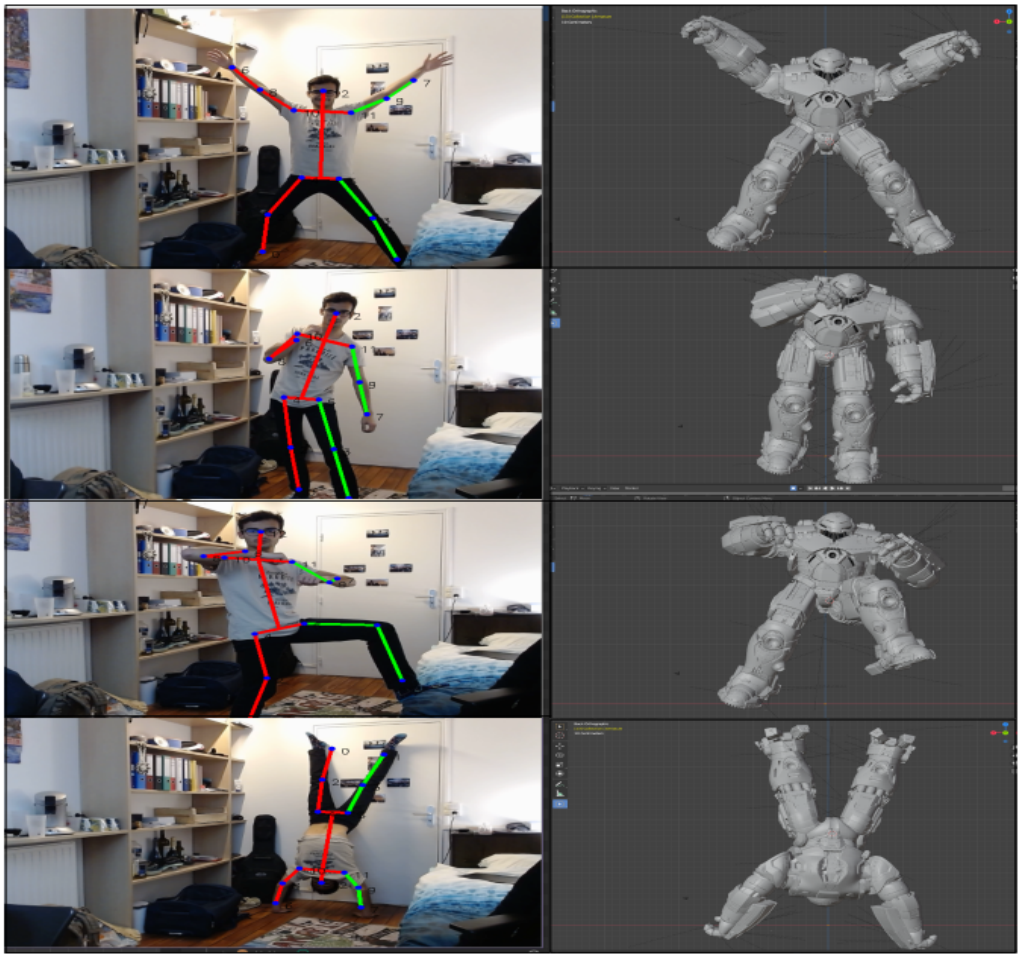
Tom Marty
Pierre Tessier

Dimanche 13 décembre 2020

PROJET INF 573
REALTIME 3D DEEP MOTION CAPTURE



1 Présentation du projet

Nous nous sommes donnés pour objectif d'implémenter un algorithme capable de relier les mouvements d'une personne au mouvement d'un modèle 3D à partir d'une vidéo 2D donnée en entrée. Ce domaine d'application s'appelle la motion capture, il est particulièrement sollicité par l'industrie du jeux vidéo ou du cinéma et permet de faire gagner beaucoup de temps d'animation aux infographistes. Le but final est de générer un ensemble de keyPoints 3D au cours du temps et localisés au niveau des articulations du personnage, cet ensemble de KeyPoints s'appelle un **rig**.

On retrouve un ancêtre du Motion capture appelé la chronophotographie dès le début du XXème siècle, mais le Motion Capture tel qu'on le connaît aujourd'hui s'est démocratisé à partir des années 2000. La plupart des techniques modernes nécessitent un système de capture vidéo très cher et encombrant utilisant plusieurs dizaines de caméras. Notre idée est donc d'implémenter un système de motion capture intelligent basé sur de l'apprentissage profond et n'utilise que le flux vidéo d'une webcam.

