthode de conception basée sur la collaboration entre un artiste animateur et un développeur ainsi que les outils facilitant cette collaboration. Nous détaillons dans ce cadre un nouvel algorithme d'adaptation d'une animation à de nouvelles contraintes géométriques.

Dans le cinquième chapitre, nous présentons les tests utilisateurs menées avec des artistes animateurs afin d'évaluer notre système d'animation paramétrique en terme d'utilisabilité du logiciel et de la qualité du résultat obtenu selon des animateurs.

Chapitre I

Contexte

A. L'expressivité du mouvement

Quand nous pensons à la communication entre deux personnes, nous pensons tout d'abord au langage, au dialogue. Pourtant, si nous voulions retranscrire pleinement une situation sociale et les échanges d'informations qui prennent place entre deux personnes, il nous faudrait aller au-delà du seul dialogue. Vinciarelli et al. (2009) citent notamment la posture, les gestes, la distance aux autres personnes ou encore la direction du regard.

C'est pour cette raison que l'auteur d'une pièce de théâtre accompagne le dialogue de didascalies. Celles-ci indiquent aux acteurs le ton d'une réplique, les gestes qu'ils doivent faire ou encore les postures qu'ils doivent prendre afin de communiquer aux spectateurs les états mentaux des personnages.

Nous nous intéresserons dans cette partie aux apports de différentes disciplines dans l'étude de la communication non verbale et plus précisément du mouvement expressif.

1. Etude du processus de communication

Une première perspective nous est apportée par l'étude du processus de communication. Le modèle de Shannon-Weaver (Shannon et al., 1969) est l'un des premiers modèles généraux du processus de communication. Bien qu'ayant été défini à l'origine pour décrire les communications intermédiées tel que le téléphone, sa généralité

a permis son utilisation dans de multiples disciplines ayant besoin de modéliser la transmission d'une information.

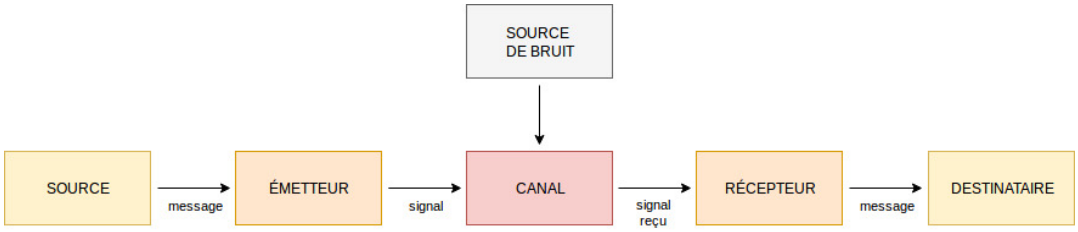


FIGURE I.1 – Modèle de Shannon-Weaver

Le modèle de Shannon-Weaver (figure I.1) décrit la transmission d'une information entre une source et une destination. La source produit un message qui est encodé par l'émetteur en un signal. Ce signal est une forme transmissible vers un récepteur au travers d'un canal de communication. Des sources de bruit peuvent altérer le signal durant cette transmission, de sorte que le signal reçu diffère du signal émis. Le récepteur doit alors décoder le signal reçu pour obtenir le message. Cette décomposition permet de mettre en exergue la nécessité de la compatibilité entre l'émetteur et le récepteur, qui doivent "parler le même langage", et l'importance de la préservation de la qualité du signal transmis. Dans certaines disciplines, comme par exemple en sémiotique, le terme "signifié" est parfois utilisé à la place de "message", le terme "signifiant" à la place de "signal" et le terme "interprétation" à la place de "décodage".

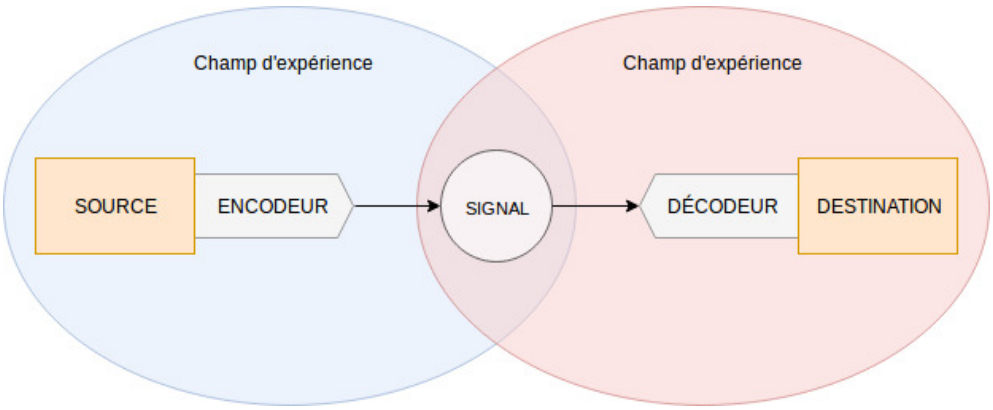


FIGURE I.2 – Modèle de Schramm

Ce modèle a été par la suite enrichi dans le cadre de la communication "humaine". Le modèle de Berlo (Berlo, 1960) détaille par exemple les facteurs psychologiques,

culturels et biologiques influençant les différents stades de la communication humaine. Le modèle de Schramm (Schramm, 1954) (figure I.2) s'appuie quant à lui sur l'importance du champ d'expérience de l'émetteur et du récepteur ainsi que sur l'existence d'un champ d'expérience commun pour qu'une communication puisse être établie.

2. Apports de la Psychologie et de l'Anthropologie

La communication non verbale chez l'Homme est étudiée depuis longtemps. Duchenne et Darwin sont considérés comme ayant initié le domaine par leurs travaux sur les émotions et les expressions faciales.

Dans *Mécanisme de la physionomie humaine*, Duchenne (1862) décrit les mécanismes à l'origine des expressions faciales. Grâce à sa méthode de contraction des muscles de la face par électrisation localisée, il génère des expressions variées pour en isoler les expressions émotionnelles. Il en tire 13 muscles ou groupes musculaires chacun responsable d'une émotion particulière. Il définit ainsi les muscles de l'attention, de la réflexion, de l'agression, de la douleur, de la joie, de la bienveillance, du mépris, de la lascivité, de la tristesse, du pleurer, du pleurnicher, de la surprise et de l'effroi.

Dans *l'Expression des Emotions chez l'Homme et l'Animal*, Darwin (1872) cherche à répondre à la question de l'origine de nos expressions faciales. En se basant sur son expérience acquise lors de son tour du monde, et des résultats de questionnaires menés à partir des photos de Duchenne, il fait l'hypothèse que l'expression de nos émotions est innée et universelle. Il propose que nos comportements expressifs sont des vestiges "*d'habitudes utiles associées*". Ces comportements auraient eu dans notre passé évolutif un lien fonctionnel avec un état mental. Ce lien fonctionnel aurait été perdu par la suite, mais l'expression de nos états mentaux devenant un avantage en soi, les comportements associés auraient été conservés.

Au 20ème siècle, Ekman est sans doute l'auteur ayant eu le plus d'influence dans le domaine. Ses travaux sur l'universalité des 6 émotions de base (joie, tristesse, colère, peur, dégoût et surprise) (Ekman, 1999), bien que débattus jusqu'à aujourd'hui (Jack et al., 2012), sont parmi les plus cités dans l'étude des émotions.

L'article fondateur d'Ekman et Friesen (Ekman and Friesen, 1969) définit une grille de lecture du répertoire des comportements non verbaux. Les auteurs y décrivent 3 dimensions permettant de classer les comportements : leur origine, leur encodage et leur usage.

L'origine se réfère à l'acquisition de ce comportement non verbal par un émetteur ou un récepteur. Il existe deux causes possibles pour l'universalité d'un comportement dans une espèce. Premièrement, lorsque celui-ci est inné car lié à un substrat biologique hérité, comme nos réflexes par exemple. Deuxièmement, lorsque celui-ci a pour origine des expériences partagées par tous les individus de cette espèce, comme utiliser ses mains pour porter de la nourriture à sa bouche. Il peut également être variable lorsqu'il est lié à l'expérience d'un individu, elle-même influencée par sa culture, son origine sociale voire son vécu individuel.

L'encodage se réfère à la relation entre le comportement (le signifiant) et son sens (le signifié). Cette relation est dite arbitraire lorsque l'action ne ressemble pas à ce qu'elle signifie, par exemple lorsque nous applaudissons pour indiquer notre admiration. A l'inverse, elle est dite iconique quand le signifiant ressemble au signifié, par exemple lorsque nous posons le doigt devant la bouche pour dire à quelqu'un de se taire. Enfin, elle est dite intrinsèque lorsque l'action est elle-même le signifié, par exemple lorsque quelqu'un frappe une autre personne.

L'usage se réfère aux conditions dans lesquelles un comportement non verbal à lieu. Cela intègre le contexte dans lequel ce comportement est effectué, sa relation au comportement verbal, si le comportement est conscient, s'il est volontaire, le retour du récepteur et enfin le type d'information transmise. Une information est informative si son sens est partagé par le groupe, communicative si l'émetteur à une volonté de la transmettre et interactive si elle influe sur le comportement d'autres personnes.

Les deux auteurs proposent également 5 catégories de comportements non verbaux, suivant une catégorisation proche de celle qu'avait développée Efron (1941) avant eux.

Emblèmes Gestes conscients qui ont un sens précis dans une communauté donnée. Ils peuvent généralement être remplacés par un mot équivalent et partagent avec les mots un encodage souvent arbitraire. Le sens d'un même geste peut ainsi varier fortement d'une culture à l'autre, au point d'être parfois inoffensif dans une culture et insultant dans une autre. Le signe effectué en formant un cercle avec l'index et le pouce est un exemple très connu d'emblème dont la signification change fortement à travers le monde. Alors qu'en France, ce geste signifie "zéro" ou "nul", il est utilisé pour dire "OK" ou "parfait" aux Etats-Unis et "monnaie" au Japon, il est utilisé dans de nombreux pays dont l'Allemagne, le Brésil ou la Grèce comme une insulte sexuelle.

Illustrateurs Gestes intentionnels accompagnant la parole. Leur sens est moins précis que les emblèmes et sont difficilement remplacés par un mot équivalent. Ekman et Friesen décrivent 8 types d'illustrateurs différents, un illustrateur pouvant appartenir à plusieurs types, en fonction notamment de leur encodage :

Bâtons Marquent l'importance d'un mot ou d'une phrase dans le discours du locuteur. Ces gestes n'ont pas de sens intrinsèque hors du propos qu'ils appuient.

Idéographes Indiquent l'organisation d'une pensée, e.g. compter les étapes d'une procédure que l'on est en train de décrire.

Mouvements déictiques Pointent vers des objets, des lieux ou des événements, présents ou imaginés, e.g. pointer du doigt un objet que l'on veut acheter dans une vitrine.

Mouvements spatiaux Décrivent une information spatiale, par exemple la taille d'un objet ou la distance entre deux objets, e.g. placer ses paumes à 20cm l'une de l'autre pour indiquer la taille du poisson que l'on a péché.

Mouvements rythmiques Décrivent une information temporelle comme la durée d'un événement, e.g. claquer des doigts pour signifier l'instantanéité d'un événement.

Kinétographes Représentent une action en la mimant, e.g. poser sa tête sur ses deux mains pour dire que l'on va se coucher.

Pictographes Décrivent un objet ou une idée en le dessinant dans les airs, e.g. un arbitre qui dessine un écran dans les airs pour signifier qu'il fait appel à l'arbitrage vidéo.

Mouvements emblématiques Emblèmes utilisés pour accompagner ou renforcer les discours, e.g. utiliser le geste de pouce levé pour accompagner un compliment.

Expressions affectives Révèlent l'état émotionnel de l'émetteur, principalement au travers des expressions faciales, e.g. expression faciale de peur.

Régulateurs Comportements interactifs ayant pour but de réguler le déroulé d'une conversation en donnant un retour à la personne qui parle, e.g. osciller la tête de haut en bas pour signifier que l'on comprend.

Adapteurs Comportements généralement non conscients et sans intention de communiquer permettant de satisfaire un besoin physique ou psychologique. Ces comportements peuvent être effectués soit envers soi-même (e.g. se gratter la tête), envers

d'autres personnes (e.g. se rapprocher de quelqu'un qu'on apprécie) ou envers des objets (e.g. tapoter avec son crayon).

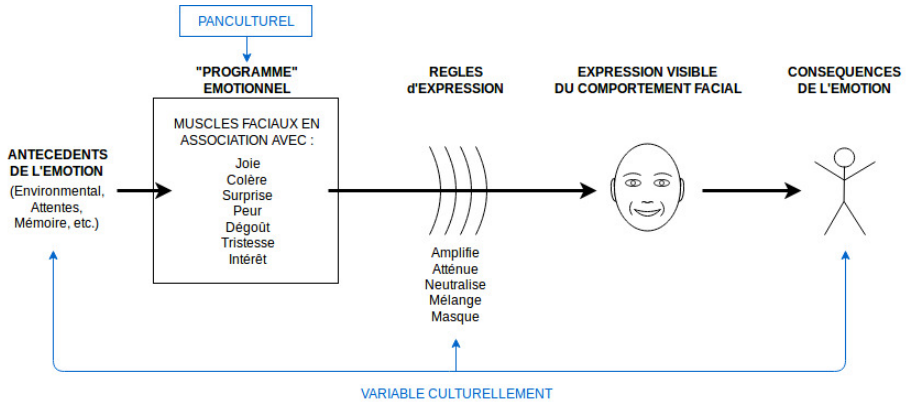


FIGURE I.3 – Expression des émotions selon Ekman

Les travaux d'Ekman sur l'universalité des émotions sont souvent utilisés pour justifier le choix d'un ensemble d'expressions émotionnelles universelles. Bien qu'ils montrent une universalité des expressions faciales associées à 6 émotions de base, ils montrent également que les cultures diffèrent dans leurs "règles d'affichage", c'est-à-dire comment et dans quel contexte une expression peut être utilisée. La figure I.3 présente la relation entre culture et expression émotionnelle d'après Ekman.

Parmi les comportements non verbaux, l'universalité des expressions faciales est une exception plutôt que la règle. Une part importante de ces comportements ne sont pas transposables d'une culture à l'autre. Un robot se limitant à un sous-ensemble de comportements universellement compris reviendrait à limiter considérablement son expressivité.

3. Apports des films d'animation

L'animation est une discipline artistique visant à créer l'illusion du mouvement par la création d'une suite d'images. Cette discipline a pris son essor au cours du 20ème siècle avec le développement du cinéma et du film d'animation. Le dessin animé fut sans doute la méthode la plus influente, notamment au travers des méthodes développées par les studios Disney.

Dans *L'Illusion de Vie* (Thomas et al., 1995), Thomas et Johnson, deux des principaux animateurs des studios Disney, énoncent 12 principes de l'animation tirés de leur expérience chez Disney. Ces 12 principes répondent à deux objectifs. Le premier répond au besoin de créer l'illusion que ce monde, dessiné sur papier puis projeté sur écran, est réel et que les objets et les personnages y sont soumis aux lois de la physique. Le deuxième est sans doute plus difficile à définir et plus encore à quantifier, car il répond à la nécessité de rendre une histoire claire et intéressante pour le spectateur.

Compression et étirement Les objets et les personnages doivent être représentés comme s'ils avaient une existence physique, notamment comme s'ils avaient une masse, un volume, une élasticité ou encore une vitesse. Ce principe est souvent considéré comme le plus important en animation. Ex. Une balle qui rebondit s'aplatit au contact du sol et s'étire en prenant de la vitesse.

Accélération et décélération Les objets et les personnages prennent du temps pour se mettre en mouvement et pour s'arrêter. Leur vitesse ne doit donc pas être constante mais passer par des phases d'accélération et de ralentissement. Ex. Une voiture qui démarre ou qui s'arrête.

Arcs Les objets suivent des trajectoires arquées. Ce principe vient du fait que naturellement, la plupart des mouvements sont soit des rotations autour d'un pivot, soit des chutes suivant une trajectoire parabolique. A l'inverse, les mouvements linéaires donnent un style mécanique à une animation. Ex. La main d'un enfant lançant une balle suit un arc centré sur son épaule.

Qualité du dessin Ce principe souligne l'importance de la qualité des dessins qui composent une animation. Le dessin doit respecter les volumes, la perspective, les ombres ou encore les reflets. Les auteurs s'appuient également sur l'importance de dessiner les personnages dans des poses naturelles, en évitant par exemple les poses symétriques qui sont naturellement très rares.

Continuité et chevauchement des actions La continuité et le chevauchement des actions permettent de représenter l'inertie des objets ainsi que leur couplage physique les uns aux autres. Le chevauchement des actions consiste à décaler les mouvements de chaque objet indépendant. Lorsqu'un personnage tourne la tête par exemple, le mouvement de la tête commencera avant celui de ses cheveux. La continuité consiste quant à elle à s'assurer que le mouvement d'un objet couplé à un personnage ne s'arrête pas en même temps que celui-ci. Les vêtements d'un personnage doivent

ainsi continuer dans le sens du mouvement du personnage après qu'il se soit arrêté. Ces principes jouent un rôle important dans le réalisme de l'animation du fait qu'ils représentent des contraintes physiques du monde réel.

Anticipation Afin de mettre en valeur l'action d'un personnage, l'animateur peut indiquer au spectateur que le personnage va l'effectuer en démarrant le mouvement par un léger mouvement dans le sens inverse. Ex. un personnage se penche en arrière juste avant de partir en courant.

Animation directe / pose à pose Il existe en animation traditionnelle deux façons d'organiser le dessin des images composant une animation. L'animation directe consiste à dessiner l'animation dans l'ordre, de la première à la dernière image. Cette méthode donne des animations plus dynamiques et plus spontanées. L'animation pose à pose (figure I.4) consiste à dessiner les images importantes correspondant à des poses clés, puis à compléter itérativement l'animation en dessinant des images intermédiaires. Cette méthode permet quant à elle des animations plus précises.

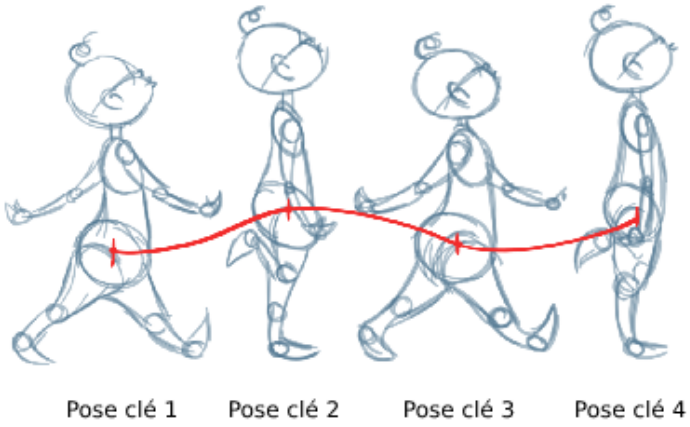


FIGURE I.4 – Animation par frames clés. (Crédit)

Action secondaire L'ajout d'une action secondaire permet de rajouter de la vie à une animation. Il est toutefois important de ne pas détourner l'attention du spectateur de l'action principale. Ex. Une personne qui marche dans la rue pourra être dessinée en train de siffler.

Exagération Une action ou une émotion peut être exagérée pour la rendre plus claire et plus intéressante aux yeux du spectateur. Cette exagération peut sembler aller

à l'encontre du réalisme qui semble être l'objectif des autres principes exposés, c'est pourquoi elle doit être utilisée avec précaution. Les émotions d'un personnage sont souvent poussées à l'extrême pour être bien visible.

Mise en scène L'idée représentée par l'animation, que ce soit une action ou une émotion, doit être claire pour le spectateur. Ceci peut nécessiter d'attirer l'attention du spectateur sur ce qui est important dans la scène, mais également de décomposer une action pour la rendre plus explicite. Ex. Un personnage montre une clé avant de la mettre dans sa poche.

Timing Ce principe met en exergue l'importance sémantique du timing d'un mouvement. Les auteurs donnent l'exemple d'une personne qui tourne la tête vers la droite. Si le mouvement est instant, la personne semble avoir reçu un violent coup à la tête. Si le mouvement est ralenti, elle semble indiquer une direction de la tête. S'il est ralenti plus encore, elle semble s'étirer la nuque. Manipuler le timing doit donc se faire avec précaution.

Charme Ce principe est sans doute le plus difficile à décrire. Les personnages doivent être conçus afin d'être intéressants voire captivants pour le spectateur.

B. Approches de la création de mouvements expressifs

Dans la section précédente, nous avons présenté différentes approches théoriques de l'expressivité du mouvement. Nous présenterons dans cette section les différentes méthodes utilisées afin de doter des personnages virtuels de mouvements expressifs. Nous nous intéressons en particulier aux approches proposées dans les domaines du jeu vidéo et du film d'animation du fait de la richesse des développements autour de l'expressivité du mouvement menés par ces communautés.

1. Approches manuelles

Un premier ensemble de méthodes fait appel à des artistes pour la création manuelle de chacun des mouvements. Ces méthodes sont toujours prépondérantes à ce jour pour la création des mouvements des personnages virtuels, en particulier pour les films d'animation.

a. Animation par frames clés

L'animation par frames clés est la méthode utilisée pour les films d'animation traditionnels (figure I.5). Le personnage est d'abord dessiné dans des poses clés correspondant à des moments importants du mouvement (ou *key frames*). L'animation est ensuite complétée en dessinant les images intermédiaires (ou *inbetweens*) faisant passer le personnage d'une pose clé à la suivante.

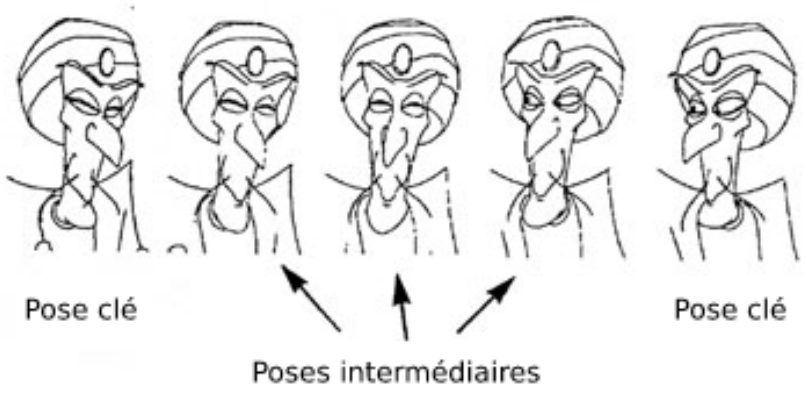


Figure I.5 – Poses clés et poses intermédiaires. (Crédit)

Les outils d'animation 3D sont une adaptation de cette méthode traditionnelle. La pose du personnage est définie à des moments clés en manipulant une armature virtuelle. Les poses intermédiaires sont quant à elles calculées par interpolation des poses clés. Le profil de vitesse de cette interpolation est généralement contrôlable par l'utilisateur au moyen de courbes paramétriques telles que des courbes de Bézier (figure I.6).

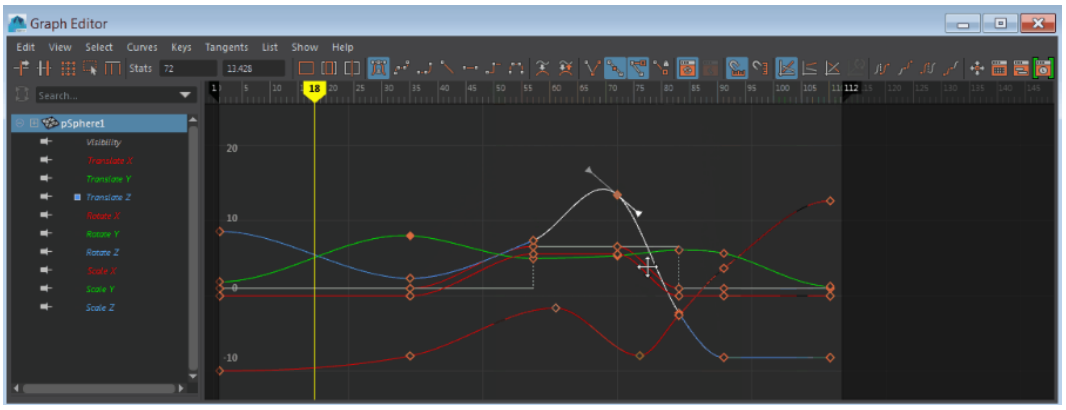


Figure I.6 – Interface du logiciel d'animation Autodesk Maya.

b. Capture de mouvement

La capture du mouvement, parfois appelée *mocap* de l'anglais *Motion Capture*, désigne l'ensemble des techniques permettant l'enregistrement numérique des mouvements d'un acteur. Ces techniques permettent de doter des personnages virtuels de mouvements très réalistes. Cette qualité fait leur succès pour l'animation des joueurs dans les jeux vidéos de simulations sportives. Elles sont également utilisées dans la création de films fantastiques pour ajouter du réalisme à des personnages imaginaires.

FIGURE I.7 – Exemple de personnage animé par capture de mouvement (Crédit).

Il existe un grand nombre de systèmes de capture de mouvement. Les systèmes les plus courants fonctionnent par localisation optique de marqueurs lumineux disposés sur le corps voire le visage de l'acteur. Ces marqueurs lumineux peuvent être passifs comme dans le cas des systèmes OptiTrack [1] ou Vicon (figure I.8) [2], ou actifs comme le système PhaseSpace [3]. Plusieurs caméras sont utilisées pour trianguler la position 3D des marqueurs. D'autres systèmes moins coûteux tel que le système Xsens [4] utilisent des centrales inertielles afin d'enregistrer les orientations et les accélérations des différentes parties du corps d'un acteur. Ces informations sont ensuite utilisées pour reconstruire le mouvement.

1. OptiTrack : https://optitrack.com - dernière visite : 01/11/2019
2. Vicon : https://www.vicon.com - dernière visite : 01/11/2019
3. PhaseSpace : https://phasespace.com - dernière visite : 01/11/2019
4. Xsens : https://www.xsens.com - dernière visite : 01/11/2019

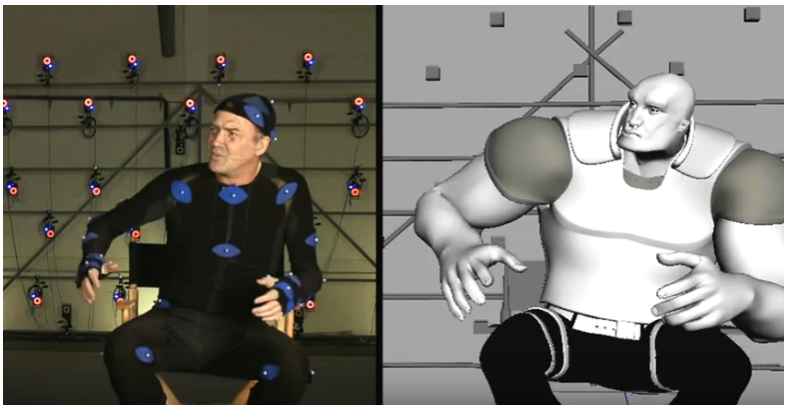


FIGURE I.8 – Vicon : Systèmes de capture de mouvement (Crédit).

c. Outils d'édition de mouvement

La capture de mouvement est un processus pouvant être long et coûteux. Si une capture ne correspond pas à la scène, il est souvent nécessaire de la refaire en entier. Plusieurs méthodes ont été proposées afin de permettre à des artistes d'éditer ces mouvements.

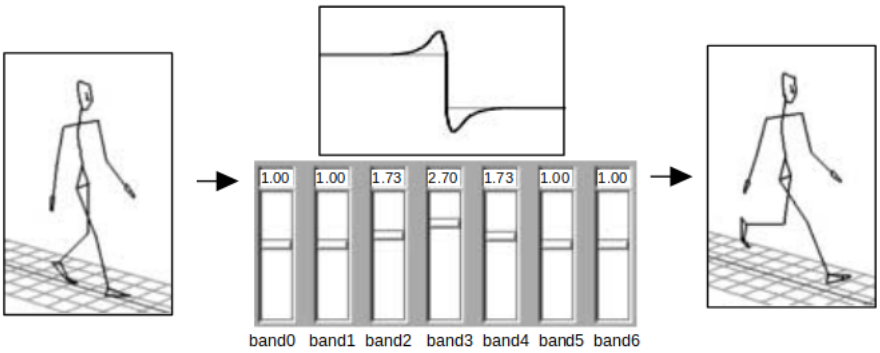


FIGURE I.9 – Motion Signal Processing (Bruderlin and Williams, 1995)

Une première catégorie de méthodes cherche à éditer l'animation sur un plan expressif. Bruderlin and Williams (1995) se sont ainsi appuyés sur les méthodes de traitement du signal pour adapter l'animation d'un point du vue expressif. Ils montrent comment le *Dynamic Time Warping* peut être utilisé pour modifier le timing d'une animation ou

encore comment la modulation de fréquence permet de changer l'émotion véhiculée par le mouvement du personnage.

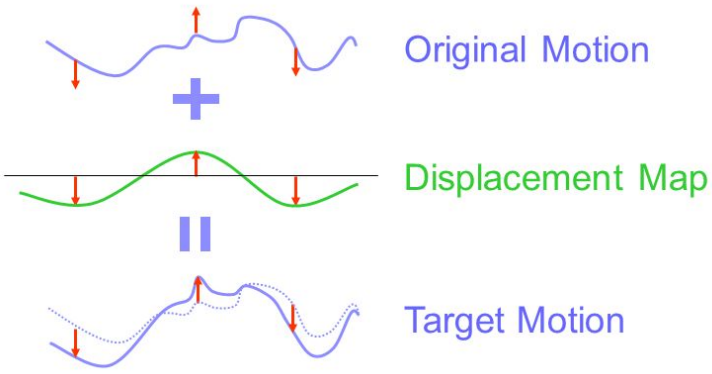


FIGURE I.10 – Displacement mapping (Crédit).

Une deuxième catégorie de méthodes d'édition a pour objectif l'adaptation de l'animation sur le plan géométrique. Elle répond ainsi au besoin d'adapter le mouvement à la géométrie de la scène environnante. C'est l'objectif du *Motion Warping* proposé par Witkin and Popovic (1995) et du *Displacement Mapping* de Bruderlin and Williams (1995). Le principe de fonctionnement de ces deux méthodes est de déformer le mouvement afin que celui-ci passe par une pose donnée.

2. Approches génératives

Un deuxième ensemble de méthodes s'intéresse à la génération de mouvements de façon procédurale. On retrouve deux approches : la programmation créative et les modèles à base de règles.

a. Programmation créative

L'un des concepteurs de Poppy a eu l'idée de le doter d'un mode "repos" où il ne fait rien, tout en étant jamais immobile. Ce mouvement de "repos" n'est que la somme de simples mouvements sinusoïdaux sur chacun de ces moteurs, à des fréquences et des phases différentes, mais crée un mouvement à la fois complexe et subtil cassant l'immobilité et l'impression de rigidité habituellement associée aux robots.

L'utilisation astucieuse de techniques mathématiques pour générer un mouvement est un exemple de programmation créative, un type de programmation qui se distingue par son but esthétique (ou plus largement expressif) plutôt que fonctionnel. La programmation créative regroupe aujourd'hui une communauté très importante et touchant à tous les médias artistiques. Cette communauté a depuis ses débuts une nature très exploratoire et intègre rapidement les dernières technologies pour en tester les limites.

La "scène démo" est sans doute le premier exemple majeur de ce type de programmation (Scheib et al., 2002). Une "démo" est une vidéo générée en temps réel par un ordinateur et parfois accompagnée de musique, générée également. Les "démos" étaient souvent utilisées comme signature dans les programmes de leurs auteurs ou dans les programmes qu'ils avaient "crackés". La génération au moyen de code informatique était nécessaire à une époque où les outils d'infographie étaient inexistants et les capacités de calcul et de mémoire des ordinateurs fortement limitées. La figure I.11 présente quelques captures d'écran de "démos" extraites d'un documentaire sur ce phénomène (Matusik, 2011).

Cette discipline a fortement influencé les débuts du jeu vidéo, alors que ce medium souffrait des mêmes limites techniques. Le bruit de Perlin (Perlin, 2002) est une des ces "astuces" mathématiques adoptées par les premiers développeurs de jeux vidéos afin d'améliorer l'esthétique de leurs jeux. Cette technique permet en effet la génération déterministe de textures naturelles très diverses (figure I.12), économisant ainsi l'utilisation de mémoire que le stockage de ces textures aurait nécessité.

