Environnement logiciel pour l'identification du mouvement

Jean-Luc Nougaret, Armel Cretual, Mathilde Vandenberghe

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Environnement logiciel pour l'identification du mouvement

Jean-Luc Nougaret — Armel Cretual — Mathilde Vandenberghe

N° 4429

Avril 2002

THÈME 3

ISSN 0249-6399 ISRN INRIA/RR--4428--FR+ENG

Environnement logiciel pour l'identification du mouvement

Jean-Luc Nougaret *, Armel Cretual †, Mathilde Vandenberghe ‡

Thème 3 — Interaction homme-machine,
images, données, connaissances
Projet Siames

Rapport de recherche n° 4429 — Avril 2002 — 29 pages

Résumé :

Ce rapport décrit un environnement logiciel dédié à une forme particulière d'acquisition de mouvement dont l'originalité est de prendre en compte une connaissance a priori sur le mouvement et sur le contexte de la mesure. Cette approche exploite de manière optimale les informations issues de capteurs de nature diverses en ajustant les paramètres du modèle qui sert à la synthèse du mouvement. Le prototype est parfaitement intégré avec Maya, un système d'animation professionnel, tant au niveau logiciel qu'à celui de l'interface utilisateur et des modes opératoires. Ce travail s'inscrit dans le cadre du projet Mouvement.

Mots-clés : acquisition de mouvement, animation basée sur les modèles physiques, identification de systèmes dynamiques, optimisation.

Le projet Mouvement est un projet soutenu par le Réseau National de Technologies Logicielles (RNTL), impliquant des partenaires académiques (projet Siames de l'INRIA-Rennes, laboratoire de l'exercice musculaire et de la performance sportive de l'Université de Haute-Bretagne) et industriels (sociétés Infogrames, Realviz et CEA-LETI).

Maya est un logiciel d'animation développé et distribué par la société canadienne Alias|Wavefront, filiale de Silicon Graphics Inc.

* Chercheur Inria au sein du projet Siames, architecte du système présenté ici.
† Ingénieur-expert Inria au sein du projet Siames.
‡ Ingénieur-expert Inria au sein du projet Siames.

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex (France)
Téléphone : +33 2 99 84 71 00 —Télécopie : +33 2 99 84 71 71

A software framework for motion identification

Abstract: This report describes a software framework suitable for a peculiar kind of motion capture, the originality of which is to take into account an a priori knowledge on both the movement and the context in which the measurements are performed. This approach optimally merges various pieces of sensory information by tuning the parameters of the motion synthesis model. The prototype blends seamlessly with Maya, a professional animation system, in terms of software integration, user interface and workflow.

Key-words: motion capture, physics-based animation, dynamic system identification, optimization.

Terminologie et abréviations

- On désigne par *sous-système d'estimation du mouvement* ou plus simplement par système d'estimation, l'ensemble des modules logiciels correspondant au système informatique à réaliser dans le cadre des sous-projets SP3et SP4.
- On désigne par le terme de *logiciel d'accueil* le logiciel d'animation choisi pour l'intégration du prototype final (Maya).
- On désigne par *degré de liberté* ou *DDL*, les paramètres de translation et de rotation déterminant la position du personnage virtuel dans l'espace, et les angles articulaires.
- On désigne par *paramètre d'un mouvement*, noté λ, un paramètre qui affecte la génération de trajectoires. Typiquement, un tel paramètre est défini sur un intervalle $[\lambda_{min}\lambda_{max}]$. Dans certains cas, λ peut être éventuellement considéré comme une variable aléatoire décrite par sa fonction densité de probabilité.
- On désigne ici par *modèle de mouvement cinématique*, un modèle de mouvement paramétré de la forme $x = c(u,t)$, c'est à dire toute fonction capable de fournir la valeur d'un certain nombre de DDLs x en fonction du temps et d'un vecteur d'entrée u. Ce type de modèle englobe les fonctions de cinématique " classiques " comme l'interpolation image-clefs, mais aussi les algorithmes de cinématique inverse auquel cas u représente la position de l'effecteur de la chaîne cinématique considéré. L'analyse de trajectoires réelles permet de dériver des modèles cinématiques empiriques. C'est par exemple le cas du modèle de marche développé à l'UHB.
- On désigne ici par *modèle de mouvement dynamique*, un modèle paramétré capable de générer des trajectoires à partir de conditions initiales connues, par intégration temporelle d'une équation d'évolution, différentielle continue de la forme $\dot{x} = f_\lambda(x,u,t)$. Le vecteur u est un vecteur d'entrée et x représente l'état du système, c'est à dire l'ensemble des variables d'état qui résument son historique et déterminent son évolution future. On considère ici une intégration à pas fixe obtenue par discrétisation temporelle de l'équation d'évolution: $x_{n+1} = f_\lambda(x_n,u_n,n)$ où n est le pas de temps. Pour les applications d'animation interactive comme les jeux vidéo, un générateur de mouvement (y compris intégrant des automates à états finis) peut être modélisé de cette façon. Pour l'étude mécanique du geste, un tel modèle peut être dérivé de manière systématique à partir d'une description du mécanisme et par application des principes généraux, ex. Lagrange (cela suppose un traitement des contraintes pour passer d'un système différentiel algébrique à un système différentiel ordinaire). Dans le domaine de la biomécanique, les lois du comportement neuromusculaire peuvent éventuellement être modélisées ainsi.

1 Introduction

Le projet RNTL Mouvement associe les laboratoires industriels des sociétés Infogrames, Realviz et CEA-LETI, le laboratoire de physiologie et de biomécanique de l'exercice musculaire de l'Université de Haute Bretagne et le projet Siames. Son objectif est d'exploiter la complémentarité de différentes sources de mesures et de tenir compte d'une connaissance a priori, afin de s'affranchir de certaines restrictions et limites des systèmes existants pour l'acquisition du mouvement.

Il s'agit avant tout d'un projet de développement logiciel basé sur une approche de prototypage rapide, justifiée par la mise au point itérative de l'interface utilisateur et des modes opératoires. Sur le plan scientifique, le modèle mathématique sur lequel se basent la conception générale du système et sa mise en œuvre algorithmique renvoie à des concepts fondamentaux de la théorie de l'automatique et du signal.

1.1 Fondements scientifiques

Sur le plan théorique, l'approche étudiée renvoie à des concepts fondamentaux du domaine de l'automatique et du traitement du signal: ceux de l'*estimation* et l'*identification*. L'objectif scientifique corollaire de ce projet de développement est d'explorer le potentiel et les limites de ces outils théoriques, en évaluant notamment l'apport des modèles statistiques: ainsi différentes politiques d'estimation sont envisageables, selon que les lois de probabilité a priori des paramètres, voire du bruit de mesure, sont connues ou non (hypothèse Bayesienne ou maximum de vraisemblance), mais aussi selon que l'on recherche un estimateur optimal (meilleur au sens de l'espérance statistique) ou un estimateur robuste (meilleur au sens du pire cas). Plus généralement, l'approche proposée se base sur un processus d'optimisation dont la formulation mathématique et la mise en œuvre algorithmique dépendent des hypothèses disponibles et de la politique d'estimation choisie.

1.2 Résumé du cahier des charges

Le sous-système d'estimation de mouvement, permet à l'utilisateur de spécifier *a priori* une connaissance sur le mouvement effectué par l'acteur et sur le contexte de la mesure (nature et placement des capteurs). Cette connaissance sert de guide et reformule le problème de l'acquisition de mouvement en un problème d'estimation des paramètres d'un modèle de mouvement que l'on cherche à identifier : il s'agit de retrouver parmi la classe de mouvements spécifiés, celui offrant la meilleure compatibilité avec les mesures observées. Comme l'illustre la figure 1, cette approche permet de palier aux difficultés inhérentes à la reconstruction des trajectoires à partir de la mesure (inversion de la fonction associée au processus de mesure), par un recours à la puissance de calcul de l'ordinateur.

Dans l'approche classique de reconstruction à partir d'une séquence acquise préalablement, le mouvement obtenu est entièrement dépendant de la mesure. Si le résultat n'est pas jugé satisfaisant par l'utilisateur, il est alors nécessaire de procéder à une nouvelle acquisition. Dans l'approche que nous proposons, le choix du modèle est antérieur au traitement des trajectoires réelles. L'introduction d'une connaissance *a priori*, qui peut être le cas échéant modifiée voire raffinée, permet, à partir d'une seule séquence acquise de parfaire le modèle en sortie de la chaîne d'édition du mouvement. Cet incrément méthodologique provient de l'utilisation de cette connaissance *a priori* dans un alogorithme itératif d'optimisation des paramètres décrivant le mouvement, qui compare la séquence acquise et une séquence simulée à partir du modèle.

Le système d'estimation du mouvement fonctionne selon un schéma itératif à deux niveaux :

- un niveau manuel comprenant l'acquisition réelle du mouvement et la spécification de la connaissance *a priori* que l'utilisateur a de ce mouvement et des conditions d'acquisition
- un niveau automatique permettant, par le biais d'un algorithme d'optimisation, l'ajustement des paramètres du mouvement

La figure 2 donne le schéma général résumant les interactions de l'utilisateur avec le processus automatique d'estimation du mouvement.

En règle générale, l'acquisition du mouvement ne se fait qu'une seule fois, mais il est bien sûr possible, que ce soit par nécessité (lorsque l'information disponible est réellement trop pauvre) ou par souhait (dans le cas où l'utilisateur cherche à valider la pertinence d'un modèle sur un type de mouvement particulier) de procéder à une nouvelle acquisition. De même, l'étape de spécification de la connaissance sur les conditions de cette acquisition n'est généralement réalisée qu'une seule fois. Cette

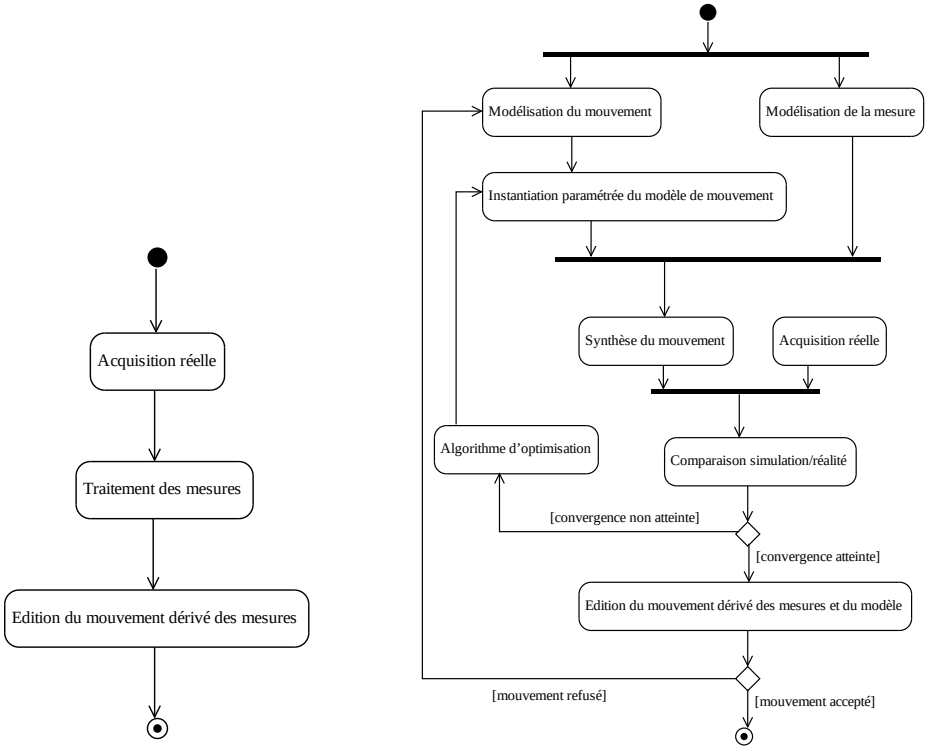

FIG. 1 – *Diagramme de flux comparatif entre une approche de reconstruction et l'approche proposée, basée sur un cycle itératif d'estimation de mouvement et sur une connaissance* a priori

étape vise à définir les capteurs virtuels qui permettront de simuler automatiquement la mesure.

