



---

## Car Race Track

---

PROJET DE BACHELOR

PRINTEMPS 2019

*Auteurs :*

Gaston COQUAND  
Mathilde DURAND  
Emma HOGGETT

*Professeur:*

Pr.Christophe SALZMANN

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Système de vision</b>	<b>4</b>
2.1	Préambule . . . . .	4
2.2	Présentation de l'interface Cortex . . . . .	4
2.3	Mise en place des caméras . . . . .	5
2.4	Calibration . . . . .	8
2.4.1	Calibration avec le L-frame . . . . .	9
2.4.2	Calibration avec la Wand . . . . .	10
2.4.3	Problème de calibration . . . . .	10
2.5	Configuration du réseau . . . . .	11
2.6	Création d'objet sur Cortex . . . . .	11
<b>3</b>	<b>Sujet</b>	<b>18</b>
3.1	Matériel . . . . .	18
3.1.1	Voiture dNano . . . . .	18
3.1.2	Batterie et chargeur . . . . .	18
3.2	Appareillage et calibration de la voiture . . . . .	19
3.2.1	Télécommande . . . . .	19
3.2.2	Calibration de la voiture . . . . .	21
3.3	Marqueurs sur la voiture . . . . .	22
<b>4</b>	<b>Émetteur</b>	<b>23</b>
<b>5</b>	<b>Interface Labview</b>	<b>24</b>
5.1	Interface utilisateur . . . . .	24
5.2	Appareillage avec l'ordinateur . . . . .	24
5.3	Télécommande intégrée . . . . .	25
<b>6</b>	<b>Contrôle</b>	<b>27</b>
6.1	Modèle physique . . . . .	27
6.1.1	Calcul de la trajectoire . . . . .	28
6.1.2	Contrôleurs . . . . .	28
<b>7</b>	<b>Améliorations</b>	<b>30</b>
7.1	Architecture . . . . .	30
7.2	Contrôle . . . . .	30
7.3	Cortex . . . . .	31
<b>8</b>	<b>Autres pistes possibles d'asservissement</b>	<b>32</b>
8.1	Première approche : Cas simplifié . . . . .	32
8.2	Deuxième approche : Contrôle par arcs de cercles . . . . .	32
<b>9</b>	<b>Conclusion</b>	<b>33</b>
<b>Appendices</b>		<b>35</b>
<b>A</b>	<b>Présentation des différents VI</b>	<b>35</b>
A.1	Programme principal . . . . .	35
A.2	Sous VI . . . . .	36



## 1 Introduction

Ce rapport présente un projet de Bachelor en Génie Mécanique. Ce projet permet la mise en pratique, sur des problèmes concrets, des compétences de domaine et transversales acquises durant les cinq premiers semestres d'études. Il est la reprise et l'amélioration formelle d'un projet de Master initié en 2012.

Ce projet vise au contrôle du parcours de voitures miniatures sur un circuit préexistant part le biais du système d'acquisition d'image en temps réel **Cortex**. Ce dernier joue le rôle de capteur de position dont les données sont récupérées et traitées par un programme informatique en langage **LabVIEW**.

Le code déjà développé par d'anciens élèves ainsi que par le Professeur C. Salzmann du Laboratoire d'Automatique de l'EPFL, s'organise autour de deux grands axes : la création d'une télécommande et la mise en place de la communication avec un système de vision. Lors de ce projet il a été modifié, ainsi la commande est mise à jour en temps réel grâce aux données récoltées par Cortex et renvoyée par liaison sans fil aux voitures, ce qui permet d'ajuster la trajectoire de ces dernières.

La mise en place de ce projet permet un développement informatique et expérimental de modèles théoriques d'automatique mais surtout de faire un premier pas vers l'implémentations d'une technologie de contrôle par voie vidéo dans un espace clos. En effet dans le contexte automobile actuel, il apparaît de plus en plus des exigences d'autonomie pour les nouvelles voitures.

Dans ce rapport, sera présenté en premier lieu la mise en place du système de vision allant de l'installation des caméras à la création d'un objet sur Cortex en passant par la calibration du système. Puis les caractéristiques du modèle de voiture ainsi que de sa télécommande seront détaillées. Finalement l'interface LabVIEW sera abordée avec d'abord un descriptif de la télécommande intégrée et enfin une explication sur la mise en oeuvre du contrôle.

Ce rapport doit constituer une notice claire et détaillée pour les étudiants qui viendraient à reprendre le projet dans les prochaines années.

## 2 Système de vision

### 2.1 Préambule

Cortex est un logiciel de motion capture. Dans ce projet, il est utilisé comme système d'acquisition de position des voitures. Quinze caméras sont installées autour de la salle, elles filment le circuit parcouru par les voitures. Il est essentiel que les caméras soient disposées de manière à couvrir l'ensemble de la zone d'étude.

Le processus de capture commence par la collecte des données de position brutes du sujet. Le succès du mouvement qui en découlera dépend non seulement de la qualité de la définition de l'objet, mais également des compétences d'organisation de l'opérateur Cortex.

Il doit s'assurer que les données de capture de mouvement sont propres et lisibles. C'est-à-dire garantir que la calibration des caméras est bonne et qu'elle le reste toute la séance. Ce dernier point est particulièrement important lorsqu'il y a plusieurs personnes autour et qu'une caméra peut être heurtée accidentellement. Il faut en outre être attentif aux reflets et aux variations des conditions lumineuses, telles que la lumière du soleil passant par une fenêtre, ou à d'autres variables externes qui peuvent affecter une capture.

Procédure générale d'installation:

1. Installation du logiciel avec le disque Motion Analysis Cortex 2.6
2. Brancher chacune des caméras au système d'alimentation avec les câbles Ethernet, puis brancher les deux systèmes d'alimentation et enfin un système d'alimentation à l'ordinateur.
3. Vérifier le champ de vision de la caméra.
4. Orienter la caméra de sorte qu'elle détecte le L-frame et couvre correctement le circuit. Il existe deux vis pour la précession et la nutation du support et un clapet de sécurité pour la rotation de la caméra.
5. Jouer sur les trois molettes au niveau de l'objectif de la caméra pour que les quatres points du L-frame apparaissent le plus net possible à l'écran.

Certains points sont développés dans les sections suivantes.

### 2.2 Présentation de l'interface Cortex

Dans le cadre de ce projet les onglets **Calibration**, **Motion Capture**, **Post Process** et **Model Edit** qui apparaissent à la Figure 1 seront utiles. En fonction de ce qui doit être observé, il peut être judicieux de consulter ce que le logiciel propose dans **Data Views**. **2-D Camera View** et **3-D View** sont particulièrement indispensables pour l'observation des petites voitures sur un circuit à plat. **Layouts** permet différentes options de configuration des panneaux.

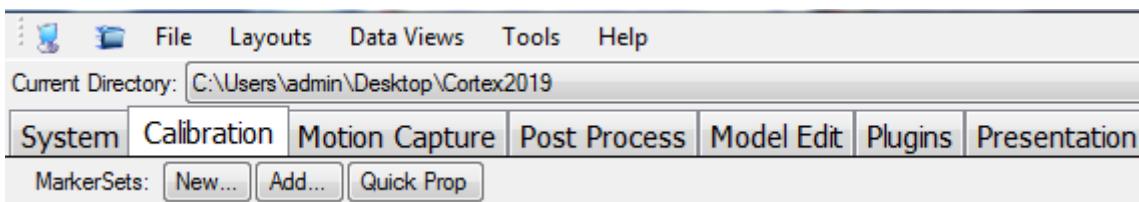


Figure 1: Interface Cortex haut

*Tracking*, *Identifying* et *Skeleton* doivent être cochés dans la Figure 2. Après avoir allumé les caméras (au moins 30s d'intervalle), il faut cliquer sur *Connect To Cameras*, puis sur le bouton **Run** pour observer quelques choses. Dans les options d'affichage il y a encore **All On** pour exposer les champs de vision de chaque caméra. Il est aussi possible de se concentrer plus spécifiquement certaines caméras en cliquant sur le numéro associé.

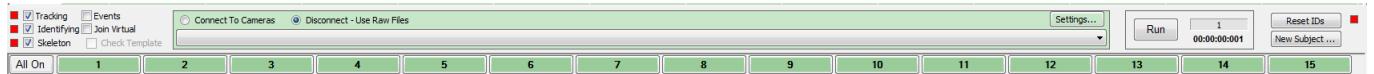


Figure 2: Interface Cortex bas

### 2.3 Mise en place des caméras

Avant de commencer il faut veiller à ce que la salle soit rangée et que les fenêtres soient recouvertes.

Liste de matériel :

- Quinze caméras Motion Analysis Osprey digital numérotées dans leur mallette étui. (Il y en a 16 mais la n°14 ne répond pas)
- Quinze supports pour caméra.
- Quinze câbles Ethernet qui relient les caméras aux systèmes d'alimentation.
- Deux systèmes d'alimentation Power Hub HCP-8 Motion Analysis et deux commutateurs Ethernet Cisco.
- Deux câbles Ethernet un qui prend en charge la connexion Cortex-LabVIEW, et l'autre celle Cortex et les caméras.



Figure 3: Caméra Analysis Osprey.

Ci-dessous quelques recommandations d'installation, afin de minimiser les possibilités de problèmes avec l'alimentation Ethernet du système Motion Analysis :

- Il faut être attentif aux étiquettes jaunes sur les Power Over Ethernet câbles. En effet il s'agit de câbles Cat5e standards, cependant quand ils sont branchés au séparateur POE ils deviennent "powered" et il ne faut surtout pas les brancher à des dispositifs Ethernet classiques.
- Pour éviter que les câbles d'alimentation ne se débranchent, un des deux placements de la Figure 4 est recommandé. En particulier si les câbles branchés aux caméras vont vers le haut, positionner le commutateur Ethernet Cisco en bas (à gauche sur la Figure 4), et si ils vont en bas positionner le en haut (à droite sur la Figure 4).



Figure 4: Placements recommandés pour le Power Hub HCP-8 et le commutateur Ethernet Cisco



Figure 5: Vue de détail du branchement

La Figure 6 présente une disposition des caméras éprouvée pour la capture 2D de notre système. Ainsi les caméras couvrent au mieux le circuit, c'est-à-dire qu'au moins trois caméras voient simultanément l'objet à chaque endroit du circuit. La prise de position est alors la plus précise possible, ce qui est crucial pour avoir un bon contrôle.