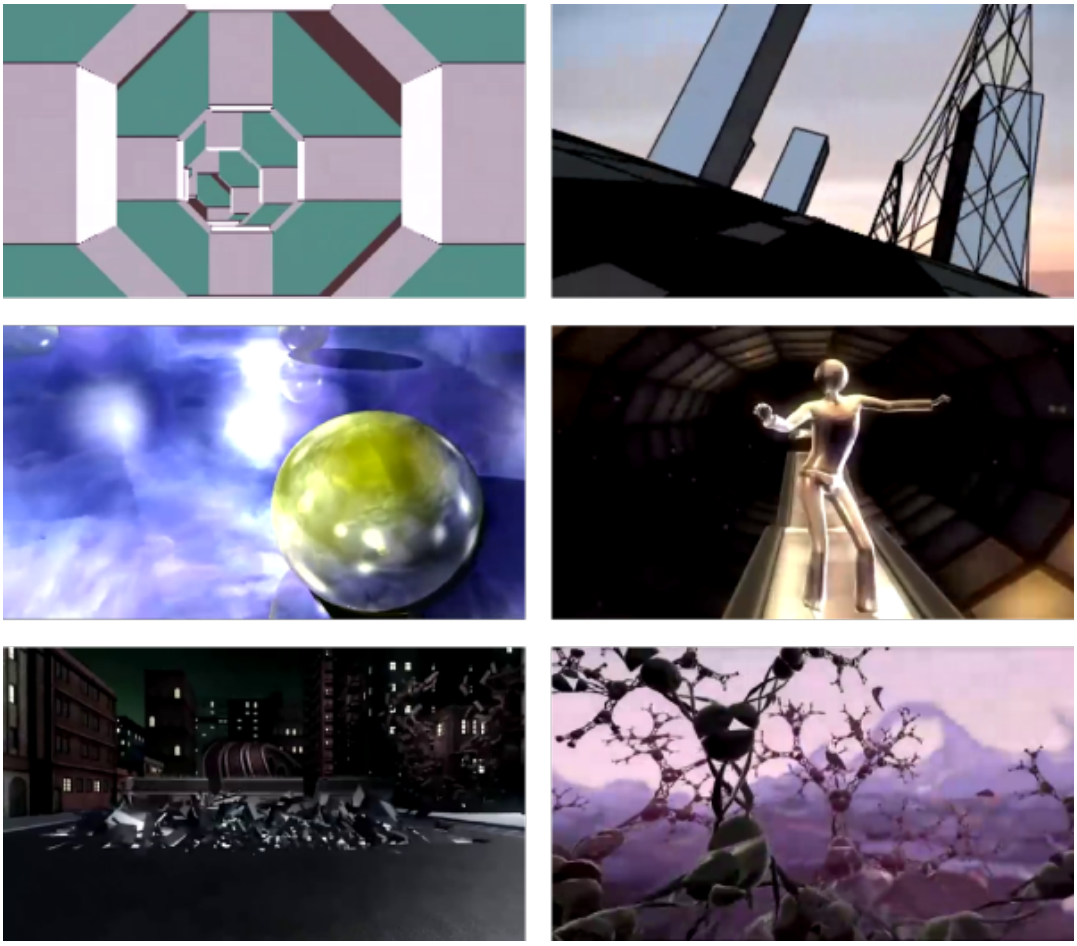


FIGURE I.11 – Captures d'écran de quelques "démos" (Matusik, 2011).

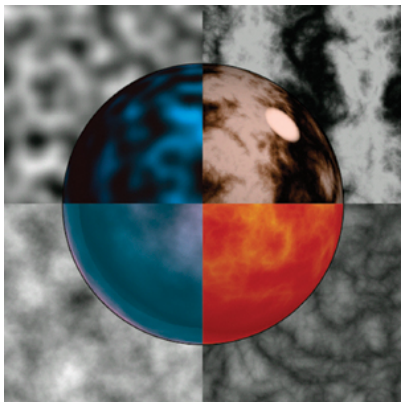


FIGURE I.12 – Exemple de textures générées grâce au bruit de Perlin (Korn and Lee, 2017).

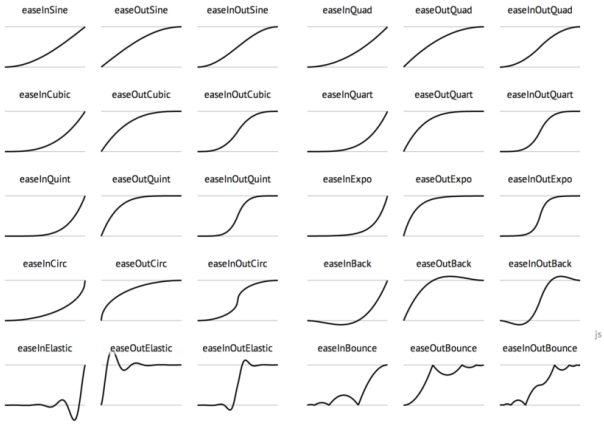


FIGURE I.13 – Exemple de Fonctions d'interpolation de Robert Penner (Crédit).

Les fonctions d'interpolation (parfois appelées "tweeners" ou fonction de "easing") popularisées par Robert Penner (Penner, 2002) est un autre exemple de cette programmation créative. Ces fonctions ont été utilisées pour la génération du mouvement des interfaces graphiques web et mobile, répondant à nouveau au manque d'outils de création et aux contraintes techniques de ces plateformes.

b. Modèles à base de règles

Comme nous l'avons vu dans la section précédente, de nombreuses disciplines ont étudié le mouvement et en particulier le mouvement expressif chez l'Homme. L'enrichissement par ses différentes disciplines se retrouve dans la diversité des modèles génératifs de mouvements basés sur des règles. Trois grandes familles de règles se dégagent : les règles artistiques, les règles linguistiques et psychologiques et les règles physiques et biomécaniques.

Règles artistiques

Le modèle EMOTE (Chi, 1999; Chi et al., 2000) reprend les notions d'efforts et de forme de la théorie de Laban sur l'analyse du mouvement de danse afin de définir un modèle paramétrique du mouvement expressif des bras et du torse d'un personnage virtuel. Pour ce faire, le modèle paramétrise des déplacements des points clés par lesquels le bras passe comparé à un mouvement original en fonction de ces efforts. Au travers de la fonction de vitesse paramétrique proposée, le modèle implémente

également des principes repris de l'animation traditionnelle comme l'anticipation, la continuité ou l'accélération/décélération.

LaViers and Egerstedt (2012) se basent également sur les travaux de Laban mais en approchant la question par la définition d'une fonction de coût paramétrée selon les efforts de la théorie de Laban. Cette fonction peut ensuite être utilisée pour générer le mouvement stylisé par contrôle optimal.

Ces méthodes ne proposent pas de correspondance entre le signifié, par exemple une émotion, et le signifiant, ici le mouvement. Cette tâche est laissée à la responsabilité de la personne créant un nouveau style de mouvement qui devra le faire de façon formelle au travers des paramètres du modèle.

Règles linguistiques et psychologiques

Une autre approche est de s'inspirer de l'étude du geste en linguistique et en psychologie. Le moteur de gestes paramétriques GRETA s'inspire des principes énoncés par McNeill (2008) et Wallbott (1998). Le modèle reprend à McNeill la décomposition en 4 phases distinctes (*preparation*, *stroke*, *hold* et *retraction*) ainsi que l'organisation de l'espace du geste en cercle concentrique (Hartmann et al., 2002). Les paramètres de modulation de la gestuelle d'un agent sont quant à eux repris de Wallbot. Le moteur permet ainsi de définir la quantité de gestes et leur répétitivité ainsi que l'amplitude, la durée, la fluidité et la tension de chaque geste (Hartmann et al., 2005).

On peut noter que ces derniers modulateurs se rapprochent de ceux de la théorie de Laban and Ullmann (1971). Le système PARSYS (Allbeck and Badler, 2002) utilise cette ressemblance en implémentant le modèle de personnalité OCEAN (*Openness*, *Conscientiousness*, *Extraversion*, *Agreeableness*, *Neuroticism*) en le reliant au moteur EMOTE présenté précédemment. Ils définissent pour cela une table de correspondance entre les efforts de Laban et les 5 grandes personnalités du modèle OCEAN.

Règles physiques et biomécaniques

Nous avons vu précédemment que plusieurs principes d'animation traditionnelle servent à s'assurer que le mouvement répond à des règles physiques, permettant ainsi de leur donner une forme de réalisme (Thomas et al., 1995). Partant de cette observation, Witkin and Kass (1988) ont développé une méthode de synthèse de mouvement basé sur la recherche d'un mouvement répondant de façon optimale à des contraintes spatio-temporelles définies par l'animateur ainsi qu'à un modèle physique.

Ce principe a donné lieu à une quantité importante de travaux afin d'améliorer la fidélité du modèle physique, de permettre différents types de contraintes ou encore d'accélérer la recherche de la solution optimale. Une autre approche, initiée par Komura et al. (1997), est de modéliser les personnages sur le plan biomécanique. Le système musculo-squelettique est ainsi modélisé comme un système mécanique dans lequel les muscles sont soumis à des contraintes d'effort ou de fatigue musculaire (Cruz Ruiz et al., 2017).

Ces méthodes sont principalement utilisées dans l'objectif de modéliser fidèlement des mouvements humains mais ont également été utilisés pour générer des mouvements vraisemblables pour des créatures imaginaires (Geijtenbeek et al., 2013).

3. Approches hybrides

Dans les deux précédentes sections, nous avons présenté les approches manuelles et les approches génératives de la création du mouvement de personnages. Nous regroupons sous le nom d'approches hybrides les méthodes de génération paramétriques de mouvement reposant sur la généralisation d'animations créées par des artistes. On distingue les différentes méthodes selon qu'elles permettent de paramétrer l'animation dans l'espace expressif ou dans l'espace fonctionnel.

a. Généralisant dans l'espace expressif

Un exemple classique d'animation paramétrique est l'animation de la marche expressive d'un personnage, permettant de faire marcher un personnage dans un style joyeux, neutre ou triste.

Amaya et al. (1996) proposent d'apprendre une fonction de transformation émotionnelle, permettant de déformer une animation pour qu'elle exprime une émotion donnée. Les auteurs proposent de modéliser l'expressivité du mouvement comme les ratios de vitesse et d'amplitude entre un mouvement expressif et un mouvement neutre. La transformation émotionnelle d'un mouvement correspond ensuite à appliquer plus ou moins fortement ces ratios à un nouveau mouvement neutre.

Rose III et al. (2001) étendent l'interpolation au cas multidimensionnel avec leur méthode des verbes et adverbes. Les verbes correspondent à un type d'action (ex. "marcher") tandis que les adverbes correspondent à la façon de faire cette action

(ex. "tristement"). Un même verbe peut être représenté par plusieurs animations pour lesquelles l'animateur définit des valeurs d'adverbes. Cela permet de localiser une animation dans un espace dont les valeurs d'adverbes sont les coordonnées. De nouvelles animations peuvent être générées pour de nouvelles valeurs d'adverbes par interpolation dans cet espace.

b. Généralisant dans l'espace fonctionnel

La généralisation d'une animation dans l'espace fonctionnel est nécessaire lorsque le mouvement est fonction de l'environnement. C'est particulièrement le cas lorsque le geste est dirigé vers une cible. Par exemple, le geste de pointer vers un objet dépend de l'emplacement de l'objet cible dans l'environnement. Ce type d'animations ne prend tout son sens que si certaines contraintes géométriques avec l'objet cible sont respectées.

Les méthodes de généralisation peuvent être divisées en deux catégories, selon qu'elles sont basées sur une seule animation originale ou sur un ensemble d'animations originales du même style. L'utilisation d'un ensemble d'animations permet une définition plus complète d'un style mais présente un coût significatif par rapport à la création d'une seule animation originale.

L'extrapolation à partir d'une animation originale unique présente une problématique spécifique : la composante stylistique et la composante fonctionnelle sont entrelacées dans l'animation originale. Les méthodes par extrapolation doivent donc intégrer un à priori sur ce qui correspond à chaque composante.

Une première approche est de contraindre les transformations à un sous-ensemble qui conserve la composante stylistique. (Gielniak et al., 2010) proposent ainsi de conserver les vitesses relatives au sein de chaque degré de liberté. (Nierhoff et al., 2016) minimisent la déformation en reprenant les principes de déformation "As Rigid As Possible" initialement développée pour la déformation de meshes (Sorkine, 2005). Bien que la conservation de la composante stylistique ne soit pas un objectif explicite des auteurs, une méthode similaire a été développée dans le cadre de l'édition de mouvements par Kim et al. (2009) et pour laquelle cette conservation du style est importante.

Une deuxième approche est de définir une fonction de coût incorporant une composante stylistique. Liu et al. (2005) partent de l'idée que l'animation originale est

une solution énergétique optimale dans un modèle physique inconnu. Ils proposent de retrouver ce modèle physique à partir de l'animation originale en utilisant une méthode d'optimisation inverse. En d'autres termes, ils apprennent une fonction de coût qui intègre la composante stylistique de l'animation originale et peut être utilisée pour générer de nouvelles animations de ce style. Récemment, (Peng et al., 2018) ont fait appel dans leur projet DeepMimic à l'apprentissage par renforcement pour généraliser une animation en entraînant un agent à imiter une animation originale. Pour ce faire, une composante d'imitation est intégrée à la fonction de récompense dans le processus d'apprentissage.

Pour leur part, les méthodes de généralisation basées sur un ensemble d'animations originales de même style tentent de trouver l'animation interpolée qui répond le mieux aux contraintes fonctionnelles. Différents méthodes ont été proposées pour définir un espace dans lequel faire cette interpolation. Huang and Kallmann (2010) proposent d'interpoler directement dans l'espace des poses clés. D'autres travaux apprennent des modèles statistiques à partir de l'ensemble des animations originales et interpolent dans l'espace latent de ces modèles. De nombreuses méthodes de modélisation statistique ont été appliquée à la modélisation de mouvement tels que les modèles de Markov à états cachés (Bowden, 2000), les analyses en composantes principales (Urtasun et al., 2004) ou en composantes indépendantes (Shapiro et al., 2006), les modèles à processus gaussiens latents (Levine et al., 2012) ou encore les auto-encodeurs (Holden et al., 2016).

C. Positionnement et objectifs de la thèse

Comme nous l'avons vu dans ce chapitre, il existe une grande variété de comportements expressifs du fait des variations culturelles et des différents rôles potentiels des robots sociaux. Se limiter à un sous-ensemble universel de comportements expressifs briderait significativement la richesse et la qualité de l'interaction. Pouvoir créer facilement et rapidement de nouveaux comportements expressifs de qualité paraît donc essentiel afin de faciliter le développement de la robotique sociale.

Parmi les approches manuelles de création de mouvements expressifs, nous avons pu voir que la capture de mouvements d'acteurs est particulièrement adaptées lorsque le but principal est le réalisme du mouvement. A l'inverse, la méthode d'animation par poses clés permet une grande liberté à la fois en terme de mouvement mais également en terme de formes des personnages à animer. La généralité de l'animation par poses

clés correspond bien au besoin d'adaptation de la forme et du comportement des robots sociaux à leurs rôles et à leurs tâches.

Nous avons également pu voir que certains mouvements expressifs nécessitent de respecter des relations spatiales. Les mouvements déictiques sont par exemple dirigés vers une cible. Les mouvements spatiaux expriment quant à eux une taille ou une distance par la localisation relative des différentes parties du corps de l'émetteur. D'autres mouvements expressifs tels que les bâtons ou les mouvements rythmiques intègrent des contraintes temporelles.

Pour un film d'animation, ces contraintes peuvent être intégrées par l'animateur lors de la conception de l'animation. Ce n'est pas le cas en revanche pour un robot social en interaction avec un ou plusieurs utilisateurs. Les localisations des objets dans l'environnement ne sont pas toujours connues à l'avance par exemple. Il est donc nécessaire de pouvoir paramétriser les animations afin de les adapter à de nouvelles contraintes pendant l'interaction. Ce besoin correspond aux méthodes d'édition de mouvement et de généralisation de l'animation dans l'espace fonctionnel que nous avons présenté précédemment.

L'objectif de cette thèse est de concevoir des outils, méthodes et algorithmes permettant d'intégrer les animateurs dans la conception des mouvements expressifs des robots sociaux et d'étendre le potentiel de l'animation en robotique en rendant ces animations adaptables à de nouvelles contraintes géométriques.

Chapitre II

Outil de prototypage de robots sociaux animés

Ce chapitre présente un système de prototypage de robots sociaux animés. Nous commençons par une description des différents robots sociaux existants dans la section B dont nous dégageons dans la section C des catégories de composants classiquement utilisés dans leur conception. Nous présentons ensuite le logiciel Blender et ses fonctionnalités dans la section D. Ces bases ainsi posées, nous décrivons le fonctionnement et l'architecture de notre système dans la section E. Enfin, nous présentons dans la section F deux exemples de réalisations permises par celui-ci.

A. Motivations

Une grande partie de l'expérience utilisateur d'un robot social est liée à sa forme et à sa façon de bouger. Ces deux dimensions sont donc majeures dans la conception d'un robot social. Concevoir l'apparence et les mouvements indépendamment peut s'avérer limité car les mouvements possibles dépendent de la forme du robot et des actionneurs le composant.

Hoffman and Ju (2014) proposent de commencer par concevoir le mouvement. Pour cela, ils définissent une méthodologie débutant par des *croquis papiers* similaires à ceux que ferait un dessinateur durant la conception d'un personnage puis à les transformer rapidement en *croquis 3D animés* permettant de définir la façon de bouger du robot.

Cette étape permet des itérations rapides entre la conception de l'apparence et des mouvements du robot.

Mais animer des robots ajoute des problématiques supplémentaires comparé à l'animation 3D. Les robots sont soumis aux lois de la physique. Ils peuvent osciller, trembler, ou encore vibrer et leurs moteurs produisent des bruits liés aux frottements. De plus, leurs moteurs ont des limites de vitesse et d'accélération. Ces contraintes spécifiques à l'animation des robots peuvent modifier l'expressivité d'un mouvement si elles ne sont pas prises en compte.

Dans la méthode de Hoffman and Ju (2014), ces contraintes sont prises en compte lors d'une seconde étape consistant à la fabrication d'un prototype du squelette du robot. Cette étape nécessite également la création d'un logiciel dédié permettant d'animer le prototype. C'est cette étape que nous proposons de simplifier grâce au système présenté dans ce chapitre.

Le principe général de notre système est de définir un ensemble de composants de base de robots sociaux et d'étendre un logiciel d'animation 3D existant pour permettre l'animation de ceux-ci. Partir d'un logiciel d'animation 3D existant a comme premier avantage de réutiliser des concepts et méthodes maitrisés par les animateurs. Ils pourront donc plus facilement prendre en main un tel système. Un deuxième avantage est de nous permettre d'évaluer si les méthodes et outils de l'animation 3D sont transposables à la robotique et quelles en sont les limites.

Une importance particulière est apportée à concevoir un système général, de sorte que différentes morphologies de robot puissent être prototypées et animées en assemblant des composants de base.

B. Robots sociaux

Les robots sociaux sont des robots conçus pour être capable d'exprimer des comportements sociaux pour communiquer avec leurs utilisateurs. Leurs designs varient grandement en fonction de leurs rôles mais également du fait de choix esthétiques et techniques de leurs designers.

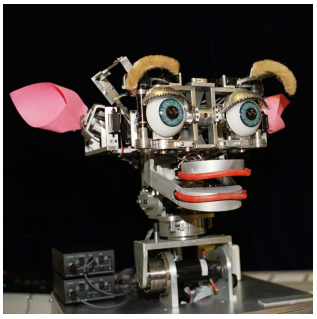
Des exemples de robots sociaux sont présentés dans figures II.1 et II.2. Nous les classons selon 3 dimensions : l'apparence, la morphologie et la mise en œuvre des expressions faciales.

(a) Pléo

(b) PaRo

(c) Leonardo

(d) Probo



(e) Kismet

(f) iCat

(g) Geminoid

(h) Furhat

(i) Zeno

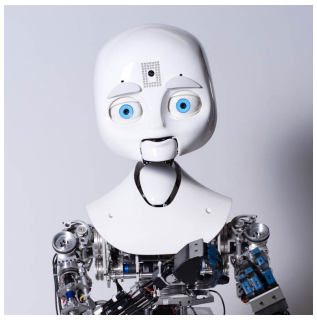
(j) Pepper

(k) NAO

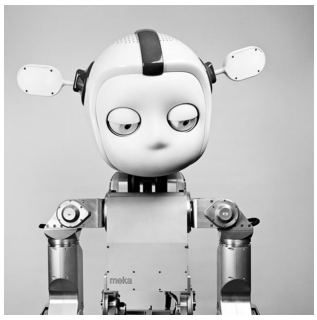
(l) iCub

FIGURE II.1 – Exemples de Robots sociaux

(a) Poppy

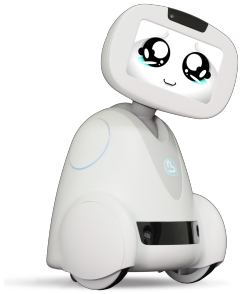


(b) Nexi



(c) Simon

(d) Jibo

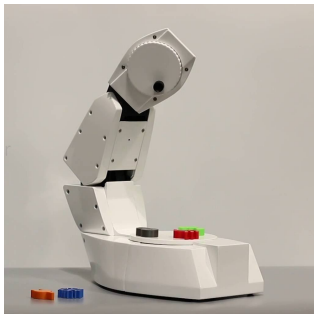


(e) Buddy

(f) Kuri

(g) ElliQ

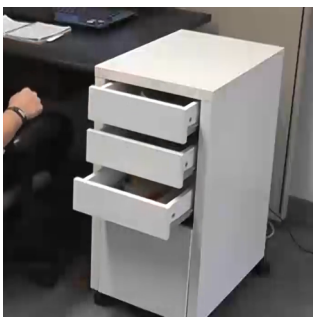
(h) Travis



(i) Vyo

(j) AUR

(k) Mechanical Ottoman



(l) Robotic Drawers

FIGURE II.2 – Exemples de Robots sociaux (cont.)

1. Apparence

Certains designers choisissent de doter leur robot d'une forme humaine ou humanoïde, avec différents degrés de réalisme. Geminoid (Nishio et al., 2007) et Furhat (Al Moubayed et al., 2012) sont ainsi conçus pour s'approcher le plus possible de l'apparence humaine, tandis que Zeno (Hanson et al., 2009) prend inspiration sur des personnages de mangas. A l'inverse, d'autres robots tel que Nexi (Fitzpatrick, 2012), iCub (Beira et al., 2006), Simon (Chao et al., 2010), Pepper [1] ou encore NAO (Gouaillier et al., 2009) et Poppy (Lapeyre et al., 2014) ont des formes inspirées de l'Homme mais sans viser le réalisme.

La forme animale est également très utilisée, parfois inspirée d'animaux imaginaires. Pléo [2] reprend ainsi la forme et l'apparence d'un dinosaure et PaRo celui d'un phoque. Probo (Saldien et al., 2008), Leonardo (Brooks et al., 2004) ou encore Kismet (Breazeal and Scassellati, 1999) ont une apparence animale sans que l'on puisse définir les animaux auxquels ils ressemblent. Enfin, Aibo (Fujita, 2001) ou iCat (van Breemen et al., 2005) ont une forme inspirée d'animaux bien définissables mais avec une apparence *cartoon*.

D'autres designers s'affranchissent plus ou moins fortement des formes et apparences humaines ou animales. Certains robots comme Buddy [3] ou Kuri [4] conservent une forme dotée d'une tête avec des yeux. D'autres comme Jibo [5], Travis (Hoffman, 2012) ou ElliQ [6] conservent uniquement une forme évoquant une tête. Enfin, d'autres comme le Vyo (Luria et al., 2016) s'abstraient plus fortement encore des formes animales en adoptant une forme plus proche d'un robot ménager. Malgré cet effort d'abstraction, sa forme reste évocatrice d'une tête.

Enfin, certains designers choisissent de robotiser des meubles, adoptant ainsi leur forme. Par exemple, AUR (Hoffman et al., 2007) prend la forme d'une lampe tandis que Mechanical Ottoman (Sirkin et al., 2015) est un repose-pied auquel a été ajouté des capacités robotiques.

1. Pepper :https://www.softbankrobotics.com/emea/fr/pepper - dernière visite : 01/11/2019
2. Pléo : http://www.pleoworld.com/ - dernière visite : 01/11/2019
3. Buddy : http://www.bluefrogrobotics.com/en/buddy/ - dernière visite : 01/11/2019
4. Kuri : https://www.heykuri.com/explore-kuri/ - dernière visite : 01/11/2019
5. Jibo : https://www.jibo.com/ - dernière visite : 01/11/2019
6. ElliQ : https://elliq.com/ - dernière visite : 01/11/2019

2. Morphologie

Les robots sociaux peuvent également être classés selon leur morphologie et plus particulièrement leur morphologie contrôlable, en ignorant donc les parties de leur "corps" n'étant qu'esthétique. Comme nous l'avons vu pour l'apparence, il peut être difficile d'éviter le vocabulaire animal pour parler de la forme de robots sociaux. Nous utiliserons donc ce vocabulaire pour notre classification des morphologies.

Une première catégorie regroupe les *têtes robotiques*, consistant essentiellement en une tête et une nuque. On retrouve dans cette catégorie les précurseurs comme Kismet ou iCat, mais également plus récemment Furhat, Jibo ou encore ElliQ et Vyo. On peut également classer le robot Travis dans cette catégorie, bien qu'il soit doté d'un pied et d'un support de smartphone mécanisés.

Une seconde catégorie regroupe les *torses robotiques*, dotés d'un torse et de bras. Les robots Probo, Leonardo et Geminoid font partie de cette catégorie. Certains robots humanoïdes comme Zeno, Poppy ou NAO existent également dans des versions "torse" restreintes au haut du corps.

Ces deux premières catégories ont une variante mobile. Kuri et Buddy peuvent ainsi être considérés comme des *têtes robotiques mobiles*, tandis que Nexi, Simon et Pepper peuvent être considérés comme des *torses robotiques mobiles*, bien qu'ils soient le plus souvent définis comme des *humanoïdes à roues*.

Une dernière catégorie regroupe les robots dotés d'un "corps" complet, soit bipède comme iCub, NAO, Poppy et Zeno, soit quadrupède comme Aibo et Pléo.

3. Visage

Le dernier axe permettant de classer les robots sociaux est le choix d'implémentation des expressions faciales, étant donné l'importance du visage pour un robot social. Certains robots comme PaRo, Travis, ElliQ et Vyo sont dépourvus d'expressions faciales. NAO, Pepper et Aibo sont également minimalistes dans le domaine, n'étant dotés que de quelques LEDs qui ne permettent pas de représenter des expressions faciales en tant que telles. iCub est quant à lui doté de panneaux de LEDs permettant de définir la forme de ses sourcils et de sa bouche.

D'autres robots comme Jibo et Buddy utilisent un écran. Furhat utilise également un écran mais retro-projeté sur une forme de visage. Enfin, certains robots ont un visage articulé. C'est la catégorie la plus représentée parmi les robots décrits précédemment avec Leonardo, Kismet, Probo, Zeno, Nexi, Simon et Geminoid.

C. Composants des robots expressifs

Une revue des différents robots sociaux fait émerger des similitudes dans les actionneurs utilisés à des fins expressives. 4 grands types d'actionneurs se démarquent : les moteurs principaux et secondaires (actionneurs dynamiques), les écrans et les LEDs (actionneurs statiques).

1. Actionneurs électromécaniques

Moteurs principaux Les moteurs principaux ont comme rôle premier le mouvement des articulations du robot. Ils doivent le plus souvent être capable de fournir un couple important, de sorte à pouvoir mettre en mouvement les parties du robot qui y sont rattachées. Les moteurs de l'épaule d'un robot humanoïde devront ainsi avoir le couple nécessaire pour porter son bras, celui du coude son avant bras et celui du poignet uniquement sa main. Leur importance fonctionnelle fait que la très grande majorité des robots sociaux en sont dotés. La gamme de moteurs Dynamixel de Robotis présentée sur la figure II.3 est souvent utilisée dans la phase de prototypage pour ce type d'actionneurs.

Figure II.3 – Gamme de servomoteurs Dynamixel de Robotis (Crédit)

Moteurs secondaires Les moteurs secondaires ont principalement une fonction expressive et sont souvent utilisés pour les mouvements des éléments du visage du robot et pour ceux de ses mains. Ils n'ont généralement pas besoin de fournir un

couple important. Dans la plupart des cas, ils n'ont pas non plus besoin d'être d'une grande précision. Kismet (figure II.4) est un exemple de robot dont les expressions faciales sont générées grâce à 15 moteurs secondaires. Des servomoteurs de véhicules radio-commandés, parfois appelés hobby servos, sont généralement utilisés dans ce but pendant la phase de prototypage. La figure II.5 présente quelques exemples de ce type de moteurs.

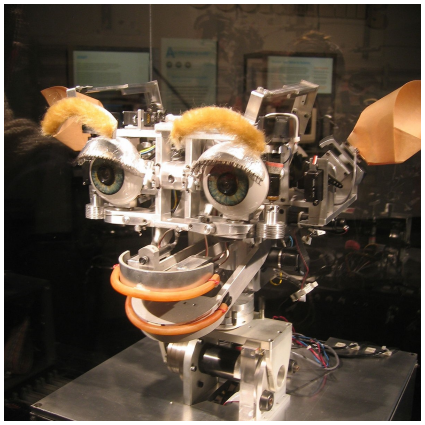


FIGURE II.4 – Robot Kismet

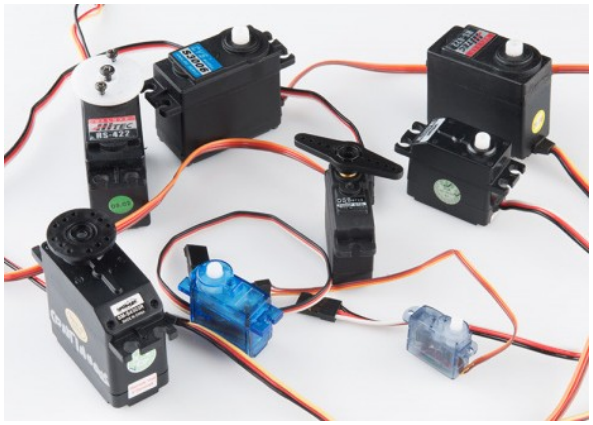


FIGURE II.5 – Servomoteurs RC

2. Actionneurs électroniques

Ecrans Ces dernières années, l'utilisation d'écrans comme composants de robots sociaux s'est multipliée. En effet, les écrans permettent par exemple de représenter des expressions faciales sans la complexité mécanique nécessaire pour générer celles-ci grâce à des moteurs. Ils ont également l'avantage d'être d'une grande versatilité, pouvant afficher successivement un visage, une icône ou même une interface graphique plus classique. De plus, le coût des écrans à fortement baissé avec la démocratisation des smartphones et des tablettes. Le robot Jibo est un exemple de robot utilisant un écran comme son premier actionneur expressif. La figure II.6 illustre les différents types d'usages de l'écran de Jibo. Le robot Pepper fait également usage d'un écran, cette fois-ci positionné sur son thorax, dont la fonction principale est d'afficher une interface graphique mais qui est parfois utilisé dans un but expressif.

FIGURE II.6 – Différentes fonctions de l'écran de Jibo. De gauche à droite, l'écran de Jibo est utilisé pour afficher le flux d'une vidéoconférence, une interface graphique, une illustration d'un conte pour enfant, une icône notifiant l'application en cours, une icône exprimant une émotion et une expression faciale.

LEDs Les LEDs sont utilisées comme actionneurs expressifs de deux façons. La première consiste à utiliser des LEDs pour représenter des expressions faciales anthropomorphes. Le robot iCub est par exemple doté de panneaux de LEDs permettant de définir la forme de ses sourcils et de sa bouche. La seconde consiste à les utiliser pour afficher une couleur représentant une émotion ou un état du robot. Le robot Simon est ainsi doté d'oreilles illuminées par des LEDs de couleur variable. Ces deux utilisations ne sont pas exclusives, les mêmes LEDs pouvant être utilisées pour les deux, comme pour les LEDs autour des yeux des robots NAO et Pepper qui sont parfois utilisées pour indiquer une émotion en variant leur couleur mais aussi pour représenter une sorte de clignement des yeux. Les LEDs peuvent également être vues comme des écrans de très faible résolution.

D. Présentation de Blender

L'outil de prototypage que nous présentons se base sur Blender, un logiciel professionnel de création 3D open-source. Les fonctionnalités de Blender sont vastes, de la modélisation 3D à l'édition vidéo en passant par l'animation.

1. Définition de formes

Modélisation géométrique Blender fournit de nombreux outils permettant de modéliser des objets ou des personnages en 3D. Le principe de fonctionnement général est de modéliser ces formes par des *maillages* (aussi appelés *meshes*). Un maillage est une approximation d'une surface par des facettes triangulaires ou quadrilatères. Chaque

facette est définie par ses sommets (ou vertices) et par ses arêtes (ou edges). Une forme est modélisée en partant d'un maillage de base (ou primitive) et en déplaçant, divisant ou encore supprimant ses sommets, arêtes et facettes, afin de le sculpter jusqu'à obtenir la forme souhaitée. La figure II.7 présente un exemple de ce processus.

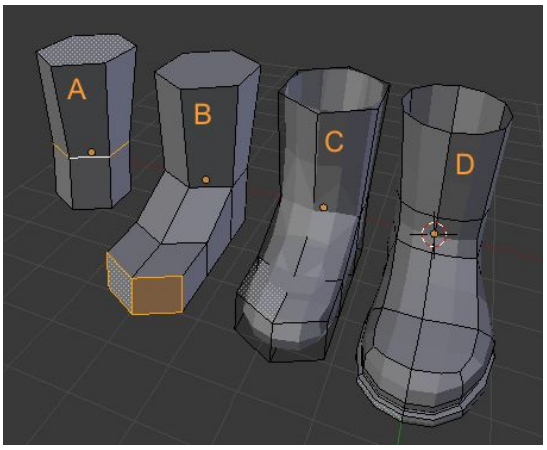


FIGURE II.7 – Modélisation d'une botte dans Blender

FIGURE II.8 – 12 exemples de formes clés pour le personnage Sintel (Crédit)

Interpolation de formes Afin d'animer des déformations d'objets ou de personnages, Blender fournit des fonctionnalités d'interpolations de formes. Cela fonctionne en créant des variantes d'un même maillage, appelées *formes clés* (ou *shape keys*) et en les interpolant avec des poids variables afin d'obtenir une *forme mélangée* (ou *blend shape*) des différentes déformations. Une contrainte importante est que ces variantes doivent conserver les mêmes sommets, arêtes et facettes, la topologie des maillages devant être identique pour pouvoir les interpoler. Cela permet par exemple d'avoir plusieurs variantes du visage d'un personnage avec des expressions différentes et d'animer ses émotions en modulant les poids de ces différentes variantes. La figure II.8 présente des exemples de formes clés conçues pour animer le personnage Sintel du court métrage d'animation éponyme de la fondation Blender.