L'introduction par l'utilisateur des connaissances sur le mouvement permet, à chaque itération de la boucle automatique d'estimation, d'instantier le modèle de mouvement, c'est-à-dire de générer une trajectoire virtuelle. Les capteurs virtuels permettent alors de générer une mesure virtuelle. L'algorithme d'optimisation cherche les paramètres qui minimise l'écart entre les mesures virtuelle et réelle, et ce jusqu'à convergence.

L'originalité de l'approche proposée ici, réside non seulement dans la possibilité donnée à l'utilisateur d'introduire une connaissance *a priori* sur le mouvement, mais également dans celle de modifier cette connaissance à l'issue de l'étape d'optimisation automatique, lorsqu'il juge que le modèle de mouvement alors obtenu ne correspond pas à ses attentes. Celà permet donc d'affiner le mouvement optimal issu des mesure sans avoir à procéder à une nouvelle acquisition, ce qui n'est pas possible avec la méthode classique de reconstruction *a posteriori*.

Cette approche présuppose donc que l'utilisateur soit en mesure d'injecter une connaissance *a priori* sur le mouvement à traiter. Dans le domaine des jeux vidéo, l'intérêt principal est de pouvoir définir dès le départ un modèle procédural de synthèse du mouvement, et de la paramétrer à partir de mesures réelles. Dans le domaine de l'analyse biomécanique du geste sportif, cette approche autorise par exemple d'évaluer la capacité d'un modèle *a priori* à expliquer les trajectoires réelles. Sur le plan pratique, la validité du concept proposé est déterminée par l'aptitude

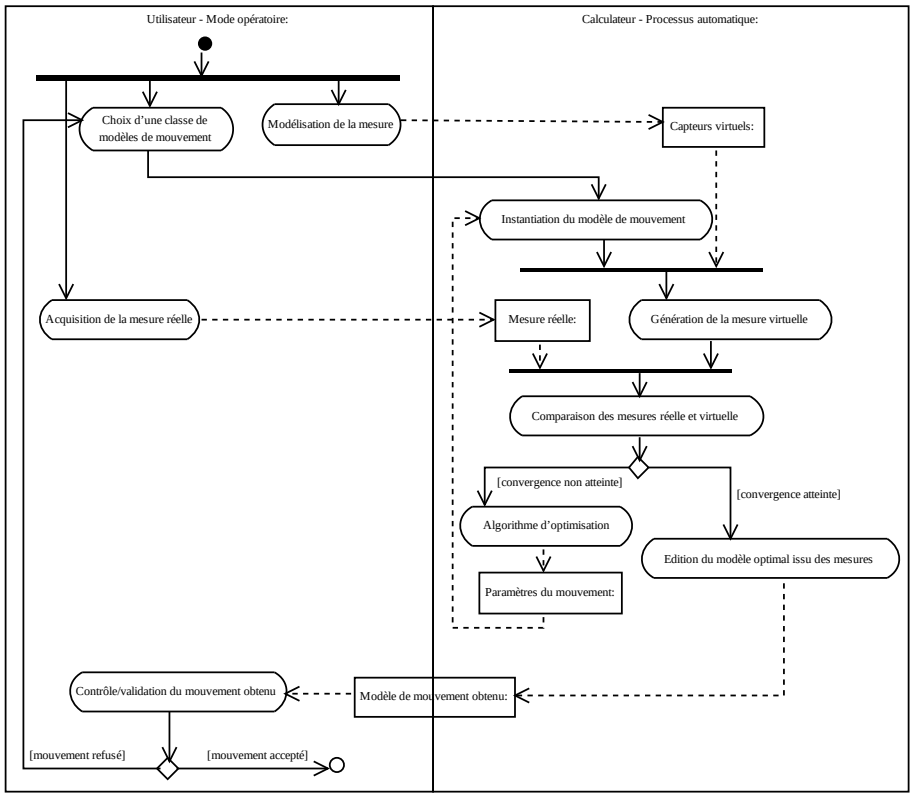

FIG. 2 – *Diagramme en "swimline" illustrant le mode d'utilisation semi-automatique du système d'estimation de mouvement*

du système à faire coopérer efficacement l'utilisateur avec la machine, de façon à tirer le meilleur parti de sa puissance de calcul.

1.3 Historique de la conception de l'architecture

L'architecture logicielle présentée ici résulte d'un cycle de prototypage itératif inspiré des méthodologies récentes de développement logiciel [1]. Cette approche, adaptée et assouplie pour les besoins du projet mené par une équipe de trois développeurs, a permis de définir d'abord l'interface utilisateur et les modes opératoires, pour s'intéresser ensuite à la nature et à la fonction des différents composants " métiers ", et finalement aboutir à une architecture logicielle stable du système. Sur une durée d'un an, trois versions du prototype ont ainsi permis de valider successivement (0.0) le concept sur lequel repose le système (1.0) son aspect fonctionnel et (2.0) son architecture logicielle. Ces itérations successives se sont accompagnées d'une évolution progressive des outils et des langages utilisés pour ce prototypage, privilégiant d'abord le caractère " malléable " du code grâce au choix de langages interprétés et d'environnements RAD (Visual Basic, Matlab, MEL[1]). On s'est ensuite attaché progressivement à des considérations sur la performance et sur le caractère générique du système, ce qui coïncide avec l'adoption de langages compilés (C++ et génération de code) et à la stabilisation de l'architecture logicielle. La figure 3 retrace cette évolution. L'architecture étant stabilisée, les développements

1. MEL est le langage se script de Maya.

ultérieurs seront confinés au développement de fonctionnalités de communication avec un éventuel serveur de mesure, ce qui peut avantageusement être réalisé par l'écriture de quelques scripts MEL, le code C++ étant définitif pour l'essentiel.

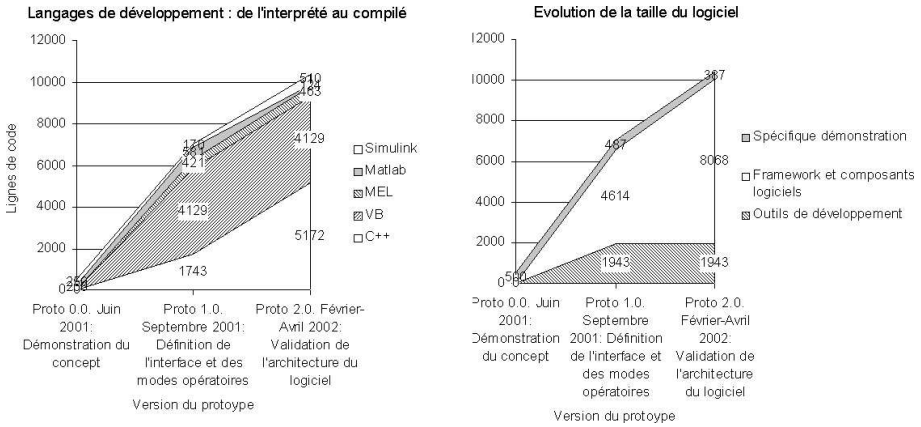

FIG. 3 – *Evolution de la taille du logiciel.*

La figure 4 représente les gains en termes de performance associés à cette évolution. Il est à noter que les gains obtenus, de l'ordre d'un facteur 10, sont dus uniquement à la mise au point progressive de l'architecture et à la migration vers des langages compilés. Des gains additionnels sont envisageables par une optimisation du code (remplacement des appels aux fonction par du code en ligne) .

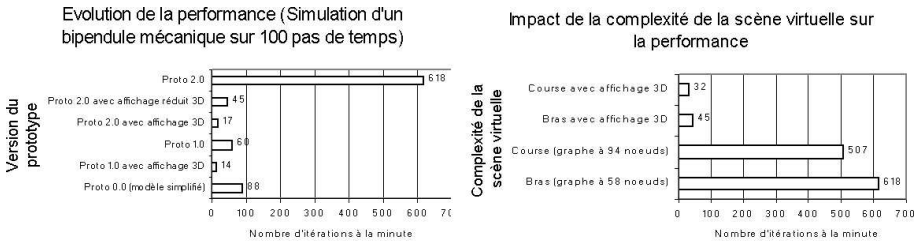

FIG. 4 – *Evolution de la performance.*

2 Une architecture modulaire et ouverte

Le choix de Maya comme plate-forme d'intégration logicielle permet de réaliser le système par assemblage de composants " métiers " spécifiques au nouveau système, mais en se basant aussi sur les fonctionnalités de base d'animation offertes par le logiciel d'accueil. Par ailleurs Maya fournit un cadre dans lequel le système s'intègre au niveau des modes opératoires et de l'interface utilisateur. La figure 5 représente l'interface utilisateur du système d'estimation de mouvement tel qu'il est implémenté dans le prototype 2.0. Les fenêtres sont gérées d'une part par Maya, d'autre part par un composant GUI, qui communiquent à travers COM.

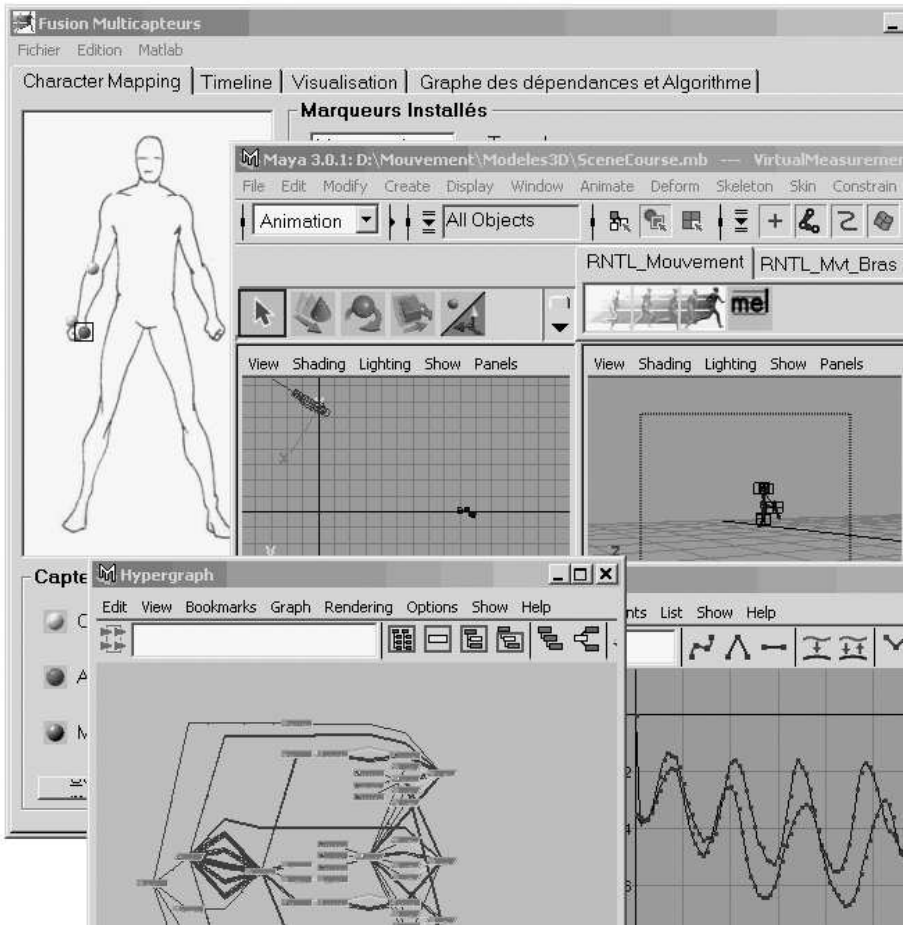

FIG. 5 – *Interface du système intégré avec le logiciel Maya.*

Composant	Type	Représentation Maya	Spécification
Estimation	Bibliothèque dynamique	Commande MEL	C++
GUI	Serveur COM (processus séparé)	Proxy	Visual Basic
Client du GUI	Bibliothèque dynamique	Commande MEL	C++
Mouvement	Bibliothèque dynamique	Nœud de l'hypergraphe	Simulink
Capteur	Bibliothèque dynamique	Nœud de l'hypergraphe	Simulink
Serveur mesures	Exécutable (processus séparé)	Gestionnaire périphérique	API en C