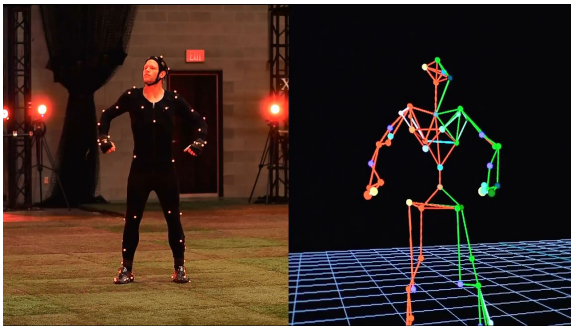


FIGURE 1 – motion capture basé sur 12 caméras.png

Il existe de nombreuses implémentations d'algorithmes d'estimation de posture à partir d'un flux vidéo, mais la plupart d'entre eux ne génère qu'une posture 2D. Dans notre cas, nous souhaitons obtenir en sortie une posture 3D afin de pouvoir animer un modèle 3D, appelé mesh dans un espace 3D. Il s'avère que les techniques d'estimation efficace de posture 3D à partir de vidéo 2D ne sont apparues que très récemment, grâce aux progrès des techniques d'apprentissage profond.

2 Détails de notre implémentation

Notre projet se divise en 2 grandes parties. La première consiste à déterminer les KeyPoints 3D en fonction du temps à partir d'un flux vidéo entrant. La deuxième partie du projet consiste enfin à utiliser ce rig et le lier à un mesh 3D sur un logiciel de rendu 3D.

Pour la première partie, nous avons décidé d'utiliser *DOPE : distillation of part experts for whole-body 3D pose estimation* . Il s'agit d'un algorithme capable de générer un squelette 2D et 3D d'une personne filmé. Sa particularité est de fournir également un squelette 3D des mains et du visage. L'algorithme se base sur un en ensemble de sous algorithmes spécialisés dans la détection respective du corps, des mains et du visage.

Pour la partie de rendu 3D, après étude des différentes solutions s'offrant à nous, nous avons décidé de travailler dans le logiciel de rendu 3D *Blender*. Il a la particularité d'avoir une API Python assez riche, et de gérer très bien les fichiers 3D pré-riggé *.FBX*. Cela nous permet de pouvoir importer un mesh 3D possédant déjà un rig relié à son mesh 3D et dont la géométrie est assez proche de celle d'un humain. Pour nos résultats, nous travaillons avec un mesh pré-riggé de *Iron Man* trouvé sur internet.

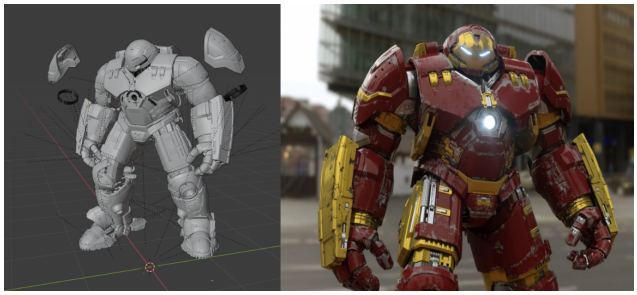


FIGURE 2 – Mesh 3D d'*Iron Man* Buster proposé par le créateur dannysgallegos sur la plateforme Sketchfab

Travailler avec un logiciel haut niveau tel que Blender possède à la fois des avantages et des inconvénients. S'il est clair que la représentation et la préparation des meshs 3D pré-riggés est plus simple, il est toujours plus compliqué d'implémenter une features dans un tel logiciel que dans une librairie de bas niveau tel que *VCL*. La prise en

main de l'API est également un facteur à prendre en compte dans le choix de la solution.