Pilotes de formes Contrôler l'interpolation de formes devient vite compliqué en se limitant à l'interface de gestion des poids des formes clés. Blender permet de créer des pilotes (ou *drivers*), c'est-à-dire des objets virtuels dont la position est associée aux poids d'une ou plusieurs clés de forme. Animer ces objets virtuels correspond alors à animer les poids des clés de forme qui y sont rattachés, et donc à animer la déformation de forme elle-même. Cette fonctionnalité permet aux animateurs de créer

leur propre interface graphique de contrôle afin de définir une forme mélangée. La figure II.9 montre un exemple d'interpolation de forme contrôlée par un pilote de forme.

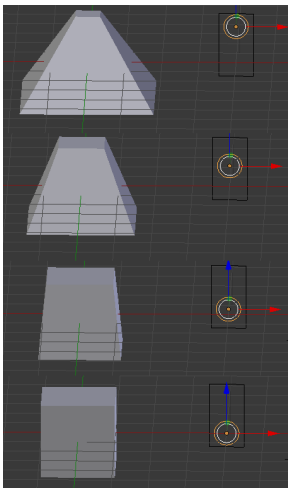


FIGURE II.9 – Pilotes de formes dans Blender

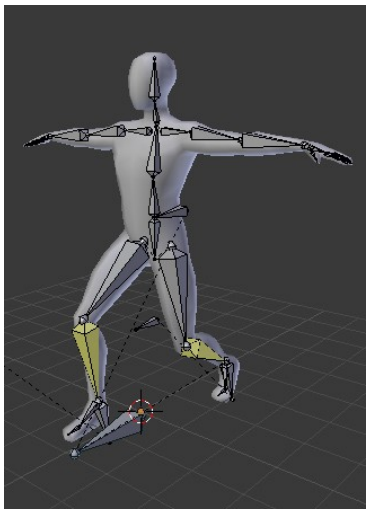


FIGURE II.10 – Armatures dans Blender (Crédit)

2. Définition de poses

Les armatures sont un autre moyen proposé par Blender pour contrôler la déformation d'un maillage ou d'un ensemble de maillages. Une armature peut être vue comme le squelette d'un personnage. Elle est composée d'un ou plusieurs *os* organisés hiérarchiquement auxquels peuvent être attachés des maillages. Ces derniers suivront alors les mouvements et les déformations de cet os. La figure II.10 montre un exemple d'armature dans Blender.

Les mouvements et les déformations d'un os peuvent être contraints en fonction des mouvements que l'animateur souhaite permettre au personnage. Une première contrainte que l'on peut souhaiter est de forcer l'os à garder la même longueur, limitant ainsi ses mouvements à une rotation. On peut également souhaiter figer une ou plusieurs rotations. L'os correspondant situé à un coude pourra ainsi être contraint à n'avoir qu'un degré de liberté. Enfin, on peut souhaiter limiter la rotation sur un intervalle donné, définissant ainsi un angle minimum et un angle maximum. La pose

d'une armature peut être modifiée de deux façons : par *cinématique directe* ou par *cinématique inverse*.

Cinématique directe La méthode par cinématique directe (ou *forward kinematics*) consiste à modifier directement l'angle entre un os et son parent. Cette méthode donne un contrôle précis mais peut rapidement s'avérer fastidieuse lorsque le nombre d'os augmente.

Cinématique inverse La méthode par cinématique inverse (ou *inverse kinematics*) consiste à définir des cibles qu'un os va chercher à approcher. Blender utilise alors un moteur de cinématique inverse pour trouver la pose de l'armature qui minimise la distance entre l'os et la cible tout en respectant les contraintes de l'armature. Nous n'entrerons pas ici dans le détail du fonctionnement d'un moteur de cinématique inverse, en revanche le problème de la cinématique inverse et des algorithmes de résolution est présenté dans le chapitre IV.

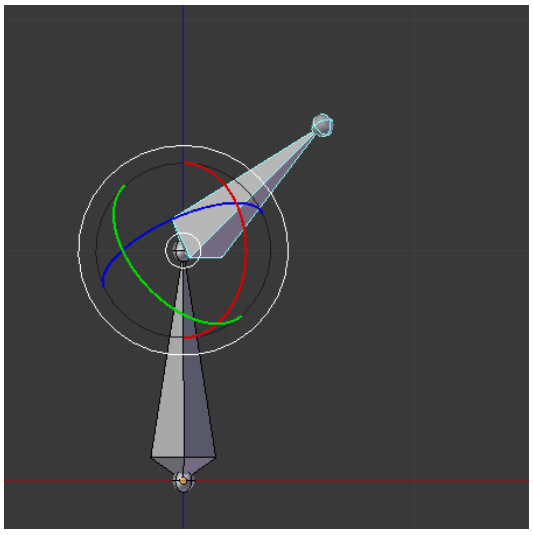


FIGURE II.11 – Cinématique directe

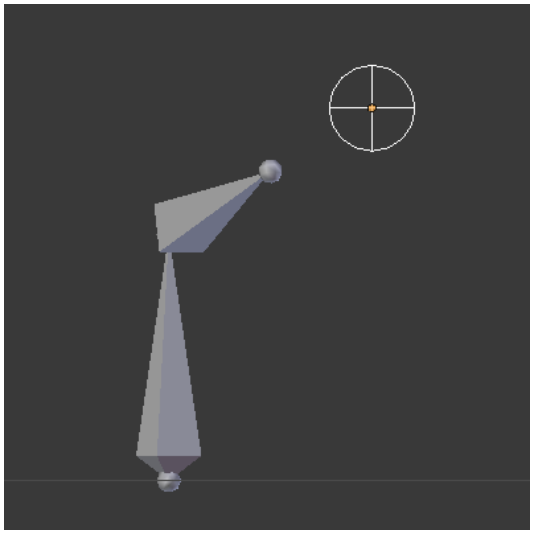


FIGURE II.12 – Cinématique inverse

3. Animation par frames clés

L'animation dans Blender est basée sur le principe des *frames clés* (ou *key frames*). Une frame clé est un enregistrement de la valeur d'un paramètre à un moment donné de l'animation. Dans Blender, tous les paramètres peuvent être animés, aussi bien la position des objets, la pose des personnages ou encore le poids des différentes formes

clés. Pour animer une action, l'animateur commence ainsi par définir des frames clés de l'animation en choisissant les paramètres à enregistrer et à quel moment le faire. Ces frames clés peuvent ensuite être décalées dans le temps sur la *feuille de clés* (ou *dope-sheet*) afin de parfaire le timing de l'animation. Enfin, le profil de vitesse du passage d'une frame clé à la suivante peut être défini grâce à l'*éditeur de courbe* (ou *F-curve*).

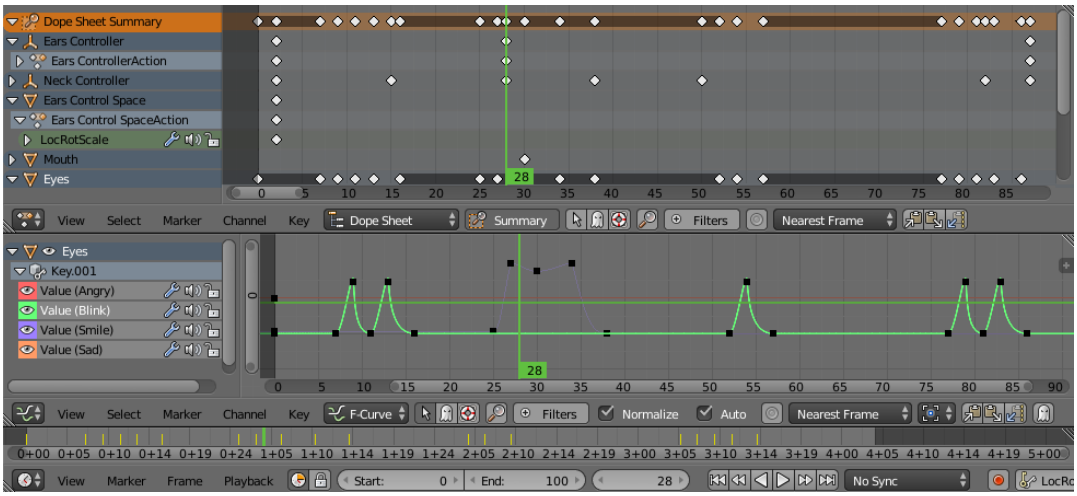


FIGURE II.13 – Interface d'animation par frames clés de Blender. La partie haute de l'image montre la *feuille de clés* (ou *dope-sheet*) et la partie basse l'*éditeur de courbe* (ou *F-curve*).

4. API Python

Les fonctionnalités de Blender peuvent être étendues grâce à l'interpréteur Python intégré. Les possibilités offertes par l'API sont nombreuses. Premièrement, l'API donne accès aux structures de données de Blender, en lecture et en écriture, permettant par exemple de récupérer la position d'un objet. Elle permet également d'attacher des fonctions au système d'événements de Blender. Enfin, elle permet d'ajouter des éléments à l'interface utilisateur de Blender.

E. Présentation de notre système

Nous avons présenté précédemment les différentes catégories de composants couramment utilisés en robotique sociale, ainsi que les différentes fonctionnalités de

modélisation, d'animation et d'extensibilité de Blender. Nous décrivons ici comment nous utilisons les fonctionnalités de Blender afin d'implémenter un système d'animation pour chaque catégorie de composants.

1. Correspondance Fonctionnalités-Actionneurs

Le fonctionnement général de notre système est de transposer l'état d'un objet virtuel animé dans Blender à des actionneurs sur le robot réel. Le contrôle des différents types d'actionneurs peut être divisé en deux groupes selon le principe de fonctionnement : l'animation des moteurs et l'animation des écrans.

a. Animation des moteurs

Parmi les fonctionnalités de Blender, nous avons présenté les *pilotes de formes* des objets virtuels dont la position dirige le poids de clés de forme. Notre système reprend le concept de pilotes afin de contrôler la position des moteurs principaux et secondaires, i.e. la position ou l'angle d'un objet virtuel animé dans Blender dirige l'angle d'un moteur physique. Deux types de *pilotes de moteurs* peuvent être utilisés : les armatures et les panneaux de contrôle.

Armatures Comme nous l'avons vu précédemment, les armatures permettent dans Blender de définir le squelette d'un personnage. Dans notre cas, elles sont une solution naturelle pour contrôler les moteurs principaux du robot. Une armature peut-être définie dans Blender pour modéliser la ou les chaînes cinématiques, chaque os représentant un moteur et reprenant ses contraintes, c'est-à-dire l'axe de rotation et l'angle minimum et maximum. Grâce aux fonctionnalités de cinématique inverse de Blender présentées précédemment, il est également possible de définir des cibles pour certains os de l'armature. Cela permet de contrôler sa pose en manipulant directement ces cibles. Le lien entre l'angle `x` de l'os `l_elbow` et un moteur principal (Dynamixel) d'id$=12$ s'écrit dans le fichier de correspondances :

```
{
    'from': {
        'path': 'l_elbow/angle/x',
        'min':  0,
        'max':  90
    },
    'to': {
        'path': 'Dynamixel/12',
        'min':  -90,
        'max':  0
    }
}
```

Panneau de contrôle Nous avons présenté précédemment des fonctionnalités de Blender permettant de définir une interface de contrôle personnalisée pour contrôler les poids des différentes formes clés. De la même façon, notre système permet de récupérer la *position* d'un objet virtuel pour contrôler l'angle d'un moteur. L'animateur peut par exemple créer un *slider* en définissant un objet virtuel contraint en `x` sur un intervalle donné et fixé sur `y` et `z`. Ce type d'interface de contrôle permet de réunir le contrôle de plusieurs éléments au même endroit, ce qui peut par exemple être utile pour les moteurs secondaires animant les expressions faciales. Le lien entre la position en `x` de l'objet virtuel `ears_slider` et un moteur secondaire (Hobby) d'id=6 s'écrit dans le fichier de correspondances :

```
{
    'from': {
        'path': 'ears_slider/position/x',
        'min': 0,
        'max': 90,
    },
    'to': {
        'path': 'Hobby/6',
        'min': 0,
        'max': 90
    }
}
```

b. Animation des écrans

L'animation des écrans fonctionne dans notre système en affichant le rendu d'une caméra virtuelle de Blender sur un écran de smartphone ou de tablette. Cela permet à l'animateur d'utiliser toutes les fonctions de modélisation et d'interpolation de formes fournies par Blender pour animer des expressions faciales, de l'iconographie ou encore du texte. Le lien entre une caméra virtuelle `face_camera` et l'écran (Screen) d'id `1` s'écrit dans le fichier de correspondances :

```
{'from': {'path': 'face_camera'}, 'to': {'path': 'Screen/1'}}
```

2. Architecture

La figure II.14 présente l'architecture de notre système implémentant la synchronisation de chaque actionneur avec l'objet virtuel correspondant animé dans Blender. L'architecture peut être décomposée en 2 parties : le *répartiteur* qui communique avec Blender, et les contrôleurs qui communiquent avec les actionneurs.

a. Répartiteur

Le *répartiteur* est le composant principal de notre système. Son fonctionnement est contrôlé par le *fichier de correspondances* décrit précédemment qui définit les associations entre objets virtuels et actionneurs. Les objets virtuels et les actionneurs sont représentés par une adresse. L'adresse d'origine contient le nom de l'objet virtuel et de l'attribut d'intérêt. L'adresse de destination est composée du nom du contrôleur devant recevoir l'information suivi d'un identificateur de l'actionneur. L'association peut également contenir une transformation à opérer sur la valeur de l'attribut avant de l'envoyer au contrôleur.

Le *répartiteur* vient se greffer au système d'événements de Blender en s'attachant comme fonction de rappel de l'événement `scene_update_post`. Chaque fois que Blender effectue une mise à jour, le *répartiteur* vérifie si la scène virtuelle a été modifiée, c'est-à-dire si un objet virtuel a été déplacé, soit par l'utilisateur, soit par le moteur d'animation de Blender.

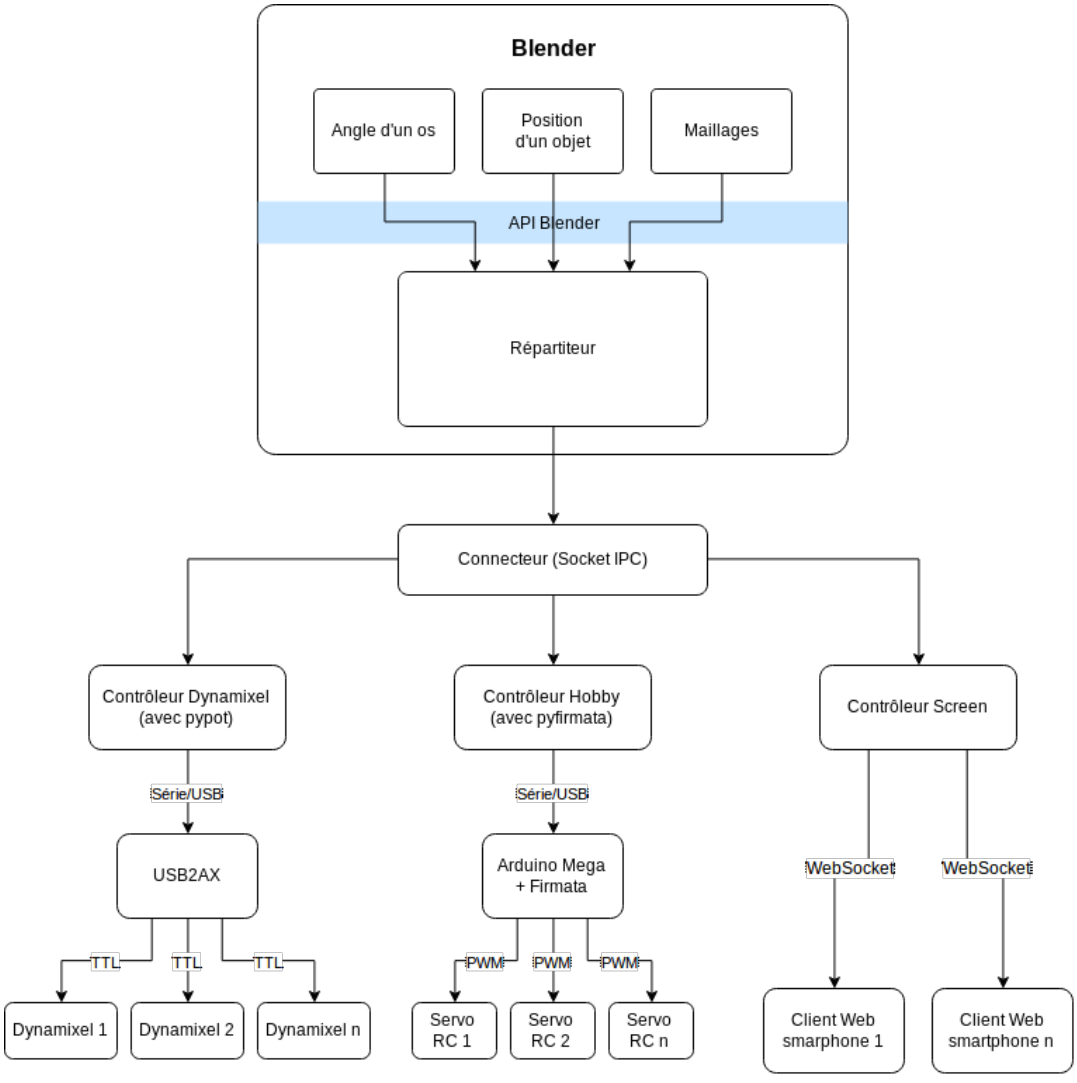


FIGURE II.14 – Diagramme de l'architecture du système

Si tel est le cas, le *répartiteur* traite chaque association définie dans le *fichier de correspondances* : il récupère l'information définie par l'adresse d'origine et la transmet au contrôleur indiqué par l'adresse de destination, accompagnée de l'identificateur de l'actionneur destinataire. Si l'objet virtuel d'origine est une caméra, l'information transmise contient les paramètres de cette caméra et les maillages de la scène. Sinon, elle contient la valeur de l'attribut d'intérêt défini dans l'adresse d'origine.

b. Contrôleurs

Un contrôleur gère l'ensemble des actionneurs d'un même type. Ces derniers sont différenciés au sein d'un même contrôleur par un identifiant unique. Ce sont ces identifiants que l'on retrouve dans les adresses de destination du fichier de correspondances. Les contrôleurs sont des processus indépendants avec lesquels le *répartiteur* communique de manière asynchrone par le biais d'un socket de communication inter-processus que nous appellerons *connecteur*. Nous utilisons l'implémentation du patron de communication *Publier-Souscrire* de la librairie *ZeroMQ*[7]. Le nom du contrôleur contenu dans l'adresse de destination correspond au *sujet* sur lequel l'information sera publiée par le *répartiteur*. Ce choix d'architecture permet d'intégrer de nouveaux actionneurs au système en ajoutant des contrôleurs. Il suffit pour cela de choisir un identifiant de *sujet* et de l'utiliser dans une adresse de destination. Le nouveau contrôleur n'aura alors qu'à souscrire au *sujet* pour se greffer au système.

Par exemple, pour ajouter le contrôle de LEDs au système :

1. Ajouter une association entre une caméra virtuelle et une adresse commençant par le sujet `LED` au *fichier de correspondances*, soit :

```
{'from': {'path': 'leds_camera'}, 'to': {'path': 'LED/1'}}
```

2. Souscrire le contrôleur de LEDs au sujet `LED` au travers du socket *connecteur*.

La souscription a dans notre implémentation une particularité qui est d'être "fainéante", c'est-à-dire qu'elle correspond à ne récupérer que le dernier message publié sur le *sujet*. Cela permet d'utiliser l'information la plus récente afin de limiter la latence entre Blender et les actionneurs. La synchronisation fainéante est possible car les moteurs utilisés sont dotés de leur propre boucle de contrôle permettant de les contrôler en position plutôt qu'en vitesse.

7. http://zero.mq - dernière visite : 01/11/2019

Chaque contrôleur gère un type d'actionneur, distribuant les commandes aux différentes instances grâce à leur identifiant. En reprenant l'exemple précédent, l'adresse $'LED/1'$ correspond à transmettre le message au contrôleur LED qui l'utilisera pour contrôler la LED numéro 1.

Le système intègre 3 contrôleurs : `Dynamixel` pour les moteurs Dynamixel de Robotis, `Hobby` pour les servomoteurs RC et `Screen` pour les écrans.

Contrôleur `Dynamixel` Les moteurs Dynamixel de Robotis sont souvent utilisés en robotique sociale dans la phase de prototypage. Ce sont des servomoteurs digitaux dotés d'une interface TTL half-duplex. Nous utilisons une carte *USB2AX*[8] et la librairie *PyPot* développée dans le cadre du projet Poppy (Lapeyre et al., 2014) afin de piloter les moteurs.

Contrôleur `Hobby` Les servomoteurs RC doivent être contrôlés grâce à un signal PWM (pour *Pulse Width Modulation*) encodant un angle cible. Nous utilisons une carte *Arduino MEGA* pour générer ce signal en utilisant la librairie *Servo* incluse dans la distribution Arduino standard. Afin de contrôler directement le signal généré par la carte depuis le contrôleur écrit en Python, nous utilisons *firmata* sur le Arduino, un firmware permettant de contrôler les fonctionnalités de la carte à distance, et la librairie *pyfirmata* côté Python. La communication entre la carte Arduino et le contrôleur se fait par une connection USB.

Contrôleur `Screen` Nous utilisons des smartphones Android pour les écrans (et potentiellement des tablettes). Ils exécutent une application web connectée au contrôleur `Screen` en *WebSocket*. Cette application est chargée de faire le rendu 3D de la scène grâce aux paramètres de la caméra et les maillages de la scène transmis par le contrôleur. Elle utilise pour cela la librairie *Three.js*, une librairie *Javascript* permettant de créer des scènes 3D et d'en générer le rendu grâce à l'API *WebGL*. Le rendu n'est donc pas généré par Blender mais directement par le navigateur sur le téléphone.

F. Réalisations

Afin d'expérimenter et valider ce système de prototypage de robots animés, nous l'avons appliqué à l'animation de deux exemples de robots sociaux : Mia, un robot

8. http://www.xevelabs.com/doku.php?id=product:usb2ax:usb2ax - dernière visite : 01/11/2019

que nous avons conçu spécifiquement dans ce but, et Poppy, un robot open-source développé par l'équipe Flowers de l'Inria Bordeaux.

1. Mia : une tête robotique expressive

Nous avons conçu le robot Mia comme exemple du système développé (figure II.17). Il est doté des 3 types d'actionneurs que nous avons identifiés comme fréquemment utilisés en robotique sociale, c'est-à-dire les moteurs principaux, les moteurs secondaires et les écrans.

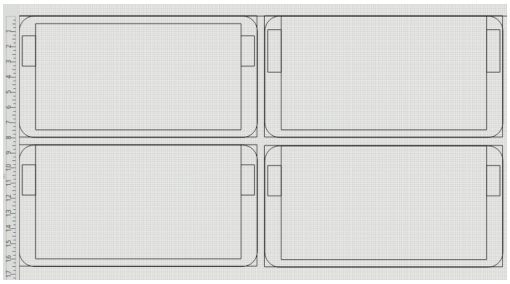


FIGURE II.15 – Capture d'écran de la conception de la structure de Mia.

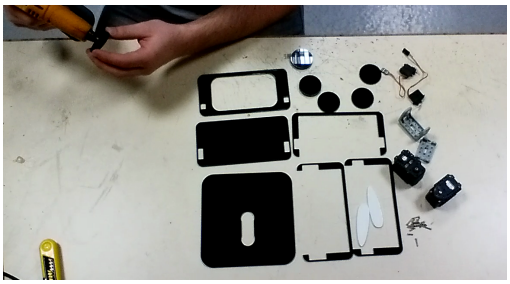


FIGURE II.16 – Capture d'écran de l'assemblage d'une version plexiglass de Mia.

Mia est doté de 4 degrés de liberté, 2 moteurs principaux pour la nuque et 1 moteur secondaire par oreille. La nuque est motorisée grâce à des servomoteurs Dynamixel *AX12A* de Robotis tandis que les oreilles utilisent chacune un servomoteur RC. La structure a été conçue à partir de pièces fournies avec les Dynamixel et de pièces découpées à la découpeuse laser dans des panneaux en fibre de bois (MDF). Du côté de Blender, les 2 servomoteurs Dynamixel sont contrôlés par une armature reprenant la structure géométrique de la nuque. Les 2 servomoteurs RC (oreilles) sont eux contrôlés grâce à un slider 2D. La figure II.19 présente ces deux interfaces de contrôle.

Mia est également doté d'un écran affichant 2 yeux et une bouche. Les yeux sont des *formes mélangées* à partir de la forme initiale (figure II.18 (a)) et de 4 *formes clés* (figure II.18 (b)) dont les poids sont contrôlés dans l'interface de Blender grâce à 4 sliders.

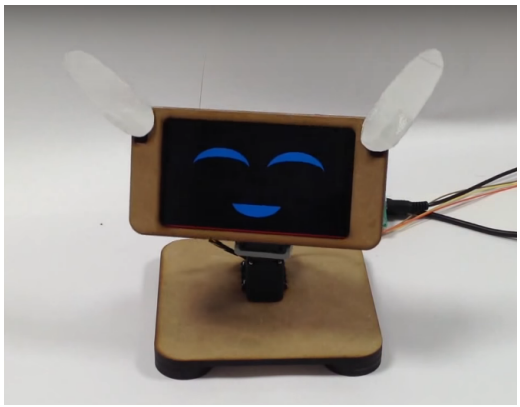


FIGURE II.17 – Présentation du robot Mia.

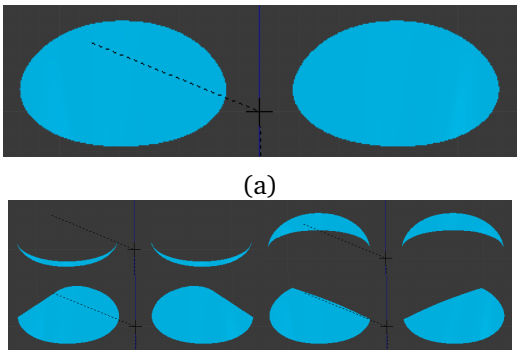


FIGURE II.18 – Les formes des yeux du robot Mia. (a) Forme initiale. (b) Formes clés.

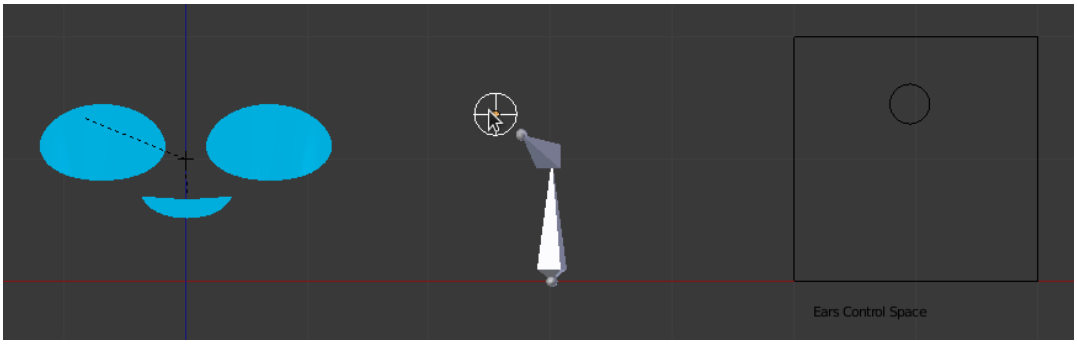


FIGURE II.19 – Interface de contrôle de Mia.

Observations et discussion

Mia peut très simplement être animé avec ce système. Le robot physique permet un retour immédiat. Le mouvement est réalisé avec toutes les contraintes de vitesse ou d'accélération du robot ainsi qu'avec le son généré par les moteurs. Ce retour permet de juger rapidement du résultat in fine, et notamment de vérifier que le mouvement conserve bien le style voulu.

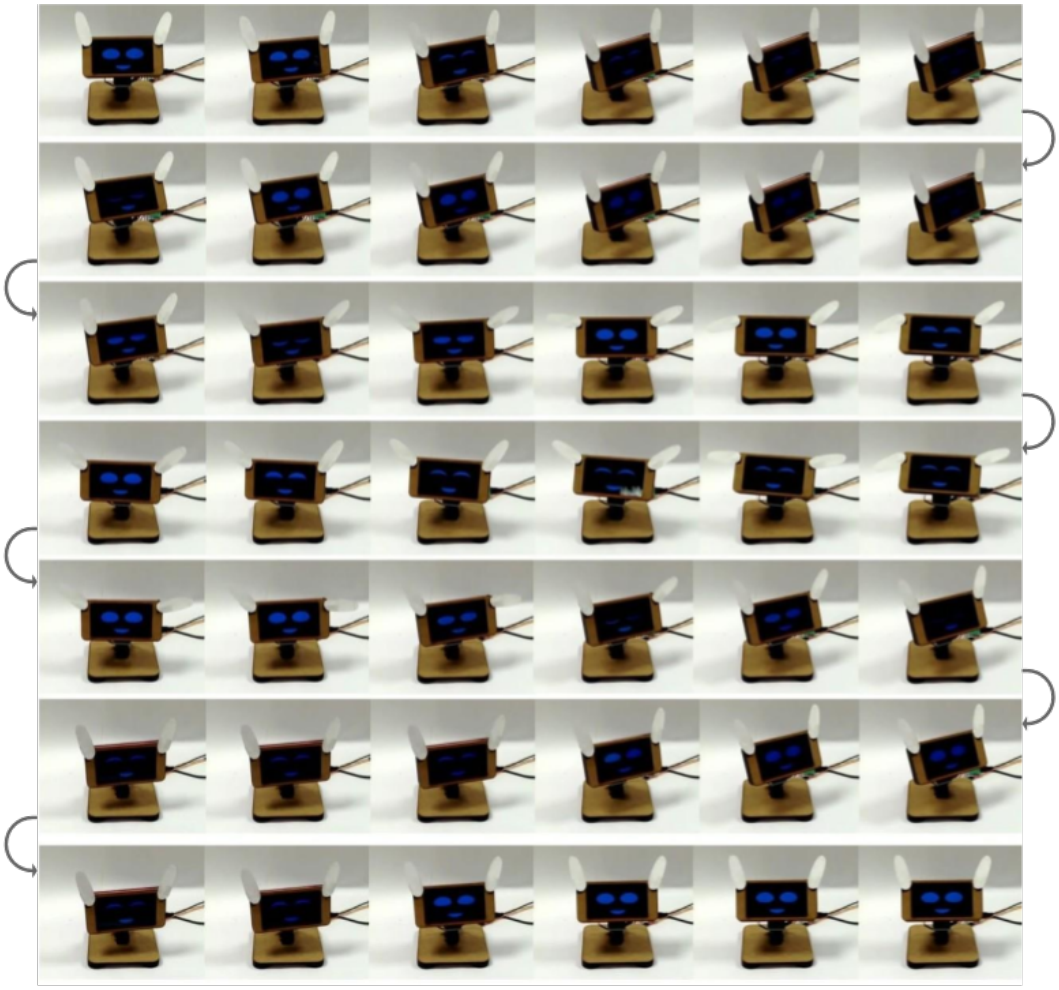


FIGURE II.20 – Exemple d'animation de Mia. La vidéo correspondante est disponible disponible sur Youtube[a].

a. https://youtu.be/aQ-g0b3mRRo

2. Poppy : un robot humanoïde

Poppy est un robot open source [9] développé par l'équipe Flowers de l'Inria Bordeaux. Initialement conçu comme plateforme expérimentale pour l'étude de l'acquisition de la marche, il a depuis été utilisé dans plusieurs performances artistiques, comme plateforme éducative ou encore comme robot compagnon pour des enfants hospitalisés.

FIGURE II.21 – Présentation du Poppy Torso (Crédit)

Nous utilisons la version *torse* du Poppy. Il est doté de 13 degrés de liberté : 2 pour la nuque, 3 pour le buste et 4 pour chaque bras. Ceux-ci sont tous motorisés grâce à des servomoteurs Dynamixel *MX28AT* de Robotis, à l'exception de la nuque qui utilise des servomoteurs *AX12A* de la même gamme. Du côté de Blender, les moteurs du robot sont contrôlés par une armature reprenant la structure géométrique du robot réel.

Observations et discussion

Bien que Poppy puisse être animé grâce à ce système, on rencontre de nouvelles problématiques liées au nombre important de degrés de liberté du robot. Premièrement, la cinématique inverse doit être utilisée pour définir la pose du robot, contrôler directement les angles devenant fastidieux. Mais utiliser la cinématique inverse pour définir la pose indirectement semble être une opportunité gâchée. En effet, utiliser la cinématique inverse correspond à simuler la possibilité de manipuler l'armature du robot. Or nous pourrions définir la pose par manipulation directe en utilisant le robot comme interface tangible. Deuxièmement, le robot virtuel doit être manipulé avec précaution pour éviter les auto-collisions sur le robot réel. Un mécanisme de détection

9. https://www.poppy-project.org/fr/ - dernière visite : 01/11/2019

des auto-collisions permettrait d'éviter de faire peser cette tâche et la charge cognitive associée sur l'utilisateur.