FIG. 6 – *Liste des composants constitutifs du système.*

2.1 Utilisation de Maya comme intergiciel

L'architecture ouverte de Maya permet également de le considérer comme un intergiciel, ou comme un middleware permettant de réaliser le collage des composants métiers développés pour les besoins spécifiques du système :

- Maya définit une interface logicielle standardisée (API) pour la connexion de nouveaux composants.
- Maya autorise la mise au point rapide de nouvelles fonctionnalités logicielles à l'aide du langage de scripts MEL. Ainsi, on peut utiliser ces scripts pour invoquer de nouvelles instances de composants et pour les connecter les unes aux autres afin de réaliser le système par simple assemblage. Cet assemblage peut d'ailleurs être géré dynamiquement.

2.2 Une architecture à base de composants

Le système est réalisé par un assemblage de composants " métiers " spécifiques à l'estimation de mouvement. Le choix du langage utilisé pour chacun des composants a été déterminé de manière à en faciliter la spécification du comportement, ainsi que le résume le tableau ci-dessous.

Le diagramme de composants représentant leurs dépendances lors de l'exécution est illustré par la figure 7. La " colle " en est fournie par la technologie COM ainsi que par l'API et le langage de script de Maya. De manière à faciliter un éventuel portage du système sur d'autres plate-formes d'accueil, les dépendances par rapport à Maya sont confinées dans des plug-ins qui réalisent une couche d'interfaçage avec les composants du type mouvement, capteur mais aussi avec le composant GUI. Pour des raisons de performance liées au contrôle des animations, à la gestion des exceptions au niveau de l'interface utilisateur et à la vitesse de transfert des résultats des animations, le composant EstimeMotion, qui est au cœur du système d'estimation du mouvement, est directement encapsulé comme un plug-in Maya. Un portage sur une autre plate-forme nécessiterait donc de revoir la structure interne de ce composant en isolant certaines classes liées à Maya.

Au niveau de l'interface Maya, les nouvelles fonctionnalités liées à l'estimation du mouvement sont associées à un ensemble de commandes et de scripts MEL, qui à leur tour, sont accessibles à travers de nouveaux boutons dans le GUI de Maya. Le diagramme de composants de la figure 8 détaille l'intégration de ces nouveaux outils. Le choix d'implémenter les trois composants qui y figurent en MEL plutôt qu'en C++ est dicté par les considérations suivantes :

- Le composant responsable de la création du graphe de la scène à partir des informations fournies au niveau du GUI doit aussi pouvoir être facilement modifié ou remplacé par l'utilisateur en fonction du type de scène (notamment représentation du squelette du personnage synthétique) dont il souhaite automatiser la construction.

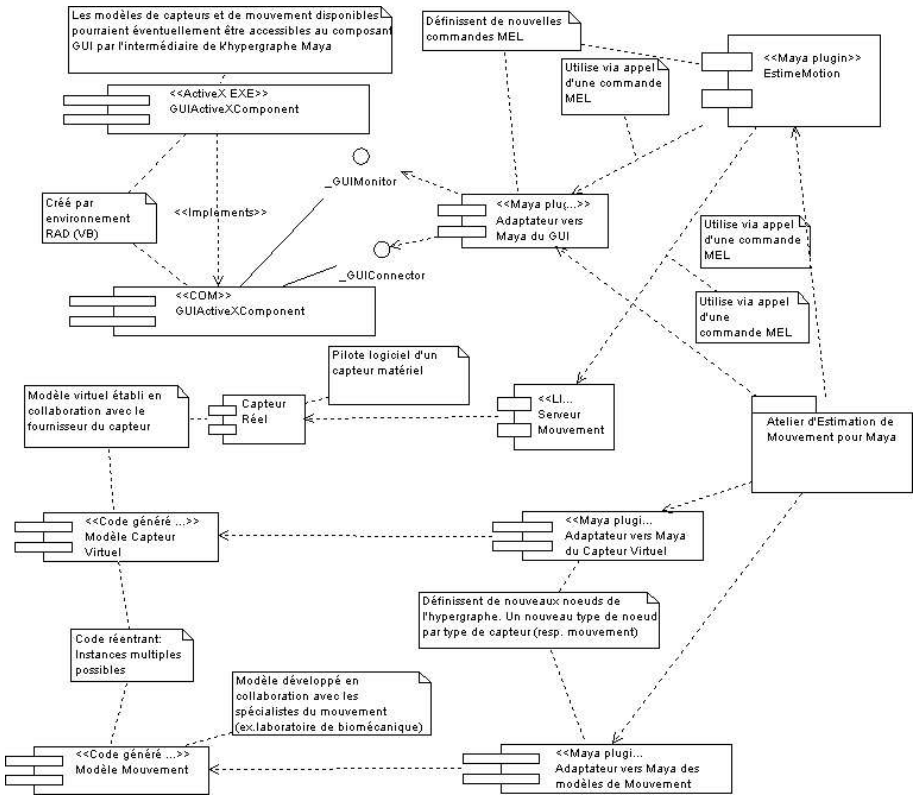

FIG. 7 – *Diagramme de composants du prototype actuel (capteur réel fictif).*

- La commande d'interface avec le GUI permet aux différents plug-ins ou scripts MEL de communiquer avec le composant GUI par simple invocation d'une commande MEL sans se soucier de la couche COM.
- Le processus d'estimation est invoqué par une commande MEL dont les paramètres permettent par exemple de spécifier le format de la source de mesures. L'intérêt de pouvoir invoquer cette commande à l'intérieur d'un script MEL personnalisé par l'utilisateur peut permettre par exemple de lancer plusieurs processus d'estimation en séquence en modifiant entre temps le graphe de la scène, c'est à dire en changeant par exemple la structure du modèle que l'on cherche à estimer.

3 Architecture interne du composant d'estimation

On décrit ici l'architecture interne du composant d'estimation qui constitue le cœur du système. Bien que son architecture essentiellement figée permette de voir ce composant comme une boîte noire lors de la phase d'intégration en aval du système, une compréhension générale de l'architecture interne de ce composant est toutefois nécessaire pour appréhender les modes opératoires de l'application et la manière dont les interactions avec les composants externes sont orchestrées. Cette description concerne successivement les aspects fonctionnel (relations entrées-sorties induites par les différents blocs fonctionnels), structurel (diagrammes de classes exprimant des relations statiques), et comportemental (diagramme de séquence exprimant les relations dynamiques entre les classes).

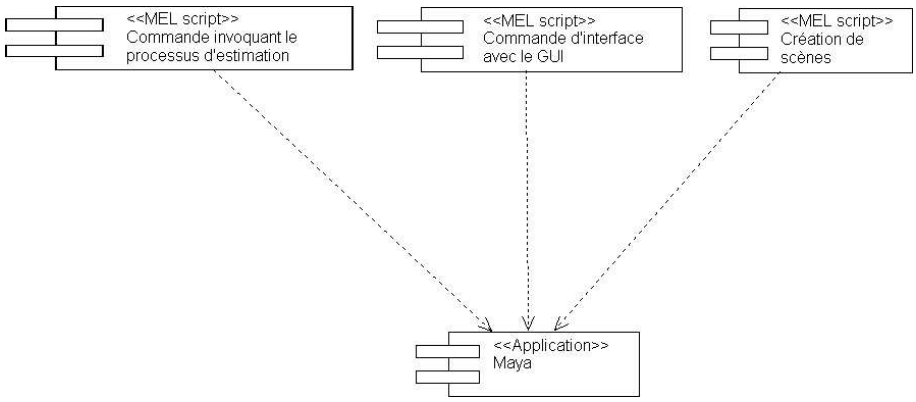

FIG. 8 – *Composants contenus dans le paquetage Atelier d'Estimation.*

3.1 Description fonctionnelle

Conformément aux objectifs affichés dans le cahier des charges [2], le système d'estimation du mouvement repose sur la simulation d'une scène virtuelle représentative du contexte réel de l'acquisition. On rappelle que cette scène virtuelle intègre différentes classes objets, notamment des objets capteurs et des objets mouvements, qui représentent la connaissance que l'utilisateur possède et injecte a priori. La représentation informatique de cette scène virtuelle correspond à un graphe de simulation tel que celui décrit sur la figure 9.

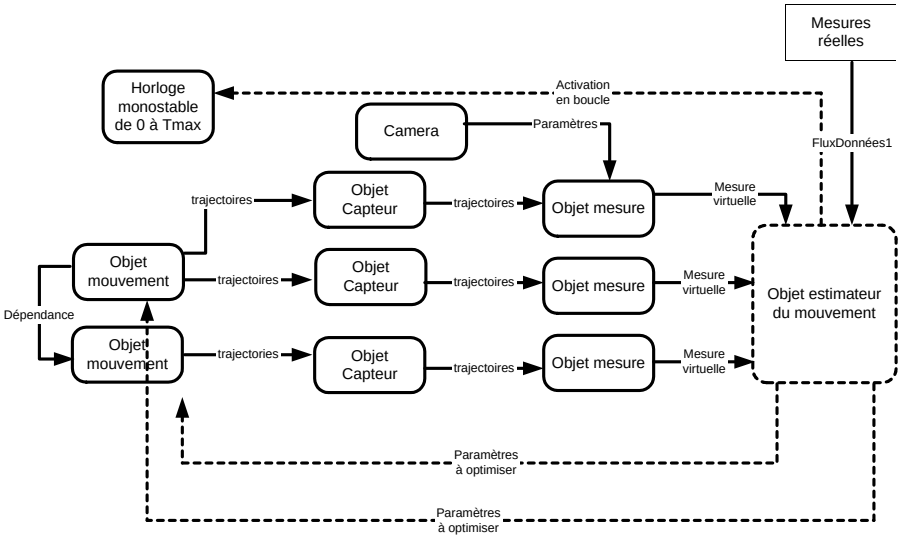

FIG. 9 – *Diagramme fonctionnel illustrant les objets et les flots de données impliqués dans l'optimisation et la simulation de la scène virtuelle.*