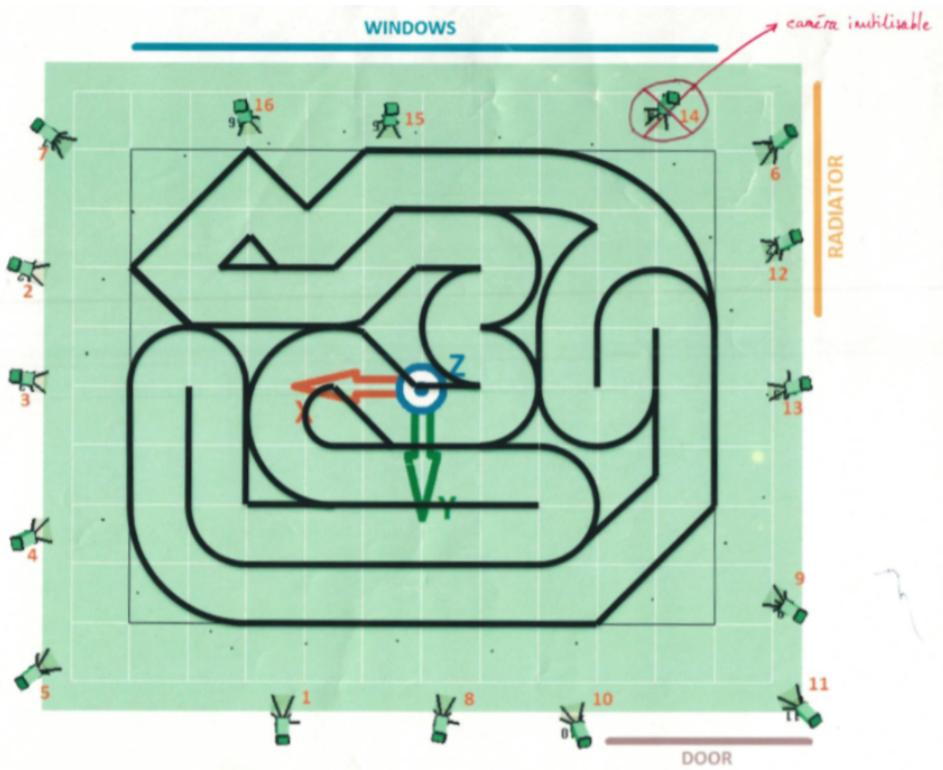


Figure 6: Plan du placement des caméras.

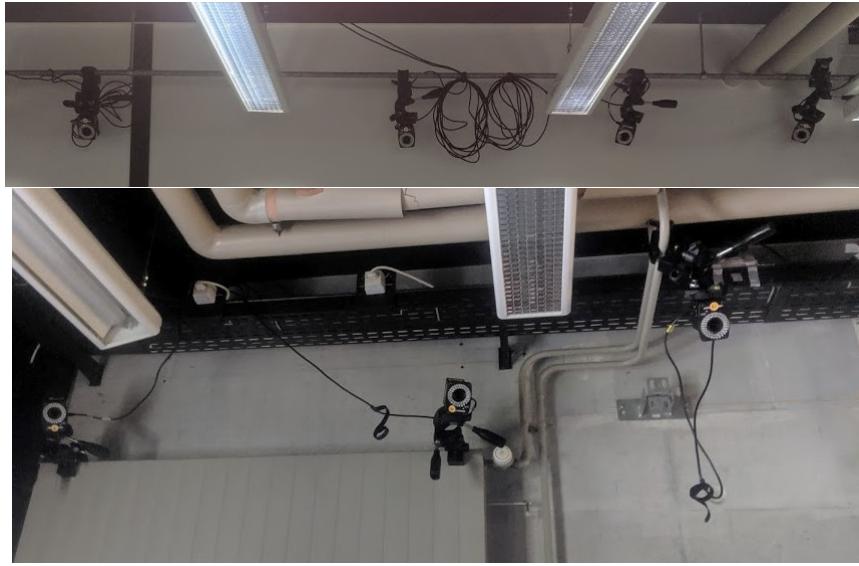


Figure 7: Vue non exhaustive du placement des caméras.

Un autre agencement est envisageable mais chaque zone du circuit doit être visible par au minimum trois caméras. Une vue 3D du champ de vision, recoupé par les caméras, peut être obtenue sur le logiciel cortex en cliquant dans la vue 3D **clique droit > Show > Show Camera Field of View**.

Placer le L-frame (cf Figure 8) au milieu de la pièce pour vérifier l'installation des caméras. Les caméras doivent détecter les quatre marqueurs (marker) du L-frame. Il joue le rôle d'un

point de repère grâce auquel elles peuvent s'orienter dans l'espace. Pendant l'installation, il est possible de vérifier sur Cortex ce que voient les caméras branchées, dans **2D Camera View** en cliquant sur **Run** après avoir sélectionné leur numéro.

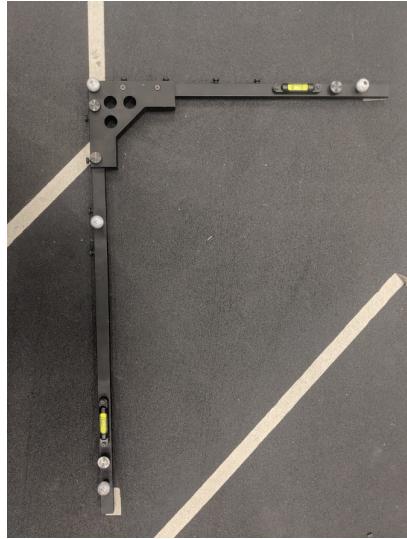


Figure 8: L-frame au centre du circuit

Une fois l'installation des caméras terminée, il peut subsister des incohérences entre leur placement réel et la façon dont Cortex les voit dans l'espace. Cela signifie que la calibration utilisée n'est plus d'actualité. Il faut donc recalibrer le système afin de concorder avec le nouveau placement des caméras.

#### 2.4 Calibration

Afin de préserver une estimation de bonne qualité de la position, le système doit être régulièrement recalibré. La calibration est une étape cruciale de la mise en place du système de vision. Sans elle les données récoltées par Cortex n'ont pas de sens. Elle se fait en deux temps. La méthode avec L-frame peut suffire, elle permet d'obtenir un premier aperçu de la position des caméras. Il est ensuite possible d'affiner le résultat trouvé en balayant l'espace avec la Wand (cf Figure 9).

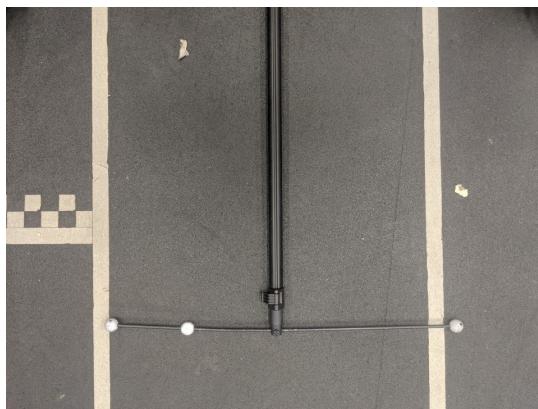


Figure 9: Extrémité de la Wand.

### 2.4.1 Calibration avec le L-frame

1. Positionner le L-frame à l'origine du circuit (marqué par 3 scotches formant un repère orthogonal au centre du circuit).
2. Configurer les fenêtres pour la calibration : **Layout > 2 panes Data View > 2D Camera View** dans la fenêtre du haut (top pane), **Data View > 3D View** dans la fenêtre du bas (lower pane).
3. Cliquer sur **Connect The Camera** puis sur **Run**.
4. Aller dans l'onglet **Calibration**, puis à droite dans **Calibrate** (Figure 11) et enfin cliquer sur *Camera Aiming*. Toutes les caméras apparaissent alors dans la vue 3D comme sur la Figure 10.

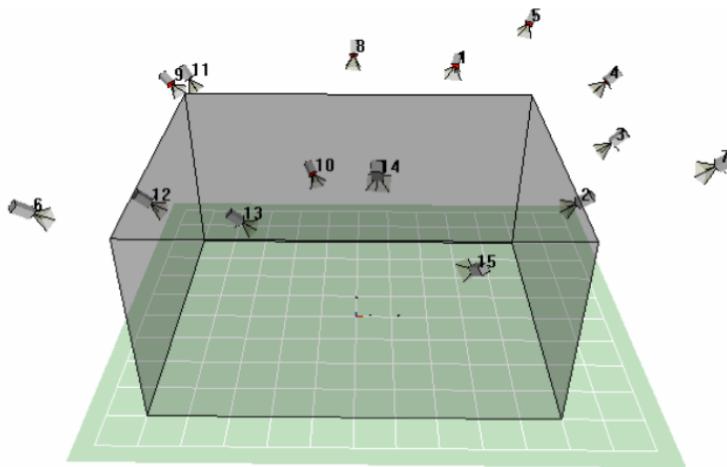


Figure 10: Vue 3D de l'emplacement des caméras.

Si toutes les étapes ont été bien suivies jusque là, le numéro associé à chaque caméra (cf Figure 2) doit apparaître en jaune.

Si la position n'est toujours pas bien déterminée, i.e. la caméra est prise de soubresauts de part et d'autre de la pièce ou son objectif reste collé face contre l'origine ou si le numéro de la caméra n'apparaît pas en couleur jaune, reportez vous à la section 2.4.3. Cependant il peut être judicieux de vérifier dans un premier temps ce qui pourrait gêner la prise de vue : les objets brillants, les possibles obstruction du champ de vision de la caméra, la connexion au boîtier d'alimentation.

5. Si une caméra apparaît à l'opposé de son emplacement physique, une explication possible est que les réglages indiqués ne correspondent pas à son inclinaison réelle. Toujours dans **Camera Aiming** (Figure 11), il faut cliquer sur **Details > length/orientation**. Puis il s'agit de sélectionner *alternate* si la caméra en question est inclinée de plus de 90° par rapport à l'axe vertical, ou bien *normal* si ce n'est pas le cas.
6. Dans la section **Calibration with L-frame**, donner un nom au fichier de calibration dans *Filename* et cliquer sur **Collect and Calibrate**. (Voir Figure 11)

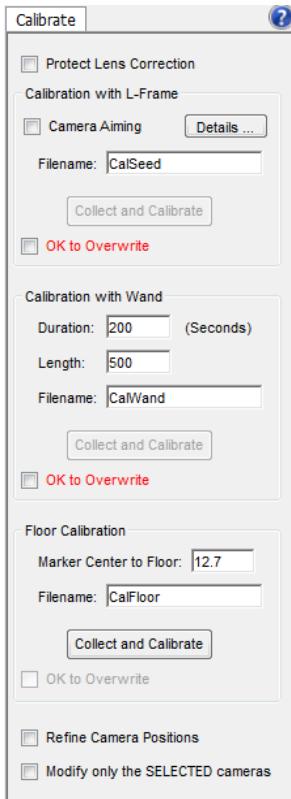


Figure 11: Options de calibration sur Cortex.