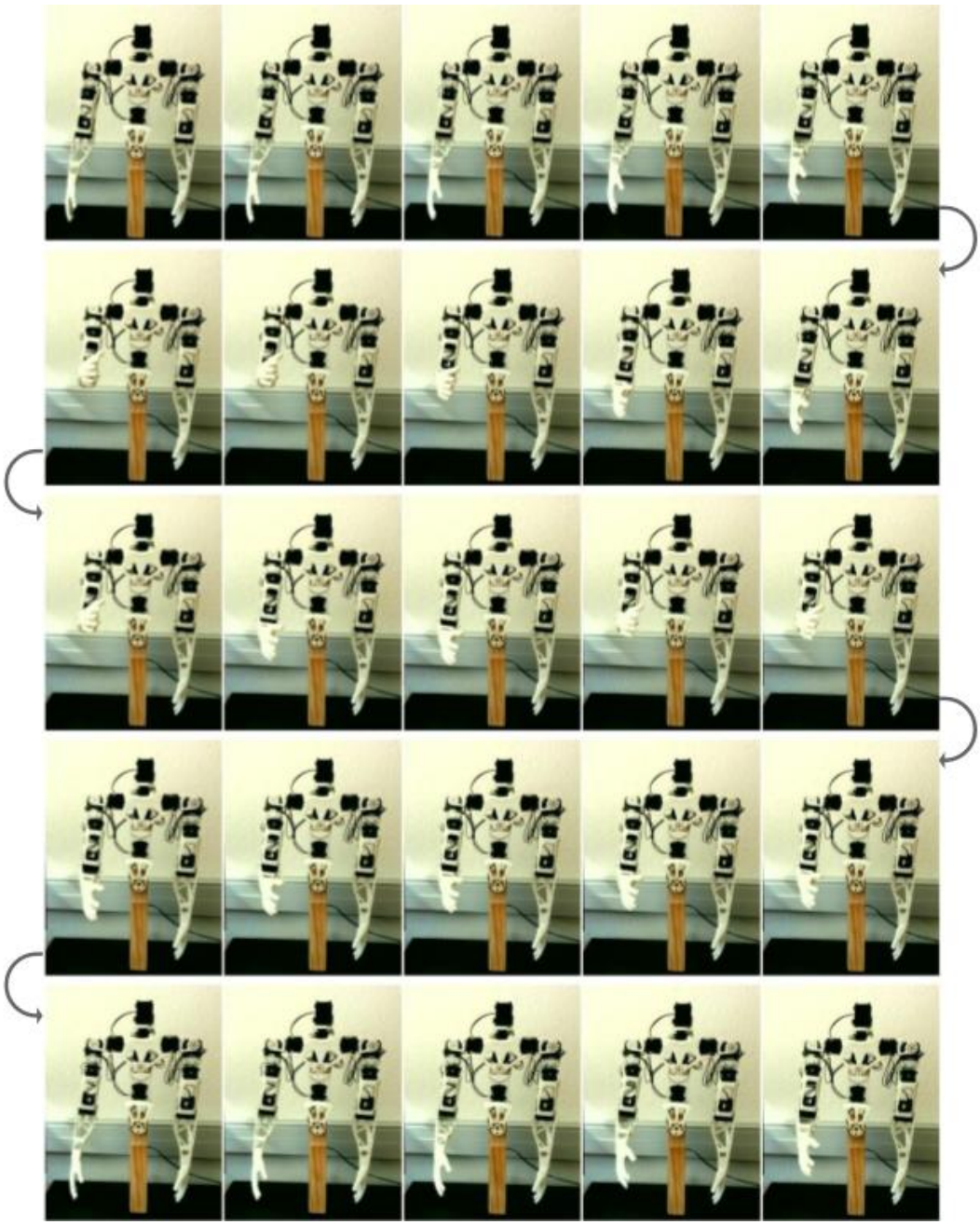


FIGURE II.22 – Exemple d'animation du robot Poppy.

G. Conclusion

Nous avons présenté dans ce chapitre un framework de prototypage de robots sociaux animés implémenté sous la forme d'un plugin au logiciel Blender, un open-source d'animation 3D. Après une analyse des différents types d'effecteurs expressifs utilisés en robotique social, nous avons montré comment ceux-ci peuvent être mis en correspondance avec les fonctionnalités d'animation de Blender.

Ce framework a été évalué expérimentalement par le prototypage du robot Mia, une tête robotique expressive dotée d'une nuque à 2 degrés de libertés, d'oreilles articulées ainsi que d'un écran pour son visage, reprenant ainsi trois types d'effecteurs utilisés en robotique sociale. Nous avons également évalué la capacité de ce framework à généraliser à des robots plus complexes en animant le torse du robot Poppy.

Si le robot Mia peut très facilement être animé au moyen de ce système, l'animation du robot Poppy s'est avérée plus difficile. En effet, la tâche de définition des poses clés est fortement complexifiée par le grand nombre de degrés de liberté à contrôler ainsi que par les risques de collisions entre les différentes parties du robot. Ces difficultés ont motivé la conception de l'outil d'animation spécialisé pour la robotique que nous présenterons dans le chapitre suivant.

Chapitre III

Spécialisation de l'interface d'animation au medium robotique

Dans le chapitre II nous avons présenté un système de prototypage de robots sociaux animés que nous avons développé en réutilisant un logiciel d'animation 3D existant et en faisant correspondre ces fonctionnalités aux besoins de l'animation de robots. Nous avons observé que cette approche est bien adaptée pour des robots dotés de peu de degrés de liberté et pour lesquels il n'y a pas possibilité d'auto-collision. Cependant, pour des robots plus complexes tels que le Poppy, une adaptation directe de l'interface utilisateur des logiciels d'animation 3D nécessite d'utiliser la cinématique inverse pour définir la pose du robot, simulant ainsi une manipulation qui pourrait dans l'idéal être faite directement sur le robot.

Une autre stratégie consiste à créer des outils pour animer les robots en utilisant le paradigme que les animateurs utilisent déjà pour l'animation 3D de films ou de jeux vidéo. La conception d'outil d'animation robotique a été explorée de multiple fois comme nous le présenterons par la suite. Mais ces logiciels sont conçus pour un robot spécifique et sont souvent propriétaires. Cela mène à une situation où chaque équipe de recherche qui conçoit un nouveau robot devrait mettre en place un outil d'animation pour bénéficier des connaissances et de l'art des animateurs. En raison de la complexité de la tâche, les outils d'animation pour un robot donné sont souvent limités ou inexistants.

Nous présentons dans ce chapitre notre proposition de résoudre ce problème en développant Open Robot Animator, un logiciel pouvant devenir une base commune pour l'animation de robots. Ce logiciel constitue un potentiel point de départ pour la communauté afin d'itérer sur l'interface utilisateur optimale pour l'animation robotique. Nous tirons également de ces expérimentations des principes généraux de cette interface utilisateur permettant de prendre en compte les contraintes et possibilités spécifiques du medium robotique.

Dans ce chapitre, nous commençons par décrire les précédents travaux sur la conception d'outil d'animation pour la robotique dans la section A. Dans la section B, nous présentons les concepts théoriques utilisés dans la conception de l'outil présenté dans ce chapitre. Nous présentons ensuite dans la section C l'outil d'animation que nous avons conçu autour de la définition de poses clés par manipulation directe.

A. Travaux connexes

Les équipes de Sony sont les premières à avoir conçu un outil d'animation (Kuroki et al., 2003) pour leur Sony Dream Robot, ou SDR, un robot humanoïde conçu pour le divertissement. Cet outil (figure III.1) est conçu autour de l'animation d'un modèle 3D du robot. Un éditeur graphique permet de définir les poses clés du mouvement par la manipulation de ce modèle 3D. Le timing de celles-ci peut ensuite être défini grâce à une frise temporelle. Différents types d'interpolation peuvent être choisis pour le passage d'une pose clé à la suivante. Le logiciel Choreonoid (figure III.2) conçu pour l'animation du robot humanoïde HRP3 (Nakaoka et al., 2010) reprend le même principe en y intégrant le respect des contraintes d'équilibre du robot.

D'autres travaux se passent de l'utilisation d'un modèle 3D pour la définition de poses. Chaque degré de liberté est alors animé indépendamment. C'est notamment le cas pour le logiciel d'animation du robot iCat (Van Breemen and Xue, 2006), une tête robotique dotée d'une nuque à 2 degrés de liberté et d'un visage expressif animé par 11 servomoteurs (Van Breemen, 2004). Ce logiciel permet de définir l'animation de chaque moteur au travers de sa propre courbe de position (figure III.3). Le logiciel d'animation développé par Saldien et al. (2014) pour le robot Probo (figure III.5) définit également l'animation indépendamment pour chaque moteur en définissant leurs positions à des moments clés qui sont ensuite interpolées linéairement.

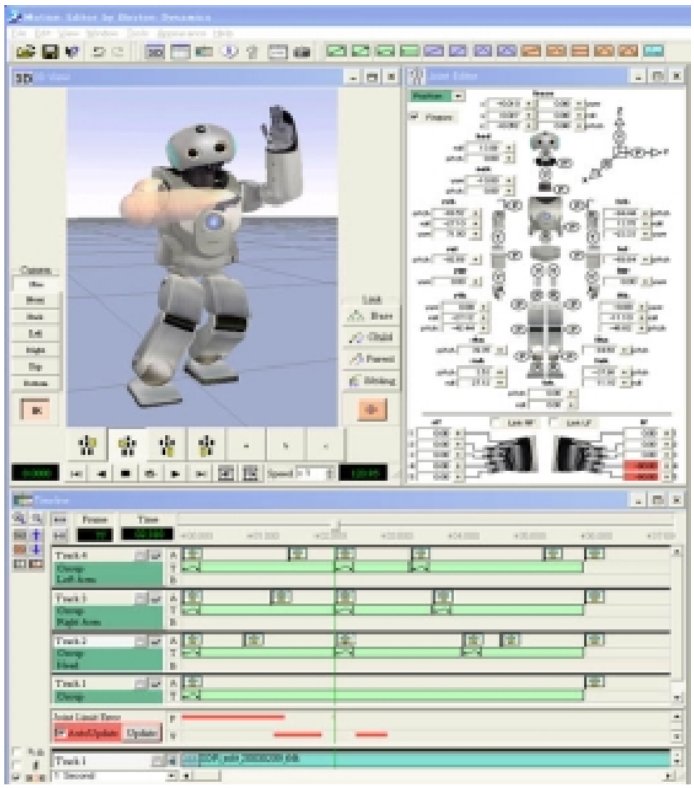


FIGURE III.1 – Interface de l'outil d'animation du Sony Dream Robot (Kuroki et al., 2003)

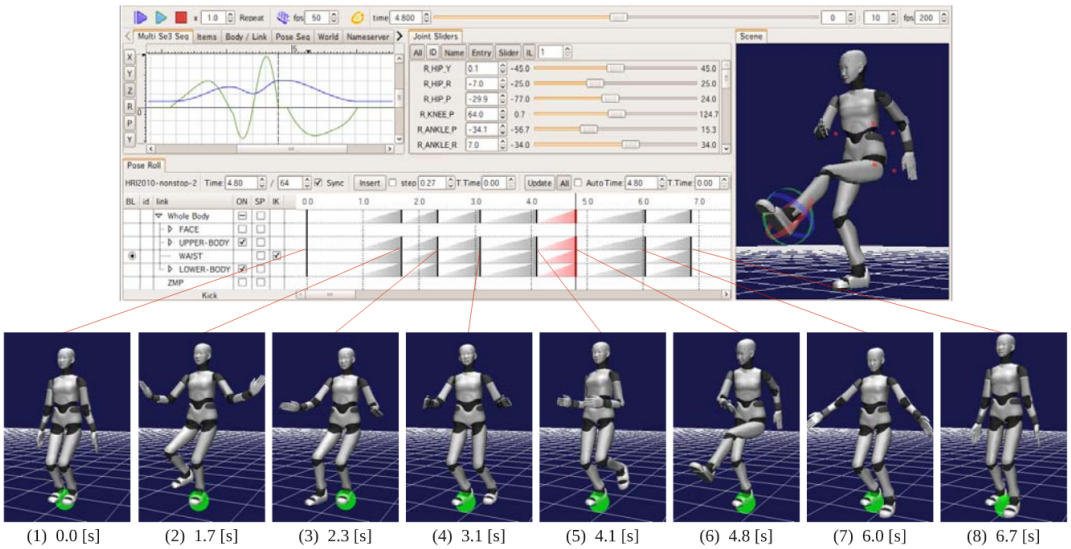


FIGURE III.2 – Interface de l'outil d'animation du robot humanoïde HRP3 (Nakaoka et al., 2010)

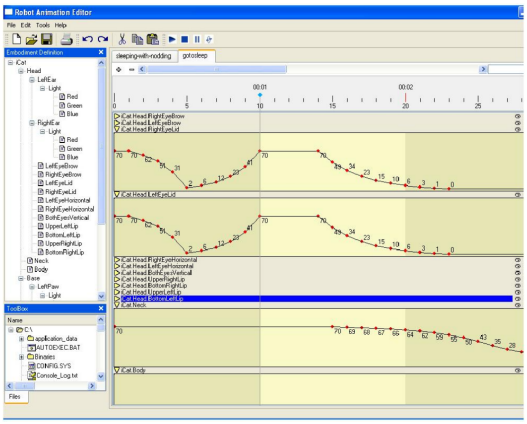


FIGURE III.3 – Interface de l'outil d'animation du robot iCat (Van Breemen and Xue, 2006)

FIGURE III.4 – Interface du logiciel Kouretes Motion Editor (Pierris and Lagoudakis, 2009)

Une autre approche de définition de pose est utilisée par le logiciel Kouretes Motion Editor (Pierris and Lagoudakis, 2009). Bien que n'étant pas conçu pour l'animation expressive, ce logiciel partage le besoin de définir une séquence de poses qui nous intéresse dans ce chapitre. Pierris and Lagoudakis (2009) proposent d'utiliser le robot comme interface d'édition de pose. L'éditeur permet ainsi de contrôler la rigidité du robot afin de pouvoir le manipuler pour enregistrer une séquence de poses clés (figure III.4). En revanche, l'interface ne permet pas de définir les profils de transition entre ces poses clés.

Le logiciel Choregraphe (Pot et al., 2009) d'Aldebaran est probablement le logiciel d'animation robotique le plus approfondi et le plus utilisé en robotique sociale du fait de son intégration au robot NAO. Son fonctionnement reprend celui des logiciels d'animation 3D tels que Blender [1], Maya [2] et 3ds Max [3]. L'éditeur d'animation est ainsi construit autour de deux vues (figure III.6) : "Worksheet" pour éditer le timing des poses clés et "Curves" pour modifier les profils d'interpolation entre ces différentes poses clés pour chacune des articulations du robot. Le principal inconvénient de Choregraphe est d'être un logiciel propriétaire dédié à l'animation des robots d'Aldebaran uniquement, ce qui limite fortement les possibilités d'expérimentation en terme d'interface.

1. https://www.blender.org - dernière visite : 01/11/2019
2. https://www.autodesk.com/products/maya/overview - dernière visite : 01/11/2019
3. https://www.autodesk.com/products/3ds-max/overview - dernière visite : 01/11/2019

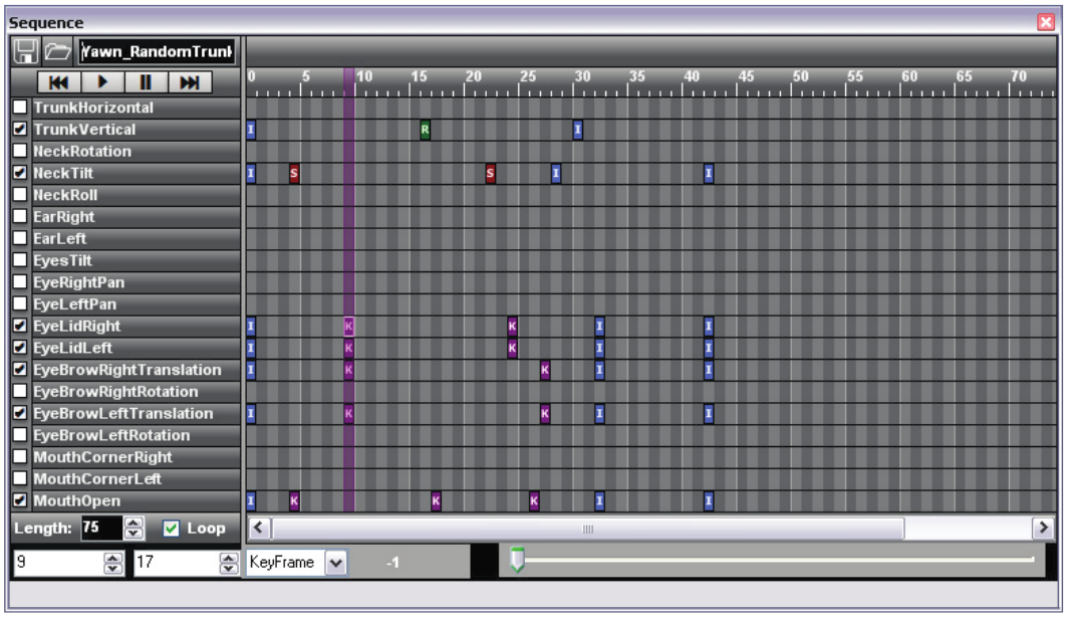


FIGURE III.5 – Interface de l'outil d'animation du robot Probo (Saldien et al., 2014)

B. Concepts théoriques

Nous présentons dans cette section quelques concepts théoriques utiles pour la conception d'un logiciel d'animation de robots. Nous présentons dans un premier temps les courbes de Bézier et en particulier leur restriction à des courbes utilisables pour définir des courbes d'animation. Nous présentons ensuite le principe de Modèle Géométrique Direct. Enfin, nous décrivons la théorie sous-tendant la détection d'auto-collisions.

1. Courbes de Bézier

Les courbes de Bézier sont des courbes paramétriques inventées en 1962 par Pierre Bézier, ingénieur chez le constructeur automobile Renault. Ces courbes ont été développées afin de spécifier numériquement la forme d'une courbe telle que pourrait la tracer un dessinateur. Depuis, ces courbes ont eu une influence importante dans l'informatisation de nombreux domaines artistiques, à la fois dans leur utilité d'origine en temps que structure de données mais également comme outil dans les logiciels de dessins.

De nombreux formats d'images vectoriels tel que PostScript ou SVG ou de polices de caractères telles TrueType ou OpenType les utilisent comme structure de données

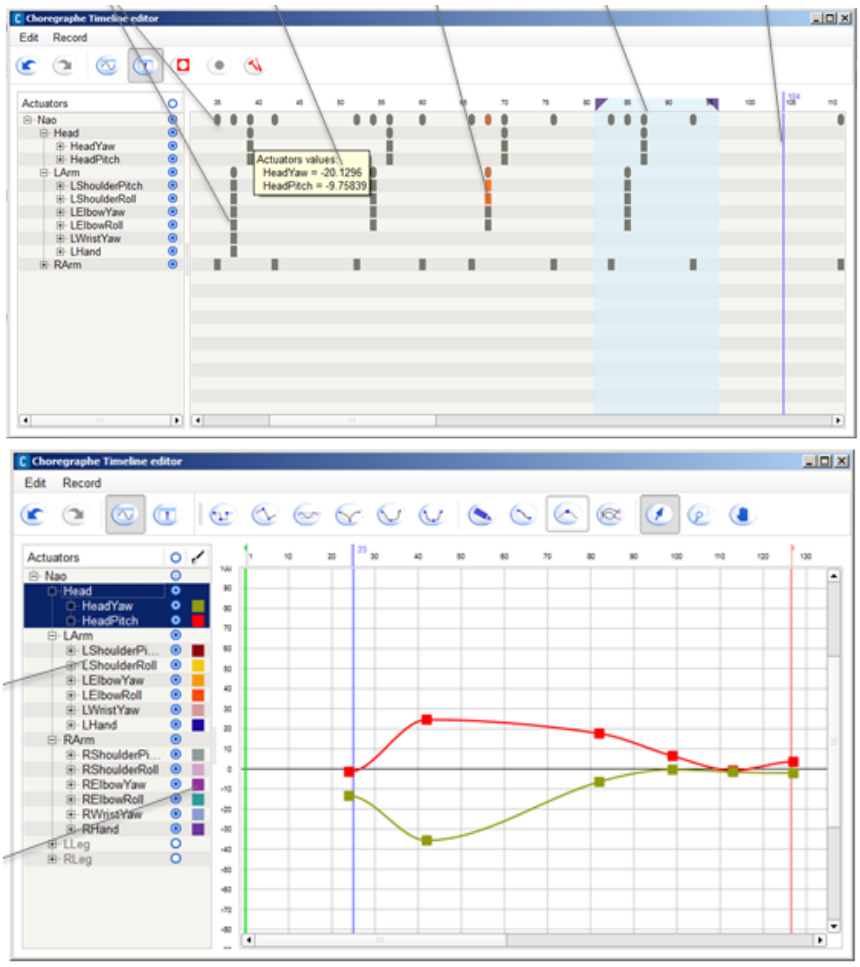


FIGURE III.6 – Interfaces des vues "Worksheet" (haut) et "Curves" (bas) du logiciel Choregraphe (Pot et al., 2009)

élémentaires. Mais leur utilisation la plus visible est sans doute comme outil de base des logiciels de création graphique et de dessin assisté par ordinateur. Les courbes de Bézier sont également utilisées dans le domaine des logiciels d'animation, domaine qui nous intéresse ici tout particulièrement.

L'intérêt de ces courbes comme interface graphique de modélisation vient du fait qu'elles sont conçues à partir de l'interpolation linéaire de points représentables dans le même espace que les points de la courbe. Concrètement, cela se manifeste par la possibilité pour l'utilisateur de manipuler ces points dits de contrôle sur l'écran pour manipuler la forme de la courbe.

La courbe de Bézier de degré n est formée par l'ensemble des points obtenus par la combinaison convexe des $n+1$ points $(P_0, P_1, ..., P_n)$ tels que :

$$P(\alpha) = \sum_{i=0}^{n} P_i \cdot b_{n,i}(\alpha) \quad \text{avec} \quad \alpha \in [0,1] \qquad \text{(Eq. III.1)}$$

Où $b_{n,i}(\alpha)$ sont les polynômes de Bernstein définis par :

$$b_{n,i}(\alpha) = \binom{n}{i} \cdot (1-\alpha)^{n-i} \cdot \alpha^i \quad \text{pour} \quad i = 0, 1, ..., n \qquad \text{(Eq. III.2)}$$

avec $\binom{n}{i}$ les coefficients binomiaux tels que

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}$$

$$\sum_{i=0}^{n} b_{n,i}(\alpha) = 1 \quad \text{avec} \quad \alpha \in [0,1] \qquad \text{(Eq. III.3)}$$

Une courbe de Bézier de degré n comporte au maximum n inflexions du fait de sa nature polynomiale. Il pourrait donc être tentant d'utiliser des courbes de Bézier de degré important afin de pouvoir exprimer des trajectoires complexes. Cependant, comme la modification d'un des points de contrôle a une influence sur la globalité des points de la courbe, cela complique rapidement le travail de l'utilisateur. En effet, à l'exception des extrémités de la courbe ($\alpha = 0$ et $\alpha = 1$), les points interpolés $P(\alpha)$ sont tous influencés par la totalité des points de contrôle P_i, l'ensemble de leurs polynômes associés $b_{n,i}(\alpha)$ étant non nuls pour α appartenant à $]0,1[$. Il est donc généralement préférable d'exprimer des courbes dotées d'un grand nombre d'inflexions grâce à des courbes de Bézier composites, ou PolyBézier, composées de plusieurs courbes de Bézier quadratiques ou cubiques misent bout à bout et partageant des points de contrôle. La figure III.7 montre ce qu'apporte les courbes de Bézier composites en terme de localité de la modification, comparées aux courbes de Bézier permettant de la représentation de courbe de complexité comparable.

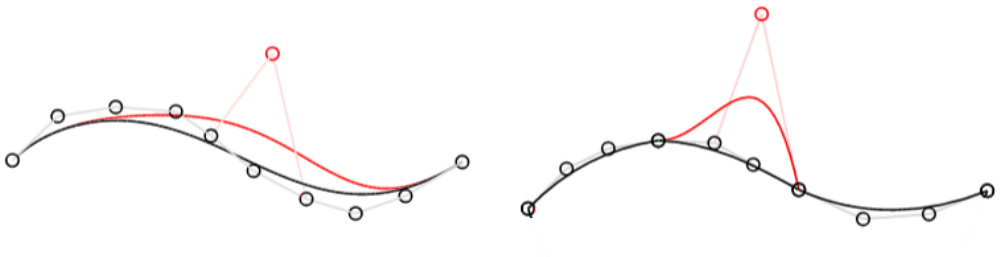


FIGURE III.7 – Différence de localité entre une courbe de Bézier de degré 10 (à gauche) et une courbe de Bézier composite (à droite).

Nous nous intéresserons plus particulièrement aux courbes de Bézier cubiques, dont les points interpolés sont définis par 4 points de contrôle P_0, P_1, P_2, P_3 tel que :

$$\begin{aligned} P(\alpha) = & P_0 \cdot (1-\alpha)^3 \\ & + P_1 \cdot 3\,(1-\alpha)^2\,\alpha \\ & + P_2 \cdot 3\,(1-\alpha)\,\alpha^2 \\ & + P_3 \cdot \alpha^3 \end{aligned} \qquad \text{(Eq. III.4)}$$

Dans le contexte qui nous intéresse dans ce chapitre, à savoir la conception d'un logiciel d'animation, les courbes de Bézier (simples ou composites) sont utiles pour permettre à un animateur de définir l'évolution d'un paramètre au cours du temps. Dans ce cas, on utilise une courbe de Bézier dont les points de contrôle appartiennent à un espace 2D dont une dimension est le paramètre animé et l'autre le temps. Ce qui nous intéresse afin d'animer le paramètre θ est de représenter une fonction $f(t) = \theta$. Sans contraindre la courbe de Bézier, celle-ci pourrait avoir deux valeurs de θ pour une valeur de t donnée, ce que l'on veut éviter.

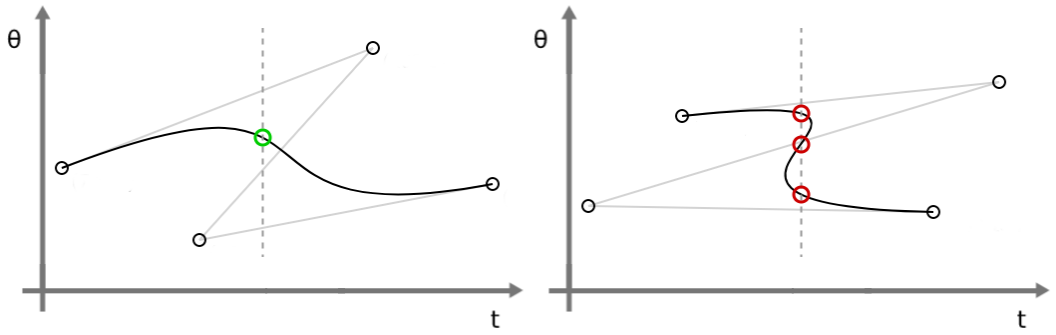


FIGURE III.8 – Courbes de Bézier valide (à gauche) et invalide (à droite) pour la représentation d'une fonction $f(t) = \theta$

Il faut donc contraindre les courbes de Bézier d'animation aux courbes pour lesquelles la composante t est strictement croissante, c'est-à-dire qui répondent à l'inéquation $t'(\alpha) > 0$. Nous pouvons utiliser pour cela le fait que la dérivée d'une courbe de Bézier cubique est une courbe de Bézier quadratique ayant pour points de contrôle $(P_1 - P_0)$, $(P_2 - P_1)$ et $(P_3 - P_2)$.

$$P'(\alpha) = (P_1 - P_0)\, b_{2,0}(\alpha) + (P_2 - P_1)\, b_{2,1}(\alpha) + (P_3 - P_2)\, b_{2,2}(\alpha) \qquad \text{(Eq. III.5)}$$

En réécrivant cette équation pour t uniquement on obtient :

$$t'(\alpha) = (t_1 - t_0)\, b_{2,0}(\alpha) + (t_2 - t_1)\, b_{2,1}(\alpha) + (t_3 - t_2)\, b_{2,2}(\alpha) \qquad \text{(Eq. III.6)}$$

Et pour $\alpha = 0$ et $\alpha = 1$:

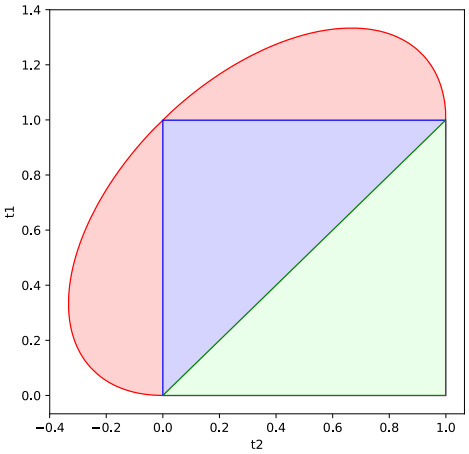
$$t'(0) = (t_1 - t_0) \quad et \quad t'(1) = (t_3 - t_2) \qquad \text{(Eq. III.7)}$$

Étant donné que les polynômes de Bernstein sont strictement positifs pour $\alpha \in]0, 1[$, une solution triviale est de contraindre t_0, t_1, t_2 et t_3 tel que :

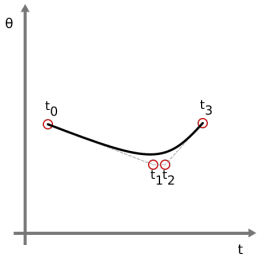
$$t_0 < t_1 \leq t_2 < t_3 \qquad \text{(Eq. III.8)}$$

Pour autant, cette contrainte est forte et limite fortement les courbes qui peuvent être exprimées par ces courbes de Bézier. En manipulant des courbes de Bézier 2D, on peut voir qu'un grand nombre de courbes que l'on peut construire représente des fonctions valides sans pour autant répondre à cette première contrainte.

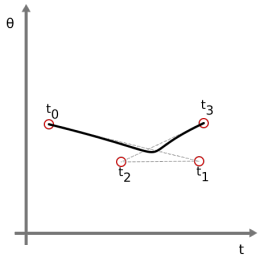
La figure III.9 représente toutes les courbes de Bézier possibles en fixant $t_0 = 0$ et $t_3 = 1$. Les courbes de Bézier représentées par les zones verte, bleue et rouge



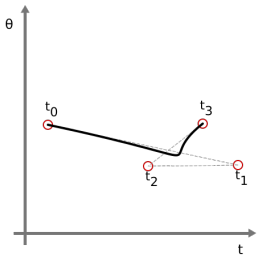
(a) Courbes de Bézier strictement croissantes sur t pour $t_0 = 0$ et $t_3 = 1$



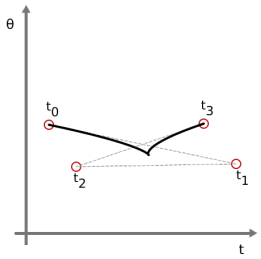
(b) Courbe valide (zone verte)
$t_0 < t_1 \leq t_2 < t_3$



(c) Courbe valide (zone bleue)
$t_0 < t_2 < t_1 < t_3$



(d) Courbe valide (zone rouge)
$t_1 > t_3$ ou $t_2 < t_0$



(e) Courbe invalide (zone blanche)

FIGURE III.9 – Schéma des contraintes applicables aux courbes de Bézier. (a) Catégories des relations entre les points de contrôle t_1 et t_2. (b) à (e) Exemples de courbes de Bézier issues de chacune des zones de l'espace des paramètres t_1 et t_2.

respectent bien le fait de ne pouvoir représenter que des courbes dont la composante t est strictement croissante.