La construction de ce graphe est effectuée à l'aide d'un programme (script) lors d'une phase d'initialisation du système. Ce script crée les instances (nœuds) des classes d'objets effectivement présents et établit des connexions entre ces nœuds qui définissent des flots de données (informations), matérialisés par des flèches pleines sur la figure ci-dessus. Du point de vue de la simulation du graphe par intégration temporelle (sur un intervalle de temps $[0T]$ correspondant à la durée de l'acquisition), ces flots de données correspondent à des dépendances dans le calcul de la

scène virtuelle à chaque instant, calcul auquel chaque nœud contribue en fonction des informations qu'il reçoit des nœuds situés en amont (formalisme classique en automatique et traitement du signal). Le processus d'estimation ajuste progressivement certains paramètres de ce graphe (les paramètres associés au nœud mouvement ou à un réseau de tels nœuds), de manière à faire coïncider mesure virtuelle et mesure réelle. Chaque itération sur l'ajustement des paramètres donne lieu à l'activation d'une simulation sur $[0T]$. Ainsi, les flèches représentées en pointillés sur la figure 9 correspondent à des flux de contrôle du processus d'estimation sur le graphe. La figure 10 représente un exemple de scène virtuelle implémentée à l'aide d'un hypergraphe Maya (la vue est partielle afin d'en faciliter la compréhension). Au cœur de ce graphe, on remarquera la présence du nœud " EstimationProxy " qui, comme son nom l'indique, est un représentant du processus d'estimation au sein de la scène virtuelle. Ce nœud possède des entrées et des sorties qui communiquent avec le reste de la scène:

- Les ports de sortie de ce nœud sont les paramètres à identifier utilisés pour la synthèse du mouvement (à ne pas confondre avec les variables d'état associées au mouvement). Ceux-ci sont mis à jour avant chaque activation d'une simulation dynamique sur $[0T]$.
- Les ports d'entrée de ce nœud correspondent aux différentes mesures virtuelles telles qu'elles sont prélevées au niveau des nœuds " capteurs " tout au long d'une simulation dynamique sur $[0T]$.La simulation engendre donc un flot de données de mesures virtuelles dont l'historique est enregistré sur $[0T]$.

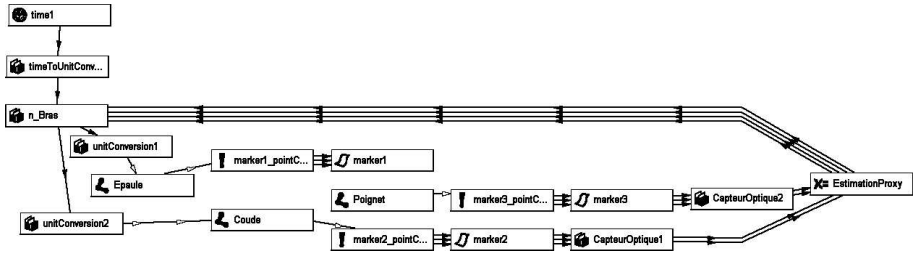

FIG. 10 – *Graphe d'une scène virtuelle permettant d'estimer le mouvement d'un bras à partir d'un modèle dynamique du mouvement et de la fusion des informations de deux capteurs optiques.*

3.2 Description structurelle

Le diagramme de classes du composant principal, représenté par la figure 11, est assemblé à partir de motifs de conception imbriqués les uns dans les autres. On y reconnaîtra notamment :

- L'utilisation centrale du patron de conception "Stratégie " qui permet de découpler le choix de l'algorithme d'optimisation du contexte auquel celui-ci est appliqué. Le choix de la stratégie employée est actuellement forcé par l'utilisateur qui spécifie le type d'algorithme à utiliser, mais il est théoriquement envisageable avec cette architecture de sélectionner automatiquement l'algorithme et les paramètres de la convergence en fonction du contexte, c'est à dire en fonction de la nature du problème posé à l'optimisation.
- L'utilisation de classes abstraites et d'interfaces permettant d'isoler l'architecture des spécificités de chaque application concrète.

La notion de framework pour l'estimation du mouvement apparaît déjà nettement dans ce diagramme de classes puisque l'on voit qu'il est possible de spécialiser

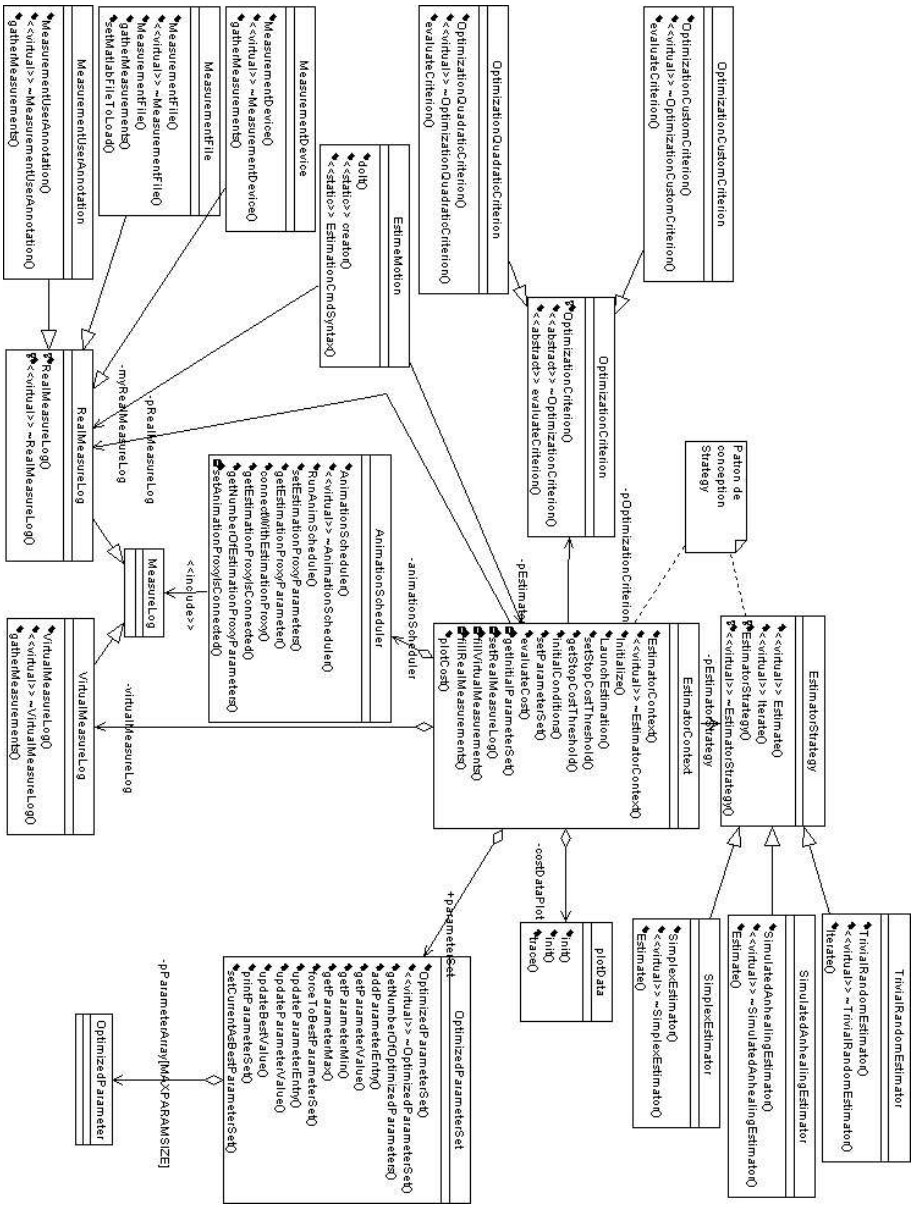

FIG. 11 – *Diagramme de classes du composant réalisant l'estimation.*

l'application au cas par cas par simple spécialisation des classes abstraites. La notion de signature de mouvement mentionnée dans le cahier des charges [2] est ici prise en compte en surchargeant la fonction de coût de manière à définir au cas par cas le niveau de ressemblance entre deux mouvements tel qu'il peut être perçu dans l'espace de la mesure.

3.3 Description dynamique

La figure 12 résume les interactions de l'utilisateur avec le processus d'estimation du mouvement. La partie supérieure du schéma représente les étapes successives de la spécification itérative d'une connaissance a priori par l'utilisateur. La partie inférieure décrit le processus d'estimation, lequel est à son tour détaillé par la figure 13. Le mode opératoire itératif de l'ensemble procède selon trois niveaux :

- Itération manuelle sur l'acquisition. En général, cette acquisition est effectuée une fois pour toutes, mais des acquisitions ultérieures peuvent être nécessaires ou souhaitées.
- Itération manuelle sur la spécification de la connaissance sur le mouvement et la mesure.
- Itération automatique sur l'ajustement des paramètres du modèle.

3.4 Classes permettant la gestion des exceptions

La mise en œuvre du processus d'estimation de mouvement en mode semi-automatique nécessite de prendre en compte différents types d'interruptions. Certaines sont liées à la gestion des différents modes de marche de l'algorithme d'optimisation (initialisation, atteinte de convergence, dépassement du nombre d'itérations autorisées), d'autres sont à l'initiative de l'utilisateur (estimation avortée volontairement parce que le résultat intermédiaire est jugé correct ou, dans le cas contraire, parce que l'utilisateur décide de réinitialiser la recherche après en avoir redéfini les conditions initiales). D'autres exceptions enfin sont liées à la gestion des erreurs susceptibles d'apparaître lors de l'exécution (bogue ou erreur système liée par exemple à l'interruption de la communication avec un module externe). Les fonctionnalités de gestion avancée des interruptions offertes par C++ ont été utilisées à cet effet au niveau de l'implémentation du composant d'estimation. Comme le représente la figure 14, celui-ci définit une hiérarchie de classes d'exceptions spécifiques à notre système.

Dans le cas où des modules externes communiquent avec le composant estimateur et génèrent leurs propres exceptions, un traitement local de ces exceptions est préférable de façon à en isoler l'algorithme d'estimation.

4 Intégration des modèles de mouvement

Conformément à la spécification du cahier des charges [2], le schéma fonctionnel d'un modèle de mouvement est représenté par la figure 15. Un booléen contrôle l'activation du modèle, ce qui permet de réduire le coût calcul lorsque le mouvement est synthétisé à l'aide de plusieurs modèles activés séquentiellement, ou lorsque le modèle est constitué de plusieurs sous-modèles associés à des niveaux de détail différents.