#### 2.4.2 Calibration avec la Wand

1. Dans la section **Calibration with Wand**, donner un nom au fichier de calibration dans **filename**, utiliser 200 et 500 pour **Duration** et **Length** respectivement et cliquer sur **Collect and Calibrate**.
2. Le système commence alors à enregistrer. Balayer l'ensemble de la zone d'étude couverte par les caméras avec la Wand. Cliquer sur **All On** pour examiner l'évolution du balayage via **2-D camera view**. Le but est de couvrir un maximum de la surface d'intérêt.
3. Si l'estimation de la **Wand length** est dans une fourchette de 0.2 mm autour de la taille de la Wand (500 mm), cliquer sur **Accept** et le processus est complété. Sinon recommencer.

#### 2.4.3 Problème de calibration

Voici une procédure non exhaustive pour résoudre les problèmes qui peuvent entraver le bon déroulement d'une calibration :

1. Positionner le L-frame de sorte qu'il soit visible par toutes les caméras.
2. Cliquer sur **Connect to Cameras > Run**.
3. Sélectionner, une à une, les caméras via leur numéro au bas de la page. Observer les données brutes affichées dans le **2-D camera view**. La fenêtre devrait seulement contenir les quatre marqueurs du L-frame. Le but étant qu'aucune perturbation n'apparaisse.

S'il existe des zones noires qui ne sont pas des marqueurs, il s'agit probablement de réflexions indésirées. Vous pouvez les masquer en mettant une **box mask** dessus à l'aide du bouton centrale de la souris.

## 2.5 Configuration du réseau

Une connexion doit être établie entre les deux ordinateurs. Cette dernière se résume par un protocole UDP, où l'adresse IP de l'autre ordinateur doit être entrée. Les deux ordinateurs communiquent bien par le biais d'un câble Ethernet [1]. La procédure de diffusion est lancée sur le second ordinateur.

La configuration du réseau sur les deux ordinateurs se fait de la façon suivante:

### Configuration sur la machine Windows

- Aller dans le panneau de contrôle > **Network and sharing center** > **Change adapter settings**
- Faire un clic droit sur la connexion Ethernet des caméras > **Properties**. Pour savoir rapidement lequel des deux est la connexion aux caméras, déconnecter le câble et regardez quelle connexion reste inchangée.
- Cliquer sur **Internet protocol Version 4 (TCP/IPv4)** > **Properties**
- Cliquer sur **Use the following IP address**, puis imposer dans **IP adress**: 10.1.1.199 et dans **Subnet mask**: 255.255.255.0. Enfin presser sur **OK** et fermer la fenêtre.
- Faire la même chose pour le second câble Ethernet, celui connecté à la machine MacOSX. L'adresse IP doit être 10.1.2.101.

### Configuration sur la machine MacOSX

- Aller dans **System preferences** > **Network**
- Sélectionner sur l'interface le port Ethernet qui est connecté à la machine avec Cortex. De même que précédemment, déconnecter le câble pour savoir lequel correspond.
- Pour configurer l'adresse IP, aller dans **Configure Iv4** puis selectionner **Manually**. L'adresse IP doit être 10.1.2.102 et **Subnet mask** 255.255.255.0.

## 2.6 Crédit d'objet sur Cortex

Avant de faire tourner une expérience, il faut être sûr que l'objet qui doit être détecté par les caméras est reconnu par Cortex. Ci-dessous un petit rappel de toutes les étapes qui ont du être accomplies avant de commencer la création de l'objet:

1. Ouvrir les deux ordinateurs.
2. Brancher le système d'alimentation des caméras.
3. Ouvrir Cortex. Il est possible d'ouvrir un fichier s'il en existe déjà un.
4. Charger les réglages de Calibration s'ils existent sinon calibrer le système. Cortex s'ouvre par défaut avec le dernier fichier de Calibration défini.
5. Vérifier que les caméras sont toutes connectées et allumées. Une lumière verte apparaît si la caméra est bien branchée et de manière générale si le système de vision ne fait pas de bruit, il n'est pas branché.

6. Positionner l'objet sur l'origine qui se trouve au centre de la salle. Pour les voitures, il faut mettre la voiture sur l'axe X et l'arrière est centré sur l'origine.
7. Aller dans **Motion Capture**. Cliquer sur **Run**. Vérifier que l'objet est visible à l'écran. Puis cliquer sur **Pause**.

Une fois que toutes ces étapes sont acquises, la création de l'objet dans Cortex peut commencer:

1. Aller dans l'onglet **Motion Capture**, puis dans la table sur la droite **Output**. Nommer la capture. Si le bouton **Record** n'est pas accessible, double-cliquer sur **Tracked Binary**. Mettre le temps d'enregistrement à 10s et lancer l'expérience.

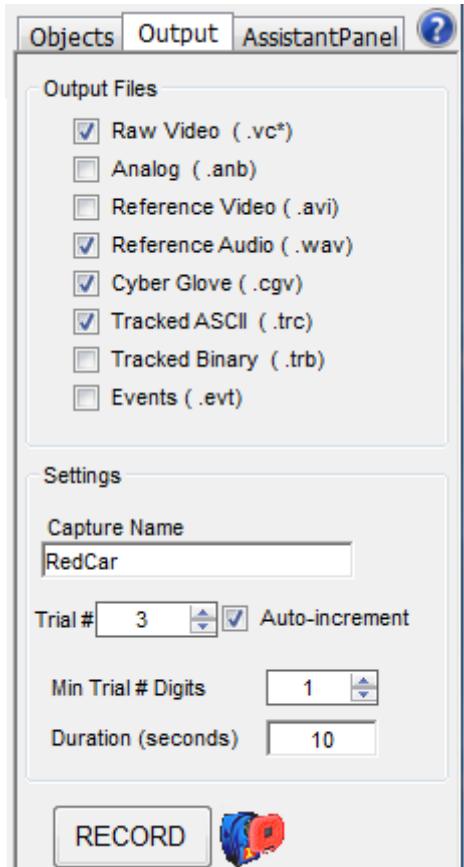


Figure 12: Lancer la capture qui permettra de définir les Markers

2. Aller dans l'onglet **Post Process**, puis dans **File** cliquer sur **Load Capture** et chercher dans la fenêtre qui vient d'apparaître le nom de la Capture précédemment enregistrée.

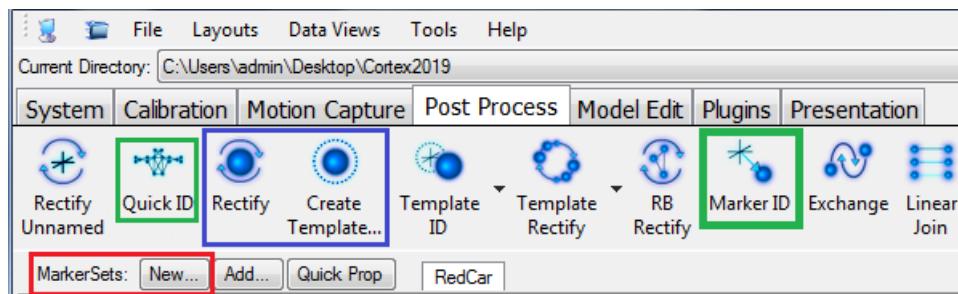


Figure 13: Onglet Post Process

3. Sélectionner les 4 Markers qui définissent la voiture dans la vision 3D. Créer un nouveau set de Marker (se référer à la Figure 13; encadré rouge).

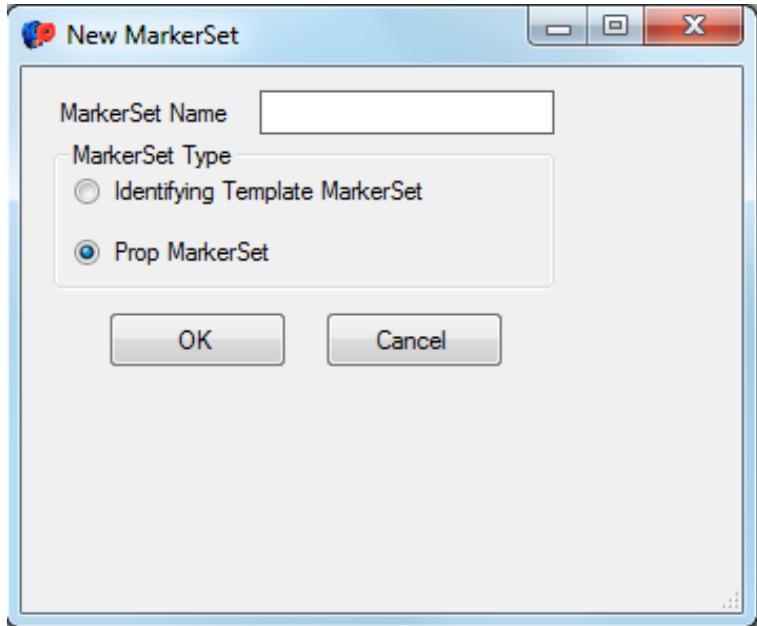


Figure 14: Fenêtre Crédation New MarkerSets

L'outil Prop object génère facilement et rapidement un objet rigide et localisé, indépendant de tout autre MarkerSet dans la capture. Il s'utilise uniquement en **Post Process** sur des données pré-enregistrées ou sur des données live (sans enregistrement). Cortex va créer un fichier ".prop" correspondant à l'objet défini. Caractéristiques des Props :

- Corps rigide, défini par 4 markers ou plus. Ces derniers sont identifiés et étiquetés en temps réel, si *Tracking* et *Identifying* sont checkés quand on est dans **Motion Capture**. Se référer à la Figure 2.
- Possède un système de coordonnées local dont l'origine correspond au centre de masse géométrique défini par les markers. L'orientation initiale est fixée par le placement du Prop dans le champ de vision au moment de sa création. Ce système de coordonnées apparaîtra si *Show Skeleton* et *Show Skeleton Axes* sont checkés dans le **3D Display Show Properties** (cf Figure 15). Ce Panneau apparaît après un clic droit dans la fenêtre de visualisation 3D.