Posons l'hypothèse d'une contrainte plus faible, contraignant seulement les deux points de contrôle centraux à être compris entre le premier et le dernier point de contrôle de début et de fin pour leur dimension temporelle, soit :

$$t_0 < t_1 < t_3 \quad et \quad t_0 < t_2 < t_3 \tag{Eq. III.9}$$

Comme nous l'avons vu ci-dessus $t'(\alpha) > 0$ si $t_0 < t_1 \leq t_2 < t_3$. Nous étudions donc le cas complémentaire tel que :

$$t_0 < t_2 < t_1 < t_3 \tag{Eq. III.10}$$

Ce cas peut être visualisé sous la forme présentée sur la figure III.10 avec :

$$\begin{aligned} X &= t_1 - t_0, \quad X > 0 \\ Y &= t_2 - t_1, \quad Y < 0 \\ Z &= t_3 - t_2, \quad Z > 0 \end{aligned} \tag{Eq. III.11}$$

On obtient ainsi le polynôme de second degré suivant :

$$\begin{aligned} t'(\alpha) &= X\ b_{2,0}(\alpha) + Y\ b_{2,1}(\alpha) + Z\ b_{2,2}(\alpha) \\ &= 3\ (1-\alpha)^2\ X + 6\ (1-\alpha)\ \alpha\ Y + 3\alpha^2\ Z \\ &= 3\ [\ (1 - 2\alpha + \alpha^2)\ X + (2\alpha + 2\alpha^2)\ Y + \alpha^2\ Z\ ] \\ &= 3\ [\ \alpha^2\ \underbrace{(X + Z - 2Y)}_{P} + \alpha\ \underbrace{(2Y - 2X)}_{Q} + \underbrace{X}_{R}\ ] \end{aligned} \tag{Eq. III.12}$$

On peut vérifier que $t'(\alpha)$ est strictement positif en étudiant le signe du discriminant Δ et du coefficient quadratique P :

$$\begin{aligned} \Delta &= Q^2 - 4PR \\ &= 4\ (Y^2 - 2XY + X^2) - 4\ (X + Z - 2Y)\ X \\ &= 4\ (Y^2 - XZ) \end{aligned} \tag{Eq. III.13}$$

Notons que dans le cas de la contrainte Eq. III.10 représenté sur la figure III.10 on a :

$$|Y| < min(X, Z) \tag{Eq. III.14}$$

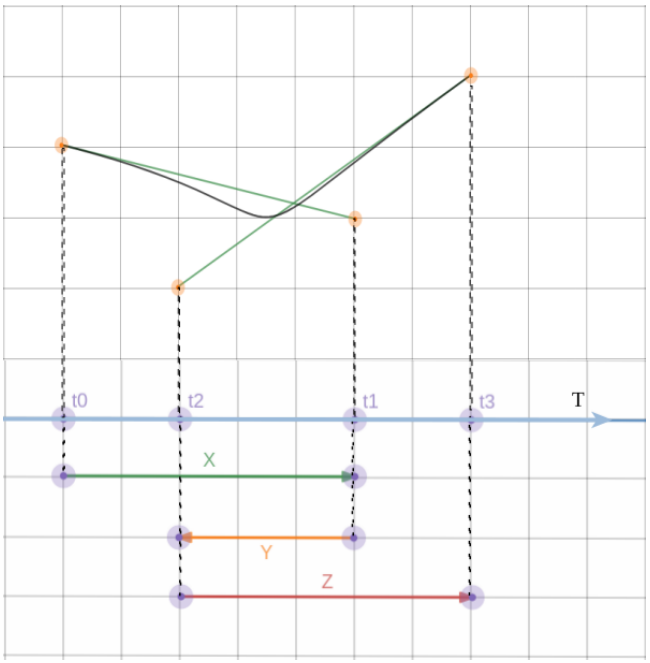


FIGURE III.10 – Visualisation d'une courbe de Bézier avec $t_0 < t_2 < t_1 < t_3$

Le discriminant Δ est donc strictement négatif et le coefficient quadratique P strictement positif, ce qui implique que :

$$t(\alpha) \geq 0, \quad \forall \alpha \in [0, 1] \tag{Eq. III.15}$$

Contraindre les deux points de contrôle centraux à être positionnés sur l'axe temporel entre le premier et le dernier point de contrôle suffit donc à garantir qu'une courbe de Bézier ne pourra représenter qu'une fonction associant à un temps t donné une unique valeur du paramètre animé θ.

Il existe des méthodes et des contraintes plus complexes permettant de limiter les courbes de Bézier à représenter des fonctions (Gangnet, 2000), et permettant notamment d'accepter les courbes représentées par la zone rouge dans la figure III.9. Cependant celle que nous proposons présente l'avantage d'être simple à implémenter et peu coûteuse en terme de calcul.

2. Modèle Géométrique Direct

Les robots auxquels nous nous intéressons ici consistent en un ensemble de corps rigides appelés liens $L = (l_1, ..., l_i)$ et connectés par des articulations de type pivot

$A = (a_1, a_2, ..., a_j)$. Un pivot est une liaison à un degré de liberté (DDL) ne permettant qu'une rotation autour de son axe et dont la configuration peut être décrite par sa position angulaire. La pose d'un robot composé de n moteurs peut ainsi être décrite dans l'espace articulaire de dimension n par les coordonnées articulaires :

$$\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)^T \in \mathbb{R}^n$$

Pour qu'un mouvement soit réalisable par un robot, l'un des impératifs est que celui-ci ne passe pas par une pose entraînant une collision entre des corps rigides composants le robot. Pour cela, il est nécessaire de connaître leur position et leur orientation. Dans l'espace tridimensionnel, on définit ainsi les coordonnées opérationnelles X_i d'un corps rigide i par :

$$X_i = (\alpha_i, \beta_i, \gamma_i, x_i, y_i, z_i,)^T \in \mathbb{R}^6 \qquad \text{(Eq. III.16)}$$

où $\alpha_i, \beta_i, \gamma_i$ définissent l'orientation du corps rigide et x_i, y_i, z_i sa position.

Le Modèle Géométrique Direct (MGD) d'un robot est une fonction permettant de faire le lien entre une position de l'espace articulaire et la position dans l'espace opérationnel d'un corps i du robot grâce à un modèle géométrique de ce dernier. On écrit donc :

$$X_i = MGD(\boldsymbol{\theta})$$

La position et l'orientation d'un corps dans l'espace euclidien peuvent également être représentées par une matrice de transformation homogène depuis la base canonique de l'espace euclidien, soit :

$$F_i = \left(\begin{array}{ccc|c} & R_i & & P_i \\ 0 & 0 & 0 & 1 \end{array}\right)$$

où P_i est la position $(x, y, z)^T$ du corps et R_i est une matrice de rotation de taille 3×3.

On peut donc définir un modèle géométrique équivalent au précédent qui retourne le cadre d'un corps i, soit :

$$F_i = MGD'(\boldsymbol{\theta})$$

Un robot peut être modélisé hiérarchiquement en utilisant une représentation arborescente (figure III.11). Cet arbre modélise les relations de dépendance géométrique

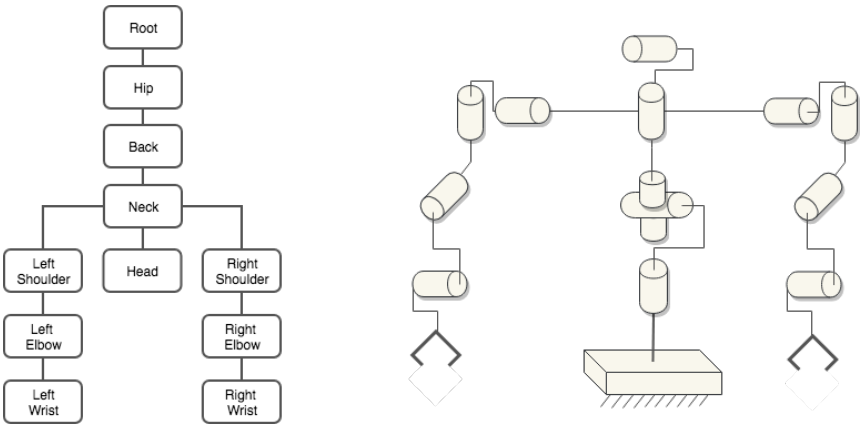


FIGURE III.11 – Modèle arborescent hiérarchique exemple qui montre les relations entre les parties du modèle du corps du robot Poppy (à gauche). Les articulations du robot Poppy et les degrés de liberté (à droite).

entre les différents corps rigides du robot. Chaque nœud de l'arbre stocke la pose (la position et l'orientation) d'une partie du corps dans un système de coordonnées locales par rapport à son nœud parent. Le nœud racine correspond à la base du robot. Son repère est parfois référé comme le repère du *monde*. Cette modélisation hiérarchique des parties du corps du robot permet d'appliquer une pose au robot en appliquant de petites transformations à chaque niveau de l'arbre en partant de la racine.

Cette modélisation s'effectue en définissant récursivement le repère F_i du composant i (lien ou articulation) par une transformation rigide T_i relative au repère F_j d'un composant parent j (figure III.12), soit :

$$F_i = F_j \cdot T_i \qquad \text{(Eq. III.17)}$$

et en définissant le repère F_R de la racine de cet arbre dans l'espace euclidien.

Un lien l_i modélise une partie inerte du robot et correspond à la combinaison d'une translation selon un vecteur $(x, y, z)^T$ constant suivie de 3 rotations successives autour des axes X, Y et Z, respectivement de ϕ, θ et γ radians, constants également. On calcule la matrice de transformation homogène telle que :

$$T_{l_i}(x, y, z, \phi, \theta, \gamma) = \text{Trans}(x, y, z) \cdot \text{Rot}_X(\phi) \cdot \text{Rot}_Y(\theta) \cdot \text{Rot}_Z(\gamma)$$

A l'inverse, une articulation modélise une transformation variable du robot. Une articulation est modélisée par une seule rotation de α radians autour de l'axe Z de

son repère (figure III.12), soit la transformation ayant pour matrice de transformation homogène suivante :

$$T_{a_i}(\alpha) = \text{Rot}_Z(\alpha)$$

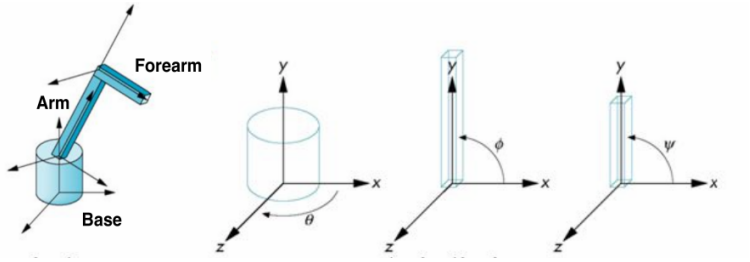


FIGURE III.12 – Exemple de bras robotique. Chaque partie du corps dans la hiérarchie a son propre repère, exprimé par rapport à son parent (Crédit).

Afin de faciliter la composition de transformations, il est pratique de toutes les exprimer sous la forme d'une multiplication matricielle. Il est donc commun de représenter un point de l'espace de coordonnées cartésiennes (x, y, z) par ses coordonnées homogènes $(x, y, z, 1)$. Ce choix de représentation permet notamment d'exprimer toutes les translations sous la forme de matrices 4×4. Afin de modéliser les liens et articulations mentionnées dans l'équation (Eq. III.17), 4 transformations rigides de base sont nécessaires : la translation et les rotations autour des axes x, y et z.

Translation selon le vecteur $(x, y, z)^T$:

$$\text{Trans}(x, y, z) = \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation de α radians autour de l'axe X :

$$\text{Rot}_X(\alpha) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation de α radians autour de l'axe Y :

$$\text{Rot}_Y(\alpha) = \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation de α radians autour de l'axe Z :

$$\text{Rot}_Z(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

La figure III.13 présente un exemple de Modèle Géométrique Direct pour un bras robotique planaire à 3 degrés de liberté. Ce bras peut être modélisé hiérarchiquement selon les principes décrits précédemment. Il est ainsi composé de 5 parties : les liens l_1, l_2 et l_3 et les articulations a_1 et a_2. Le point (x_e, y_e, z_e) correspondant à l'extrémité du bras est calculé récursivement selon l'équation (Eq. III.17).

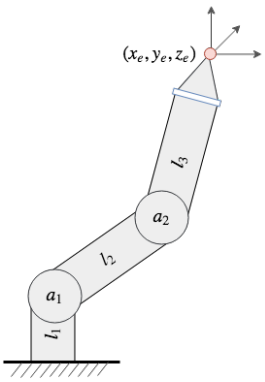


FIGURE III.13 – Exemple de Modèle Géométrique Direct

En appliquant cette formule de façon récursive et en pré-calculant les transformations constantes T_{l_1}, T_{l_2} et T_{l_2} nous pouvons écrire le Modèle Géométrique Direct de la façon suivante :

$$\begin{aligned} \text{MGD}(\alpha_1, \alpha_2) &= (x_e, y_e, z_e, 1)^T \\ &= (0,0,0,1)^T \cdot T_{l_1} \cdot \text{Rot}_Z(\alpha_1) \cdot T_{l_2} \cdot \text{Rot}_Z(\alpha_2) \cdot T_{l_3} \end{aligned}$$

Nous utilisons cette formulation pour calculer, pour une pose du robot, la matrice de transformation homogène correspondant au repère de ses différents liens. Ceux-ci sont notamment nécessaires afin de pouvoir détecter les auto-collisions entre deux parties du robot.

3. Détection des auto-collisions

Une pose du robot n'est pas forcément réalisable notamment parce qu'elle peut correspondre à une pose où deux parties du robot s'intersecteraient, ce que l'on appelle *auto-collision*. Une technique économe pour détecter les auto-collisions est de définir des capsules englobants les parties du robot et de vérifier que celles-ci ne s'intersectent pas (Ericson, 2004).

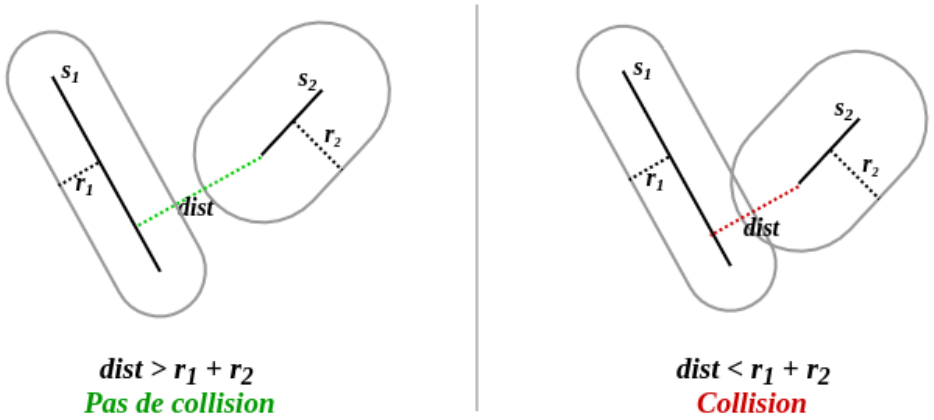


FIGURE III.14 – Principe du test de collision en 2D.

Une capsule, aussi appelée stadium de révolution (Figure III.14), est une forme géométrique pouvant être décrite comme un cylindre dont les extrémités sont des hémisphères. Une capsule est définie par un segment s et un rayon r et comprend tous les points p_i respectant l'inéquation :

$$dist_{\text{Point-Segment}}(p_i, s) <= r$$

L'intérêt des capsules pour la détection de collisions réside dans l'efficacité du calcul de l'existence d'une intersection entre deux capsules. Étant donné deux capsules C_1 et C_2 de segments centraux s_1 et s_2 et de rayons r_1 et r_2, tester l'existence d'une

intersection entre ces deux capsules revient à vérifier l'inéquation :

$$dist_{\text{Segment-Segment}}(s_1, s_2) > r_1 + r_2$$

Grâce à cette formulation, évaluer l'auto-collision revient à calculer la distance entre chaque paires de capsules, à l'exception des paires partageant une articulation. Les collisions entre parties du robot partageant une articulation peuvent quant à elles être évitées en définissant des limites de rotation de celle-ci. Nous utilisons cette méthode dans notre outil d'animation pour prévenir les auto-collisions.

C. Open Robot Animator (ORA)

Nous allons maintenant présenter notre proposition d'outil d'animation pour la robotique. La méthode d'animation dominante dans les logiciels d'animation 3D est l'animation par frames clés. D'après cette méthode, l'animateur définit les poses des personnages à des frames clés de l'animation ainsi qu'un profil de transition entre celles-ci. Ces profils de transition seront utilisés par le logiciel pour calculer automatiquement la pose du personnage aux frames intermédiaires (les *inbetweens*).

Les deux composants principaux des interfaces de logiciel d'animation sont donc les interfaces de définition de poses clés et de définition des profils de transition. L'interface utilisateur pour la définition de ces derniers fait le plus souvent usage de courbes paramétriques, tel que les courbes de Bézier composites présentées dans la section B par exemple. Notre outil ORA reprend ce principe pour la définition des profils de transition en basant l'interface autour de l'édition de courbes de Bézier composites. Nous nous concentrons dans ce chapitre sur la présentation des adaptations spécifiques à l'animation robotique, à savoir la manipulation directe et la représentation graphique des contraintes physiques du robot. Les figures III.16 et III.15 présentent respectivement l'interface graphique du logiciel ORA et un exemple d'animation générée.

Nous commencerons par motiver notre choix d'intégrer la manipulation directe à notre outil. Nous expliquerons ensuite comment la manipulation directe peut être utilisée pour définir des poses clés et créer des esquisses dynamiques des animations. Puis nous montrerons une façon de représenter les contraintes physiques du robot dans une interface graphique. Nous terminerons enfin par la présentation de quelques détails sur la mise en oeuvre de cet outil.

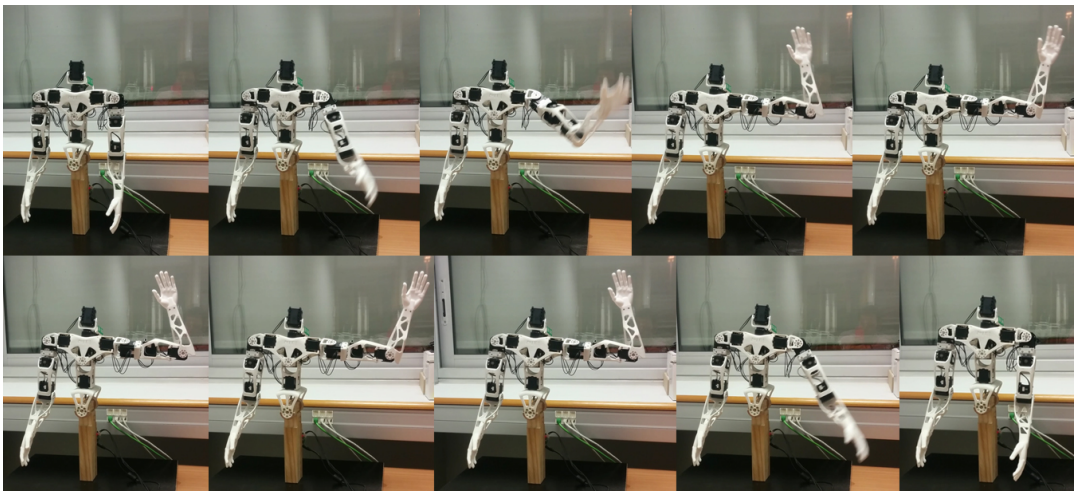


FIGURE III.15 – Animation d'un salut de la main de Poppy.

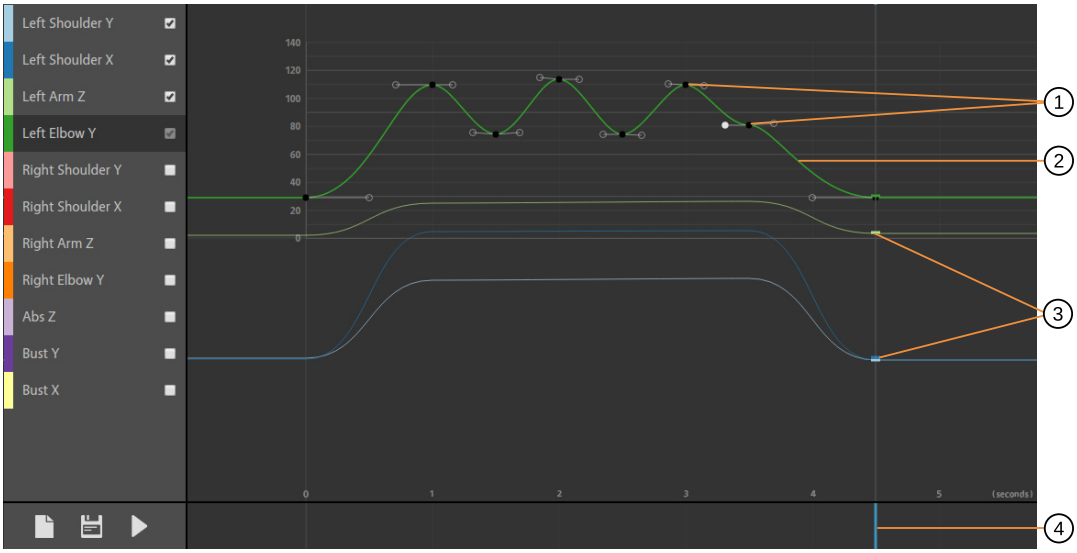


FIGURE III.16 – Etat de l'interface correspondant à cette animation. (1) Points de contrôle de la courbe de Bézier. (2) Courbe de Bézier reprenant la couleur du moteur concerné. (3) Curseurs de position de 2 moteurs. (4) Curseur temporel de l'animation.

1. Interface tangible et manipulation directe

Le paradigme d'Interaction Homme-Machine (IHM) dans lequel les logiciels d'animation 3D se placent est celui des Interfaces Graphiques. L'interaction se fait par manipulation d'objets virtuels dessinés sur un écran par l'intermédiaire d'une souris ou d'un clavier. Dans le cas de la définition de la pose d'un personnage 3D, cette virtualité présente l'avantage essentiel de la généralité. Différents personnages dotés de différentes morphologies peuvent être représentés à l'écran et manipulés au travers de cette interface graphique.

Définir la pose d'un personnage nécessite de placer chaque partie de son corps dans un espace 3D. Cela engendre deux problèmes pour les interfaces graphiques. Le premier problème est que l'écran comme la souris sont des dispositifs essentiellement 2D. La manipulation d'objets 3D se fait donc au travers d'une représentation 2D ce qui crée des ambiguïtés dans la manipulation. Le second est que la manipulation se fait au travers d'un seul pointeur et donc élément par élément ce qui devient vite laborieux pour placer chaque partie du corps d'un personnage. Des solutions à ces deux problèmes existent et sont implémentés dans les logiciels d'animation 3D.

Le problème d'ambiguïté de l'interaction 2D est résolu en contraignant les manipulations pour qu'elles concernent au maximum deux degrés de liberté à la fois. Ces manipulations peuvent ainsi être représentées à l'écran sans ambiguïté. On manipulera par exemple la position d'un objet le long d'un seul axe à la fois, ou encore sur un plan définit par deux axes et orthogonal au troisième. De même, la manipulation de l'orientation d'une partie du corps d'un personnage se fera par rotation autour d'un axe à la fois. Bien que fonctionnelle, cette solution décuple le second problème en multipliant le nombre d'interactions successives nécessaires pour obtenir le résultat souhaité.

Le second problème est résolu au travers de ce qu'on appelle la cinématique inverse. L'idée est de modéliser les contraintes physiques entre les différentes parties du personnage, comme la longueur des os, les degrés de libertés des articulations ou encore les angles minimums et maximums de ces derniers. Ces contraintes peuvent ainsi être simulées par le logiciel, ce qui permet de manipuler la position d'une partie du personnage et que les autres parties se déplacent en fonction.

Les Interfaces Tangibles forment un paradigme d'IHM alternatif à celui des Interfaces Graphiques. Dans ce paradigme, l'information est représentée et manipulée par des

objets présents dans l'environnement physique de l'utilisateur. Cela permet selon Ishii (2008), pionnier du domaine, de tirer avantage des capacités des humains à saisir et manipuler des objets physiques (*"taking advantage of human abilities to grasp and manipulate physical objects"*). *Kouretes Motion Editor* (Pierris and Lagoudakis, 2009) à implémenter ce paradigme pour la création de mouvement sur le NAO.

Les articulations des robots étant motorisées, leurs poses peuvent être contrôlées informatiquement, faisant des robots des périphériques de sortie, et donc une partie de l'IHM. Mais pour de nombreux robots, et notamment le robot Poppy, les moteurs peuvent également être manipulés et sont dotés de capteurs d'angles et de force, faisant également de ces robots des périphériques d'entrée. Ce type d'interface physique capable de représenter et de capter une information est à la base des Interfaces Tangibles, faisant de ce paradigme une solution naturelle à la définition de la pose d'un robot.

2. Définition de poses clés par manipulation directe

Nous souhaitons permettre à l'animateur de définir une nouvelle pose au robot en le manipulant. Une méthode possible serait de n'utiliser les moteurs que comme des capteurs d'angle, en désactivant le contrôle en position des moteurs, le transformant en un pantin articulé. Cependant, cette méthode est peu pratique car elle laisse à l'animateur la tâche de maintenir en position les différentes parties du robot. Il est difficile pour un utilisateur de manipuler précisément avec ses 2 mains plus de 2 objets physiques à la fois. Il est donc important de pouvoir définir la pose en manipulant successivement les différentes articulations. Nous voudrions pour cela que le robot tienne la pose sur toutes ses articulations à l'exception de celles actuellement manipulées par l'animateur tel que présentée dans la figure III.17.

Certains robots sont capables de tenir une pose tout en étant manipulables. Cette fonctionnalité est généralement appelée *compensation de gravité*. Elle nécessite de pouvoir contrôler en force les moteurs et de pouvoir calculer la force de gravité à laquelle on s'attendrait que chaque moteur soient soumis. Le calcul de ces forces nécessite de connaître la répartition de la masse du robot et sa pose actuelle (telle que capturée par ses capteurs d'angle). Les servomoteurs utilisés dans les robots tels que le Poppy sont généralement contrôlés en position et non en force, ne permettant pas d'implémenter une *compensation de gravité* en tant que telle.

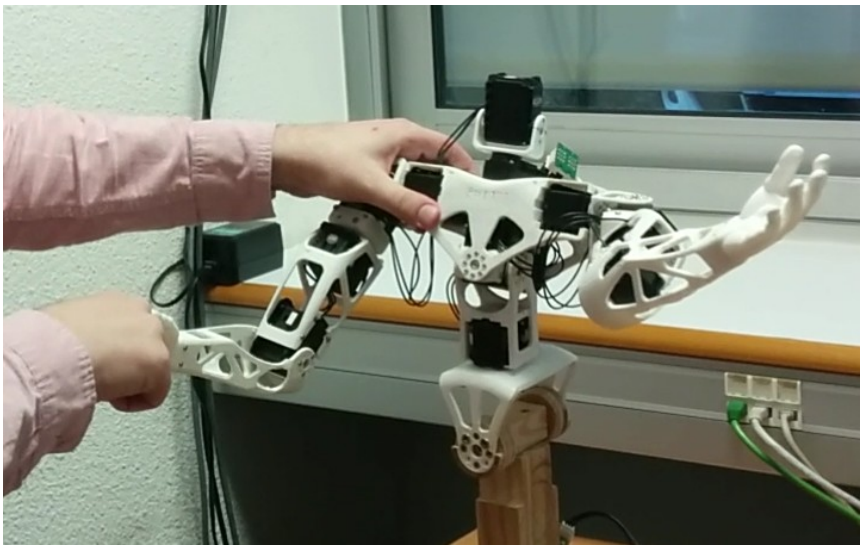


FIGURE III.17 – Définition d'une pose par manipulation directe.

Pour les robots fonctionnant sur le principe du contrôle en position, il faut pouvoir désactiver ce contrôle pour les moteurs des articulations que l'animateur souhaite manipuler. Une première méthode est de demander à l'animateur de sélectionner manuellement les moteurs manipulés. Cette sélection peut se faire au travers une interface graphique mais avec l'inconvénient de forcer l'animateur à passer plusieurs fois de l'interface graphique à l'interface tangible au cours de la définition d'une pose. Celle-ci peut également se faire grâce à des boutons physiques présents sur les articulations du robot lorsque ceux-ci existent comme c'est le cas sur le robot NAO. Nous utilisons une troisième méthode consistant à utiliser les capteurs de force des moteurs pour détecter s'ils sont manipulés par l'animateur.

Les moteurs ne sont généralement pas manipulés indépendamment mais par groupes formant une articulation. Les moteurs `r_shoulder_x`, `r_shoulder_y`, `r_arm_z` forment par exemple l'articulation de l'épaule droite (figure III.18). Nous avons donc conçu notre méthode de détection de manipulation d'une articulation autour de ces groupes de moteurs. Pour chacun de ces groupes, nous calculons la somme des forces subies au sein du groupe telles qu'estimées par ses moteurs. Pour les moteurs Dynamixel utilisés pour le robot Poppy, cette estimation est obtenue par mesure du courant. Le fait que cette estimation soit bruitée ne permettrait pas un contrôle en force mais est en pratique suffisant pour notre méthode de détection. Nous définissons ensuite empiriquement un seuil sur cette somme de forces au-dessus

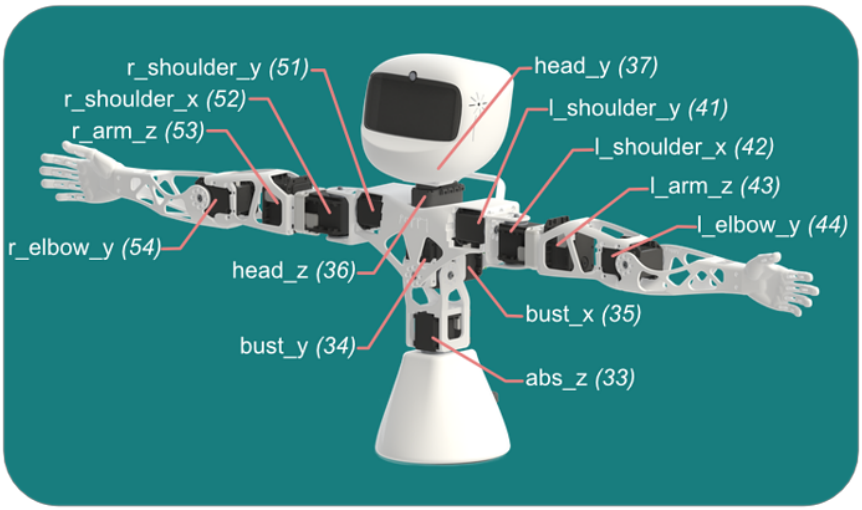


FIGURE III.18 – Moteurs du Poppy Torso. Chaque moteur est indiqué par son nom (ex : head_z) et son identifiant numérique (ex : 36) (Crédit)

duquel nous désactivons le contrôle en position de ces moteurs. Lorsque cette somme repasse en dessous du seuil pendant plus de 500 millisecondes, nous enregistrons la position courante de chacun des moteurs du groupe comme leur nouvelle position cible respective et réactivons leur contrôle en position.

3. Gestion des contraintes

Animer un robot présente des contraintes particulières comparé à l'animation d'un personnage physique. Il faudra par exemple éviter à tout prix les collisions entre deux parties du robot aussi appelées auto-collisions. Les moteurs du robot ont également des contraintes angulaires à prendre en compte, ainsi que des limites de vitesse et d'accélération. L'outil d'animation peut faciliter le travail de l'animateur en l'informant de la violation d'une contrainte physique ou en représentant cette contrainte dans l'interface graphique pour l'aider à la respecter. Nous recherchons les auto-collisions pour chaque frame en utilisant la méthode décrite page 83.

Une première dimension permettant de choisir la manière de gérer les violations des différentes contraintes est leur niveau de criticité respectif. Dans le cas de l'évitement des auto-collisions, une violation signifie des dommages potentiels pour le robot. C'est donc une contrainte extrêmement critique et la détection de sa violation devra empêcher l'exécution du mouvement. A l'inverse, la violation des contraintes en vitesse

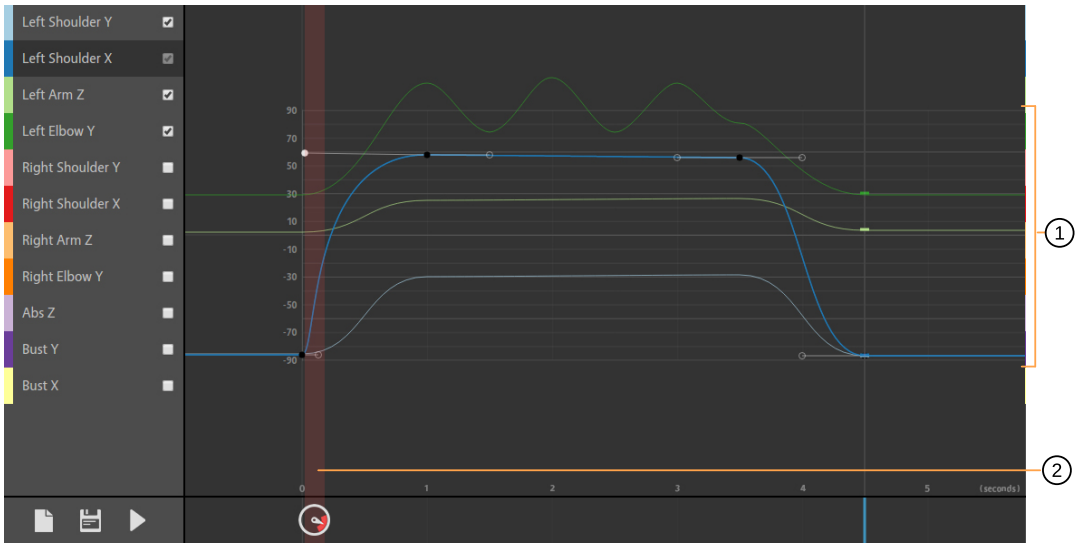


FIGURE III.19 – Communication des violations de contraintes physiques au travers de l'interface. (1) Limites angulaires du moteur sélectionné (Left Shoulder X). (2) Dépassement de la vitesse maximum pour le moteur sélectionné (Left Shoulder X).

ou en accélération ne risquent pas d'endommager le robot mais seulement d'altérer le mouvement exécuté comparé au mouvement représenté à l'écran. Il peut être utile à l'animateur de savoir que ces contraintes sont ne sont pas respectées et que le mouvement produit déviera du mouvement représenté, mais il n'est pas nécessaire pour autant d'empêcher l'exécution du mouvement. Nous appelons respectivement ces deux types de violation *erreurs* et *avertissements*, reprenant le vocabulaire couramment utilisés dans le domaine des environnements de développement.

Une seconde dimension est la possibilité ou non de représenter graphiquement l'intervalle d'un paramètre dans lequel une ou plusieurs contraintes sont respectées. Le cas le plus manifeste est dans notre système les contraintes de position de chaque moteur. L'intervalle des positions possibles, ou à l'inverse les angles minimum et maximum, peuvent être représentés par un guide visuel dans l'interface graphique. Cela permet à l'animateur de savoir en permanence si une modification entraînerait une violation de contrainte ou non. Cet avantage fait que cette solution de gestion de contrainte doit être privilégiée lorsqu'elle est possible.

4. Esquisse par manipulation directe

Dans la section 2, nous avons présenté une interface d'animation de robot basée sur la méthode *Pose à Pose*. Cette interface divise le travail de l'animateur entre une interface tangible pour la définition des poses clés et une interface graphique pour la définition de la dynamique du mouvement. La tangibilité de l'interface n'est donc que partielle. La définition de la dynamique d'une animation reste fondée sur l'espacement des poses clés et les profils de transition définis par des courbes de Bézier, deux étapes devant être effectuées pour chaque degré de liberté du robot.