Un modèle de mouvement peut éventuellement ne concerner qu'un sous-ensemble de DDLs pour se concentrer par exemple sur une sous-chaîne articulaire comme le bras. Dans ce cas, le mouvement peut être déterminé par celui associé à d'autres DDLs comme celui du point d'attache de la sous-chaîne (en l'occurrence ici, le mouvement de l'épaule peut affecter celui du bras). Ainsi, le schéma fait apparaître

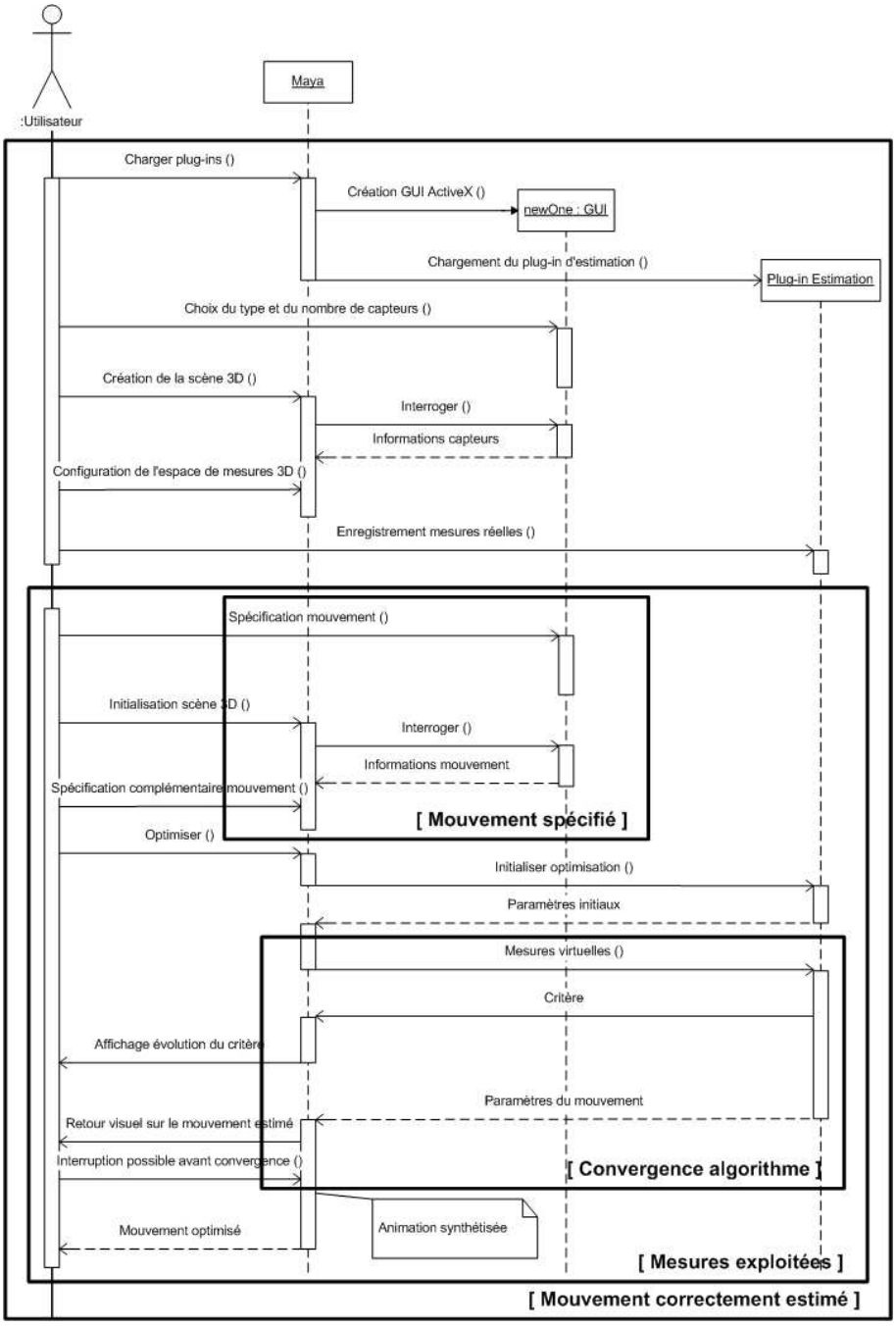

FIG. 12 – *Diagramme de séquence résumant le mode opératoire itératif.*

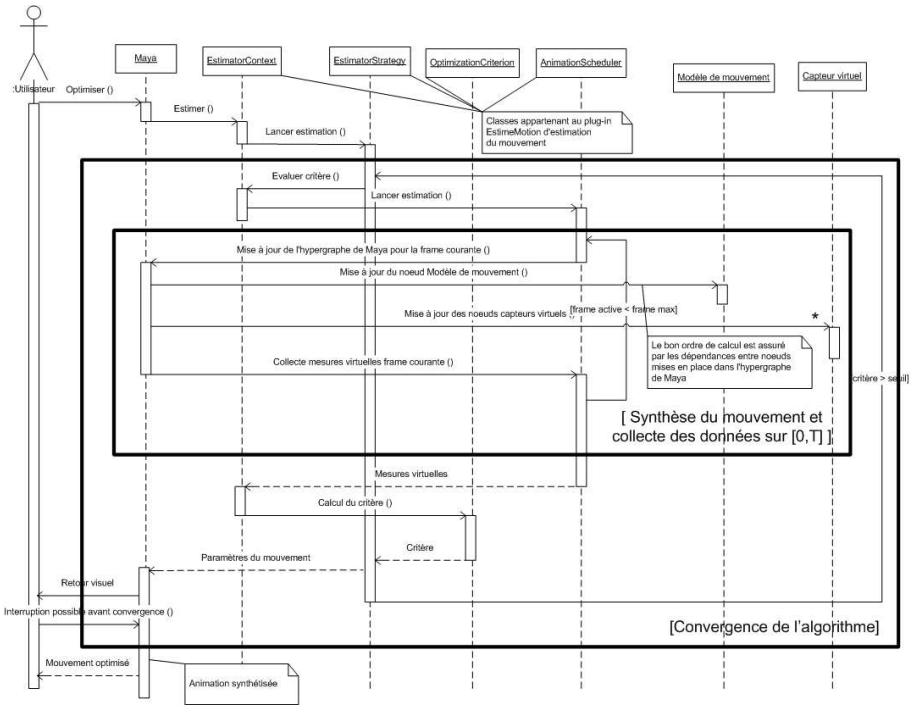

FIG. 13 – *Diagramme de séquence détaillant le processus d'estimation.*

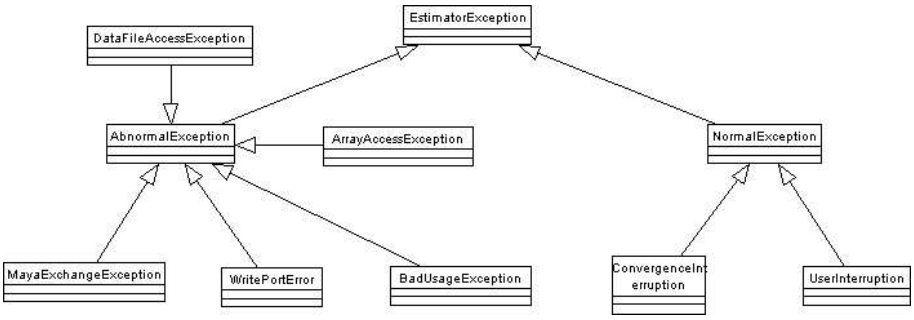

FIG. 14 – *Diagramme de classe des conditions d'exception.*

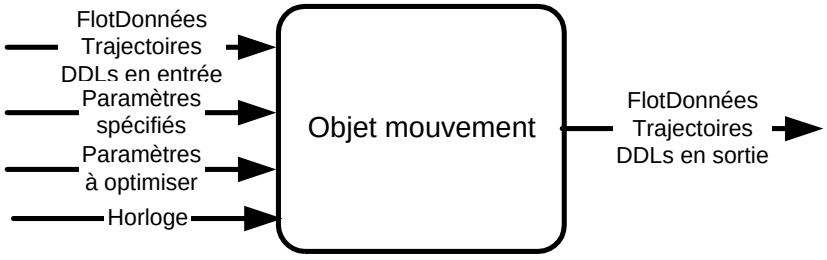

FIG. 15 – *Schéma fonctionnel d'un modèle de mouvement.*

un flot de données en entrée associé à des trajectoires extérieures au modèle de mouvement.

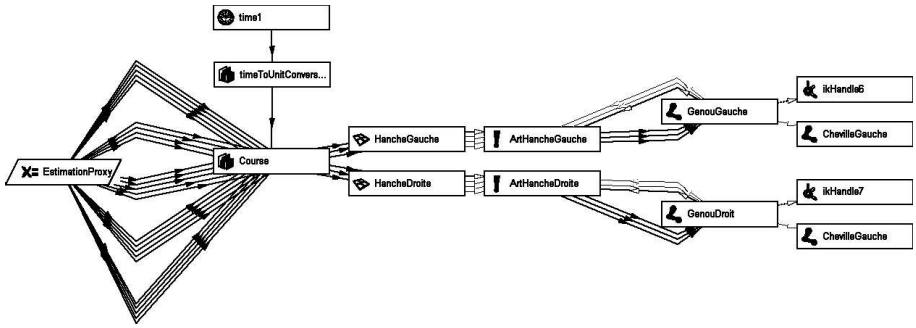

FIG. 16 – *Sous-graphe associé à la synthèse du mouvement de course, mixant des modèles dynamiques et cinématiques (un sous-graphe similaire est associé à la jambe droite).*

A titre d'exemple, le graphe représenté sur la figure 16 montre que la notion de synthétiseur de mouvement est à prendre au sens large, et n'est nullement restreinte à la notion de simulation dynamique. En effet, on voit ici que le graphe de calcul synthétise le mouvement d'un personnage grâce à un assemblage de nœuds d'animation qui intègrent des modes d'animation complémentaires:

- Le graphe contient un modèle procédural et paramétré de course dont on cherche à identifier les paramètres à l'aide d'un processus d'estimation (dont on rappelle que le nœud EstimationProxy joue le rôle de représentant au sein du graphe de la scène). Ce nœud génère les trajectoires des hanches et des chevilles à partir d'un modèle à base de règles issues de connaissances biomécaniques (modèle développé en collaboration avec le laboratoire d'étude de l'exercice musculaire de l'UHB).
- Le graphe contient aussi un modèle d'animation hiérarchique pour gérer la cinématique de chacune des deux jambes (les chaînes cinématiques hanche - genou - cheville).
- Le graphe contient enfin des nœuds de calcul permettant de déduire les rotations des genoux par application d'un algorithme de cinématique inverse.

En conclusion, le graphe de la figure 16 montre que l'on est capable de contraindre le problème de l'identification du modèle de mouvement choisi a priori (ici la course), en incorporant dans le graphe de la scène un ensemble de nœuds d'animation qui représentent autant de contraintes cinématiques, elles aussi connues a priori. On voit donc que l'utilisateur du système garde à sa disposition l'ensemble des outils d'animation traditionnels, c'est à dire basés sur l'interpolation entre images-clefs et la cinématique inverse. En plus des modèles dynamiques et procéduraux, celui-ci peut faire appel à tous les outils classiques de l'animation afin de spécifier la connaissance a priori qu'il souhaite incorporer à la description du mouvement, avant même de faire appel au processus d'estimation pour en retrouver les paramètres inconnus.