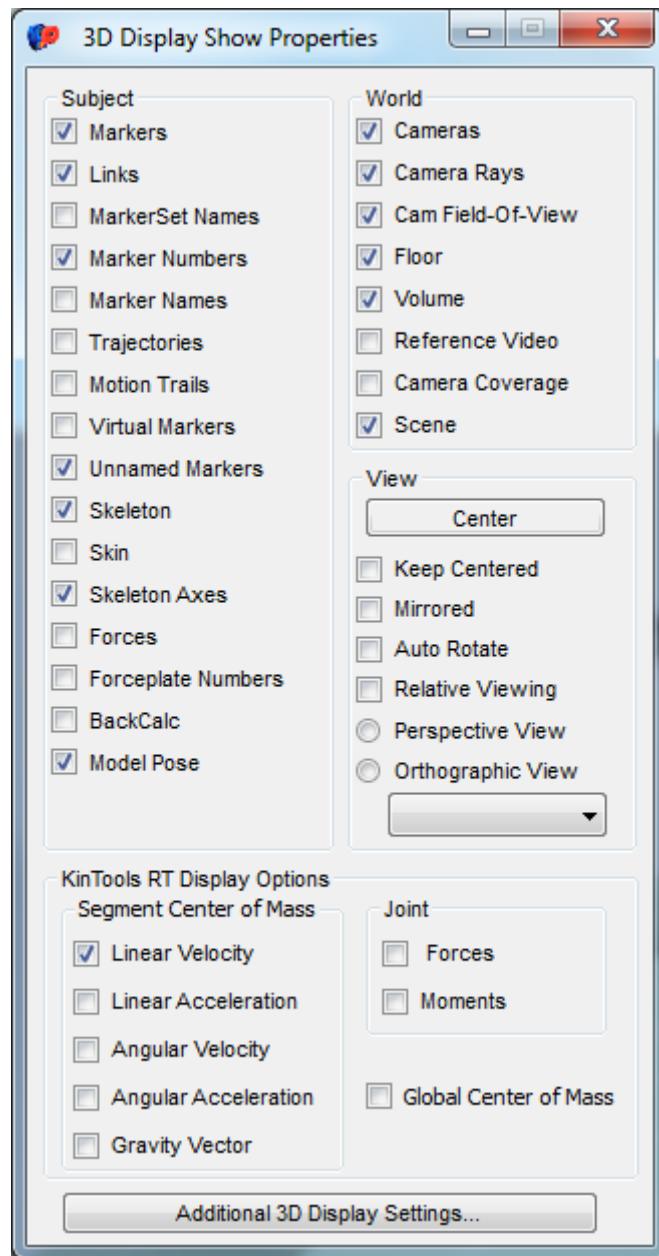


Figure 15: 3D Display Show Properties

- Le bouton **Quick Prop** de la Figure 13 génère un Prop à partir de la sélection de markers de la vue 3D. Les markers et le Prop sont nommés par défaut, cela peut être modifié dans **Model Edit**. Si l'ordre des markers est important il peut également être modifié dans l'onglet **Tree View** dans **Model Edit**.
- Les Prop peuvent s'utiliser de la même manière que les ".mars". Il s'agit type de fichier de base des MarkerSet. Il s'obtient en cochant *Identifying Template MarkerSet* à la place de *Prop MarkerSet* dans la Figure 14.

Ci-dessous à la Figure 16, le Prop obtenu.

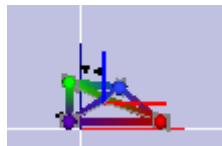


Figure 16: Aperçu Vision 3D; Objet défini

4. Aller dans **Model Edit**, puis dans la fenêtre sur la droite dans l'onglet **Markers**. Dans la liste **MarkerNames** insérer le nom des trois marqueurs. Cliquer sur **Marker ID** (cf Figure 13; encadré vert). Cliquer sur un nom entré dans la liste, puis sur le Marker dans la vision 3D qui correspond. La même procédure peut être suivie via **Quick ID**.

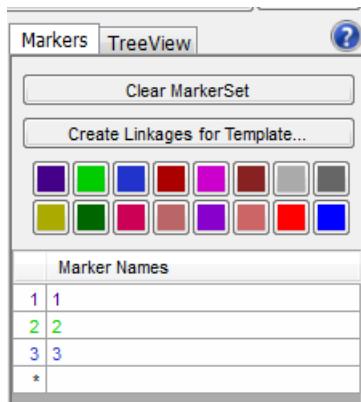


Figure 17: Onglet Markers dans Model Edit

5. Si des segments doivent être créés (certains vont être générés automatiquement par le Prop), cliquer sur **Create Linkages for Template...** (cf Figure 17). Attention la fenêtre de la Figure 18 apparaît, il ne faut ni s'en préoccuper ni la fermer. Poursuivre l'étape indiquée ci-dessous. Sélectionner une couleur, cliquer sur le premier marker puis glisser jusqu'au suivant. À répéter pour chaque segment.

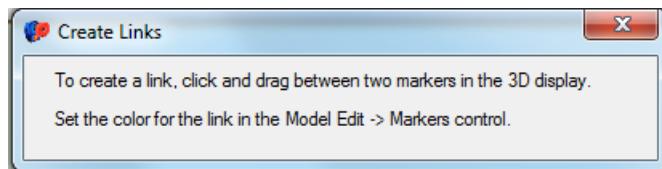


Figure 18: Pop-up à ne pas fermer

Il est possible de vérifier dans **TreeView** le set de Markers créé.

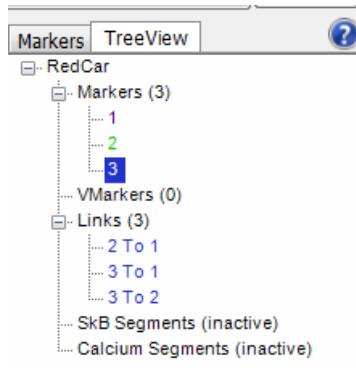


Figure 19: Onglet Tree View dans Model Edit

6. Afin d'enregistrer l'objet ainsi créé, sélectionner **Rectify** puis **Create Template**.

Au cours des modifications, il est à tout moment possible de faire un clic droit sur l'onglet du nom du MarkerSet et de l'enregistrer (cf Figure 13; *RedCar*).

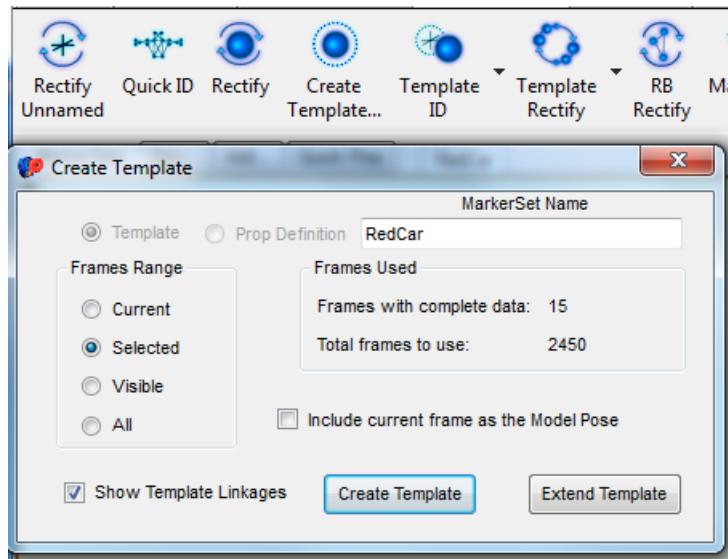


Figure 20: Enregistrer l'objet

Une fois cette procédure terminée, l'expérience peut être lancée. Dans un premier temps il faut s'assurer que l'ordinateur sur lequel LabVIEW tourne n'est pas connecté à internet. Cela empêcherait la communication avec l'ordinateur sur lequel Cortex fonctionne. Sur Cortex, il faut aller dans l'onglet **Motion Capture**. Sélectionner l'onglet **Objects** dans le panneau à droite, puis cliquer sur **Open Fusion Object Loader**.

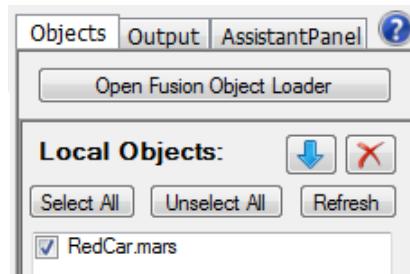


Figure 21: Lancer l'expérience : Ouvrir la fenêtre de lancement

La fenêtre de la Figure 22 apparaît. Il faut alors *Load* l'objet puis cliquer sur **Run**

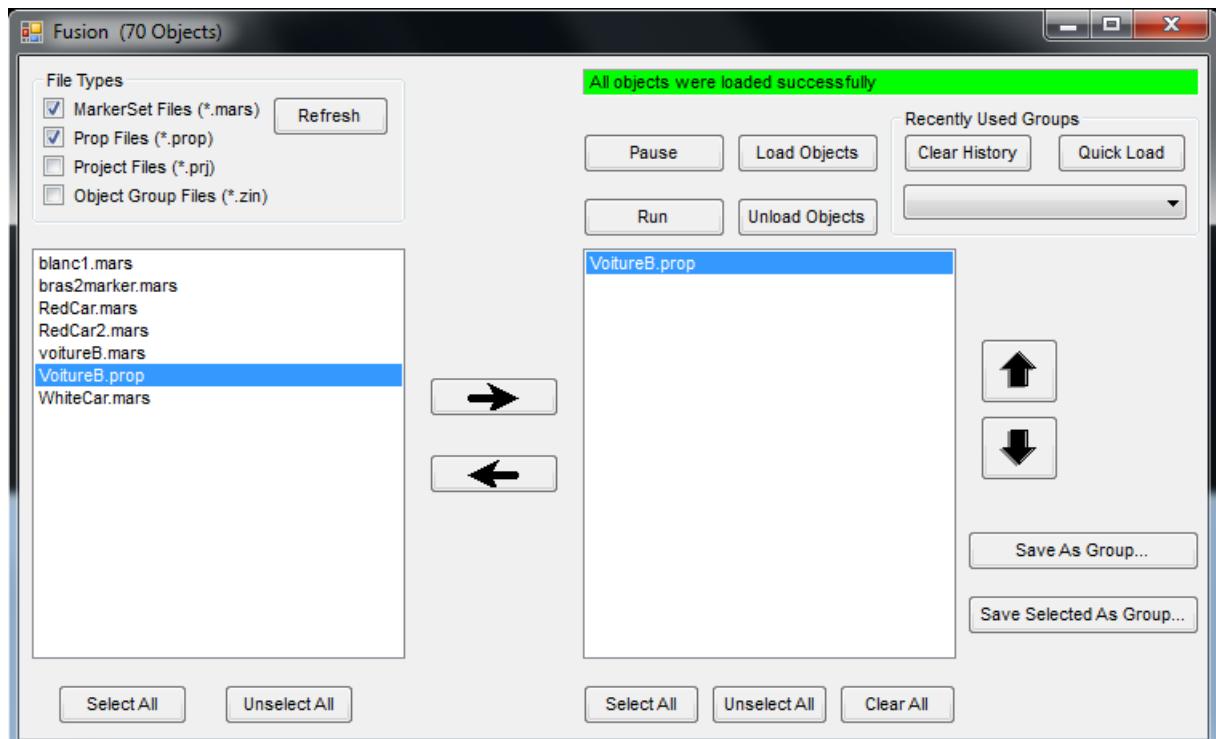


Figure 22: Lancer l'expérience : chargement de l'objet et Run

Enfin sur l'autre ordinateur lancer le programme LabVIEW correspondant.