Comme nous l'avons vu dans le chapitre I (page 26), il existe une deuxième méthode d'animation appelée *Animation Directe*. Alors que l'animation pose à pose aide à la création d'animations précises, l'animation directe est quant à elle conseillée lorsque l'on souhaite obtenir des animations plus spontanées et dynamiques. Dans cette section, nous détaillons une interface tangible inspirée de la méthode d'animation directe permettant de créer l'esquisse d'une animation. Cette interface utilise toujours le robot comme périphérique d'entrée mais cette fois-ci en enregistrant le mouvement du robot (ou d'une de ses parties) pendant que ce dernier est manipulé telle une marionnette par l'animateur.

Manipulation directe couche par couche Nous avons vu que la définition d'une pose clé grâce à une interface tangible nécessitait de décomposer cette tâche en manipulations successives d'une à deux parties du robot à la fois. La même contrainte liée au nombre d'objets manipulables simultanément s'est posée pour la tâche de définition de la dynamique d'une animation. L'enregistrement de mouvement par couches successives permet de répondre à cette limite. L'animateur commence dans un premier temps par manipuler un premier groupe de moteurs, les autres restant fixes. Dans un second temps, l'animateur manipule un second groupe de moteur pendant que le premier groupe rejoue le mouvement enregistré auparavant.

Édition par déformation de trajectoire Bien que cette méthode d'animation permette de rapidement définir une dynamique complexe, cette facilité se fait au détriment de la précision de l'animation et notamment la précision des poses intermédiaires. Ce compromis entre précision de l'animation et qualité de sa dynamique est analogue à celui mentionné par Thomas et al. (1995) dans leurs descriptions de l'*Animation Directe* et de l'*Animation Pose à Pose*.

Afin de pallier à ce manque de précision des poses intermédiaires, il est utile de pouvoir corriger la pose du robot à un temps donné de l'animation mais également de propager cette correction au reste de l'animation. Cette problématique est similaire à celle de l'adaptation des animations obtenues par capture de mouvement. Nous intégrons un mécanisme d'édition des poses intermédiaires par déformation de mouvement. Nous décrirons cette méthode plus en détail dans le chapitre IV dans le cadre de la création d'animations paramétriques.

5. Implémentation

Notre système est implémenté avec pour objectif de créer un logiciel suffisamment générique pour que la communauté puisse l'adapter à divers cas d'utilisation et de le développer sur cette base.

L'interface utilisateur de notre système est implémentée en utilisant les standards des plateformes web - plus spécifiquement JavaScript, HTML et SVG - en raison de leur caractère multi-plateforme. De plus, cela permet au logiciel d'être facilement adaptable en tant qu'application de bureau ou en tant qu'application web servie directement depuis l'ordinateur embarqué du robot.

Le backend du système est quant à lui développé en Python et utilise la bibliothèque pypot du projet Poppy pour contrôler les moteurs. Cette bibliothèque est principalement destinée à la commande des moteurs Robotis Dynamixel mais peut être étendue à d'autres actionneurs.

Nous avons choisi de ne pas baser notre système sur un middleware comme ROS afin de garder l'architecture la plus simple et modulaire possible. Le backend pourrait ainsi être remplacé par un noeud ROS si nécessaire.

D. Conclusion

Dans ce chapitre, nous avons présenté ORA, un logiciel d'animation pour la robotique intégrant le principe de la manipulation directe à une interface graphique reprenant les principes des logiciels d'animation 3D. La manipulation directe du robot y est utilisée pour faciliter la définition des poses clés de l'animation ainsi que pour la définition d'une esquisse dynamique de l'animation. Nous avons également proposé des principes pour le retour d'informations sur les contraintes physiques du robot par

le biais de l'interface graphique. Des tests utilisateurs de ce logiciel seront présentés dans le chapitre V.

Le logiciel ORA permet de créer des animations dites simples, au sens où ces animations sont rejouées à l'identique par le robot. Cependant, de nombreux cas de gestes que l'on souhaiterait animer pour un robot nécessite d'adapter l'animation à l'environnement du robot, et notamment à la position des objets concernés par ces gestes. Le prochain chapitre s'intéresse à ce type d'animation, dites paramétriques, et présente un outil basé sur le logiciel ORA ainsi qu'une méthode de collaboration entre animateurs et développeurs.

Chapitre IV

Outil et méthode d'animation robotique paramétrique

Dans le chapitre II, nous avons présenté un outil de prototypage de robots expressifs animés basé sur un logiciel d'animation existant. Dans le chapitre III, nous avons présenté un outil d'animation de robot conçu autour de l'utilisation du robot comme interface tangible.

Ces outils permettent de concevoir des animations pouvant être enregistrées et ré-exécutées par le robot par la suite. Ce type d'animation peut être utile dans certaines situations. Les danses exécutées par le robot NAO correspondent à ce type d'animation. Le robot joue une série de mouvements pré-enregistrés, et ce dans des conditions prévues lors de la conception du comportement, à savoir être sur un sol plat et ne pas être gêné dans ses mouvements par un obstacle. Ce type d'animation peut également être utilisé pour certains comportements expressifs indépendants de l'environnement du robot, par exemple pour le faire se gratter la tête, s'étirer ou encore bailler.

Dans certains cas, il est utile voire nécessaire de pouvoir adapter l'animation à l'environnement du robot. Pour saluer une personne de la main, il est ainsi préférable que le geste soit orienté vers celle-ci. De même, pointer vers un objet n'a de sens que si le geste est adapté à la position de l'objet cible. Le sens que prennent ces gestes est fonction du respect de contraintes géométriques par la pose du robot à certains moments clés du mouvement.

En faisant appel à un Modèle Géométrique Inverse, il est possible de trouver une pose répondant à des contraintes géométriques. Le principe général est de chercher la pose du robot minimisant une fonction de coût, laquelle encode les contraintes géométriques. Par exemple, si nous voulons que le robot tende le bras vers un objet, nous pourrions chercher la pose du robot qui placerait sa main le plus près possible de cet objet.

L'objet de ce chapitre est de présenter une méthode permettant d'intégrer des animateurs dans la conception d'animations paramétriques pour des robots. Nous commencerons par présenter dans la section A le concept de Modèle Géométrique Inverse et les principales méthodes du domaine. Nous présenterons par la suite dans la section 1 des méthodes d'édition d'animation ainsi que leurs limites. Nous détaillerons ensuite la méthode que nous avons conçue dans la section C. Nous finirons par décrire dans la section D, 2 exemples d'animations paramétriques réalisés en utilisant la méthode développée.

A. Modèle Géométrique Inverse

Dans le chapitre précédent, nous avons présenté le Modèle Géométrique Direct, permettant de trouver la position des différentes parties du robot à partir d'une configuration de ses moteurs. Cette méthode nous a permis d'expliquer le fonctionnement de notre détection d'auto-collisions. Pour pouvoir généraliser un mouvement à une nouvelle cible, il est nécessaire de résoudre le problème inverse, c'est-à-dire trouver la configuration des moteurs du robot qui permet de mettre une de ses parties à une position donnée de l'espace, soit un Modèle Géométrique Inverse.

Pour une configuration des moteurs $\boldsymbol{\theta}$ et une position des parties du robot $\mathbf{x}$, on peut poser les Modèles Géométriques Directs et Inverses tel que :

$$\mathbf{x} = MGD(\boldsymbol{\theta})$$

$$\boldsymbol{\theta} = MGI(\mathbf{x}) = MGD^{-1}(\mathbf{x})$$

Dans le cas de chaînes cinématiques comportant peu de degrés de liberté (DDLs), il est possible de trouver un Modèle Géométrique Inverse de façon analytique. Cependant, nous nous intéressons ici à des robots dont les chaînes cinématiques comportent potentiellement un nombre important de DDLs et nous nous concentrerons donc sur les méthodes dites itératives.

Les méthodes itératives sont basées sur une approximation locale du problème, autour de la configuration actuelle du robot. Plutôt que de chercher à trouver directement la configuration $\boldsymbol{\theta}$ correspondant à la position cible $\mathbf{x_{cible}}$, elles permettent de trouver une modification $\Delta\boldsymbol{\theta}$ qui minimise l'erreur $\vec{e}$ entre la position cible $\mathbf{x}_{\text{cible}}$ et la position actuelle $\mathbf{x_t}$.

$$\vec{e} = \mathbf{x}_{\text{cible}} - \mathbf{x_t}$$

Pour cela, on utilise la jacobienne $J(\boldsymbol{\theta})$ associée à la fonction $MGD(\boldsymbol{\theta})$, soit la matrice $m \times n$ des dérivées partielles de MGD :

$$J(\boldsymbol{\theta}) = \begin{pmatrix} \dfrac{\partial x_1}{\partial \theta_1} & \cdots & \dfrac{\partial x_1}{\partial \theta_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial x_m}{\partial \theta_1} & \cdots & \dfrac{\partial x_m}{\partial \theta_n} \end{pmatrix}$$

Nous utiliserons par la suite J pour indiquer la jacobienne en $\boldsymbol{\theta}$. Celle-ci nous permet d'exprimer les variations des positions des parties du robot en fonction des variations de la configuration de ses moteurs :

$$\Delta\mathbf{x} = J\Delta\boldsymbol{\theta}$$

Trouver la modification $\Delta\boldsymbol{\theta}$ rapprochant les positions des parties du robot de leur position cible revient à chercher $\Delta\boldsymbol{\theta}$ tel que $\Delta\mathbf{x} = \vec{e}$. On veut trouver la solution de l'équation suivante :

$$J\Delta\boldsymbol{\theta} - \vec{e} = 0$$

Dans les cas où la jacobienne n'est pas carrée, ce qui est généralement le cas pour les chaînes cinématiques qui nous concernent, le problème que nous avons posé précédemment est sur-déterminé et il n'est en général pas possible de le résoudre exactement. Le mieux que l'on puisse faire est de minimiser le résiduel $r = J\Delta\boldsymbol{\theta} - \vec{e}$. On utilise classiquement la minimisation de la norme euclidienne du résiduel $\|r\|_2$, soit :

$$\min_{\Delta\boldsymbol{\theta}} \| J\Delta\boldsymbol{\theta} - \vec{e} \|_2$$

$\|r\|^2$ étant convexe, son minimum est obtenu lorsque son gradient est nul, ce qui correspond à la solution des équations normales suivantes :

$$J^T\vec{e} = J^T J\Delta\boldsymbol{\theta}$$

qui a pour solution

$$\Delta\boldsymbol{\theta} = (J^T J)^{-1} J^T\vec{e} = J^+\vec{e}$$

La matrice $J^+ = (J^T J)^{-1} J^T$ correspond à la pseudo-inverse de Moore-Penrose J^+, qui peut être calculée plus efficacement en évitant l'inversion de la matrice de Gram $J^T J$.

Bien que l'on puisse résoudre le problème en utilisant la pseudo-inverse, elle souffre d'instabilité lorsque la jacobienne est singulière ou quasi-singulière. Whitney (1969, 1972) a montré que l'adjonction d'un terme de régularisation permettait de résoudre ces problèmes d'instabilité de la pseudo-inverse autour des singularités du robot. Sa méthode, appelée Damped Least-Squares ou méthode des moindres carrés avec amortissement, correspond au problème d'optimisation suivant :

$$\min_{\Delta\boldsymbol{\theta}} \parallel J\Delta\boldsymbol{\theta} - \vec{e} \parallel_2 + \lambda^2 \parallel \Delta\boldsymbol{\theta} \parallel_2$$

qui correspond à la résolution des équations normales :

$$J^T \vec{e} = (J^T + \lambda^2 I) J\Delta\boldsymbol{\theta}$$

ayant pour solution :

$$\Delta\boldsymbol{\theta} = (J^T J + \lambda^2 I)^{-1} J^T \vec{e}$$

Wampler (1986) puis Maciejewski and Klein (1989) ont montré le fonctionnement de l'adjonction du terme d'amortissement grâce à la décomposition en valeurs singulières (ou SVD) de la jacobienne, définie par :

$$J = U\Sigma V^T$$

où :

— J est la matrice jacobienne $m \times n$

— U est une matrice unitaire $m \times m$

— Σ est une matrice diagonale $m \times n$ contenant les valeurs singulières $\sigma_1, \ldots, \sigma_m$ de J.

— V est une matrice unitaire $n \times n$

En réécrivant la pseudo-inverse avec la SVD de la jacobienne, on obtient :

$$(J^T J)^{-1} J^T = V\Sigma^{-1} U^T$$

où Σ^{-1} est une matrice diagonale dont les coefficients sont de forme

$$\frac{1}{\sigma_i}$$

En faisant de même pour la méthode des moindres carrés avec amortissement, on obtient :

$$(J^T J + \lambda^2 I)^{-1} J^T = V \left[\left(\Sigma^T \Sigma + \lambda^2 I \right)^{-1} \Sigma \right] U^T = V S U^T$$

où $S = \left(\Sigma^T \Sigma + \lambda^2 I \right)^{-1} \Sigma$ est une matrice diagonale dont les coefficients sont de forme

$$\frac{\sigma_i}{\sigma_i^2 + \lambda^2}$$

La comparaison de ces deux décompositions en valeurs singulières permet de mieux comprendre la différence de comportement des deux méthodes à proximité des singularités de la jacobienne.

Lorsque la matrice jacobienne J est singulière ou quasi-singulière, une ou plusieurs de ces valeurs singulières σ_i va approcher de 0. Dans le cas de la pseudo inverse, les coefficients diagonaux correspondants dans la matrice Σ^{-1} vont prendre de valeurs importantes. A l'inverse, dans le cas de la méthode des moindres carrés avec amortissement, les coefficients diagonaux correspondants dans la matrice S vont tendre vers 0 également.

On peut également voir que lorsque σ_i est grand par rapport à λ, les deux méthodes sont similaires. En effet, on peut voir que dans ce cas :

$$\frac{\sigma_i}{\sigma_i^2 + \lambda^2} \approx \frac{1}{\sigma_i}$$

Il est important de noter que la méthode des moindres carrés avec amortissement gagne en stabilité mais prend plus de temps à converger. De nombreuses techniques ont été proposées pour moduler λ afin d'accélérer la convergence. On notera que la méthode des moindres carrés avec amortissement est un cas particulier de la régularisation de Tikhonov pour $\Gamma = \lambda I$, la forme générale correspondant au problème suivant :

$$\min_{\Delta \boldsymbol{\theta}} \parallel J \Delta \boldsymbol{\theta} - \vec{e} \parallel_2 + \parallel \Gamma \, \Delta \boldsymbol{\theta} \parallel_2$$

ayant pour solution :

$$\Delta \boldsymbol{\theta} = (J^T J + \Gamma^T \Gamma)^{-1} J^T \vec{e}$$

Na et al. (2008) proposent de modifier la méthode DLS pour y ajouter le respect des contraintes angulaires, une position de confort et la rigidité variable d'une articulation. L'idée générale de leur méthode, appelée Improved Damped Least Squares, est de moduler la rigidité d'une articulation en pondérant dynamiquement le coût de la

dimension de $\Delta\boldsymbol{\theta}$ correspondante. Pour cela, ils remplacent λI par une matrice Λ qui est fonction de la position actuelle $\boldsymbol{\theta}$.

$$\min_{\Delta\boldsymbol{\theta}} \parallel J\Delta\boldsymbol{\theta} - \vec{e} \parallel_2 + \parallel \Lambda(\boldsymbol{\theta}) \cdot \Delta\boldsymbol{\theta} \parallel_2$$

ayant pour solution :

$$\Delta\boldsymbol{\theta} = (J^T J + \Lambda(\boldsymbol{\theta})^2)^{-1} J^T \vec{e}$$

où $\Lambda(\boldsymbol{\theta})$ est définie par :

$$\Lambda(\boldsymbol{\theta}) = \begin{pmatrix} \lambda_0(\theta_0) & & \\ & \ddots & \\ & & \lambda_n(\theta_n) \end{pmatrix}$$

Un terme de régularisation λ_i est donc défini pour chaque articulation i en fonction de la position actuelle θ_i de celle-ci. Na et al. (2008) proposent d'intégrer les différentes contraintes physiques de la chaîne cinématique à cette fonction.

Une alternative aux méthodes basées sur l'inversion de la jacobienne du Modèle Géométrique Direct est de formuler le problème de géométrie inverse comme un problème de minimisation d'une fonction de coût pouvant être résolu grâce à des méthodes d'optimisation non linéaires sous contraintes telles que SLSQP (Kraft, 1988) (pour Sequential Least Square Quadratic Programming). Cet algorithme fonctionne de façon itérative en minimisant des approximations quadratiques de la fonction de coût.

L'algorithme SLSQP permet de résoudre des problèmes d'optimisation sous contraintes de la forme :

$$\begin{array}{lllll} \text{minimiser} & f(x) & & & \\ \text{soumis à} & g_i(x) & = & 0 & \forall i \in \\ & h_j(x) & \leq & 0 & \forall j \in I \end{array}$$

où $g_i(x)$ et $h_j(x)$ sont respectivement les contraintes d'égalités et d'inégalités à respecter et $f(x)$ la fonction de coût à minimiser.

Les contraintes de limites angulaires peuvent ainsi être naturellement exprimées comme des contraintes d'inégalités sur le vecteur de contrôle. Le problème de géométrie inverse présenté précédemment correspond à :

$$\begin{array}{lll} \text{minimiser} & \|MGD(\boldsymbol{\theta}) - \mathbf{x}_{\text{cible}}\| & \\ \text{soumis à} & \theta_i - \sup_{\theta_i} \leq 0 & \text{pour } i = 1, \ldots, m \\ & \inf_{\theta_i} - \theta_i \leq 0 & \text{pour } i = 1, \ldots, m \end{array}$$

où $\inf_{\theta_i}$ et $\sup_{\theta_i}$ sont respectivement les limites angulaires inférieure et supérieure de l'articulation θ_i.

B. Algorithme de déformation d'animation

Nous avons vu que pour certaines animations, les contraintes géométriques étaient aussi essentielles sur le plan sémantique que le style de celles-ci. L'objectif de cette section est de détailler un algorithme de déformation d'animation permettant de déformer une animation afin de respecter ces contraintes géométriques tout en conservant le style de l'animation originale.

Nous proposons de décomposer cette déformation en deux étapes. Une première étape consiste à déformer la pose originale à chacun des moments clés de l'animation afin de respecter les contraintes géométriques. Une fois les poses définies pour les moments clés, la seconde étape consiste à déformer l'animation afin qu'elle passe par ces nouvelles poses.

1. Méthodes connexes

Le domaine de l'animation 3D présente une problématique similaire d'adaptation d'une animation à de nouvelles conditions. Les mouvements créés par des techniques de Motion Capture doivent être édités après coup afin de correspondre aux positions des autres objets de la scène. Plusieurs techniques d'édition d'animation ont ainsi été proposées afin de faciliter et d'accélérer ce type de tâches.

Nous allons tout d'abord présenter les méthodes de *Motion Displacement Mapping* et de *Motion Warping* qui partage la même idée générale : l'animateur modifie la pose d'un personnage 3D à des moments clés d'une animation. Une déformation lisse est ensuite calculée de sorte à transformer l'animation afin qu'elle passe par ces nouvelles poses. Ces deux méthodes partagent également la spécificité de traiter indépendamment chaque degré de liberté de l'animation. On notera ainsi θ_t la position d'un degré de liberté au temps t.

a. Motion Displacement Mapping (Bruderlin and Williams, 1995)

Cette première technique est une adaptation de la méthode du *displacement mapping* à l'animation. La figure IV.1 montre les différentes étapes de cette technique. Dans cette technique, la déformation lisse est obtenue en ajoutant à l'animation les décalages (offsets) souhaités par l'animateur aux moments clés (noté b_t) et en utilisant une

spline pour interpoler le décalage à appliquer pour le reste de l'animation :

$$\theta'_t = \theta_t + b_t$$

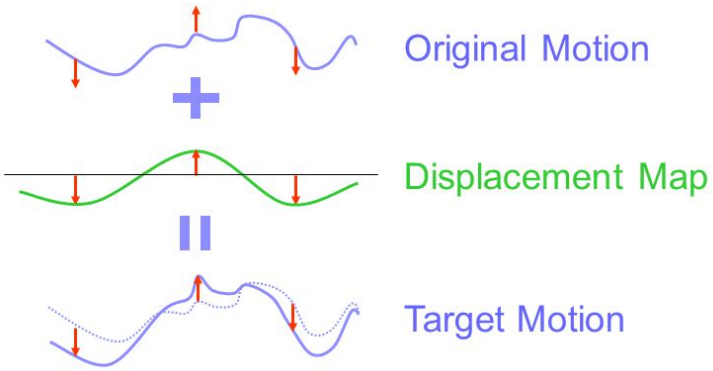


FIGURE IV.1 – Displacement mapping (Crédit).

b. Motion Warping (Witkin and Popovic, 1995)

Cette seconde technique ajoute à la précédente la possibilité de déformer l'animation originale de façon multiplicative. L'animateur peut en effet choisir de modifier la position du degré de liberté θ_t à un moment clé soit y en ajoutant un décalage b_t (offset), soit en la multipliant par un facteur d'échelle a_t (scaling factor) :

$$\theta'_t = a_t\theta_t + b_t$$

Comme dans la méthode précédente, les valeurs de a_t et b_t aux moments clés de l'animation sont interpolées à l'aide d'une spline afin de définir les fonctions a et b pour le reste de l'animation.

La figure IV.2 montre un exemple d'utilisation de cette méthode afin de déplacer la position de la raquette dans une animation d'un joueur de tennis.

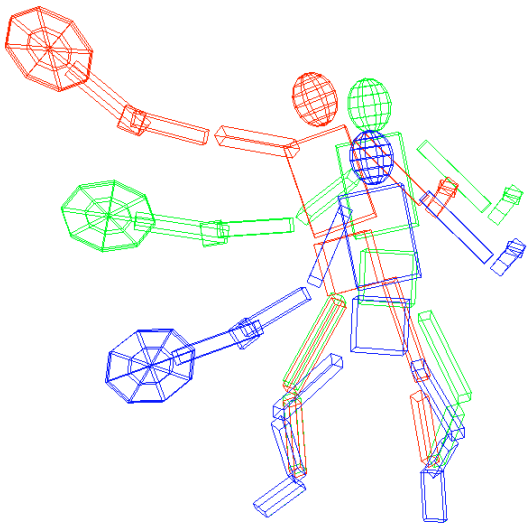


FIGURE IV.2 – Motion Warping (Witkin and Popovic, 1995).

c. Stylized Motion Generalization

Bien que le travail d'adaptation soit grandement simplifié et accéléré par les deux méthodes précédentes, ces techniques sont conçues pour une utilisation "offline" par l'animateur, pendant la phase de conception d'un film d'animation ou d'un jeu vidéo par exemple. La meilleure déformation est laissée à l'appréciation de l'animateur et fait partie de son choix artistique. Celui-ci peut également détecter que l'adaptation ne fonctionne pas et faire manuellement les modifications nécessaires. Le pire cas étant de devoir les retoucher manuellement une par une.

A l'inverse, la méthode de Gielniak et al. (2010), conçue pour la robotique, a pour but de déformer un mouvement de façon automatique, pour une utilisation "online", au moment de son utilisation.

Les auteurs souhaitent déformer un mouvement original $\theta(t)$ entre une pose initiale θ_0 et une pose finale θ_n pour générer un nouveau mouvement $\theta'(t)$ entre les poses θ'_0 et θ'_n.

Pour ce faire, ils proposent de fixer la nouvelle pose initiale θ'_0 et d'intégrer les vitesses $\dot{\theta}(t)$ du mouvement original modulées par un facteur constant a tel que :

$$\theta'(t+1) = \theta'(t) + a\dot{\theta}(t)$$

Cette méthode revient à générer un nouveau mouvement par déformation affine du mouvement original soit :

$$\theta'(t) = a(\theta(t) - \theta_0) + b$$

avec :

$$b = \theta'_0$$

Cette formulation rapproche cette méthode d'un cas particulier de la méthode de Motion Warping présentée précédemment dans laquelle les fonctions d'amplification $a(t)$ et de décalage $b(t)$ seraient constantes.

d. Limites

Les trois méthodes de déformation d'animations présentées précédemment sont une composition de deux types d'opérations de déformation : la déformation par addition d'un décalage, ou *offsetting*, et la déformation multiplicative, ou *scaling*. Nous allons ici détailler les cas limites causées par ces opérations.

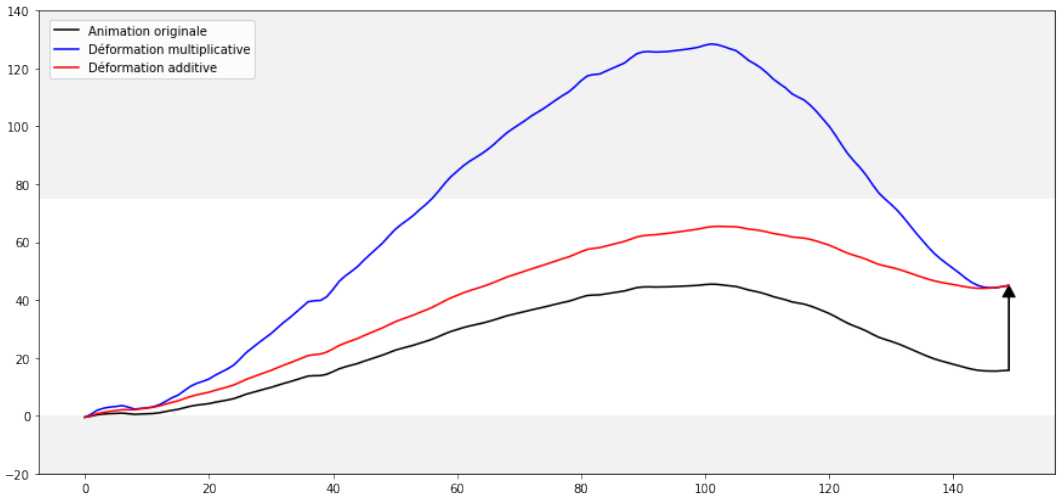


FIGURE IV.3 – Sortie des limites angulaires

Sortie des limites angulaires Ce premier cas limite se pose quand la déformation nécessaire pour répondre aux contraintes de poses engendre une ou des poses intermédiaires hors des limites angulaires pour une articulation donnée. Ce cas limite est particulièrement important en robotique étant donné que les limites angulaires ne sont pas seulement des contraintes virtuelles choisies pour un but de réalisme mais des

contraintes matérielles. Le premier cas limite touche potentiellement les déformations additives comme multiplicatives. Toutefois, il est plus rapidement atteint dans le cas de la déformation multiplicative comme l'illustre la figure IV.3.

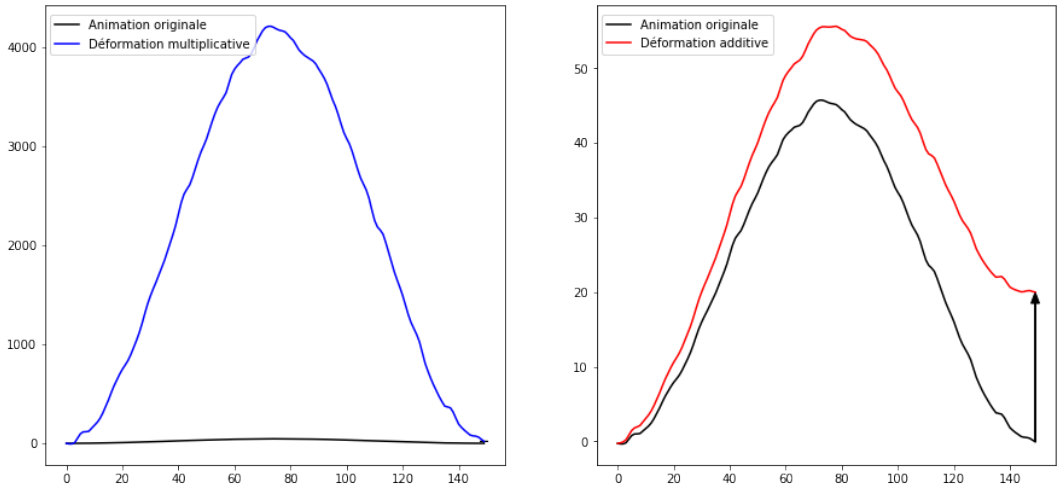


FIGURE IV.4 – Positions de début et de fin identiques

Positions de début et de fin identiques Ce cas limite se présente quand la position d'une articulation au départ et à la fin du mouvement sont identiques. Cette situation se produit lorsque le mouvement original est cyclique, c'est-à-dire que les poses de début et de fin du squelette sont identiques. Elle peut également se produire lorsque l'articulation ne bouge pas dans le mouvement original. Dans ce cas, la déformation multiplicative ne permet pas de modifier la différence entre les positions de début et de fin de l'articulation et donc de répondre à la contrainte de position finale. Cette limite est due au fait que cette déformation est symétrique entre les vitesses positives et négatives. La figure IV.4 présente un exemple de ce cas limite.

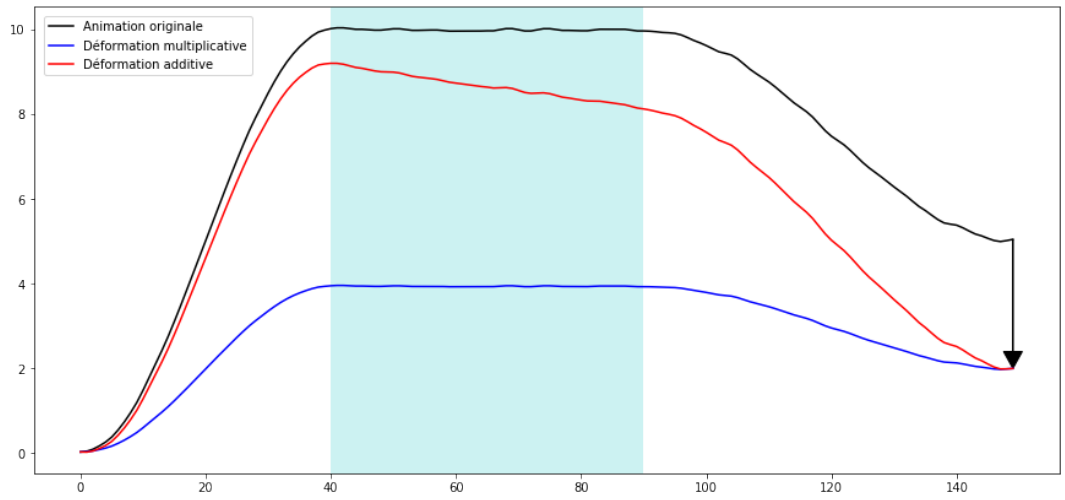


FIGURE IV.5 – Pause au milieu d'un mouvement

Pause au milieu d'un mouvement Ce dernier cas critique se pose lorsqu'une pause est contenue dans le mouvement sur une ou plusieurs articulations du robot. Dans ce cas, la déformation additive générera du mouvement là où il n'y en avait pas en déformant les parties immobiles d'une animation (figure IV.5). En pratique, nous avons noté que le moindre mouvement ajouté lors d'une partie immobile de l'animation originale modifiait le style de l'animation et la rendait peu naturelle. Notre interprétation est que la déformation est cachée par le mouvement existant dans l'animation originale. Lorsqu'il n'y a aucun mouvement dans une partie de l'animation originale, la moindre déformation est directement visible.

Conclusion Le cas de la pause au milieu de l'animation implique que l'animation doit être répartie aux moments de déplacement de l'animation originale. Le cas des positions identiques de début et de fin nous montre quant à lui que déformer symétriquement l'animation rend la méthode incompatible avec certaines animations. Ce cas limite nous indique également que la déformation doit se faire en priorité sur les degrés de liberté utilisés dans l'animation originale.

2. Méthode proposée

Notre méthode peut être décomposée en deux sous-parties, la première étant l'algorithme de déformation du profil de position d'un degré de liberté et la deuxième la

recherche d'une pose respectant les contraintes géométriques tout en étant similaire à la pose finale d'origine.

a. Déformation de courbes

Nous avons vu précédemment que la déformation multiplicative présente l'avantage de conserver les parties immobiles de l'animation, tandis que la déformation par addition progressive d'un décalage présente à l'avantage d'être plus conservateur pour la sortie des limites angulaires et de ne pas être sensible au cas des positions de début et de fin identiques.

Nous proposons de modifier la déformation additive en modulant la répartition du décalage pour le concentrer dans les périodes de mouvement de l'animation originale. Pour cela, nous proposons de moduler cette répartition en fonction de la répartition du déplacement absolu dans l'animation originale.

L'algorithme de déformation peut-être décomposé en 4 étapes :

a. Calculer pour chaque frame le déplacement absolu cumulé depuis la première frame.

$$D(t) = \int_0^t |v_t| \, dt$$

b. Normaliser pour chaque frame ce déplacement absolu cumulé par sa valeur à la dernière frame. Cela donne la répartition du déplacement absolu au cours du temps dans l'animation originale.

$$R(t) = \frac{D(t)}{D(f)}$$

c. Calculer le déplacement total à ajouter pour respecter la contrainte de position finale.

$$\Delta_\theta = (\theta'_f - \theta'_0) - (\theta_f - \theta_0)$$

d. Calculer la nouvelle position pour chaque frame en intégrant le déplacement final proportionnellement à sa part du déplacement absolu dans l'animation originale.

$$\theta'(t) = \theta(t) + \left(\theta'_0 - \theta_0\right) + \Delta_\theta \cdot R(t)$$

En plus de limiter l'occurrence des cas de sorties des limites angulaires (l'offset ne peut dépasser à aucun moment de l'animation celui que l'on veut ajouter) et de résoudre

le cas limite des positions identiques de début et de fin, cette formulation permet de répartir le déplacement final sur un degré de liberté en utilisant la répartition du déplacement absolu sur les degrés de libertés associés. Prenons l'exemple du robot Poppy et des degrés de liberté de son épaule gauche. Si l'un de ces moteurs est immobile dans l'animation originale, il est possible d'utiliser la répartition du déplacement absolu des deux autres moteurs comme fonction de répartition du déplacement à ajouter.