2.1 Estimation de posture 3D

Pour estimer la posture 3D d'une personnage filmé nous utilisons l'algorithme *DOPE*. Cet algorithme fait appel à un ensemble de sous algorithmes spécialisés dans la détection du corps, des mains et du visage. Chaque sous-algorithme d'estimation du corps, du visage et des mains est entraîné sur un dataset spécifique et qui est propre à sa fonction. *DOPE* se charge de sélectionner les portions d'images pertinentes pour chacun des sous-algorithmes. *DOPE* se charge de localiser grossièrement les mains et le visage dans l'image, puis d'envoyer une requête locale qui ne contient qu'une partie spécifique de l'image à chacun des algorithmes experts pour estimer la position 3D locale des éléments de la requête. *DOPE* se charge ensuite de tout regrouper pour proposer une estimation du rig 3D, main et visage compris.

FIGURE 3 – Photo en extérieur et rendu 3D par *DOPE*

Si la position relative des agents entre eux semble assez approximative, la posture 3D de chaque personnage est assez fidèle et robuste à l'occlusion.

Une fois que *DOPE* tournait parfaitement sur nos machines, il nous as fallu comprendre le code et la logique derrière *DOPE* afin de pouvoir extraire les composantes utiles pour continuer notre projet. Ces étapes sont parfois ingrates, car nous devons comprendre le code d'une tierce personne qui ne communique souvent pas de documentation à son sujet.

Nous avons tout de même réussi à nous approprier assez bien le code du projet, pour nous débarrasser de nombreuses étapes de calculs intermédiaires peu pertinentes pour notre projet, comme le calcul précis des mouvements du visage. Nous avons également dépouillé le code de toutes ses fonctions de capture et de visualisation pour implémenter les nôtres à la place.

In fine, le code original nous sert exclusivement à calculer les positions 3D des keypoints dans un repère abstrait, et à calculer les positions des keypoints 2D sur l'image pour l'affichage en temps réel, en parallèle de *Blender*.

2.2 API Blender

Une grosse partie du travail a consisté à implémenter un script Python de communication en temps réel entre *DOPE* et *Blender*. L'API blender étant très riche, la très grande liberté d'action de l'API était contre-balancée par un formalisme assez opaque et qui nécessitait un certain niveau de pratique pour son utilisation.

Le script final devait être capable de :

- Démarrer le flux caméra et récupérer notre rig 3D généré par *DOPE* à chaque frame de calcul
- Instancier un Objet *Blender* de type Operator capable d'effectuer des modifications sur les autres objets de Blender.
- Convertir le rig généré par *DOPE* pour l'adapter au rig de notre mesh 3D en utilisant un ensemble de quaternions bien déterminés.
- Appliquer ces transformations à notre rig *Blender* en temps réel par l'ajout de Keyframes sur la timeline de *Blender*.

Cette partie fut particulièrement laborieuse étant donné notre peu de connaissance sur l'API. Cela était d'autant plus complexe car nous souhaitions avoir un rendu en temps réel, ce qui nous a obligé à travailler avec des objets bien spécifique de *Blender*.

L'API se sépare en deux partie : une partie Setup où l'on définit un objet de type *WorkSpaceTool* qui n'est autre qu'un bouton d'enregistrement qui appelle un autre script. Une partie Loop où l'on défini un objet de type *Operator* qui sera appelé par le *WorkSpaceTool* précédemment créé. Lorsque l'on appuie sur le bouton d'enregistrement, celui-ci lance une fonction d'exécution de l'opérateur qui lance le script *DOPE*, récupère les Keyframes en boucle, effectue la bijection entre

les Keyframes d'entrée et le rig d'Iron Man, transforme le rig à l'aide des quaternions calculés, et déplace le mesh en conséquence.

2.3 Couplage des Rigs

Afin de pouvoir animer notre Iron Man, il est nécessaire de pouvoir établir une bijection entre le rig donné par *DOPE* et le rig de l'Iron Man. En effet on verrait mal utiliser un rig humanoïde pour animer une araignée à 8 pattes. Un des critères d'utilisation du mesh d'Iron Man était donc d'avoir un rig avec le même nombre d'articulation, la même topologie que celui généré par *DOPE*. Ci-dessous, une comparaison des deux rigs que nous utilisons.