Si jamais certains problèmes persistent, il ne faut pas hésiter à solliciter le support de Cortex. L'équipe est assez réactive et disponible à l'adresse suivante : [support@motionanalysis.com](mailto:support@motionanalysis.com).

### 3 Sujet

#### 3.1 Matériel

##### 3.1.1 Voiture dNano

L'émetteur génère une fréquence de 2.4GHZ, tout type de voiture fonctionnant à cette fréquence peut donc être potentiellement utilisé. Dans ce projet, des voitures de modélisme **dNano FX-101 2.4GHz** de la marque japonaise **Kyosho** ont été choisies.



Figure 23: Voiture dNano complète sans carrosserie.

Il y a trois châssis accompagnés de leur carrosserie (blanche, rouge et noire) sur lesquelles sont collés les marqueurs (markers) de motion capture. Ce modèle a un prix de l'ordre de 200 chf justifié par la taille réduite de sa technologie embarquée et sa réactivité haute performance.

##### 3.1.2 Batterie et chargeur

Chacune des voitures fonctionne à l'aide d'une batterie **dNano Lipo 3.7V-200mAh 0.74Wh**. Elles peuvent être rechargées par le biais d'un des deux **KYOSHO dNano PORTABLE QUICK CHARGER (For 3.7V Lipo)** [3]. Il en existe un fonctionnant avec quatre piles AA et un autre qui se branche sur secteur. Les deux modèles sont à disposition pour le projet.



Figure 24: Batterie dNano et chargeur à piles associé.

Il est important à veiller à recharger régulièrement les batteries car celles-ci s'épuisent très vite. Pour cela il faut suivre la procédure suivante:

1. Enlever la carrosserie. Il faut commencer par l'écarter sur les côtés avant de la déclipser à l'avant.

2. Détacher la batterie. Elle se trouve sur le dessus de la voiture.
3. Positionner la batterie sur le chargeur en veillant à faire correspondre les parties métalliques entre elles.

La Led du chargeur suit le code couleur suivant :

- verte si la batterie est en charge.
- éteinte si la batterie est chargée.
- clignote Vert/Rouge si la batterie de la pile du chargeur est insuffisante.
- rouge si il y a une erreur.

### 3.2 Appareillage et calibration de la voiture

#### 3.2.1 Télécommande

Avant d'appareiller la voiture à l'ordinateur, Il est recommandé de connecter la voiture avec une télécommande **ASF 2.4GHz System Perfex KT-18** [4] afin de s'assurer qu'il n'y a aucun problème et de calibrer la voiture.



Figure 25: Télécommande dNano. .

La télécommande fonctionne à l'aide de 4 piles AAA. L'étui à pile s'ouvre à l'aide du bouton "release" sur la face principale de la télécommande. Il en existe trois fonctionnelles.

Pour appareiller une voiture à une télécommande il faut suivre le protocole suivant:

1. Avec une télécommande éteinte, appuyer sur **[P]** du Trim de Gaz et le maintenir enfoncé.
2. Mettre la télécommande sous tension (ON) à l'aide du bouton "power" tout en maintenant **[P]** enfoncé.
3. Vérifier que la LED arrière soit allumée. Lorsque la LED s'éteint après trois secondes, relâcher le bouton **[P]**. La préparation de l'émetteur est terminée.

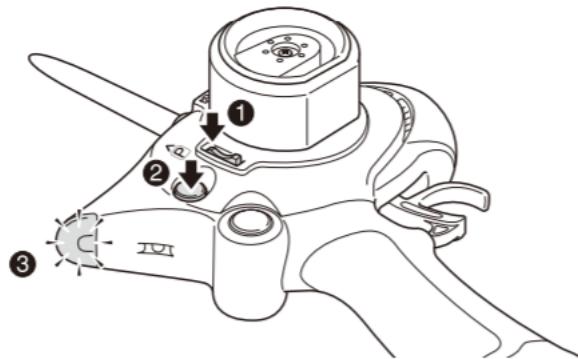


Figure 26: Procédure pour l'émetteur.

4. Sur la voiture, à l'aide d'un objet pointu presser le bouton de paireage au centre de la plaquette bleue située sur le dessous de la voiture (un clic peut être entendu).
5. Mettre sous tension la voiture à l'aide de l'interrupteur en bas à droite de la plaquette bleue en maintenant le bouton enfoncé.

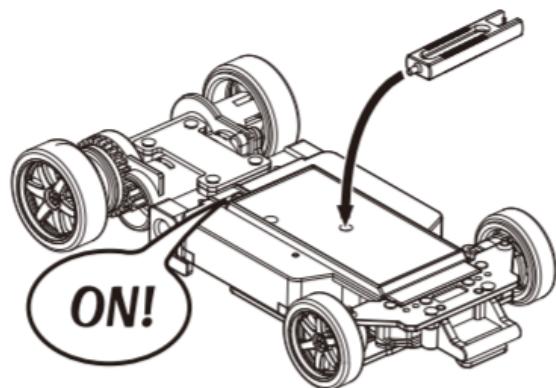


Figure 27: Procédure pour la voiture.

6. Une fois que la LED sur la plaquette bleue cesse de clignoter pour briller d'un rouge intense, relâcher le bouton de liaison. L'appareillage s'est bien effectué.
7. Mettre hors tension la voiture puis la télécommande. À la prochaine mise en tension, l'appareillage sera effectif. **Attention!** Il est important de d'abord éteindre la voiture avant la télécommande quand il est souhaité d'arrêter d'utiliser l'ensemble voiture et télécommande.

Il se peut que le processus de connexion ne soit pas concluant. Dans ce cas, réitérer la procédure complète. Voici une liste non exhaustive de cause possible à un problème de "paireage":

- Une autre connexion est encore effective.
- Un autre appareil fonctionne également sur du 2.4 GHz.
- Des appareils émettant de fortes ondes électriques sont en marche.
- Un autre émetteur utilisant du 2.4 GHz est en fonctionnement.

### 3.2.2 Calibration de la voiture

Il peut subsister des décalages dans le fonctionnement de la voiture qu'il est nécessaire de régler avant toute utilisation avec une télécommande.

Si la gâchette de gaz est en position neutre et que la voiture continue à rouler c'est qu'il un décalage en accélération. Dans ce cas appuyer autant de fois que nécessaire sur [A], si la voiture recule ou sur [B], si la voiture avance. Activer régulièrement la gâchette des gaz pour enregistrer les modifications et tester l'importance de l'erreur.

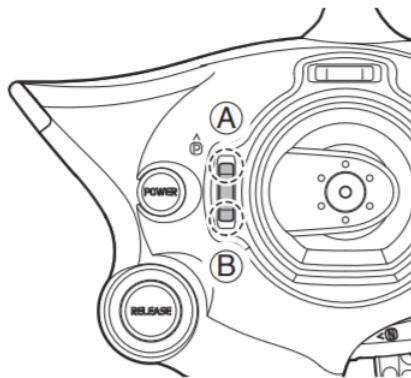


Figure 28: Calibration de l'accélération.

Si la voiture ne roule pas droit alors que la manette de direction est en position neutre, il existe un décalage en direction. Dans ce cas appuyer autant de fois que nécessaire sur [R], si la voiture dévie naturellement à gauche ou sur [L], si la voiture va naturellement à droite. Activer régulièrement sur la gâchette des gaz pour enregistrer les modifications et tester l'importance de l'erreur.

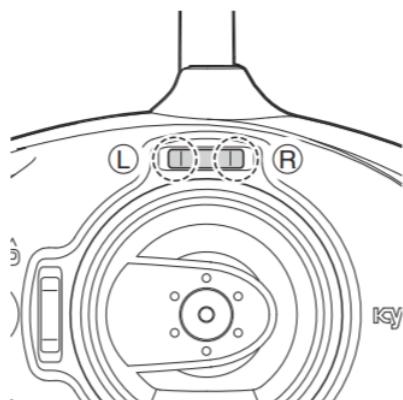


Figure 29: Calibration de la direction.

Si jamais la calibration n'est pas suffisante pour supprimer les erreurs, se référer au protocole de calibration avancée contenu dans le document [4].

### 3.3 Marqueurs sur la voiture

Afin que les caméras détectent la voiture, des marqueurs (*markers*) sont collés sur la carrosserie comme présenté à la Figure 30.

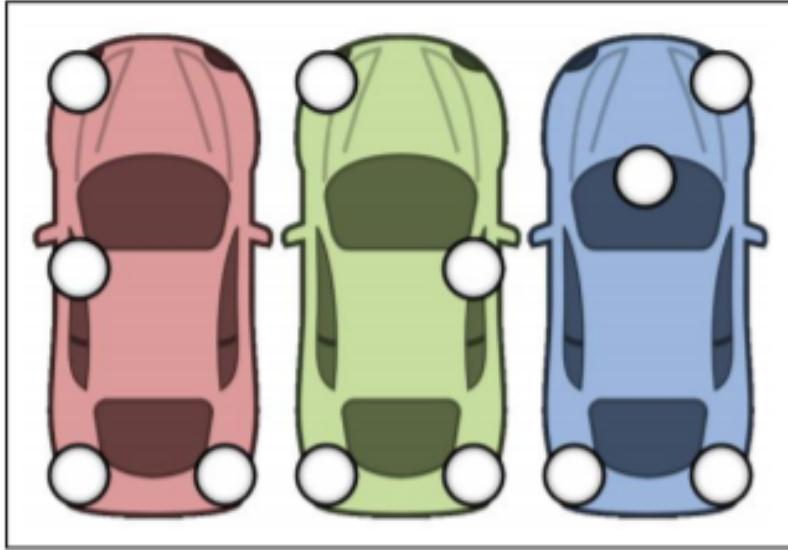


Figure 30: Disposition des marqueurs sur les voitures [6].