4.1 Format standard

On suppose que les mouvements sont échantillonnés avec une période fixe et commune à tous les modèles, indépendamment de leur nature. Généralement cette fréquence sera prise égale à celle des systèmes optiques, soit 60 Hz. Lorsqu'on utilise le logiciel de conception et de simulation de systèmes dynamiques Simulink, un

modèle de mouvement peut être assemblé à partir des blocs standards en respectant les contraintes suivantes :

- Taille des ports d'entrées et sorties : éventuellement les entrées et sorties vectorielles peuvent être utilisées à l'intérieur du modèle mais elles doivent être multiplexées (resp. dé-multiplexées) au niveau de ses ports de connexion externes.
- Type des paramètres d'entrée : toutes les données appliquées aux ports d'entrée du modèle sont des flottants 32 bits. Aucune distinction n'est faite entre les entrées mises à jour à chaque pas de temps, et les entrées correspondant à des paramètres constants sur toute la durée du mouvement. De même, il n'y a aucune distinction entre les paramètres spécifiés par l'utilisateur, les constantes vraies, et ceux susceptibles d'être modifiés par l'algorithme d'optimisation. La nature " spécifiée " ou " optimisée " d'un paramètre peut donc être déterminée librement par l'utilisateur ou modifiée dynamiquement à l'aide d'un script. Du point de vue du concepteur du modèle de mouvement, cela signifie aussi que les plages de variation autorisées pour les différents paramètres d'entrée ne peuvent être garanties à moins d'associer une opération de seuillage à l'intérieur des bornes autorisées pour chacun des ports d'entrée.
- Type des sorties : toutes les données issues des ports de sortie du modèle sont des flottants sur 32 bits. Ces sorties sont sans type au sens où aucune distinction n'est faite par exemple entre les DDLs en translation et les DDLs en rotation. C'est pourquoi on prendra soin de nommer les nœuds avec des préfixes explicites quant à la nature du DDL concerné, afin de faciliter la tâche de l'utilisateur qui souhaite éditer le graphe de la scène manuellement plutôt qu'à l'aide de scripts. Par convention également, on considère que les paramètres de rotation doivent être exprimés ou convertis en degrés avant d'être appliqués sur un port de sortie du modèle.
- Discrétisation temporelle du modèle à la période d'échantillonnage normalisée. On suppose donc que chaque modèle continu est préalablement discrétisé temporellement. A cet effet, on peut remplacer les opérateurs de dérivation temporelle par des différences finies. De même, on peut remplacer les intégrations temporelles par des incrémentations sur un registre.
- Réinitialisation de l'état interne du modèle implicite du point de vue du système. Le processus de génération de composants de type mouvement rajoute au modèle une entrée horloge, dont la remise à zéro par le contrôleur d'animation du logiciel d'accueil provoque la réinitialisation du vecteur d'état du modèle.

Des exemples de modèles de mouvement créés avec Simulink sont représentés sur les figures 17 et 18. Ces exemples font apparaître des flots de données. D'un autre côté, la gestion des événèments est la prise en compte d'automates d'état peut être prise en compte à l'intérieur de ces modèles.

4.2 Génération de code

Par invocation de nos outils (basés sur l'atelier de génération de code de Simulink [5]) des composants directement enfichables dans Maya sont compilés à partir des modèles de mouvement spécifiés à l'aide d'un progiciel commercial dédié à la conception et à la simulation de systèmes dynamiques [4]. Le processus de génération de code est résumé par la figure 19. A l'instar de la solution retenue pour les modèles de capteurs, les fonctions du code généré sont réentrantes, ce qui signifie que plusieurs instances du modèle peuvent coexister à l'exécution, ce qui laisse ouverte la possibilité de travailler avec des mouvements composés à partir de plusieurs instances initialisées avec des paramètres différents (voir ci-dessous).

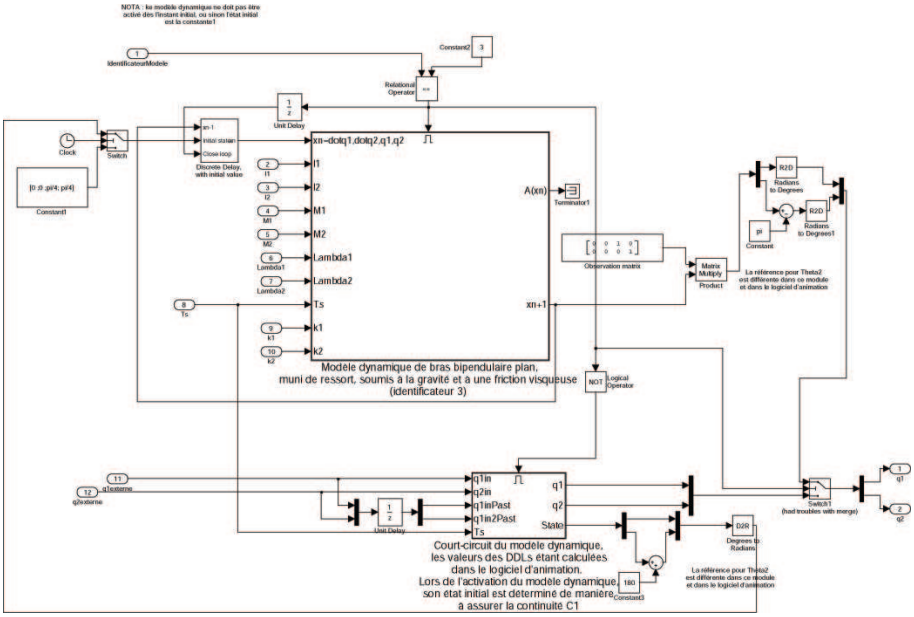

FIG. 17 – *Modèle dynamique, échantillonné, de bras créé avec Simulink.*

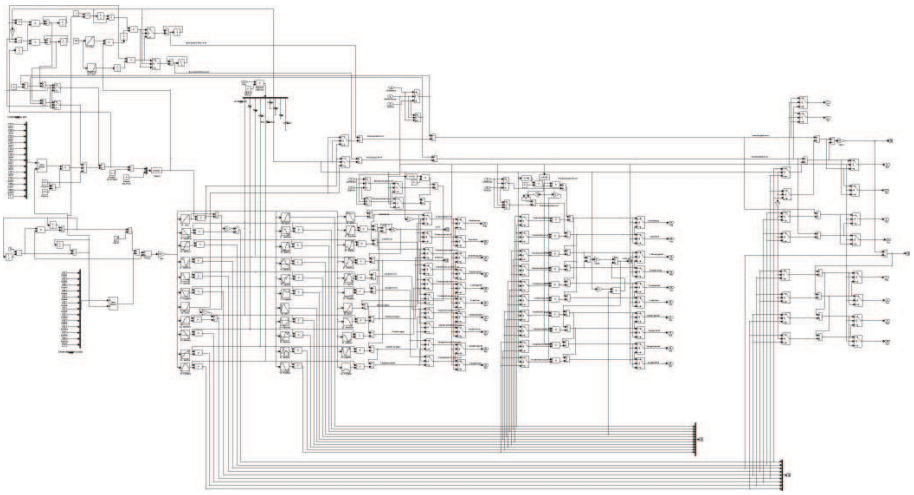

FIG. 18 – *Modèle procédural de course créé avec Simulink.*

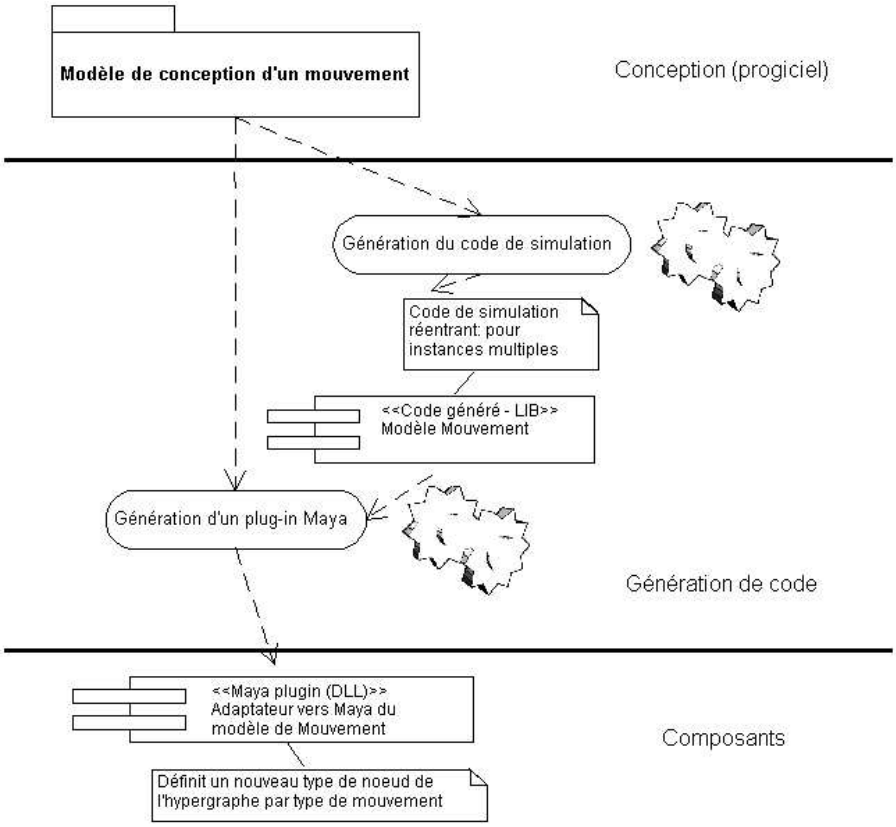

FIG. 19 – *Processus de génération des composants de mouvement.*

4.3 Mouvements composés

Ainsi que nous l'avons mentionné dans le cahier des charges [2], la notion de mouvement complexe décomposable en mouvements de base reste un domaine de recherche ouvert. Pour explorer cette voie, le logiciel fournit des mécanismes élémentaires de composition de mouvement basés sur les fonctionnalités de construction du graphe de scène du logiciel d'accueil. Il suffit ainsi d'établir les connections entre les différents nœuds associés à des mouvements élémentaires dans l'hypergraphe de la scène. La composition de modèles de mouvements associés à des DDLs différents est triviale, la composition de mouvement s'exécutant séquentiellement dans le temps peut être réalisée en validant ou invalidant l'activation des nœuds concernés, ce qui suppose aussi l'écriture d'un nœud de script assurant cette orchestration. Dans tous les cas, cela suppose que l'utilisateur a une connaissance du logiciel significative, lui permettant de réaliser des connections manuellement, ou d'écrire des scripts MEL afin d'automatiser la création de tels graphes composés.

5 Intégration des modèles de capteurs virtuels

On rappelle que la vocation d'un capteur virtuel est de fournir une mesure de référence issue d'un capteur idéal placé dans la scène de synthèse. C'est pourquoi les capteurs virtuels simulent les capteurs réels au niveau fonctionnel et non pas au niveau physique. Ainsi les anomalies des caméras et le bruit de mesure ne sont pas pris en compte dans la simulation.

5.1 Format standard

Au niveau fonctionnel, une mesure correspond à un nœud du graphe de la scène virtuelle, avec en entrée un flot de données échantillonnées à la fréquence nominale correspondant aux trajectoires du capteur matériel dans la scène, elles-mêmes déterminées en fonction des trajectoires du squelette du personnage. Ainsi l'élaboration de la mesure correspond typiquement à la mise en série d'un objet capteur et d'un objet mesure. La figure 20 illustre cette structure. Certains capteurs complexes comme les capteurs optiques impliquent un réseau d'objets plus complexe reliés par des flots de données intermédiaires. Généralement les flots de données en entrée de la mesure proviennent d'objets attachés au squelette en mouvement pour matérialiser des marqueurs ou des capteurs. En sortie d'un nœud de mesure, un flot de donnée (scalaire ou vectoriel), échantillonné lui-aussi à la fréquence nominale, représente la mesure virtuelle (mono ou multidimensionnelle).

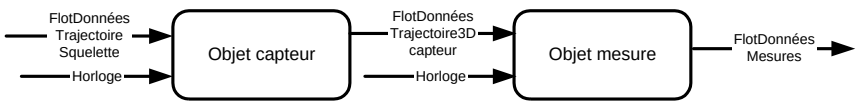

FIG. 20 – *Modèle de flot de données d'un capteur.*

Pour davantage de flexibilité (déclenchement à une date donnée), l'horloge générale du système fait aussi partie des entrées d'un objet capteur.