La sélection de la répartition à utiliser nécessite de définir un arbre des dépendances entre moteurs. Cet arbre est différent du modèle géométrique du robot, bien que les deux soient liés. L'objectif de cette arbre est de modéliser le fait que des moteurs mettent en mouvement les mêmes parties du robot. Cela permet d'utiliser un mouvement pré-existant d'une partie du robot pour cacher l'intégration du déplacement final souhaité.

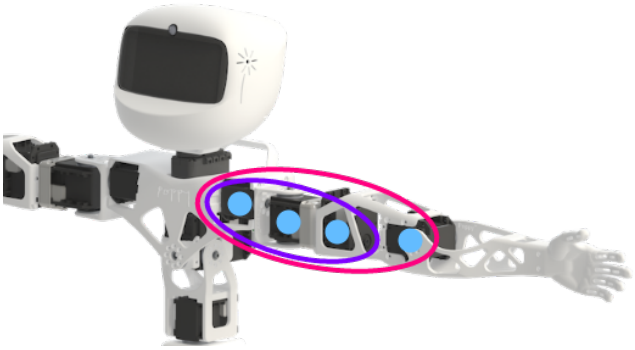


FIGURE IV.6 – Relations entre les moteurs du bras du robot Poppy

Reprenons l'exemple du bras du robot Poppy. Celui-ci est doté de trois moteurs au niveau de l'épaule et d'un moteur au niveau du coude. Les trois moteurs de l'épaule mettent en mouvement exactement la même partie du robot. Ils peuvent donc constituer un premier niveau de parenté. Le moteur du coude mettant en mouvement une partie du bras, un second niveau de parenté peut être constitué pour regrouper tous les moteurs du bras. La figure IV.6 illustre cette modélisation.

b. Déformation de pose

Nous souhaitons trouver la déformation minimale d'une pose permettant de respecter une contrainte géométrique tout en conservant le style original. Nous posons cette recherche comme une optimisation d'une fonction de coût qui peut être décomposée en deux : un coût stylistique, dont le rôle est de favoriser les poses préservant le style original, et un coût géométrique quant à lui responsable du respect des contraintes géométriques.

Coût géométrique Le coût géométrique est responsable du respect des contraintes géométriques et sa fonction de coût est donc dépendante de la sémantique du moment clé de l'animation. La fonction de coût géométrique la plus courante est la distance euclidienne entre la position d'une cible et d'une partie du robot, par exemple sa main, soit :

$$C_{\text{geom}}(P'_i) = \|\text{cible}_{\text{pos}} - \text{main}_{\text{pos}}\|$$

D'autres distances peuvent être utilisées en fonction de la tâche. Nous avons par exemple utilisé une fonction de coût modélisant l'orientation du "regard" du robot vers la cible.

$$C_{\text{geom}}(P'_i) = \angle\left(\overrightarrow{\text{dir}}, \overrightarrow{\text{tête}_z}\right)$$

avec :

$$\vec{\text{dir}} = \text{cible}_{\text{pos}} - \text{tête}_{\text{pos}}$$

Coût stylistique Le coût stylistique est quant à lui responsable de la ressemblance de la nouvelle pose à celle de l'animation originale. Nous avons vu dans la déformation de courbe que lorsqu'un degré de liberté est immobile dans l'animation originale, nous étions obligé d'utiliser le profil de position d'autres degrés de liberté afin de "cacher le mouvement" de ce degré de liberté.

Pour autant, cette solution est imparfaite et il est donc préférable de limiter son utilisation. Nous souhaitons donc prendre la quantité de déplacement de chaque degré de liberté dans l'animation originale dans le cadre de la déformation de pose.

Nous définissons pour cela le coût stylistique pour une pose P'_i en fonction de la distance à la pose originale P_i modulée par l'utilisation de chaque degré de liberté dans l'animation originale jusqu'au moment i de cette pose, soit :

$$C_{\text{style}}\left(P'_i\right) = \left(P'_i - P_i\right)^T \cdot W_i \cdot \left(P'_i - P_i\right)$$

où W_i est donc une matrice diagonale de taille $n \times n$ avec n le nombre de degrés de liberté du robot.

Cette matrice W_i a pour objectif de pénaliser davantage les modifications de la position des degrés de liberté les moins mobiles jusqu'au moment i de l'animation originale. Pour cela, on pose W_i tel que :

$$W_i = \begin{pmatrix} w_i^0 & & \\ & \ddots & \\ & & w_i^n \end{pmatrix}$$

avec w_i^n tel que :

$$w_i^n = \frac{|\theta_{\max} - \theta_{\min}|}{\int_0^i \left|\dot{\theta}_t^n\right| dt + \epsilon}$$

où ϵ permet de se prémunir d'une division par 0.

C. Méthodologie et interfaces

1. Principe de fonctionnement

La méthode de conception d'une animation paramétrique développée repose sur l'organisation de la collaboration entre animateur et développeur. L'objectif principal est de permettre à l'animateur de s'occuper de l'expressivité de l'animation et au développeur de s'occuper de la généralisation de celle-ci.

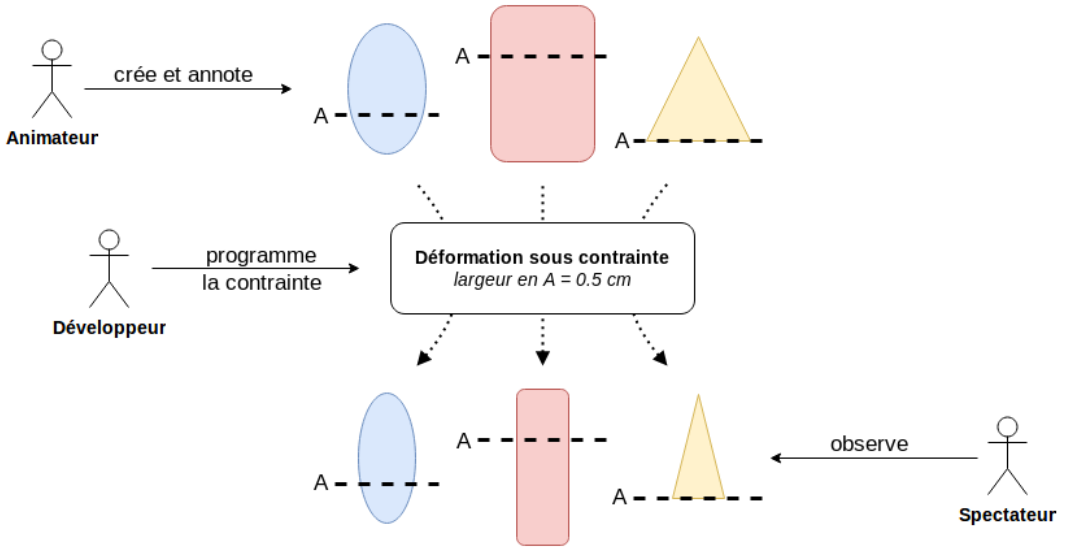


FIGURE IV.7 – Principe de notre méthode de conception d'animation paramétrique

2. Rôle de l'animateur

Le rôle de l'animateur consiste à concevoir une animation originale et de l'enrichir d'informations sur les contraintes géométriques qu'il juge importantes. Pour cela, l'animateur marque les moments clés auxquels la pose du robot doit respecter une ou plusieurs contraintes géométriques et décrit ces dernières sous forme textuelle. Par exemple, pour une animation dans laquelle le robot tape un objet avec la paume de la main droite, l'animateur marquerait le moment correspondant au contact avec l'objet avec comme commentaire "la paume droite du robot touche l'objet".

Cette partie de la conception se fait au travers d'un outil d'animation tel que nous en avons présenté dans les chapitres précédents auxquels nous avons ajouté des fonctionnalités d'annotations. L'animateur peut créer un marqueur au moment sélectionné. Il peut ensuite lui attribuer un nom ainsi qu'une description à destination du développeur.

3. Rôle du développeur

Le rôle du développeur est quant à lui de généraliser l'animation originale, premièrement en traduisant les descriptions textuelles des contraintes géométriques fournies par l'animateur en critères d'optimisation du Modèle Géométrique Inverse, et deuxiè-

mement en combinant les parties d'une ou plusieurs animations originales pour composer un comportement complet. Il utilise pour cela l'interface de programmation permettant de manipuler les animations. Nous décrirons premièrement les concepts de cette interface de programmation puis nos choix d'implémentation en Python.

a. Interface de programmation

Cette interface de programmation est conçue autour de deux classes d'objets, les classes Pose et Animation. On pose tout d'abord les types suivants :

Identifiant de frame (`IdFrame`)
Un identifiant de frame est soit un **entier** correspondant au numéro de la frame dans l'animation, soit une **chaîne de caractères** correspondant au nom d'un marqueur de l'animation défini préalablement par l'animateur.

Fonction de coût
Une fonction de coût est une **Fonction** que le développeur doit fournir. Celle-ci prend en entrée une pose du robot et une cible (dont le type est défini par le développeur), et retourne un **Flottant** coût.

Classe Pose

Attribut

`angles`
Dictionnaire associant un moteur, identifié par une **chaîne de caractères** unique, à un angle en radians.

`reperes`
Dictionnaire associant à chaque parties du robot, identifié par une **chaîne de caractères** unique, un repère sous la forme d'une matrice homogène de taille 4×4.

Méthode

`deformer(fcout: FonctionCout, cible: Cible) -> Pose`
Retourne une nouvelle pose calculée en déformant la pose courante pour minimiser la fonction `fcout` paramétrée par la cible `cible`.

Classe Animation

Attributs

`frames`
Liste contenant les poses du robot à chaque pas de temps de l'animation. Par défaut, les animations sont échantillonnées à 50 frames par secondes.

`marqueurs`
Dictionnaire contenant les marqueurs définis par l'animateur. Chaque marqueur associe un nom, c'est-à-dire une **chaîne de caractères** unique, à un **entier** correspondant au numéro d'une frame de l'animation.

Méthodes

`retournePose(idframe: IdFrame) -> Pose`
Retourne la pose prise par l'animation à la frame indiquée par l'identifiant de frame *idframe*.

`couper(premiere: IdFrame, derniere: IdFrame) -> Animation`
Retourne une nouvelle animation reprenant la portion de l'animation courante comprise entre les frames *premiere* (par défaut la première de l'animation courante) et *derniere* (par défaut la dernière de l'animation courante).

`concatener(anim2: Animation) -> Animation`
Retourne une nouvelle animation construite en concaténant l'animation `anim2` à la suite de l'animation courante.

`deformer(debut: Pose, fin: Pose) -> Animation`
Retourne une nouvelle animation calculée en déformant l'animation courante pour qu'elle commence par la pose `debut` et finisse par la pose `fin`. Différentes méthodes de déformation d'animation, dont celle utilisée *in fine*, sont décrites dans la section B.

b. Implémentation

Syntaxe Python

Afin de pouvoir utiliser ses opérations de façon succincte, notre implémentation utilise les possibilités syntaxiques offertes par le langage Python. Nous nous sommes

inspirés de la syntaxe Python pour la manipulation des chaînes de caractères afin d'implémenter la classe `Animation`. `retournePose` est ainsi implémenté comme une indexation, `couper` comme une opération de *slicing* et la concaténation est comme une addition de deux objets de classe `Animation`.

Voici quelques exemples de l'usage de cette syntaxe ainsi que leur correspondance dans l'interface de programmation présentée précédemment :

Interface : `animation.retournePose(0)`
Syntaxe Python : `animation[0]`
Description : Retourne la première pose de l'animation.

Interface : `animation.retournePose(-1)`
Syntaxe Python : `animation[-1]`
Description : Retourne la dernière pose de l'animation.

Interface : `animation.retournePose('start')`
Syntaxe Python : `animation['start']`
Description : Retourne la pose au moment clé correspondant au marqueur 'start'.

Interface : `animation.couper(0, 50)`
Syntaxe Python : `animation[0:50]`
Description : Retourne une nouvelle animation créée à partir des 50 premières poses de l'animation.

Interface : `animation.couper('start', 'touch')`
Syntaxe Python : `animation['start':'touch']`
Description : Retourne une nouvelle animation créée à partir des poses contenues entre les marqueurs 'start' et 'touch'.

Interface : `animation1.concatener(animation2)`
Syntaxe Python : `animation1 + animation2`
Description : Retourne une nouvelle animation créée en concaténant les deux animations.

Contraintes géométriques

Déformer une pose du robot afin de respecter des contraintes géométriques nécessite de traduire celles-ci en un critère d'optimisation pour un problème de géométrie

inverse. L'objectif de notre implémentation est de permettre au développeur de créer une fonction de coût qui compare la position et/ou de l'orientation d'une partie du robot à une cible.

Comme nous l'avons présenté précédemment, la fonction de coût prend comme entrée une cible de type définie par le développeur et une pose du robot. Cette pose du robot est définie comme les positions et orientations des différentes parties du robot. Nous représentons la position et l'orientation de chaque partie du robot sous la forme d'une matrice de transformation homogène par rapport à la base canonique de l'espace euclidien, comme décrit dans le chapitre précédent (page 78).

Nous utilisons alors la méthode d'optimisation SLSQP (pour *Sequential Least SQuares Programming*) afin de résoudre le problème de géométrie inverse ainsi défini. Les dérivées partielles de la fonction de coût géométrique en fonction de la configuration du robot peuvent être obtenues en utilisant une méthode d'auto-différenciation ou être approximées par la méthode des différences finies.

D. Exemples d'animations paramétriques

Nous avons procédé à une première évaluation de cette méthodologie et des outils développés en concevant des animations paramétriques. Nous allons en décrire deux ici. La première consiste à permettre au robot Poppy de jouer du Glockenspiel en généralisant à partir d'une démonstration. Ce premier exemple se concentre sur la déformation de courbes, les différentes poses intermédiaires étant données comme entrée. La deuxième consiste à programmer la déformation d'une famille d'animations où Poppy touche une cible avec son maillet et intègre la déformation de pose et la déformation de courbes.

1. Jouer du Glockenspiel

L'objectif de ce premier test est de concevoir une animation paramétrique faisant la démonstration de l'algorithme de déformation de courbe ainsi que la composition d'une animation complexe à partir d'animations élémentaires permise par l'interface de programmation.

Pour cela, nous utilisons la tâche de jouer au Glockenspiel une mélodie paramétrable. Cette mélodie est donnée comme une succession de poses correspondant aux notes

sur le Glockenspiel. Utiliser les poses de chaque note comme contrainte permet de se concentrer pour cette expérience sur la déformation et la composition des animations.

a. Animations originales

L'animation complète que nous souhaitons créer peut être décomposée en deux démonstrations. La première démonstration, visualisée sur la figure IV.8, consiste à faire jouer à Poppy deux notes sur le Glockenspiel. Cette démonstration pourra être ensuite annotée par 4 marqueurs : 2 marqueurs de début et de fin du mouvement d'intérêt ("Debut" et "Fin") et 2 marqueurs pour les 2 moments où le maillet frappe le Glockenspiel ("Frappe1" et "Frappe2"). La deuxième démonstration, visualisée sur la figure IV.9, permet quant à elle à Poppy de saluer son public après avoir joué la mélodie.

FIGURE IV.8 – Démonstration en jouant deux notes sur le glockenspiel (8 images/seconde)

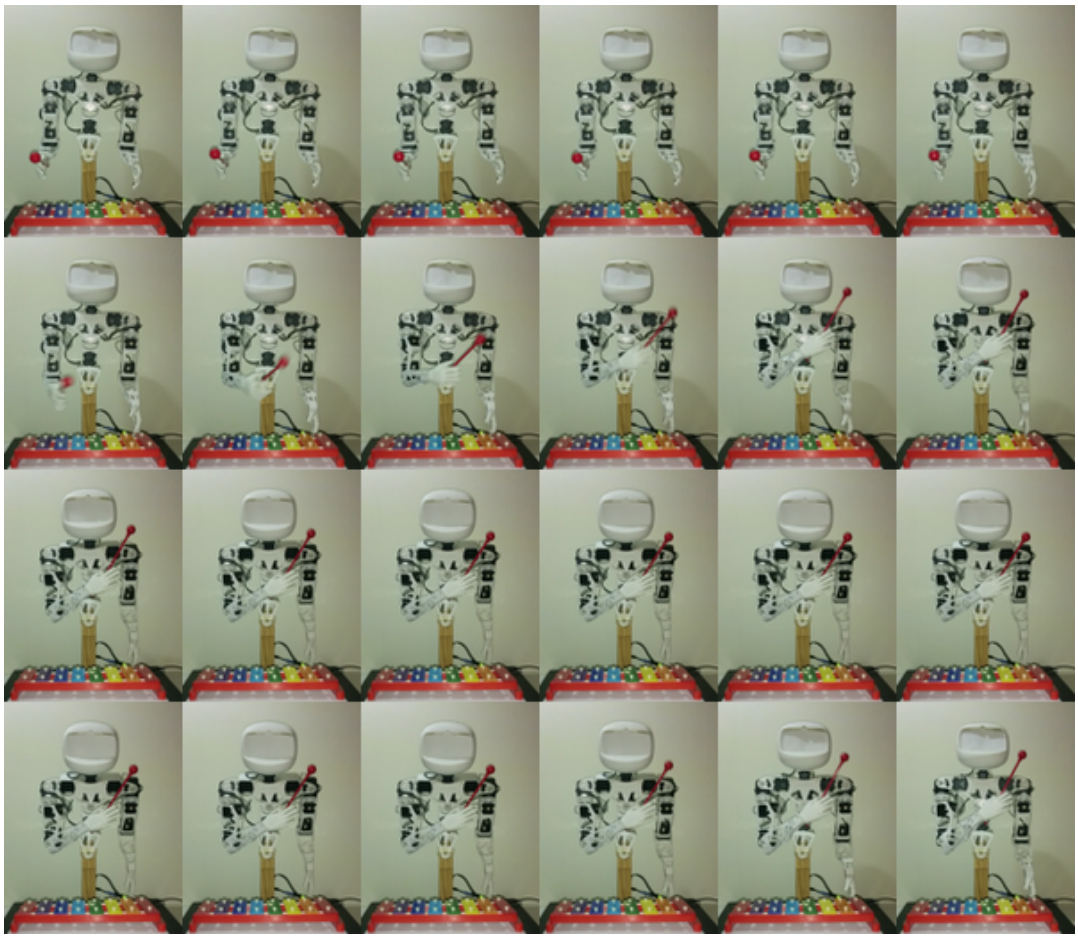


FIGURE IV.9 – Animation originale du salut final (8 images/seconde)

b. Code applicatif

Nous détaillons ici le code applicatif permettant la création d'une animation du Poppy en fonction d'une mélodie.

La première étape consiste à importer les démonstrations et à les découper si nécessaire en animations élémentaires. Dans cet exemple, la démonstration de frapper deux touches peut être décomposée en trois sous-parties : le début de l'animation jusqu'à frapper la première note (début), le passage d'une note à l'autre (cycle) et le retour à la pose de repos (fin).

```
demo = Animation.charger_json(chemin='demo.json')
salut = Animation.charger_json(chemin='salut.json')

debut = demo['Debut':'Frappe1']
cycle = demo['Frappe1':'Frappe2']
fin = demo['Frappe2':'Fin']
```

Comme nous l'avons dit précédemment, la mélodie est donnée comme une séquence de poses correspond à chaque note. Nous enregistrons donc les poses du robot dans lesquels celui frappe les différentes touches. Nous enregistrons dans cet exemple la note C, la même opération étant effectuée pour D, E, F, G, A, B et C2, ce qui correspond à la notation anglaise des notes de musique. Nous enregistrons également une pose de silence "hold" afin de pouvoir prolonger la note précédente.

```
C = retourne_pose_actuelle()
```

Ces 9 poses enregistrées, nous pouvons concevoir le coeur de cet exemple, à savoir la fonction de génération de l'animation complète.

```
def jouer_melodie(poses, debut, cycle, fin, salut=False):
    anim = debut.deformer(fin=poses[0])

    for precedent, suivant in zip(poses[:-1], poses[1:]):
        anim += cycle.deformer(debut=precedent, fin=suivant)

    anim += fin.deformer(debut=poses[-1])

    if salut:
        anim += salut.deformer(debut=anim[-1])

    return anim
```

Cette fonction peut ensuite être appelée pour générer une animation correspondant à jouer la suite de notes passées en paramètres en utilisant les 4 animations élémentaires *debut*, *cycle*, *fin* et *salut*, elles-aussi paramétrables.

```
jouer_melodie(
    [ C, D, E, F, hold, G, A, B, C2 ],
    debut, cycle, fin, salut
    )
```

Des mélodies plus complètes peuvent ensuite être jouées de la même façon, en passant en paramètre la mélodie sous la forme d'une séquence de poses. La mélodie *Ah ! vous dirai-je, maman* est par exemple encodée par la séquence de poses suivante :

```
ah_vous_dirai_je_maman = [
    C, C, G, G, A, A, G, hold,
    F, F, E, E, D, D, C, hold,
    G, G, F, F, E, E, D, hold,
    G, G, F, F, E, E, D, hold,
    C, C, G, G, A, A, G, hold,
    F, F, E, E, D, D, C
    ]
```

L'exécution de cette séquence donne la séquence animée représentée sur la figure IV.10. La vidéo correspondante est disponible sur Youtube [1].

1. https://youtu.be/ixWB3UTgRhY - dernière visite : 03/11/2019

FIGURE IV.10 – Animation du Poppy jouant "Ah vous dirai-je maman" au glockenspiel (1 image/seconde). La vidéo est disponible ici.

2. Toucher une cible

Cette application a été développée pour les tests utilisateurs du système d'animation paramétrique. A l'inverse de l'exemple précédent, la façon de toucher la cible n'est pas connue à l'avance. Il n'est donc pas possible d'enregistrer un ensemble de poses permettant au robot de toucher les différentes cibles. Le code applicatif doit donc déformer la pose correspondant au marqueur "Touche" avant de pouvoir déformer le reste de l'animation. Ce code applicatif fait appel pour cela aux fonctionnalités de cinématique inverse détaillées précédemment.

a. Animation originale

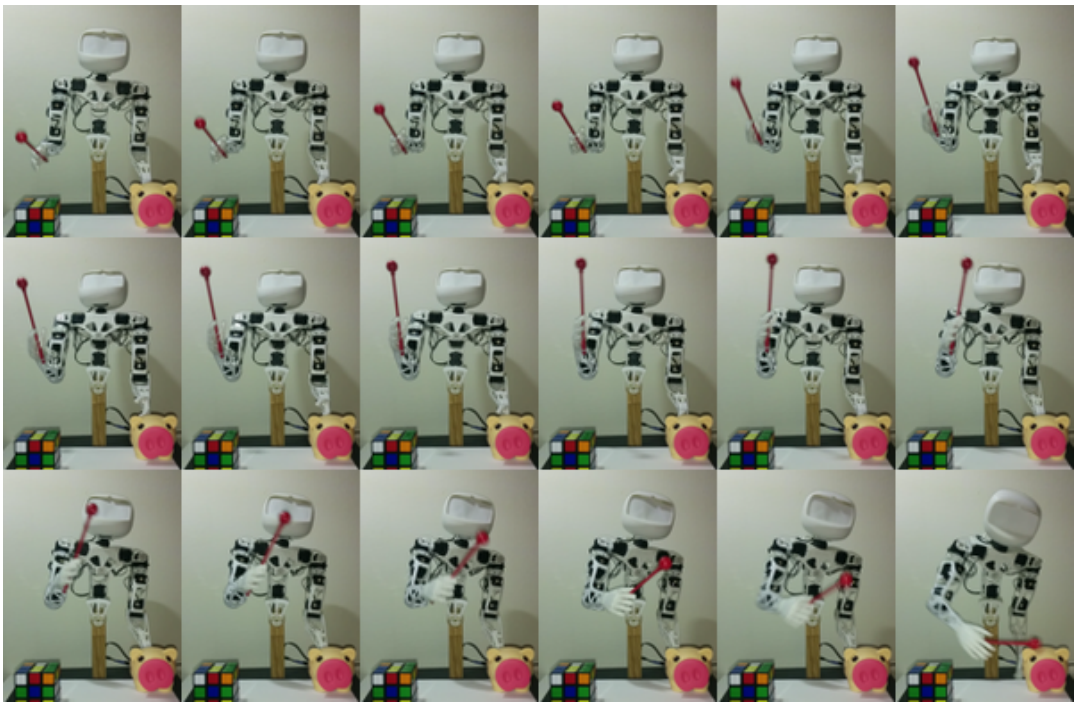


FIGURE IV.11 – Animation originale (8 images/seconde)

b. Code applicatif

Tout comme dans la première expérience, la première étape consiste à importer la démonstration et à découper la partie que l'on souhaite utiliser.

```
demo = Animation.charger_json(chemin='demo.json')
demo_touche = demo['Debut':'Touche']
```

La seconde étape est de définir la fonction de coût géométrique permettant d'implémenter la contrainte correspondant au marqueur "Touche", à savoir la distance euclidienne entre la position du maillet droit du robot et la position de la cible.

```
def position_maillet(cible_pos, pose_robot):
    maillet_pos = pose_robot.reperes['maillet_droit'][0:3, 3]
    return np.linalg.norm(maillet_pos - cible_pos)
```

Cette fonction peut ensuite être utilisée pour déformer la pose finale en vue de déformer l'animation originale. La figure IV.12 présente un exemple d'extrapolation de l'animation afin de toucher une nouvelle cible.

```
def toucher_cible(cible_pos, demo_touche):
    pose_finale = demo_touche['Touche']
    nouvelle_pose = pose_finale.deformer(position_maillet, nouvelle_cible)
    return demo_touche.deformer(fin=nouvelle_pose)
```

FIGURE IV.12 – Animation extrapolée pour toucher une nouvelle cible (8 images/seconde)

E. Conclusion

Dans ce chapitre, nous avons proposé une méthodologie de création d'animations paramétriques fondée sur la collaboration entre un animateur et un développeur. L'animateur décrit les moments clés de l'animation devant respecter des contraintes géométriques. Le développeur crée ensuite une fonction génératrice de l'animation paramétrique grâce à l'interface de programmation dédiée.

Nous avons également présenté un algorithme de déformation d'animation composé de deux étapes : une première étape de déformation de la pose du robot au moment clé et une deuxième étape de déformation de la courbe afin qu'elle passe par la pose déformée. Le principe de cette seconde étape est de répartir la déformation suivant la répartition du déplacement dans l'animation originale.

Enfin, nous avons présenté deux exemples d'animations paramétriques développées afin de tester cette méthodologie. Toutefois, l'objectif final de cette méthodologie est qu'elle soit utilisée par des animateurs, ce que nous avons évalué au moyen de tests utilisateurs présentés dans le chapitre suivant.

Chapitre V

Evaluation

Dans le chapitre IV, nous avons présenté une méthode pour la conception d'animations robotiques paramétriques. Nous présentons dans ce chapitre des tests utilisateurs que nous avons réalisés avec des étudiants en animation de l'Ecole Ariès de Meylan. Cette évaluation a été mise en place avec l'aide de Nadine Mandran, ingénieur CNRS en production et analyse des données du Pôle d'Ingénierie Multidisciplinaire du LIG (PIMLIG).

L'objectif de ces tests utilisateurs étaient d'évaluer le système développé selon deux axes. Le premier axe correspond à l'évaluation de l'utilisabilité du système pour des utilisateurs ayant une formation artistique et non informatique. Le deuxième axe correspond à l'évaluation de la qualité de la généralisation des animations telles que générées par notre système.

A. Plan expérimental

1. Participants

Les tests utilisateurs ont pu être organisés grâce au concours des Ecoles Ariès [1], un réseau de cinq écoles spécialisées dans la création digitale. 4 étudiants de l'école de Grenoble ont participé à cette expérimentation, 2 femmes et 2 hommes âgés de 21 à 24 ans. Ils étaient tous en 2ème année de Bachelor Conception 3D, VFX et Animation. 2 des participants avaient des bases en programmation informatique.

1. http://www.ecolearies.fr/ - dernière visite : 01/11/2019

2. Installation

L'expérimentation a eu lieu au sein de l'Ecole Ariès. La figure V.1 montre le système tel qu'installé lors de l'expérimentation et le positionnement de ses différents éléments, à savoir un ordinateur doté du prototype, le robot Poppy et 4 cibles numérotées. Les participants étaient debout lors de l'expérimentation, afin de faciliter le passage de l'utilisation de l'interface graphique à la manipulation du robot. Le robot Poppy était muni d'un maillet attaché à sa main gauche. La position de chaque cible avait été calibrée au préalable afin que le système connaisse la position dans l'espace des différentes cibles.

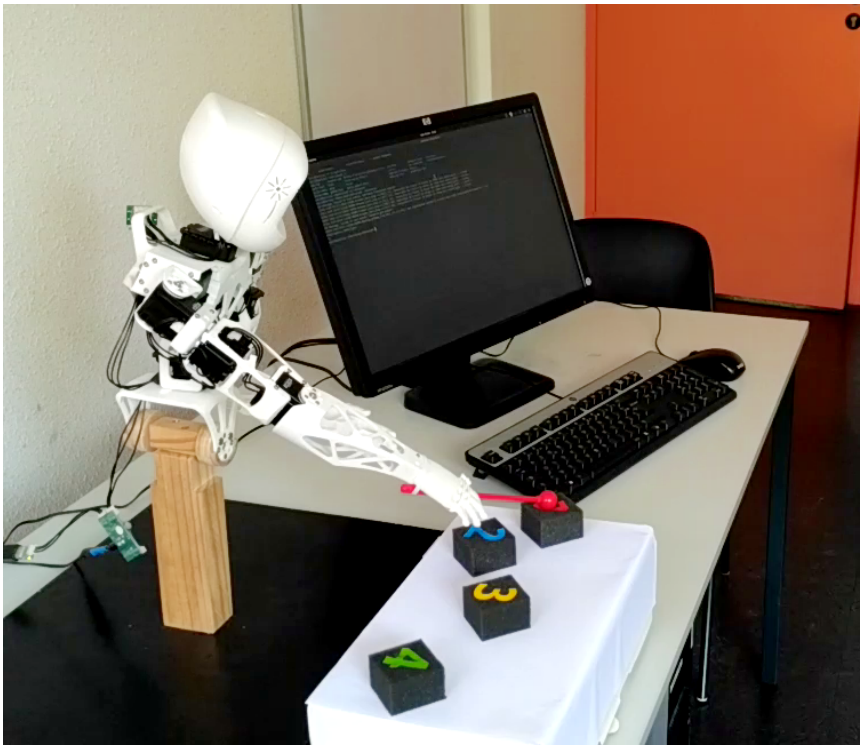


FIGURE V.1 – Installation expérimentale.

3. Protocole

Chaque session de test utilisateur suit le même protocole experimental. La session débute par une présentation du contexte de l'expérimentation, puis un moment est laissé au participant pour lire et signer le formulaire de consentement.

Une démonstration du système et de ses fonctionnalités est faite au sujet suivant un scénario d'usage similaire à celui de la tâche expérimentale en explicitant les différentes actions effectuées.

Pour des raisons de temps d'expérimentation, la partie animation de ces tests utilisateurs s'est concentrée sur la fonctionnalité d'esquisse de l'animation par manipulation directe.

Les participants effectuent ensuite une tâche de test pendant laquelle le participant est encouragé à poser des questions afin de faciliter la prise en main du logiciel. La tâche de test est composée des trois sous-tâches suivantes :

1. Enregistrer une animation dans laquelle le robot touche la cible 1 avec le maillet, en utilisant le torse et le bras droit du robot.
2. Éditer une frame de l'animation pour préciser le mouvement.
3. Annoter les frames correspondant au début souhaité de l'animation exportée ("Start") et au moment où le robot touche la cible ("Touch").

A la suite de cette tâche de test, un débriefing a lieu afin de recueillir les premières impressions du participant et de s'assurer que le fonctionnement du système est bien compris.

Une fois cette phase d'introduction au logiciel terminée, les participants passent à la tâche réelle consistant à concevoir 3 animations différentes. Le robot doit toucher la cible 1 avec le maillet dans ces 3 animations, mais en utilisant 3 styles différents. Comme dans la tâche de test, le participant doit également annoter les frames correspondant au début souhaité de l'animation exportée ("Start") et au moment où le robot touche la cible ("Touch").

Un second débriefing est alors effectué avec les participants grâce à une série de questions ouvertes pour recueillir leur perception du système, ainsi qu'au questionnaire SUS, pour System Usability Scale (Brooke et al., 1996), devenu un standard pour évaluer l'utilisabilité d'un système. Il a l'avantage de pouvoir être utilisé tant pour l'évaluation d'un système matériel que pour celle d'un logiciel.

La seconde partie de l'expérimentation est dédiée à l'évaluation de l'extrapolation. Pour chaque animation originale proposée par le participant, nous commençons par la rejouer (en visant donc la cible 1). Nous faisons ensuite exécuter au robot les animations générées par extrapolation pour les cibles 2, 3 et 4. Après chaque animation

exécutée, le participant évalue la qualité de l'extrapolation. Chaque participant évalue donc 9 animations extrapolées.