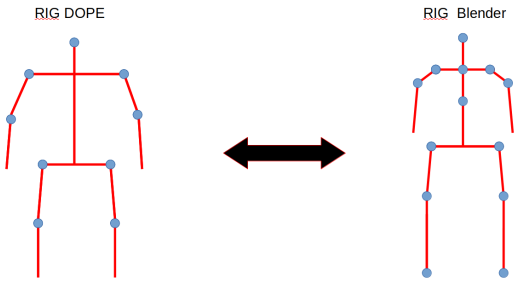


FIGURE 4 – Rig généré par *DOPE* et rig du mesh d'Iron Man

Si la topologie des deux meshs doit correspondre, nous avons néanmoins une certaine marge de manoeuvre sur la forme générale des rigs. A partir de notre simple rig composé d'un tronc, de deux bras et de deux jambes, nous sommes logiquement capable d'animer n'importe quel rig possédant un tronc, deux bras et deux jambes mais dans des proportions différentes. Si toutefois les proportions sont trop éloignées, la démarche - humaine - du mesh 3d finirait par perdre en réalisme car ne serait pas en accord avec la physiologie du personnage que l'on souhaite animer. Ci-dessous, des exemples de meshs 3D humanoïdes qui conviennent parfaitement à notre méthode.

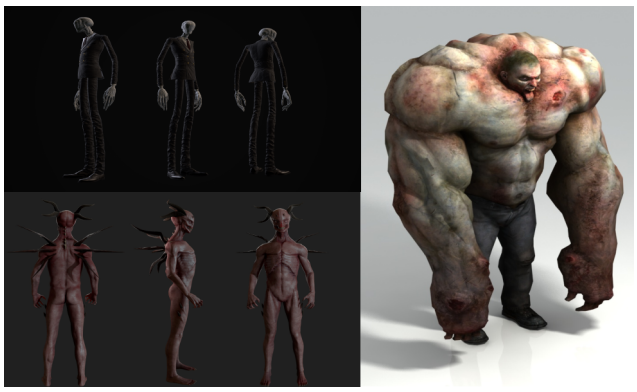


FIGURE 5 – 3 exemples de meshs topologiquement identiques mais géométriquement éloignés

La prochaine étape consiste alors à appliquer les bonnes transformations sur le rig d'Iron Man, à partir des transformations observées sur le rig *DOPE*. Nous avons décidé dans un premier temps de nous contenter des rotations entres les différentes parties du rig hiérarchique, pour modéliser les rotations nous utilisons des quaternions, qui sont déjà implémentés dans Blender. L'interpolation de deux quaternions est relativement aisée, et on pourrait ainsi à l'avenir rendre le mouvement de notre personnage plus fluide entre deux frames générées par *DOPE* à l'aide de la formule suivante :

$$q_t = q_0 * (q_0^{-1} q_1)^t$$

Avec q_0 et q_1 les deux quaternions à interpoler, et t le scalaire d'interpolation entre 0 et 1.

Cependant, cette partie du projet ne fut pas simple en pratique. Des différences de repères entre *DOPE*, *Blender* et même au sein des différentes interfaces de *Blender*, ainsi qu'un bug identifié sur l'implémentation des quaternions ont rendu le travail très laborieux.

Le rig renvoyé par *DOPE* est assez sommaire. Il ne s'agit que d'un ensemble de points reliés entre-eux. Ils n'y a aucune notion de repère local permettant de définir des rotations le long d'une articulation. Nous avons donc dans un premier temps défini une position de base du mesh et fixé à l'identité toute les rotations le long des articulations du rig. Cette position est appelé *T-Pose* et constitue la position de référence largement utilisée dans le milieu de l'infographie 3D.

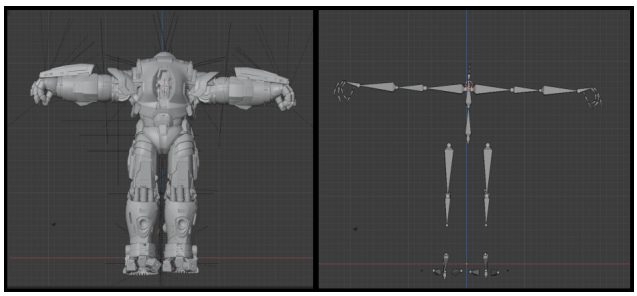


FIGURE 6 – mesh 3D d'Iron Man en T-pose / rig seul

Dans un second temps nous avons libéré certains axes de rotation - notamment celui de l'avant bras ou du torse et introduit de nouvelles équations basées sur des contraintes physiologogiques du corps humain pour obtenir un système d'équations résolvables.