Le placement des marqueurs doit obéir à plusieurs règles pour que l'objet soit bien défini par Cortex. Le logiciel est capable de détecter des marqueurs jusqu'à 5 mm de diamètre. Des marqueurs de 1.3 cm ont été utilisés. Ce choix de taille devrait garantir une détection de bonne qualité, tout en conservant une certaine précision sur l'emplacement des marqueurs sur la voiture. Il faut éviter toute symétrie de point pour rendre possible l'orientation de la voiture. Le nombre minimum de marqueurs à avoir pour que Cortex caractérise bien l'objet est donc de quatre. Dans le cas où plusieurs voitures sont employées, leurs marqueurs respectifs devront être positionnés avec des motifs différents, voir la Figure 30.

Les marqueurs qui ont été positionnés sur la voiture blanche dans le cadre du projet ne tiennent pas très bien, car fixés avec des autocollants double face. Le pistolet à colle serait une bonne alternative pour les attacher correctement.

## 4 Émetteur

Un émetteur 2.4 GHz assure la communication entre l'ordinateur avec LabVIEW et la voiture. La construction de ce dernier est basée sur le même principe que la télécommande livrée avec le modèle réduit. Il a été implémenté à partir d'un circuit imprimé sur lequel sont soudés deux connecteurs. Le module visible à la Figure 31 à gauche est issu d'une manette qui a été démontée. Il s'agit d'un RF transmitter module appelé KTSS-701, il est connecté au premier connecteur du circuit imprimé. Le deuxième est utilisé pour connecter l'émetteur à la NI card de l'ordinateur.

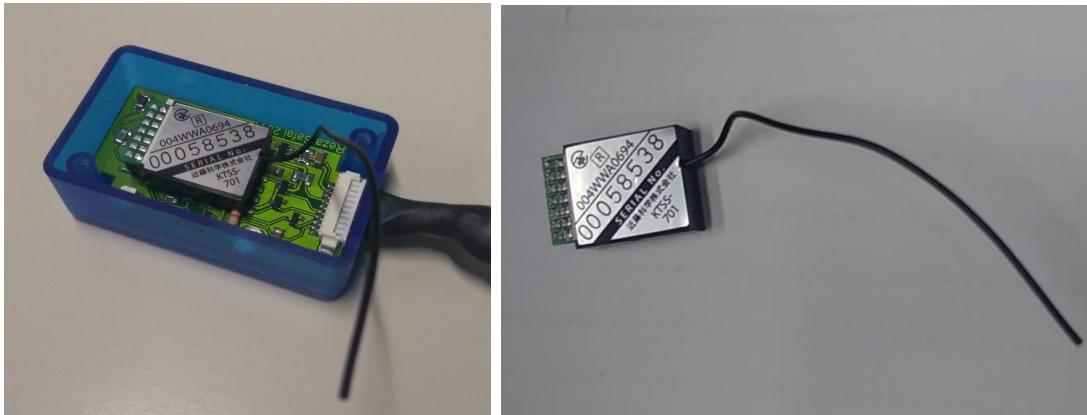


Figure 31: Intérieur du boîtier d'émission à gauche. Module d'émission à droite.

L'indicateur audio implémenté dans la télécommande a été transformé en indicateur visuel avec une LED verte. Une LED jaune indique si le pairage a bien fonctionné, elle clignote tant que l'émetteur et la voiture ne sont pas connectés et elle brille avec un intensité faible si la connexion est bien établie.

L'émetteur construit est facilement démontable et modulable, il peut être modifié en cas de problème de liaison. Il existe un problème de faux contact au niveau de la soudure à la partie blanche. L'ensemble de l'émetteur est présenté à la Figure 32.



Figure 32: Emetteur complet à gauche. Connexion avec l'ordinateur à droite. [6].

## 5 Interface Labview

Pour contrôler la voiture à partir des données fournies par Cortex, un code Labview est élaboré. Ce dernier détermine la position en temps réel de la voiture sur le circuit et calcule une trajectoire de référence à partir de celle-ci puis envoie les commandes à la voiture. Le fonctionnement de l'interface utilisateur dans sa globalité et un protocole de mise en route seront exposés. Le contrôle par le biais du Logitech RumblePad sera également explicité.

Deux VIs ont été créés, le "main.vi" qui est le VI sur lequel le contrôle est implémenté et "main\_2.vi" qui lui ressemble fortement mais qui teste les valeurs reçues par Cortex. Le premier VI sera explicité ici, le deuxième étant une version moins aboutie et composée de VIs équivalents.

### 5.1 Interface utilisateur

Une fois que Cortex est lancé, le programme LabVIEW peut être à son tour démarré. Avant de le lancer, il faut bien contrôler que la connexion Wi-Fi est éteinte et qu'elle n'interférera pas avec la connexion Ethernet entre les deux ordinateurs. Il faut également vérifier que le nom de l'objet, i.e. celui du ".prop" utilisé dans Cortex est entré.

De plus, un chemin a été ajouté vers une image du circuit et il faut veiller à ce qu'il soit complété. Son nom est "trackfinal.png" et il est sur le poste de travail. Il faut faire attention au fait que son orientation est mauvaise par rapport à la réalité. La trajectoire de référence peut être modifiée à tout moment par l'utilisateur en faisant glisser les points sur le graphique à l'aide de la souris.

Une fois toutes ces vérifications faites, allumer l'émetteur grâce au bouton **Power**. Il est possible d'arrêter le contrôle à tout instant avec le bouton **Quit**.

À partir d'ici, les données de contrôle peuvent être changées par l'utilisateur, notamment au niveau des gains. La trajectoire de référence et celle de la voiture sont dessinées en temps réel sur l'interface.

Les boutons **Load Trajectories** et **Save trajectories** servent à analyser les données envoyées à la voiture. Il ne sont pour le moment pas fonctionnels.

Les commandes **ST-L**, **ST-R**, **TH-INC** et **TH-DEC** ont pour but de calibrer la voiture. Si la voiture dévie naturellement à gauche, cliquer sur **ST-R** autant de fois que nécessaire. Inversement si la voiture dévie naturellement à droite. Dans le cas où la voiture est trop rapide, cliquer sur **TH-DEC** autant de fois que nécessaire. Inversement si la voiture va trop lentement. **Reset** réinitialise la calibration sur les commandes de direction et de vitesse.

### 5.2 Appareillage avec l'ordinateur

Pour connecter la voiture à l'ordinateur, il est nécessaire de passer par l'émetteur de 2.4GHz présenté à la section 4. Il présente des faux contact, il est donc possible que le système ne soit pas fonctionnel.

Une procédure de démarrage du programme LabVIEW est proposée ci-dessous :

1. Mettre en route le projet "HumainTest2019". Avant de le mettre en route vérifier que le programme ne présente pas de fil cassé. Si le fil problématique est lié à une boucle for,

modifier l'auto-indexing. **Attention!** Modifier ce paramètre peut changer l'action de la boucle.

2. Cliquer sur le bouton **Pair** dans l'interface utilisateur.
3. Attendre 3.5 s, jusqu'à ce que la LED jaune de l'émetteur s'allume. Elle doit briller avec moins d'intensité.
4. Éteindre la voiture. L'interrupteur se trouve sous la voiture.
5. Appuyer sur le bouton d'accouplement sous la voiture et le maintenir pressé.
6. Allumer la voiture.
7. Lâcher le bouton d'accouplement. La lumière rouge sous la voiture doit rester allumée.
8. Éteindre puis allumer la voiture et l'émetteur. Pour l'émetteur, presser le bouton **Stop** ou désactiver le bouton **Power** dans l'interface utilisateur de Labview.

La connexion peut ne pas bien s'établir pour différentes raisons :

- Un autre appareil est pairé en même temps. Dans ce cas, réitérer la procédure à un autre moment.
- Un autre appareil qui utilise une fréquence de 2.4GHz est proche. Il faut donc éteindre cet appareil.
- L'émetteur ou la voiture n'ont pas été éteints puis rallumés dans la dernière étape.

### 5.3 Télécommande intégrée

La commande utilisateur se fait par l'intermédiaire d'un Logitech Rumble GamePad visible sur la figure 33. Le joystick droit de la manette permet de gérer la vitesse de la voiture. Le joystick gauche quant à lui contrôle de l'orientation des roues. Néanmoins, la commande de direction envoyée par ce joystick à la voiture est erronée et fausse le contrôle de la voiture. Il serait intéressant de revenir sur le code pour résoudre le problème.

Le programme va établir une connexion avec la manette lors du lancement du programme grâce au bouton **Manual Command**. Il est important d'entrer au démarrage les valeurs maximales du joystick en allant chercher ses positions extrêmes, et ainsi éviter toute erreur de calibration lors de la prise en main.

Ensuite, la position du joystick est lue à chaque itération, puis convertie pour être transmise à la voiture sous les variables **Steering** et **Speed**.

La manette peut être désactivée à tout moment depuis l'interface utilisateur. Les commandes seront alors toujours lues et converties mais elles ne seront pas envoyées à la voiture.



Figure 33: Manette servant au contrôle de la voiture, Logitech Rumble GamePad F510 [5]

## 6 Contrôle

Dans ce projet, le contrôle de la voiture est possible sous deux formes via l'interface LabVIEW. Il se trouve dans le VI **main.vi**. Le premier cas est explicité à la section 5.3, l'utilisateur gère la trajectoire et contrôle la voiture par l'intermédiaire une manette. Le deuxième contrôle se fait par la création d'une trajectoire de référence, les commandes de la voiture sont ajustées pour qu'elle se conforme au mieux à ce parcours sans autre intervention.

Le deuxième contrôle sera exposé dans cette section, les étapes intermédiaires seront explicitées. Le principe est de suivre une trajectoire de référence définie au début du programme.

### 6.1 Modèle physique

Le contrôle de la voiture doit être réalisé en adéquation avec le modèle physique établi. Le modèle suivant est proposé, les calculs présentés ici sont issus d'un ancien rapport [6].