Pour des raisons de simplicité de mise en œuvre, on considère que tous les systèmes de mesure virtuels fonctionnent à une fréquence d'échantillonnage commune, identique à celle utilisée pour la simulation du mouvement. Généralement, cette fréquence sera prise égale à celle des systèmes optiques, c'est à dire 60Hz. La comparaison avec les mesures réelles présuppose donc une opération de décimation (ou plus généralement un changement de fréquence par décimation-interpolation) associée à un filtrage passe-bas des mesures acquises.

5.2 Génération de code

Grâce aux outils de génération de code employés (basés sur l'atelier de génération de code de Simulink [5]), des composants directement enfichables dans Maya sont compilés à partir des modèles de capteurs spécifiés à l'aide d'un progiciel commercial dédié à la conception et à simulation de systèmes dynamiques [4]. La figure 21 résume le processus de génération de code. A l'instar de la solution retenue pour les modèles de mouvement, les fonctions du code généré sont réentrantes ce qui signifie que plusieurs instances du modèle peuvent coexister à l'exécution, ce qui est en général une nécessité.

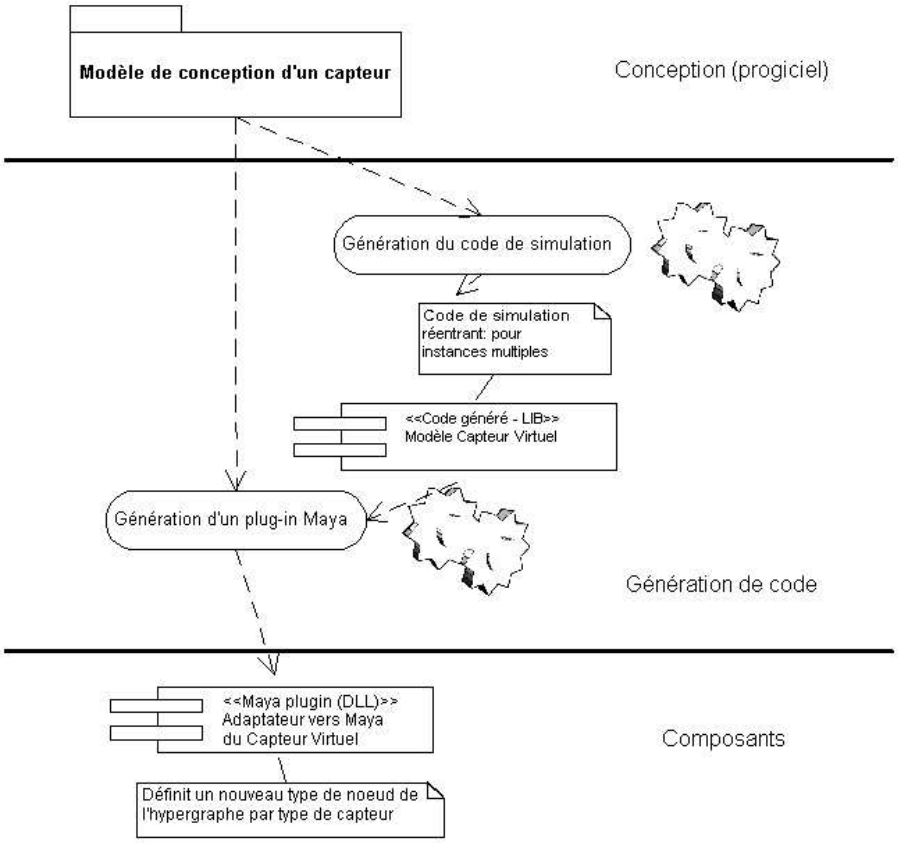

FIG. 21 – *Processus de génération d'un composant de capteur virtuel.*

La liste des capteurs utilisables avec le sous-système d'estimation du mouvement est théoriquement extensible. En principe, celui-ci est donc susceptible d'exploiter des sources diverses, allant du simple détecteur sonore jusqu'à une information sémantique décrite par des annotations en langage naturel par l'utilisateur. Pour les besoins du projet, les capteurs utilisés dans le prototype final comportent des capteurs optiques, magnétiques et des accéléromètres. La simulation de ces différents capteurs au niveau fonctionnel est prise en compte de la façon suivante :

- La simulation des capteurs optiques implique l'objet mesure optique - qui correspond à une projection dans le plan image, mais également des objets auxiliaires qui sont les objets marqueurs et les objets caméras, ainsi que le montre la figure 22. Les objets marqueurs sont attachés au squelette du personnage virtuel et leur trajectoire dans le repère global de la scène est transmis comme un flot de données à un objet réalisant la projection dans le plan image de la caméra. Les paramètres de la caméra peuvent être connus a priori mais

dans le cas général, leur évaluation automatique nécessite une phase de calibration. Les fonctions de calibration évoluées, notamment celles autorisant un suivi temporel relèvent des compétences de Realviz.

- La simulation des accéléromètres repose sur un modèle mathématique mettant en jeu les forces d'interaction entre la base du capteur et une masselotte mobile. Ces forces comprennent des facteurs d'inertie dont il faut s'affranchir afin de dériver l'accélération de la base du capteur en fonction des grandeurs électriques de sortie. Les équations du comportement du capteur sont fournies par le CEA-LETI.
- La simulation des magnétomètres sera basée sur un modèle simple permettant de dériver à chaque pas de simulation l'orientation du capteur dans le repère global. Les inhomogénéités du champ magnétique pourront éventuellement être prises en compte par un échantillonnage spatial tabulé et interpolé.

5.3 Cas particulier des acquisitions optiques

On présente ici quelques graphes de scènes comprenant des capteurs optiques virtuels qui sont les plus complexes à mettre en œuvre en termes de nœuds et de connexions. La figure ci-dessous représente le graphe associé à un capteur optique et une seule caméra.

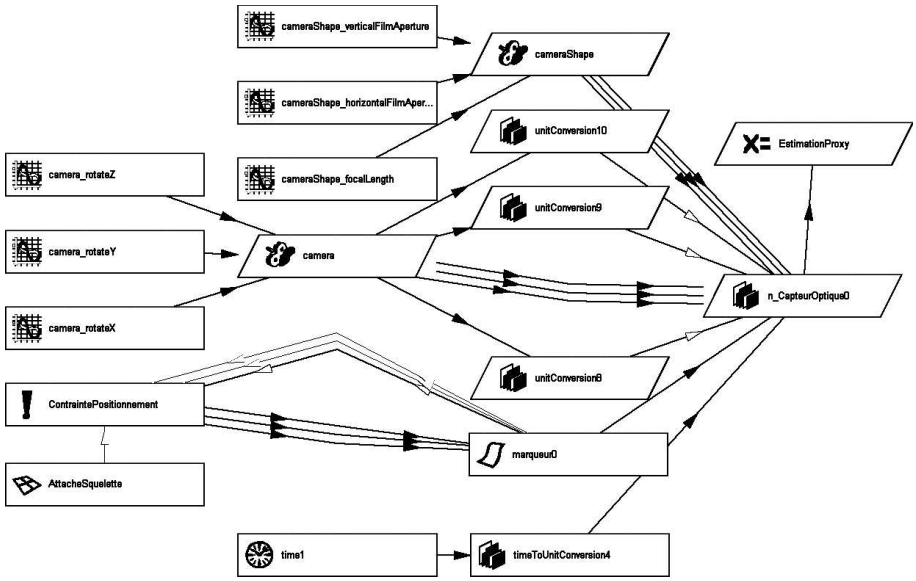

FIG. 22 – *Sous-graphe d'une scène virtuelle associé à un seul marqueur optique en vision monoculaire.*

Il est à noter que les paramètres de la caméra, qu'ils soient intrinsèques (distance focale) ou extrinsèques (position et orientation) sont susceptibles d'évoluer dans le temps. Ainsi, le graphe représenté ci-dessus contient des nœuds qui définissent le mouvement et éventuellement le facteur de zoom de la caméra à l'aide d'une interpolation entre des clefs par des courbes splines (nœuds de base fournis par Maya pour l'animation). Bien que rien n'empêche de considérer les paramètres de ces nœuds comme autant de paramètres à optimiser, donc susceptibles d'être concaténés avec ceux relatifs à l'estimation du mouvement, le coût calcul serait rédhibitoire. On doit donc considérer que le mouvement de la caméra a déjà été calculé lorsque l'on initialise le processus d'estimation, ce qui suppose que les trajectoires correspondantes sont chargées au préalable. Un outil tel que MatchMover de Realviz peut à cet effet

être utilisé en amont du système pour déduire les trajectoires de la caméra par analyse des séquences vidéo, et les exporter sous un format de fichier compatible pour une importation dans Maya.

Le graphe de la figure 23 montre comment le graphe de la figure 22 peut être modifié de manière à prendre en compte plusieurs marqueurs. De la même façon, il suffirait de rajouter un nœud de type caméra pour une vision stéréoscopique.

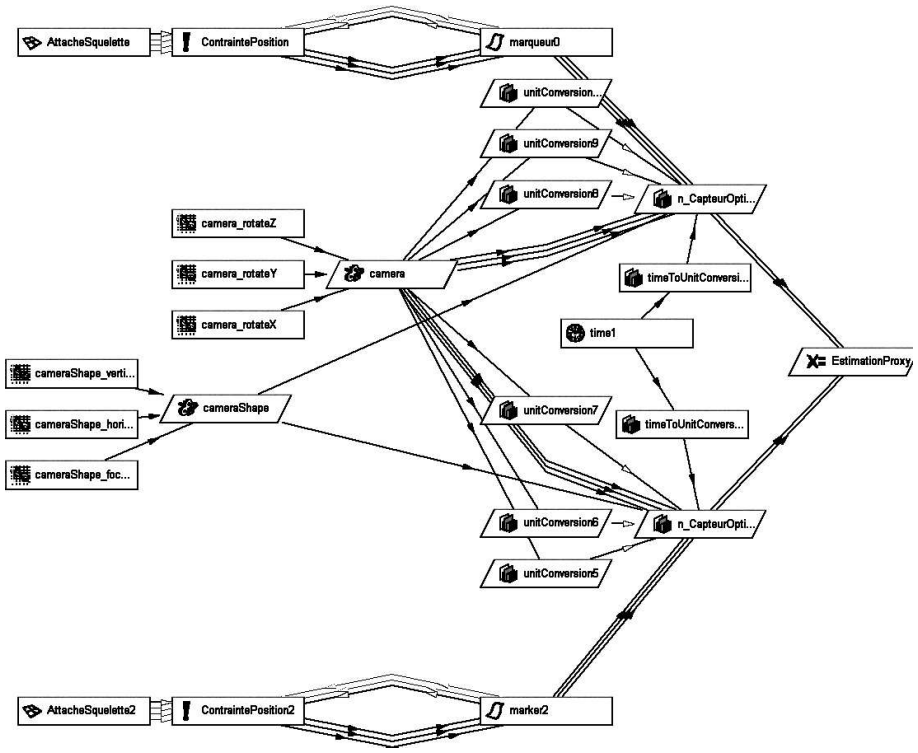

FIG. 23 – *Sous-graphe associé à deux marqueurs optiques et une caméra.*

Il convient de rappeler que dans la pratique, ces graphes ne sont pas créés manuellement par l'utilisateur (possible mais fastidieux), mais construits à l'aide de scripts MEL à partir des spécifications entrées par l'utilisateur dans un GUI spécifique lui permettant de définir la nature et le placement des capteurs.