Nous illustrons cette notion avec l'exemple du coude : le biceps est dirigé par le vecteur Y_b. Le vecteur X_b est défini comme orthogonal au plan formé par l'épaule, le coude et le poignet. Le vecteur Z_b complète l'ensemble pour former une base directe. L'avant bras est dirigé par le vecteur Y_{ab}. Le coude est modélisé par une liaison pivot à un seul degré de liberté (on néglige son léger rotulage), **le vecteur X_{ab} est donc colinéaire à X_b.** Le vecteur Z_{ab} complète l'ensemble pour former une base directe.

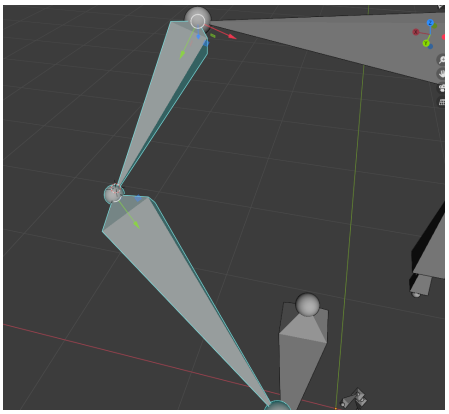


FIGURE 7 – Vue spécifique de l'articulation du coude

Nous n'avons hélas pas eu le temps de nous intéresser aux translations sur l'ensemble du mesh. Mais les résultats auraient été rapidement limités par l'imprécision sur la position 3D globale des rigs générés par *DOPE*.