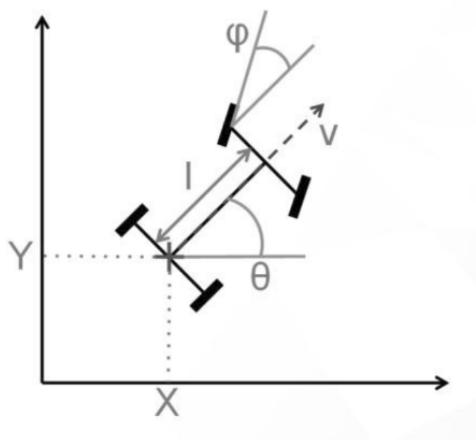


Figure 34: Représentation de la voiture dans l'espace 2D: l'origine représente la référence.

Ce modèle présente donc le modèle mathématique suivant:

$$\begin{cases} \dot{X} = v \cdot \cos(\theta) \\ \dot{Y} = v \cdot \sin(\theta) \\ \dot{\theta} = v \cdot \tan(u_2)/l \\ \dot{v} = K_v \cdot u_1 - K_s \cdot v \cdot |u_2| \end{cases} \quad (1)$$

où  $u_1$  et  $u_2$  sont la vitesse et la direction envoyées à la voiture. La vitesse de la voiture est comprise entre 0 et 5[m/s]. La direction est comprise entre  $-22^\circ$  qui est l'angle maximal des roues pour tourner à droite et  $22^\circ$  qui est l'angle maximal pour tourner à gauche.

L'angle de la voiture peut-être résumé par la formule suivante.

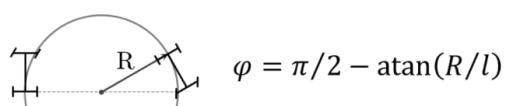


Figure 35: Formulation de l'angle de direction de la voiture.

### 6.1.1 Calcul de la trajectoire

- La trajectoire de référence de la voiture est définie par le biais de quatre points. Ces cinq points définissent là où la voiture doit passer. Le temps est aussi défini pour chaque point.
- Une évaluation de la trajectoire est ensuite faite grâce à une interpolation polynomiale. Il s'agit de définir une fonction polynomiale d'ordre quatre.

La fonction est déterminée grâce à la formule suivante:

$$\begin{bmatrix} x_0^4 & x_0^3 & x_0^2 & x_0 & 1 \\ x_1^4 & x_1^3 & x_1^2 & x_1 & 1 \\ x_2^4 & x_2^3 & x_2^2 & x_2 & 1 \\ x_3^4 & x_3^3 & x_3^2 & x_3 & 1 \\ x_4^4 & x_4^3 & x_4^2 & x_4 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \quad (2)$$

$$y(x) = a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x + a_0 \quad (3)$$

Une fois la fonction de trajectoire définie, les dérivées première et seconde sont établies à partir du polynôme. Cette trajectoire est calculée au début par le programme, puis elle sera réévaluée lors des itérations du contrôle. La voiture est supposée suivre cette trajectoire de référence.

### 6.1.2 Contrôleurs

Le contrôle s'articule en trois axes: un filtre de Kalman, un contrôleur "Jet scheduling" et enfin un PID pour la vitesse accompagné d'une commande linéaire pour la direction.

1. Les positions mesurées par Cortex sont traitées à travers un filtre de Kalman où l'état initial de la voiture est redéfini. Pour de plus ample information, se référer à l'annexe B.
2. En parallèle, la fonction de trajectoire et ses dérivées sont évaluées comme expliqué précédemment.
3. Ces deux informations sont transférées au contrôleur "Jet-Scheduling". Ce dernier calcule les tangentes qui seront utiles afin de coller au mieux à la trajectoire entrée par l'utilisateur. Pour plus de détails se référer au document [2].
4. La vitesse ainsi que la direction de référence sont enfin converties en commande. Pour la direction, la conversion est linéaire. La vitesse passe par un régulateur PID.

**Direction** La conversion de la direction ce fait ainsi:

$$\varphi_{com} = -\varphi_{ref} \cdot \frac{1.5}{0.384} + 1.5 \quad (4)$$

0.384[rad] correspond à l'angle maximal et 1.5 à la commande médiane.

**Vitesse** Pour la vitesse, l'erreur est quantifiée grâce à la référence et aux mesures prises par Cortex. L'erreur est définie par l'expression suivante:

$$e = |x_{ref} - x_{mes}| \quad (5)$$

Les anciennes valeurs de l'intégrateur et du déivateur sont conservées lors du calcul afin de les redéfinir. Les paramètres du contrôleur sont définis par l'utilisateur.

Le terme intégrateur s'exprime de la façon suivante:

$$u_i^i = u_i^{i-1} + \frac{h}{T_i \cdot e^i} \quad (6)$$

et le déivateur ainsi:

$$u_d^i = \frac{T_d}{1 + T_d} \cdot (u_d^{i-1} + (e^i - e^{i-1})) \quad (7)$$

pour obtenir la vitesse suivante:

$$v = K_p \cdot (e + u_i^i + u_d^i) \quad (8)$$

où  $K_p$ ,  $T_i$ ,  $T_d$  sont les constantes de contrôle entrées par l'utilisateur. Les  $i$  en exposant présentent si il s'agit d'un terme antécédent ou non.

Une commande ARW est ajoutée afin d'empêcher la présence d'un terme intégrateur trop grand ou trop petit qui saturerait la commande. Ainsi, si la commande est trop grande ou trop petite, celle-ci est mise à son maximum ou minimum et le terme intégrateur est redéfini de cette façon:

$$u_i = \frac{v_{max,min}}{K_p - e - u_d} \quad (9)$$

## 7 Améliorations

À la fin de ce projet, les problèmes suivants ont été observés. Pour des raisons de temps et de logistique, les améliorations proposées dans les prochaines sections n'ont pas pu être réalisées.

### 7.1 Architecture

La plupart des problèmes qui ont été rencontré lors de ce projet sont principalement des problèmes d'architecture. C'est la cause principale du retard, elle rend vain le contrôle. Une liste non exhaustive des éléments à améliorer lors d'un futur projet est présentée ci-dessous. Des compteurs de boucle ont été ajoutés au sein du code afin de contrôler l'exécution des boucles for.

- Le code est actuellement sur LabVIEW 2011. Une mise à jour du programme sera nécessaire pour la suite de ce projet. Il faudra néanmoins faire attention à ce que le programme soit bien traduit dans la nouvelle version.
- La boucle de contrôle pour la manette influence les performances du code lors de son utilisation. Elle tourne trop vite par rapport à ses capacités réelles, ce qui provoque une surcharge de tâches à traiter et provoque du retard dans le programme. Une attente a été implémentée afin de lui permettre de tourner mais le retard reste trop important.
- La boucle de contrôle de la manette est tout le temps active même quand elle n'est pas utilisée. L'empêcher d'être en route lorsqu'elle n'est pas utilisée permettrait un gain de performance.
- Il faut aussi optimiser la boucle de contrôle de la voiture afin d'éviter les retards sur la commande. Pour ce faire, il faudrait donc séparer l'envoi des commandes ainsi que la réception des données. Il est fort probable que les deux actions se font simultanément, car elles ne sont pas reliées entre elles par un fil d'erreur. Actuellement, une attente est imposée afin d'éviter que le programme n'ait trop de données à gérer. La voiture a donc tendance à accélérer puis s'arrêter net.
- Le programme de contrôle n'est pas écrit de façon optimale. En effet, certain transfert de données se font par des boucles alors que l'utilisation de l'index améliorerait les performances de ce dernier.

### 7.2 Contrôle

- Certains problèmes de contrôle peuvent être dus à une mauvaise calibration de la voiture, qu'il est conseillé de régulièrement vérifier.
- Les valeurs de PID et du ARW n'ont pas été testées. Il serait assez intéressant de les tester afin de rendre le modèle plus robuste.
- La présence d'un terme déivateur peut provoquer de fortes instabilités. Mettre un contrôleur PI permettrait de garantir au mieux cette stabilité. Il serait donc nécessaire de passer par un diagramme de Nyquist.
- Lorsque la boucle est lancée, la voiture reçoit bien les commandes, mais ses dernières ne sont pas assez puissantes pour vaincre les frottements et faire avancer la voiture. Le problème semble toutefois plus du fait des voitures que du contrôle.

- Le code fonctionne grâce aux trajectoires qui sont calculées tout au long du programme. Dans le programme originel, la trajectoire doit être calculée grâce à la photo du circuit. La construction de la trajectoire doit encore être approfondie. Le départ étant fourni par Cortex et l'arrivée par l'utilisateur.
- Dans le programme originel ("Control Platform.vi"), les données de chacune des boucles étaient sauveées afin de les analyser. Pour affiner le contrôle, cette fonction pourrait être utile à ajouter au code.
- Si par la suite, l'utilisateur désire réaliser le circuit, il devra se munir d'un caméra supplémentaire pour détecter le chemin à parcourir. La boucle est présente dans un ancien code ("Control Platform.vi"). Il faut faire attention à ce que la bonne librairie soit utilisée (ici "LV\_PS3\_Eye\_USB.framework") sinon le programme va planter.

### 7.3 Cortex

Suite à plusieurs tests, la détection des marqueurs sur la voiture semble prendre un peu trop de temps et être un peu fluctuante. Cette latence est probablement causée soit par la taille des marqueurs, soit par les réglages physiques de la caméra. Il faudrait tester le contrôle avec des marqueurs plus imposants, si le problème persiste modifier les réglages des caméras et recalibrer le système de vision.

## 8 Autres pistes possibles d'asservissement

Dans cette partie sont exposées les alternatives qui ont été envisagées mais non explorées pour établir le contrôle de la voiture. Il est souhaité ici de présenter d'autres pistes de réflexion aux prochains étudiants et de permettre à terme de contrôler la voiture depuis sa position initiale à un point de référence en suivant le parcours du circuit.

### 8.1 Première approche : Cas simplifié

Pour arriver à terme à un contrôle optimal de la voiture, le modèle d'asservissement doit être construit par étape.

Dans un premier temps un cas simple a été considéré : une marche avant depuis sa position jusqu'à un point de référence, la voiture est supposée orientée dans la bonne direction. Pour affiner la commande et gagner en précision, il est envisageable de définir différents cas et de définir un facteur correctif proportionnel. La vitesse serait ajustée en fonction de la norme de la distance entre la voiture et la position de référence. La norme serait régulièrement recalculée et la vitesse adaptée à chaque itération. La voiture connaîtrait une phase d'accélération au-delà d'une certaine distance et une phase de décélération en deçà de cette limite.