6 Intégration des pilotes logiciels des matériels de mesure

L'implémentation du sous-système d'estimation comprend un certain nombre de modules complémentaires, qui s'intègrent dans le logiciel d'animation. Cette intégration va au delà d'un simple interfaçage logiciel au niveau d'une API. Elle doit aussi prendre en compte la cohérence des modes opératoires au sein du système d'accueil.

6.1 Format des sources de mesure

Indépendamment du fait que le système permette déjà de combiner différents types de capteurs afin d'en réaliser une fusion des informations, l'architecture prévoit aussi que les mesures puissent être enregistrées selon trois formats complémentaires :

- Fichier d'enregistrement des mesures synchrones et échantillonnées à la fréquence nominale (fréquence vidéo).
- Serveur de mesure centralisant les mesures des différents pilotes pour les matériels d'acquisition et de traitement et communiquant avec Maya par l'intermédiaire d'une socket.
- Annotations de l'utilisateur au niveau du GUI (transmises via COM).

Quelque soit le format choisi, les mesures peuvent être issues de capteurs de natures différentes (optique, magnétique ou accéléromètre) et l'on travaille toujours à la fréquence nominale (en ayant préalablement réalisé toute éventuelle transposition de la fréquence d'échantillonnage par le biais d'une opération de décimation-interpolation). Le diagramme de classes représenté sur la figure 24 permet de découpler le processus d'estimation des choix d'implémentation concrets liés à l'acquisition des mesures réelles. La classe MeasurementDevice permet d'associer le flot des mesures réelles à un serveur externe qui centralise les mesures issues des différents pilotes des périphériques et les transmet au système au travers d'une communication par socket. Pratiquement la connexion d'une instance de cette classe à un serveur de mesures externes peut être réalisée à l'aide d'un script MEL.

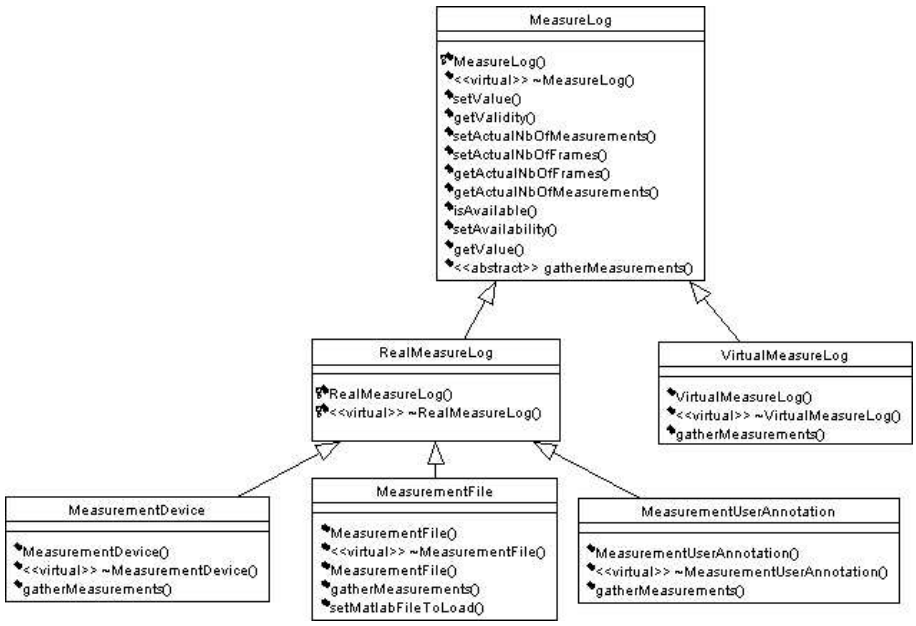

FIG. 24 – *Hiérarchie des classes utilisées pour les mesures.*

Enfin, il faut noter que les classes concrètes associées aux trois différents formats de mesure sont actuellement disjointes. Ainsi, la notion de format composite (permettant de combiner un dispositif de mesure avec des annotations de l'utilisateur) pourra ultérieurement nécessiter la mise en œuvre d'un héritage multiple.

6.2 Déploiement au sein du système intégrant un serveur de mesure

L'architecture logicielle proposée a vocation à être intégrée au cœur d'un système complet comprenant des équipements matériels pour l'acquisition et la mise en forme de la mesure. Pour découpler les tâches d'acquisition (qui nécessitent une gestion " dure " du temps-réel) de celles du système d'estimation de mouvement (qui peut se contenter d'un temps réel " mou " permettant l'interactivité), des processus séparés, voire des processeurs distincts sont idéalement affectés aux pilotes des différents périphériques d'acquisition et au programme responsable de leur orchestration. La figure 25 représente le diagramme de déploiement du système visé. En amont de la chaîne d'acquisition, un PC centralise les acquisitions et effectue les pré-traitements à partir desquels un flot de mesures synchrones et échantillonnées à la fréquence nominale est transmis au PC en aval. Idéalement le PC amont exécute un OS temps-réel tandis que le PC aval exécutera Windows2000. Bien que ces problèmes d'intégration en aval avec les périphériques d'acquisition de la mesure sortent du cadre de ce document, on peut noter que le choix d'utiliser Maya comme logiciel d'accueil facilite la mise en œuvre du diagramme de déploiement de la figure 25 en apportant une bibliothèque de communication par socket [3] que l'on pourra spécialiser pour notre application.

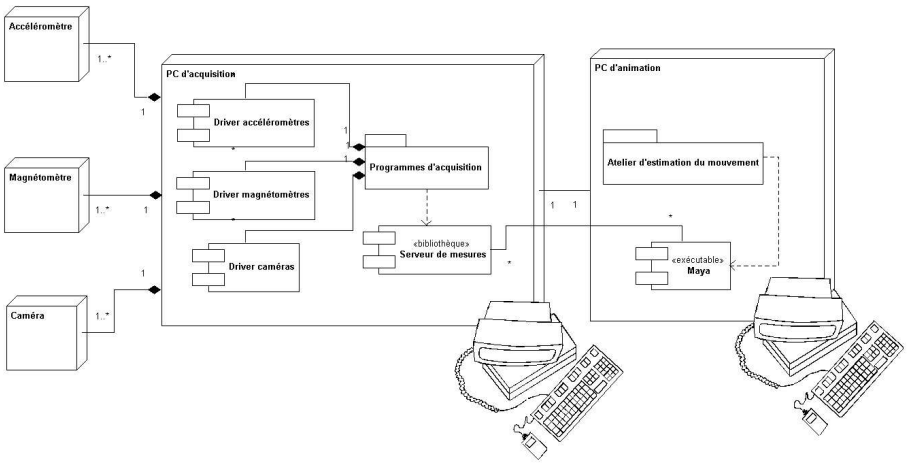

FIG. 25 – *Diagramme de déploiement au sein de l'architecture matérielle.*

7 Intégration avec le logiciel d'analyse d'images

Dans le cadre du projet Mouvement, l'intégration avec le logiciel de Realviz est potentiellement complexe et reste à définir : au delà de l'aspect purement informatique, se pose aussi la question de la cohérence des modes opératoires. L'objectif de cette section est d'abord de suggérer les objectifs que l'on souhaite se donner en termes de services mutuellement offerts entre les deux logiciels. Une fois que ceux-ci auront été fixés, plusieurs solutions sont envisageables en ce qui concerne concrètement l'interfaçage des deux logiciels. Si les traitements des mesures optiques sont prépondérants en temps calcul par rapport à la durée d'une itération de l'algorithme d'estimation, alors une communication par simple échange de fichiers sera suffisante en termes de performances. Pour référence, les mesures de performances obtenues sur le prototype 2.0 avec un modèle de course simplifié sont de l'ordre de 500 itéra-

tions à la minute, chaque itération impliquant la synthèse complète de la séquence de mouvement sur 100 pas de temps.

7.1 Exploitation des mesures issues du logiciel d'analyse d'images

Le logiciel de Realviz peut fournir deux types d'information au système d'estimation du mouvement :

- Trajectoires et évolution des paramètres de la caméra : un logiciel tel que MatchMover étant capable de retrouver le mouvement de la caméra à partir d'une analyse vidéo et d'en exporter les trajectoires au format des animations Maya, l'écriture d'un simple script MEL (réalisant le chargement des fichiers et l'application des trajectoires aux nœuds d'animation concernés) pourrait suffire à exploiter ce service.
- Trajectoires 3D des marqueurs optiques : ces trajectoires peuvent être transmises à l'intérieur du flot vectoriel des mesures réelles avant de lancer le processus d'estimation des paramètres du mouvement. Dans ce cas une simple transmission par fichier peut s'avérer suffisante. Alternativement, on peut aussi envisager, comme on le fait dans la section suivante, le cas ou le logiciel de Realviz exploite aussi les résultats du processus d'estimation pour raffiner par étapes successives son analyse de la vidéo. Dans ce cas, ont peut envisager que les informations fournies par le logiciel de Realviz soient périodiquement rafraîchies et transmises. L'intégration de cette transmission au niveau du serveur de mesure est envisageable. Dans le cas contraire, il faudrait mettre en œuvre un mécanisme de notification des mises à jour pour déclencher par exemple un chargement des fichiers de mesure correspondants au niveau de Maya.

7.2 Remontée des mesures virtuelles pour l'analyse d'images

Le modèle de caméra virtuelle déjà implémenté dans le prototype 2.0 est capable de synthétiser les trajectoires 2D de chaque marqueur dans le plan image des différentes caméras, en fonction du mouvement synthétique associé au jeu de paramètres courant. Dans ce cas, cela suppose que les trajectoires 2D de chaque marqueur telles qu'elles sont vues par la (ou par les différentes) caméra(s) sont aussi fournies par le logiciel de Realviz afin d'être ensuite injectées en entrée du système d'estimation. Dans le cadre du service d'assistance à la mesure tel qu'il est proposé dans le cahier des charges [2], il serait alors envisageable d'exploiter la connaissance des trajectoires synthétisées afin d'accélérer la recherche des patrons dans les images acquises en localisant la fenêtre de recherche autour de ces trajectoires de référence.

Ces coopérations entre les deux logiciels peuvent se faire soit par échange de fichiers, soit par l'intermédiaire d'un serveur de mesure basé sur la bibliothèque de communication par socket avec Maya. Dans les deux cas, la classe spécialisée qui est responsable de l'acquisition de la mesure pourra se contenter d'invoquer un script MEL pour gérer le transfert de la source de données externes vers les structures internes utilisées par la classe responsable.

Références

1 The Unified Software Development Process. I. Jacobson, G. Booch, J. Rumbaugh. Addison-Wesley. 1999

2 Cahier des charges du sous-système d'estimation de mouvement basé sur l'utilisation de modèles de mouvement et sur la fusion des informations capteurs

CDC-SP3 /4. V1.0. Jean-Luc Nougaret, Armel Cretual et Mathilde Vandenberghe. 5 octobre 2002. Document interne. 5 octobre 2001

3 Documentation de la boîte à outils de développement de Maya. V3.01.Alias|Wavefront. 2001

4 Documentation de Matlab 6.0 (R12)/ Simulink. V4.0 (R12). The Mathworks Inc. 2001

5 Documentation du Real-time Workshop pour Simulink. V4.0 (R12). The Mathworks Inc. 2001

Table des matières

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399