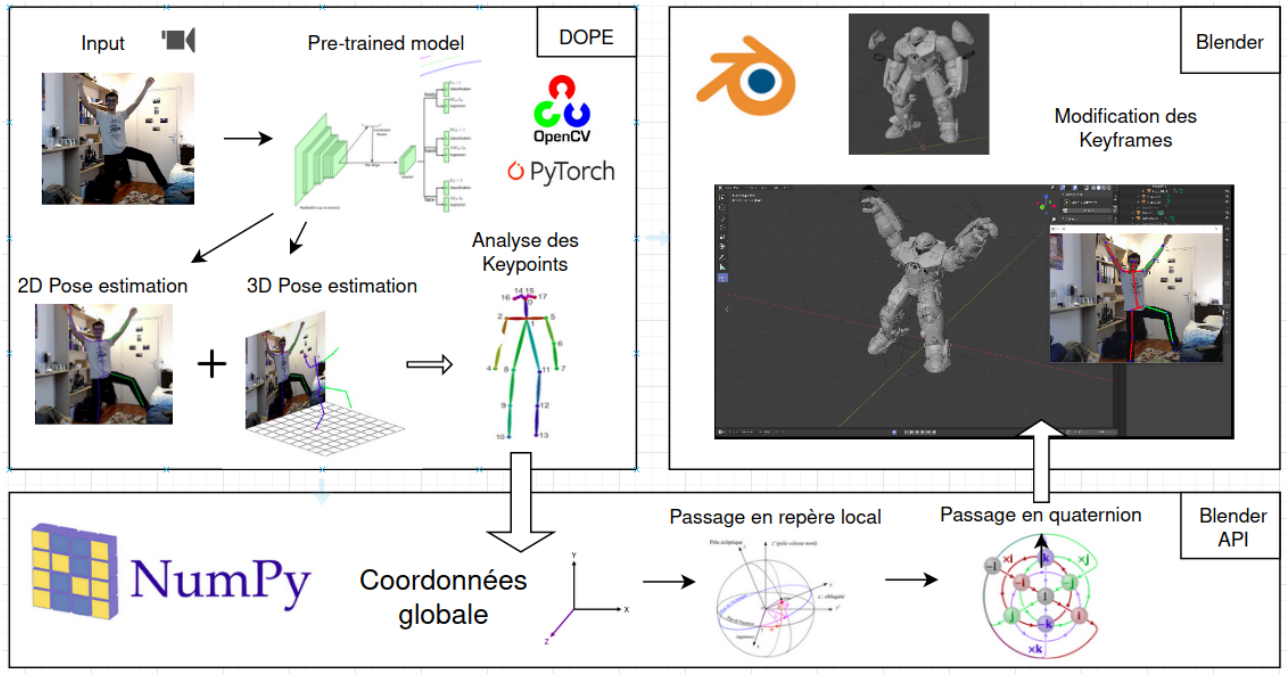


FIGURE 8 – Pipeline complet du Motion Capture

3 Résultats

3.1 Résultats de l'estimation de posture 3D

Les résultats obtenus par *DOPE* sont globalement bons, la localisation 3D est assez fidèle et robuste aux changements de luminosité et à l'occlusion. Nous avons néanmoins noté des soucis de performances sur une de nos machines n'étant pas équipée d'une carte graphique moderne. Ceci est un problème récurrent des réseaux de neurones convolutifs. Ce problème est particulièrement préoccupant dans la mesure où nous souhaitions obtenir un rendu en temps réel. Sur notre machine principale les résultats étaient tout à fait acceptable, avec une moyenne de 100 ms de calcul par frame. Cependant avec un tel taux de rafraîchissement, le taux d'allocation de VRAM de la carte graphique atteint ses limites, jusqu'à nous empêcher de prendre une capture vidéo en parallèle.

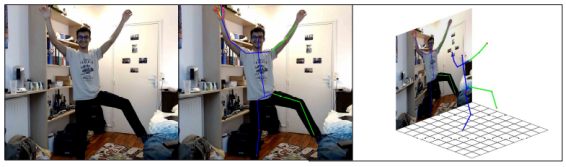


FIGURE 9 – Pipeline d'estimation *DOPE*

3.2 Préparation du Mesh 3D sur Blender

Il a également fallu un peu de travail pour importer le mesh pré-riggé et le préparer. Certains éléments ont été retirés et le rig a été simplifié pour se rapprocher le plus possible du rig généré par *DOPE*.

3.3 Résultats du pipeline complet

Le pipeline complet est fonctionnel. Blender est capable de démarrer le flux vidéo de notre webcam, de lancer *DOPE*, de récupérer les positions du rig, de traduire ces positions en quaternions à appliquer au rig de l'*Iron Man*, et enfin d'appliquer en temps réel ces modifications à notre mesh 3D. Ces positions peuvent être enregistrées et reproduites par la suite pour animer le modèle.

Le mesh suit assez fidèlement les mouvement du rig estimé par Blender. Le modèle est ainsi capable de s'asseoir, de tourner ses bras, de lever ses jambes une à une, de mettre ses mains sur sa tête, ou de faire un poirier. Nous avons cependant observé certaines limitations de notre méthode, et cela est principalement dû aux différences entre nos hypothèses physionomiques simplificatrices et la réalité des mouvements. Il est ainsi impossible au rig d'opérer une rotation de l'épaule dans l'axe du bras. Le mapping des mains permettrait de lever un degré de liberté sur la position du poignet et d'affiner ainsi le positionnement angulaire de tout le bras.

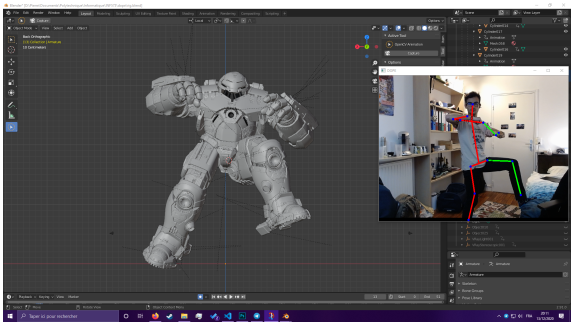


FIGURE 10 – Exemple de résultat obtenu en temps réel

4 Conclusion

En conclusion, nous avons particulièrement aimé travailler sur un projet aussi appliqué et avec un résultat aussi graphique. Nous avons particulièrement apprécié travailler sur ce projet car il nous a permit d'approfondir nos connaissances dans de nombreux domaines connexes.

Nous pensons avoir atteint les objectifs que nous nous étions fixés, à savoir de prouver la faisabilité d'une plateforme de motion capture à bas coût basé sur des techniques récentes d'apprentissage profond. Il reste cependant du travail pour implémenter une solution prenant en compte les mains et le visage, mais le travail est assez similaire à celui ayant déjà été accompli.

Enfin, il aurait été tout à fait possible avec un peu plus de temps de développer un algorithme capable de meilleurs calculs de quaternions à partir du RIG de *DOPE*, pour mieux s'adapter aux articulations humaines. Nous sommes également convaincus que l'interpolation de quaternions serait un très bon remède au faible taux de rafraîchissement.