La qualité de l'extrapolation est évaluée au moyen d'un questionnaire formé de 5 énoncés pour chacun desquels le participant donne leur niveau d'accord sur une échelle de 1 à 5, sur le modèle de celle utilisée dans le questionnaire SUS.

Les trois premiers énoncés concernent la qualité de l'extrapolation en elle-même, et plus exactement de sa conservation des propriétés de l'animation originale :

1. L'animation extrapolée a conservé le style de l'animation originale.
2. L'animation extrapolée a conservé l'esthétique de l'animation originale. [2]
3. L'animation extrapolée a conservé les contraintes géométriques de l'animation originale.

Les deux derniers énoncés concernent quant à eux la qualité intrinsèque de l'animation extrapolée, sans prendre en compte la conservation des propriétés de l'animation originale :

4. L'animation extrapolée est esthétiquement de bonne qualité.
5. L'animation extrapolée est fonctionnellement de bonne qualité.

B. Résultats

1. Qualitatifs

Comme nous l'attendions, les participants ont trouvé inhabituel d'enregistrer une animation par manipulation du robot, leur formation les ayant habitués à une méthode d'animation par poses clés. Malgré ce caractère inhabituel, ils ont trouvé très intéressant de pouvoir animer le robot en interagissant directement avec un objet "qu'on peut toucher" (P1 [3] et P2) et "d'initier soi même le mouvement" (P3). L'un d'eux a décrit l'expérience comme "très ludique" (P2) et le fonctionnement de la capture "assez intuitif" (P2).

La plupart des difficultés que les participants ont rencontrées sont liées à des considérations matérielles. Certains ont trouvé que le robot manquait de certaines articulations

2. Le terme "esthétique" est parfois utilisé pour décrire le style d'une animation. Permet de vérifier que les deux concepts sont bien corrélés pour les animateurs.
3. Participant 1

pour pouvoir exprimer ce qu'ils souhaitaient lui faire exprimer. Ils évoquent en particulier le manque d'articulations des poignets. Bien que la tête soit articulée, elle leur a semblé "un petit peu rigide" (P2) comparée aux autres moteurs, au point d'avoir "l'impression qu'on va casser le robot" (P3). Une autre difficulté matérielle exprimée par les participants est l'imprécision du robot due aux jeux mécaniques entre les différentes pièces qui le composent.

Certains participants ont exprimé leur difficulté à prévoir les mouvements du robot lors du ré-enregistrement du mouvement par superposition au mouvement existant. L'un d'eux avait du mal à voir quand le robot allait "se mettre à bouger" (P3), tandis qu'un autre évoquait l'importance de se rappeler du mouvement que le robot va faire pour ne pas "lui arracher le bras" (P1).

Cette difficulté à se représenter les mouvements du robot se retrouve également dans l'utilisation des fonctionnalités nécessitant de sélectionner une frame, comme l'édition de l'animation par déformation de la pose du robot à une frame donnée, ou l'annotation sémantique de l'animation. Ils nous ont par exemple dit qu'il était compliqué de "tomber sur le bon moment à éditer" (P1) ou de savoir le "moment précis [où le robot] fait les gestes" (P4).

L'imprécision dans la sélection d'une frame et la difficulté à prévoir le mouvement que le robot va faire sont tous les deux à relier au besoin d'une fonctionnalité de visualisation du mouvement frame par frame évoqué par les participants lorsque nous leur avons demandé quelles fonctionnalités leurs avaient manqué. Ils nous ont par exemple demander "que le robot puisse bouger en même temps que le curseur" (P1) et que l'on puisse voir "à quel [moment du] mouvement est le robot à la frame sélectionnée" (P4).

Malgré ces difficultés à sélectionner une frame précise, les participants ont trouvé les outils d'édition fonctionnels et l'annotation utile et claire.

2. Quantitatifs

a. System Usability Score (SUS)

Le score SUS évalue l'utilisabilité du système sur une échelle de 0 à 100, 0 étant le pire résultat possible. Les participants ont attribué au système un score SUS moyen de 75,5 avec un minimum de 62,5 et un maximum de 92,5.

Le questionnaire SUS est composé de 10 énoncés. Il est demandé à l'utilisateur d'évaluer chaque énoncé sur une échelle de 5 points, de 1 pour "Pas du tout d'accord" à 5 pour "Tout à fait d'accord". Les énoncés posés couvrent l'efficacité, l'efficience et la satisfaction procurées par le système (figure V.2).

Le questionnaire est construit de sorte que les énoncés pairs sont positifs et les énoncés impairs sont négatifs. Chaque énoncé donne de 0 à 10 points en fonction de la réponse du participant. Le maximum de point pour un énoncé positif est obtenu lorsque le participant est Tout à fait d'accord avec l'énoncé et inversement pour les énoncés négatifs (tableau V.1).

	Énoncé pair	Énoncé impair
1) Pas du tout d'accord	0,0	10,0
2) Plutôt pas d'accord	2,5	7,5
3) Ni d'accord, Ni en désaccord	5,0	5,0
4) Plutôt d'accord	7,5	2,5
5) Tout à fait d'accord	10,0	0,0

TABLEAU V.1 – Barème des points par énoncé

b. Qualité de l'extrapolation

Pour chaque animation extrapolée, les participants ont évalué les 5 énoncés sur une échelle d'accord de 1 à 5, sur le modèle du questionnaire SUS. L'analyse de ces notes montre un effet important de la position de la cible (tableau V.2). Plus la cible est proche de la cible originale (cible 1), meilleures sont les notes. Cet effet est d'autant plus fort pour les notes accordées à la conservation des contraintes géométriques. Comme nous l'attendions, les scores de qualité "intrinsèque" sont légèrement supérieurs aux scores de conservation des propriétés correspondantes. On note également une faible dispersion des notes, en particulier pour les cibles 2 et 3, et ce pour les 5 énoncés.

1. Je pense que je voudrais utiliser ce logiciel fréquemment.
2. J'ai trouvé ce logiciel inutilement complexe.
3. J'ai trouvé ce logiciel facile à utiliser.
4. Je pense que j'aurai besoin de l'aide d'un technicien pour être capable d'utiliser ce logiciel.
5. J'ai trouvé que les différentes fonctions de ce logiciel ont été bien intégrées.
6. Je pense qu'il y a trop d'incohérence dans ce logiciel.
7. J'imagine que la plupart des gens serait capable d'apprendre à utiliser ce logiciel très rapidement.
8. J'ai trouvé ce logiciel très lourd à utiliser.
9. Je me sentais très en confiance en utilisant ce logiciel.
10. J'ai besoin d'apprendre beaucoup de choses avant de pouvoir utiliser ce logiciel.

FIGURE V.2 – Questionnaire SUS (System Usability Score)

C. Discussion

1. Utilisabilité du système

Différents barèmes ont été proposés dans la littérature pour interpréter les scores SUS. Sauro (2011) propose ainsi un barème construit à partir de 500 tests, les systèmes obtenant une note B s'ils ont un score SUS supérieur à 74 et A s'il est supérieur à 80,3. Bangor et al. (2008) proposent quant à eux que les bons systèmes obtiennent des scores SUS entre 70 et 80, en se basant sur 2324 tests qu'ils ont administrés. Dans une étude suivante, les mêmes auteurs proposent une échelle d'adjectifs qu'ils ont fait correspondre aux scores SUS sur la base de 1000 tests administrés, un score SUS de 71,4 correspondant à "Bon" et un score de 85,5 à "Excellent" (Bangor et al., 2009). Par ailleurs, selon Sauro (2013), le score SUS moyen reste une mesure fiable même avec

	Cible 2	Cible 3	Cible 4
1) Conserve le style	4,7	4,2	4,0
2) Conserve l'esthétique	4,7	4,3	3,8
3) Conserve les contraintes géométriques	4,6	3,9	3,1
4) Esthétiquement de bonne qualité	4,8	4,4	3,9
5) Fonctionnellement de bonne qualité	4,8	4,6	4,0

TABLEAU V.2 – Notes moyennes données par les participants aux animations extrapolées en fonction de la cible visée.

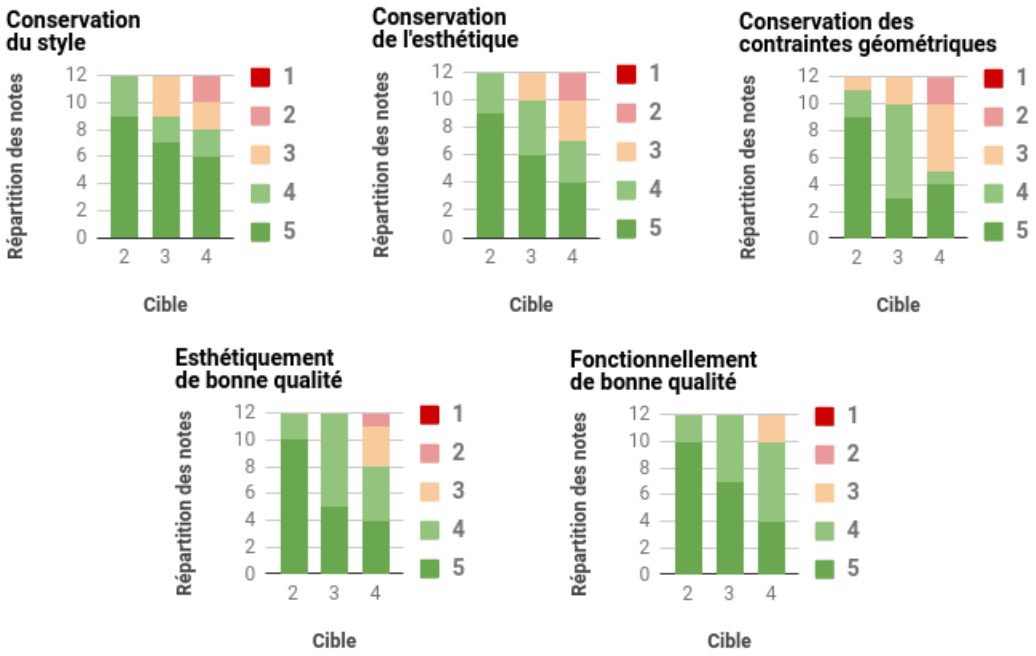


FIGURE V.3 – Répartition des notes en fonction de la cible visée. Il y a 3 extrapolations à évaluer par cible et par participant, soit 12 évaluations au total par cible

un faible nombre d'utilisateurs ("*Five is often a magic number for early-phase usability studies.*").

Avec un score SUS moyen attribué par les participants de 75,5, notre système a été jugé par les participants comme ayant une bonne utilisabilité selon ces barèmes. Le résultat est donc encourageant, bien qu'un nombre d'utilisateurs plus important permettrait de l'affiner.

Le détail des résultats qualitatifs permet de contextualiser ces scores SUS encourageants. Premièrement, ces résultats sont obtenus malgré la nouveauté que la méthode d'animation proposée par notre système représente pour les utilisateurs et malgré le faible temps de prise en main de celui-ci. C'est particulièrement le cas pour les difficultés liées à l'aspect mécanique du robot et donc aux spécificités du medium robotique. Deuxièmement, les difficultés liées au fonctionnement du logiciel qu'ils déclaraient avoir rencontrées sont principalement dues à des fonctionnalités manquantes ou à des détails d'implémentation qui sont aisément corrigeables dans une prochaine version du système.

2. Qualité de l'extrapolation

L'évaluation de la qualité de l'extrapolation par les participants est globalement positive, à la fois sur la conservation du style de l'animation et sur la conservation des contraintes géométriques. La faible dispersion des notes notamment pour les cibles 2 et 3 montre que la qualité de l'extrapolation est stable et n'est pas altérée en fonction du style de l'animation originale (tableau V.2).

Il est intéressant de noter que les participants s'attendaient à ce que le robot touche la nouvelle cible au même endroit que ce qu'ils ont montré dans l'animation originale. Or la fonction de généralisation définissant la contrainte géométrique était programmée pour toucher le milieu des cibles. Cela a donc diminué les notes données lorsque les animateurs souhaitaient que le robot touche une partie précise de la cible, par exemple le bord gauche.

Cela peut expliquer la différence entre les notes de conservation des contraintes géométriques et celles de qualité fonctionnelle de l'animation et montre l'importance de la communication entre l'animateur et le développeur. Annoter et commenter l'animation permet de s'assurer que le développeur programme bien la contrainte géométrique que l'animateur avait en tête.

Cet effet s'est également manifesté pour une des animations du participant 3. Dans son animation originale, le robot visait d'abord la cible avec la main gauche, puis touchait la cible avec le maillet, tenu dans la main droite, dans un second temps. La contrainte sur la main gauche n'étant pas programmée dans cette expérience, l'extrapolation ne pouvait respecter cette contrainte géométrique souhaitée par l'animateur que "par accident", et donc d'autant moins probable que la cible visée était éloignée de la cible originale.

Nous avons vu que l'extrapolation est de meilleure qualité quand la nouvelle cible est proche de la cible originale. Notre méthode d'extrapolation doit donc être principalement considérée comme une méthode locale dans la tâche proposée. Afin d'étendre le champ d'extrapolation utilisable, il est possible de multiplier les exemples et de sélectionner celui dont la cible visée était la plus proche de la nouvelle cible au moment de la génération d'une animation extrapolée. Il pourrait par exemple être utile de donner deux exemples, l'un où la cible du geste serait à gauche du robot et l'autre où elle serait à sa droite.

La méthode d'extrapolation présentée peut être utilisée pour assister l'animateur à étendre l'espace dans lequel une animation de bonne qualité peut être générée. Le système peut générer une animation par extrapolation pour une cible donnée. Si la qualité de l'extrapolation convient à l'animateur, celle-ci est rajoutée aux exemples, sinon, l'animateur crée un nouvel exemple manuellement. L'espace pour lequel l'animation peut être extrapolée est ainsi maximisé tout en minimisant la quantité de travail pour l'animateur.

Conclusion

L'animation en robotique a le potentiel de rendre les mouvements des robots plus expressifs et de permettre ainsi de créer des interactions plus engageantes pour les utilisateurs. Cependant, il est actuellement difficile de prototyper un robot animé ou encore de créer des animations qui soient réutilisables dans de nouvelles conditions. Cette thèse adapte les outils d'animation et les méthodes d'animation paramétrique au medium robotique afin d'intégrer les animateurs dans la conception des mouvements des robots.

A. Contributions

Les domaines du jeu vidéo et du film d'animation ont développé un grand nombre d'outils et de méthodes afin de créer les mouvements de personnages virtuels. Dans cette thèse, nous nous sommes inspirés de ces outils et méthodes afin de les adapter à la création de mouvements de robots.

Le chapitre II présente un outil de prototypage de robots expressifs animés. Celui-ci est développé comme une surcouche au logiciel d'animation 3D Blender et a été évalué au travers du prototypage de la tête robotisée Mia et de l'animation du robot humanoïde Poppy. Ces expérimentations nous ont permis de mettre en relief les possibilités offertes par une adaptation directe des outils d'animation 3D à la robotique ainsi que les limites que cela présente en terme d'ergonomie.

Le chapitre III présente l'outil d'animation robotique développé au cours de cette thèse. Nous montrons comment la manipulation directe s'intègre naturellement dans la méthode pose à pose implémentée par les outils d'animations 3D. Nous présentons également une fonctionnalité d'esquisse d'animation par manipulation directe s'inspi-

rant de l'Animation Directe utilisée en animation traditionnelle. L'ergonomie de notre outil est évalué par des tests utilisateurs dans le chapitre V.

Le chapitre IV présente nos travaux sur la création d'animation paramétrique pour un robot. A partir d'une analyse des limitations des méthodes existantes de déformation d'animation, nous formulons une heuristique pour la préservation du style d'une animation. Cette heuristique nous permet de définir une nouvelle méthode de déformation d'animation répartissant la déformation de pose finale en fonction de la répartition du mouvement dans l'animation originale. Enfin nous définissons une méthode et des outils de collaboration entre animateurs et développeurs pour la création d'animations paramétriques. Les tests utilisateurs présentés dans le chapitre V montrent le potentiel de cette méthode pour la création d'animations paramétriques par des animateurs.

B. Futures directions

1. Dictionnaire de contraintes géométriques

La méthodologie de création d'animation paramétrique développée dans le chapitre IV s'organise autour de la collaboration entre développeurs et animateurs. Les animateurs annotent l'animation pour indiquer les contraintes géométriques devant être respectées aux différents moments clés de l'animation. Ces indications textuelles permettant dans un second temps au développeur de créer la fonction génératrice de l'animation paramétrique. Pour cela, il transforme les contraintes géométriques en une fonction de coût à optimiser par le moteur de cinématique inverse. Ce choix permet à la méthode d'être généraliste mais présente l'inconvénient de ralentir les itérations dans la création d'une animation paramétrique.

Un dictionnaire de contraintes géométriques pourrait être créé en réutilisant les contraintes conçues pour des animations précédentes. De cette façon, l'animateur pourrait itérer rapidement en choisissant les contraintes parmi ce dictionnaire lorsque les contraintes géométriques sont similaires à celles utilisées précédemment.

2. Création interactive de l'animation paramétrique

La méthode d'extrapolation de l'animation présentée dans le chapitre IV se base sur un seul exemple. Cela permet de rapidement créer une animation paramétrique en

minimisant le nombre d'animations à créer. A l'inverse, le fait de se baser sur un exemple unique donne peu d'information sur ce qu'il est important de conserver dans l'animation. L'algorithme d'extrapolation encode un certain choix sur ce qui doit être conservé pour préserver le style de l'animation.

En permettant à l'animateur de corriger une première extrapolation de son animation originale, une méthode interactive d'animation permettrait d'inférer ce qu'il faut conserver de l'animation originale. Une approche possible serait de définir une forme paramétrique de l'a priori utilisé par notre algorithme d'extrapolation et de mettre à jour ses paramètres afin de correspondre à l'extrapolation corrigée.

3. Généralisation de l'animation par renforcement

Récemment, l'apprentissage par renforcement a été appliquée avec un grand succès à l'imitation de mouvement (Peng et al., 2018) pour des personnages virtuels. Cette méthode présente des points communs avec la méthode d'adaptation d'animation que nous présentons. Elle permet ainsi d'imiter un mouvement à partir d'un seul exemple et de l'adapter afin de répondre à des variations de l'environnement telles que la position d'une cible. Elle présente également des avantages supplémentaires comme la prise en compte de l'équilibre du personnage et potentiellement des collisions mais également la possibilité d'imiter un ensemble de mouvements plutôt qu'un unique exemple.

L'intégration de cette méthode d'adaptation du mouvement est donc une des directions à explorer pour la conception d'outils d'animation paramétrique pour la robotique. Cela nécessiterait de reformuler la méthode d'encodage des contraintes géométriques présentée dans le chapitre IV sous la forme de la création d'une fonction de récompense voire de curriculum permettant d'entraîner l'agent par renforcement.

Bibliographie

Al Moubayed, S., Beskow, J., Skantze, G., and Granström, B. (2012). Furhat : a back-projected human-like robot head for multiparty human-machine interaction. In *Cognitive behavioural systems*, pages 114–130. Springer.

Allbeck, J. and Badler, N. (2002). Toward representing agent behaviors modified by personality and emotion. *Embodied Conversational Agents at AAMAS*, 2 :15–19.

Amaya, K., Bruderlin, A., and Calvert, T. (1996). Emotion from motion. In *Graphics interface*, volume 96, pages 222–229. Toronto, Canada.

Badeig, F., Wargnier, P., Pino, M., Lopes, P. D. O., Grange, E., Crowley, J. L., Rigaud, A.-S., and Vaufreydaz, D. (2016). Impact of head motion on the assistive robot expressiveness-evaluation with elderly persons. In *1st International Workshop on Affective Computing for Social Robotics Workshop at the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*.

Bangor, A., Kortum, P., and Miller, J. (2009). Determining what individual sus scores mean : Adding an adjective rating scale. *Journal of usability studies*, 4(3) :114–123.

Bangor, A., Kortum, P. T., and Miller, J. T. (2008). An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, 24(6) :574–594.

Beira, R., Lopes, M., Praça, M., Santos-Victor, J., Bernardino, A., Metta, G., Becchi, F., and Saltarén, R. (2006). Design of the robot-cub (icub) head. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 94–100. IEEE.

Berlo, D. K. (1960). *The process of communication ; an introduction to theory and practice*. New York, Holt, Rinehart and Winston.

Bowden, R. (2000). Learning statistical models of human motion. In *IEEE Workshop on Human Modeling, Analysis and Synthesis, CVPR*, volume 2000.

Breazeal, C. and Scassellati, B. (1999). How to build robots that make friends and influence people. In *Intelligent Robots and Systems, 1999. IROS'99. Proceedings. 1999 IEEE/RSJ International Conference on*, volume 2, pages 858–863. IEEE.

Brooke, J. et al. (1996). Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194) :4–7.

Brooks, A. G., Gray, J., Hoffman, G., Lockerd, A., Lee, H., and Breazeal, C. (2004). Robot's play : interactive games with sociable machines. *Computers in Entertainment (CIE)*, 2(3) :10–10.

Brown, L. N. and Howard, A. M. (2014). The positive effects of verbal encouragement in mathematics education using a social robot. In *2014 IEEE Integrated STEM Education Conference*, pages 1–5. IEEE.

Bruderlin, A. and Williams, L. (1995). Motion signal processing. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 97–104. ACM.

Campeau-Lecours, A., Lamontagne, H., Latour, S., Fauteux, P., Maheu, V., Boucher, F., Deguire, C., and L'Ecuyer, L.-J. C. (2017). Kinova modular robot arms for service robotics applications. *International Journal of Robotics Applications and Technologies (IJRAT)*, 5(2) :49–71.

Chao, C., Gielniak, M., Yoo, J. W., and Thomaz, A. L. (2010). Interactive learning by demonstration with the simon robot. In *Proceedings of the 9th AAAI Conference on Enabling Intelligence Through Middleware*, pages 2–2. AAAI Press.

Chi, D., Costa, M., Zhao, L., and Badler, N. (2000). The EMOTE model for effort and shape. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00*, pages 173–182, Not Known. ACM Press.

Chi, D. M. (1999). A Motion Control Scheme for Animating Expressive Arm Movements. *IRCS Technical Reports Series*, page 80.

Cruz Ruiz, A., Pontonnier, C., Pronost, N., and Dumont, G. (2017). Muscle-Based Control for Character Animation : Muscle-Based Control for Character Animation. *Computer Graphics Forum*, 36(6) :122–147.

Darwin, C. (1872). The expression of emotion in animals and man. *London : Methuen.(1877), A biographical sketch of an infant. Mind*, 2 :285–294.

Duchenne, G.-B. (1862). *Mécanisme de la physionomie humaine ou Analyse électrophysiologique de ses différents modes d'expression.* P. Asselin.

Efron, D. (1941). *Gesture and environment.* King's Crown Press.

Ekman, P. (1999). Basic emotions. *Handbook of cognition and emotion*, pages 45–60.

Ekman, P. and Friesen, W. V. (1969). The repertoire of nonverbal behavior : Categories, origins, usage, and coding. *semiotica*, 1(1) :49–98.

Ericson, C. (2004). *Real-Time Collision Detection*. Morgan Kaufmann series in interactive 3D technology. Taylor & Francis.

Fitzpatrick, R. J. (2012). *Designing and constructing an animatronic head capable of human motion programmed using face-tracking software*. PhD thesis, Worcester Polytechnic Institute.

Forlizzi, J. and DiSalvo, C. (2006). Service robots in the domestic environment : a study of the roomba vacuum in the home. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 258–265. ACM.

Fujita, M. (2001). Aibo : Toward the era of digital creatures. *The International Journal of Robotics Research*, 20(10) :781–794.

Gangnet, M. (2000). Parametric function curve editing. US Patent 6,154,221.

Geijtenbeek, T., van de Panne, M., and van der Stappen, A. F. (2013). Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics*, 32(6).

Gielniak, M. J., Liu, C. K., and Thomaz, A. L. (2010). Stylized motion generalization through adaptation of velocity profiles. In *RO-MAN, 2010 IEEE*, pages 304–309. IEEE.

Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., and Maisonnier, B. (2009). Mechatronic design of nao humanoid. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 769–774. IEEE.

Hanson, D., Baurmann, S., Riccio, T., Margolin, R., Dockins, T., Tavares, M., and Carpenter, K. (2009). Zeno : A cognitive character. In *Ai magazine, and special proc. of aaai national conference, chicago*.

Hartmann, B., Mancini, M., and Pelachaud, C. (2002). Formational parameters and adaptive prototype instantiation for mpeg-4 compliant gesture synthesis. In *Computer Animation, 2002. Proceedings of*, pages 111–119. IEEE.

Hartmann, B., Mancini, M., and Pelachaud, C. (2005). Implementing expressive gesture synthesis for embodied conversational agents. In *International Gesture Workshop*, pages 188–199. Springer.

Hoffman, G. (2012). Dumb robots, smart phones : A case study of music listening companionship. In *RO-MAN, 2012 IEEE*, pages 358–363. Citeseer.

Hoffman, G. et al. (2007). *Ensemble : fluency and embodiment for robots acting with humans*. PhD thesis, Massachusetts Institute of Technology.

Hoffman, G. and Ju, W. (2014). Designing robots with movement in mind. *Journal of Human-Robot Interaction*, 3(1) :89–122.

Hoffman, G., Kubat, R., and Breazeal, C. (2008). A hybrid control system for puppeteering a live robotic stage actor. In *Robot and Human Interactive Communication, 2008. RO-MAN 2008. The 17th IEEE International Symposium on*, pages 354–359. Citeseer.

Holden, D., Saito, J., and Komura, T. (2016). A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (TOG)*, 35(4) :138.

Huang, Y. and Kallmann, M. (2010). Motion parameterization with inverse blending. In *MIG*, pages 242–253. Springer.

Ishii, H. (2008). Tangible bits : beyond pixels. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages xv–xxv. ACM.

Jack, R. E., Garrod, O. G., Yu, H., Caldara, R., and Schyns, P. G. (2012). Facial expressions of emotion are not culturally universal. *Proceedings of the National Academy of Sciences*, 109(19) :7241–7244.

Jeong, S., Breazeal, C., Logan, D., and Weinstock, P. (2017). Huggable : Impact of embodiment on promoting verbal and physical engagement for young pediatric

inpatients. In *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 121–126. IEEE.

Kim, M., Hyun, K., Kim, J., and Lee, J. (2009). Synchronized multi-character motion editing. *ACM transactions on graphics (TOG)*, 28(3) :79.

Komura, T., Shinagawa, Y., and Kunii, T. L. (1997). A Muscle-based Feed-forward Controller of the Human Body. *Computer Graphics Forum*, 16(s3) :C165–C176.

Korn, O. and Lee, N. (2017). *Game Dynamics*. Springer.

Kraft, D. (1988). A software package for sequential quadratic programming. *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*.

Kuroki, Y., Blank, B., Mikami, T., Mayeux, P., Miyamoto, A., Playter, R., Nagasaka, K., Raibert, M., Nagano, M., and Yamaguchi, J. (2003). Motion creating system for a small biped entertainment robot. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, pages 1394–1399. IEEE.

Laban, R. and Ullmann, L. (1971). *The mastery of movement.* ERIC.

Lapeyre, M., Rouanet, P., Grizou, J., Nguyen, S., Depraetre, F., Le Falher, A., and Oudeyer, P.-Y. (2014). Poppy project : open-source fabrication of 3d printed humanoid robot for science, education and art. In *Digital Intelligence 2014*, page 6.

LaViers, A. and Egerstedt, M. (2012). Style based robotic motion. In *American Control Conference (ACC), 2012*, pages 4327–4332. IEEE.

Leite, I., Pereira, A., Castellano, G., Mascarenhas, S., Martinho, C., and Paiva, A. (2011). Social robots in learning environments : a case study of an empathic chess companion. In *Proceedings of the International Workshop on Personalization Approaches in Learning Environments*, volume 732, pages 8–12.

Levine, S., Wang, J. M., Haraux, A., Popović, Z., and Koltun, V. (2012). Continuous character control with low-dimensional embeddings. *ACM Transactions on Graphics (TOG)*, 31(4) :28.

Liu, C. K., Hertzmann, A., and Popović, Z. (2005). Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics (TOG)*, 24(3) :1071–1081.

Luria, M., Hoffman, G., Megidish, B., Zuckerman, O., and Park, S. (2016). Designing vyo, a robotic smart home assistant : Bridging the gap between device and social agent. In *Robot and Human Interactive Communication (RO-MAN), 2016 25th IEEE International Symposium on*, pages 1019–1025. IEEE.

Maciejewski, A. A. and Klein, C. A. (1989). The singular value decomposition : Computation and applications to robotics. *The International journal of robotics research*, 8(6) :63–79.

Matusik, S. (2011). Demoscene : The art of the algorithms.

McNeill, D. (2008). *Gesture and thought*. University of Chicago press.

Mori, M. (1970). Bukimi no tani [the uncanny valley]. *Energy*, 7 :33–35.

Na, M., Yang, B., and Jia, P. (2008). Improved damped least squares solution with joint limits, joint weights and comfortable criteria for controlling human-like figures. In *Robotics, Automation and Mechatronics, 2008 IEEE Conference on*, pages 1090–1095. IEEE.

Nakaoka, S., Kajita, S., and Yokoi, K. (2010). Intuitive and flexible user interface for creating whole body motions of biped humanoid robots. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1675–1682. IEEE.

Nierhoff, T., Hirche, S., and Nakamura, Y. (2016). Spatial adaption of robot trajectories based on laplacian trajectory editing. *Autonomous Robots*, 40(1) :159–173.

Nishio, S., Ishiguro, H., and Hagita, N. (2007). Geminoid : Teleoperated android of an existing person. In *Humanoid robots : New developments*. InTech.

Peng, X. B., Abbeel, P., Levine, S., and van de Panne, M. (2018). Deepmimic : Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 37(4) :143.

Penner, R. (2002). Motion, tweening, and easing. *Programming Macromedia Flash MX*, pages 191–240.

Perlin, K. (2002). Improving noise. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 681–682. ACM.

Pierris, G. and Lagoudakis, M. G. (2009). An interactive tool for designing complex robot motion patterns. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 4013–4018. IEEE.

Pot, E., Monceaux, J., Gelin, R., and Maisonnier, B. (2009). Choregraphe : a graphical tool for humanoid robot programming. In *Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on*, pages 46–51. IEEE.

Rose III, C. F., Sloan, P.-P. J., and Cohen, M. F. (2001). Artist-directed inverse-kinematics using radial basis function interpolation. In *Computer Graphics Forum*, volume 20, pages 239–250. Wiley Online Library.

Saldien, J., Goris, K., Yilmazyildiz, S., Verhelst, W., and Lefeber, D. (2008). On the design of the huggable robot probo. *Journal of Physical Agents*.

Saldien, J., Vanderborght, B., Goris, K., Van Damme, M., and Lefeber, D. (2014). A motion system for social and animated robots. *International Journal of Advanced Robotic Systems*, 11(5) :72.

Sauro, J. (2011). Measuring usability with the system usability scale (sus).

Sauro, J. (2013). Things to know about the system usability scale (sus).

Scheib, V., Engell-Nielsen, T., Lehtinen, S., Haines, E., and Taylor, P. (2002). The demo scene. In *ACM SIGGRAPH 2002 conference abstracts and applications*, pages 96–97. ACM.

Schramm, W. E. (1954). *The process and effects of mass communication.* University of Illinois Press.

Shannon, C. E., Weaver, W., and Burks, A. W. (1969). *The mathematical theory of communication*. Urbana : Univ. of Illinois Pr.

Shapiro, A., Cao, Y., and Faloutsos, P. (2006). Style components. In *Proceedings of Graphics Interface 2006*, pages 33–39. Canadian Information Processing Society.

Sirkin, D., Mok, B., Yang, S., and Ju, W. (2015). Mechanical ottoman : how robotic furniture offers and withdraws support. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, pages 11–18. ACM.

Sorkine, O. (2005). Laplacian mesh processing. In *Eurographics (STARs)*, pages 53–70.

Thomas, F., Johnston, O., and Thomas, F. (1995). *The illusion of life : Disney animation*. Hyperion New York.

Urtasun, R., Glardon, P., Boulic, R., Thalmann, D., and Fua, P. (2004). Style-based motion synthesis. In *Computer Graphics Forum*, volume 23, pages 799–812. Wiley Online Library.

Van Breemen, A. (2004). Bringing robots to life : Applying principles of animation to robots. In *Proceedings of Shapping Human-Robot Interaction workshop held at CHI 2004*, pages 143–144.

Van Breemen, A. and Xue, Y. (2006). Advanced animation engine for user-interface robots. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 1824–1830. IEEE.

van Breemen, A., Yan, X., and Meerbeek, B. (2005). icat : an animated user-interface robot with personality. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 143–144. ACM.

Vinciarelli, A., Pantic, M., and Bourlard, H. (2009). Social signal processing : Survey of an emerging domain. *Image and vision computing*, 27(12) :1743–1759.

Wallbott, H. G. (1998). Bodily expression of emotion. *European journal of social psychology*, 28(6) :879–896.

Wampler, C. W. (1986). Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1) :93–101.

Westlund, J. K., Gordon, G., Spaulding, S., Lee, J. J., Plummer, L., Martinez, M., Das, M., and Breazeal, C. (2015). Learning a second language with a socially assistive robot. *Almere, The Netherlands*.

Whitney, D. E. (1969). Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on man-machine systems*, 10(2) :47–53.

Whitney, D. E. (1972). The mathematics of coordinated control of prosthetic arms and manipulators. *ASME Journal of Dynamic Systems, Measurement and Control*, 20(4) :303–309.

Witkin, A. and Kass, M. (1988). Spacetime constraints. *ACM Siggraph Computer Graphics*, 22(4) :159–168.

Witkin, A. and Popovic, Z. (1995). Motion warping. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 105–108. ACM.