### 8.2 Deuxième approche : Contrôle par arcs de cercles

Une autre option, pour permettre à la voiture de rejoindre la position de référence, serait de calculer à chaque itération un arc de cercle que la voiture doit effectuer depuis sa position. Il faudrait déterminer le rayon et l'angle à chaque fois, puis en déduire le rayon de courbure à suivre et corriger la vitesse et la direction par un facteur proportionnel défini empiriquement.

## 9 Conclusion

La communication entre les caméras et LabVIEW en passant par Cortex a été rétablie et fonctionne. La démarche à suivre a été explicitée tout au long du rapport. Des conseils techniques ont également été présentés, pour résoudre les problèmes qui ont été ou qui pourraient être rencontrés. Cortex est un outil d'acquisition et d'édition de capture du mouvement qui s'est révélé puissant, il transmet des données de bonne qualité en temps réel. Toutes les étapes de la capture sont gérées dans Cortex : configuration initiale, calibration, suivi et post-traitement. Le processus d'identification unique de Cortex est un moyen rapide et flexible d'identifier les données de capture de mouvement.

Le logiciel LabVIEW propose une approche de programmation graphique qui aide à visualiser tous les aspects du projet, y compris la configuration de la télécommande, la communication avec Cortex, la gestion des données envoyées par ce dernier et la mise au point du contrôle. Cette visualisation facilite l'intégration au matériel de mesure de n'importe quel fournisseur, le développement des algorithmes d'analyse de données et permet de concevoir des interfaces utilisateurs personnalisées.

Plusieurs perspectives d'amélioration s'ouvrent à la fin de ce projet de Bachelor. Les principales modifications à apporter sont d'ordre architecturale. Les boucles sur LabVIEW connaissent en effet trop de latence rendant ainsi le contrôle quasiment impossible. Une fois seulement le code LabVIEW rectifié, les différentes options de contrôle pourront être testées et affinées. Ensuite le fichier ".trc "envoyé par Cortex affiche des espaces vides dans les positions des markers quand il ne détecte pas bien ces derniers. Il y a différentes manières de palier à ce problème. Les caméras sont accrochées très haut par rapport à la taille de l'objet observé et surtout que le mouvement se fait dans un plan au sol. Elles pourraient être posées sur des pieds à fin de n'être qu'à un peu plus d'un mètre du sol, les risques qu'elles soient heurtées seraient plus grands il faudrait donc recalibrer plus souvent, mais elle distinguerait mieux l'objet. Une autre possibilité est d'utiliser de plus gros markers, mais la carrosserie de la voiture a une taille limitée. Il est également possible de gérer ces valeurs vides depuis le LabVIEW (LabVIEW affiche 1E-7 quand il n'arrive pas à lire une valeur), en signalant un fichier trop parsemé ou en complétant ces valeurs par d'autres interpolées ou bien encore en construisant un nouveau vecteur dont elles ont été retranchées.

## References

- [1] Motion Analysis. *Cortex, Version 2.5 Quick-Start Guide*. Motion Analysis, 2011.
- [2] Davide Buccieri. “Jet-scheduling control for flat systems”. PhD thesis. École Polytechnique Fédérale de Lausanne, 2008. URL: [https://infoscience.epfl.ch/record/114764/files/EPFL\\_TH3996.pdf?version=1](https://infoscience.epfl.ch/record/114764/files/EPFL_TH3996.pdf?version=1).
- [3] Kyosho. *DNaNo FX-101 Series Complete Chassis Set*. URL: [http://www.kyosho.com/jpn/support/instructionmanual/dnano/pdf/DNFX\\_T13\\_dNaNo\\_CCS\\_FX\\_101\\_m.pdf](http://www.kyosho.com/jpn/support/instructionmanual/dnano/pdf/DNFX_T13_dNaNo_CCS_FX_101_m.pdf).
- [4] Kyosho. *PERFEX, Digital Proportional radio Control System, KT-18*. URL: [https://www.kyosho.com/jpn/support/instructionmanual/digital\\_prcs/pdf/82001E\\_PERFEX\\_KT-18\\_IM.pdf](https://www.kyosho.com/jpn/support/instructionmanual/digital_prcs/pdf/82001E_PERFEX_KT-18_IM.pdf).
- [5] Logitech. *Rumble Gamepad F510*. URL: [https://support.logitech.com/en\\_us/product/rumble-gamepad-f510](https://support.logitech.com/en_us/product/rumble-gamepad-f510).
- [6] Reza Safai-Naeeni. *Real time control of race cars from a vision system*. 2011.
- [7] Wikipedia. *Kalman Filter*. URL: [https://en.wikipedia.org/wiki/Kalman\\_filter](https://en.wikipedia.org/wiki/Kalman_filter).

# Appendices

## A Présentation des différents VI

### A.1 Programme principal

leône	Nom	Appelle	Définition
	Setup Remote Control.vi	-	Définie les sorties de la voiture : la vitesse, la conduite et les interrupteurs.
	Reset.vi	-	Remet à zéro les interrupteurs, la conduite et la vitesse.
	Pair.vi	-	Établie la connexion avec la voiture.
	Send to Remote Control.vi	-	Envoie les commandes aux sorties de la voiture.
	Close Remote Control.vi	-	Arrête et initialise les commandes de sortie.
	Setup Cortex.vi	« Cortex Decode PK.vi »	Ouvre la connexion UDP avec l'adresse IP du second ordinateur.
	Read Cortex.vi	« Cortex Decode PK.vi »	Établie et lis les données envoyées par la connexion UDP. Il retourne Segments.
	Close Cortex.vi	-	Ferme la connexion UDP avec l'autre ordinateur.
	Elapsed Time.vi	-	Sort le temps écoulé depuis que le système est en route.
	Multi Car Control.vi	« EKF Multi Car.vi » « Steering Controller.vi » « Velocity Controller Multi Car.vi » « Jet-Scheduling Controller Multi Car.vi »	Contrôle la voiture. Il comprend un filtre de Kalman, plusieurs autres contrôleurs.
	External Command.vi	-	Active la commande de la voiture et transmet les trajectoires de référence.
	New Trajectories.vi	-	Remplace la trajectoire de référence par l'état de la voiture.
	Init Trajectories.vi	« Compute Reference Trajectory.vi »	Calcule les trajectoires de références grâce à des points définis dans le temps et l'espace.

Figure 36: Tableau résumant la fonction de chacun des VIs dans la boucle principale, pour le contrôle de la voiture. Il faudra sûrement séparer les VIs propres à Cortex et ceux de la Remote Control

Icône	Nom	Appelle	Définition
READ	Read Gamepad.vi	« HID_Read.vi » « Conv_255-3.3.vi »	Lit les commandes envoyées par la manette.
SETUP	Setup Gamepad.vi	« HID_FindProduct.vi» « HID_GetDevices.vi»	Mets en place la connexion avec la manette.

Figure 37: Tableau résumant la fonction des VIs pour les commandes reçues par la manette. Cette boucle tourne trop vite par rapport à ce qu'elle est capable de faire.

## A.2 Sous VI

Icône	Nom	Définition
255 3.0	Conv_255-3.3.vi	Convertie un nombre avec 8 bits en un nombre compris entre 0 et 3.
Read HID	HID_Read.vi	Lis une valeur spécifique propre au dispositif choisi.
Prod. Q + Dev HID	HID_FindProduct.vi	Cherche un système spécifique et retourne la liste.
[Dev] HID	HID_GetDevices.vi	Retourne la liste des dispositifs HID connectés ainsi que leurs informations associées.

Figure 38: Tableau résumant la fonction de chacun des VIs présent dans "Read Gamepad.vi" et "Setup Gamepad.vi"

Icône	Nom	Définition
	Evaluate Reference Trajectory Time.vi	Evalue la trajectoire de référence à un temps données
	EKF Multi Car.vi	Implémente un filtre de Kalman.
	Steering Controller.vi	Contrôle la direction linéairement
	Velocity Controller Multi Car.vi	Contrôle de la vitesse grâce à un PID
	Jet-Scheduling Controller Multi Car.vi	Implémente un contrôleur « jet-scheduling » pour plusieurs voitures.

Figure 39: Tableau résumant les différents contrôleurs et fonctions dans "Multi Car Control.vi". Cette dernière devra être optimisée au niveau de son architecture ainsi qu'au niveau du contrôle.

Icône	Nom	Définition
	Cortex Decode PK.vi	S'occupe d'extraire les données envoyées par la connexion UDP avec Cortex

Figure 40: Tableau résumant la fonction du VI "Cortex Decode PK.vi" appelé par "Setup Cortex.vi" et "Read Cortex.vi". Les VIs qui sont présent dans ce VI mais non expliqués ne sont pas utilisés.

Icône	Nom	Définition
	Extract I32.vi	Extrait d'un string un entier non-signé avec 32 bits ce qui correspond à 9 décimales
	Cortex Get DOFs.vi	Prends les degrés de liberté de chacun des markers sur la voiture
	Cortex Get Segments.vi	Prends les coordonnées du segment représentant la voiture
	Extract SGL.vi	Extrait d'un string un nombre à virgule flottante à simple précision
	Cortex Get Markers.vi	Prends la position dans l'espace de chacun des markers sur la voiture
	Extract CStr.vi	Extrait d'un string un « Single-Precision Floating-Point »
	Extract DBL.vi	Extrait d'un string en un double.
	ExtractI16.vi	Extrait un entier non-signée avec 16 bits ce qui correspond à 4 décimales.

Figure 41: Tableau des fonctions qui sont appelées par "Cortex Decode PK.vi"

Icône	Nom	Définition
	<i>HID_BuildFrameworkError.vi</i>	Construit un message d'erreur compréhensible.
	<i>HID_Init.vi</i>	Initialise la librairie HID et retourne le nombre de dispositif HID reconnu.
	<i>HID_GetDeviceInfo.vi</i>	Retourne les informations « DeviceIdx » du dispositif. Retourne le nombre associé d'élément.
	<i>HID_GetDeviceElemInfo.vi</i>	Retourne les informations sur “DeviceIdx/DeviceElemIdx” du dispositif.

Figure 42: Tableau résumant la fonction des VIls appelés par "HID\_Read.vi", "HID\_FindProduct.vi" et "HID\_GetDevices.vi"

## B Filtre de Kalman

Le filtre de Kalman est une méthode de contrôle itérative qui se base sur une connaissance du modèle physique et sur la mesure en temps réel du système. Elle se présente de la façon suivante. L'état initial du système est connu ainsi que les paramètres initiaux. Ses paramètres sont ensuite implémentés dans le modèle physique qui est comparé à la mesure. Il peut donc être résumé par la figure 43.

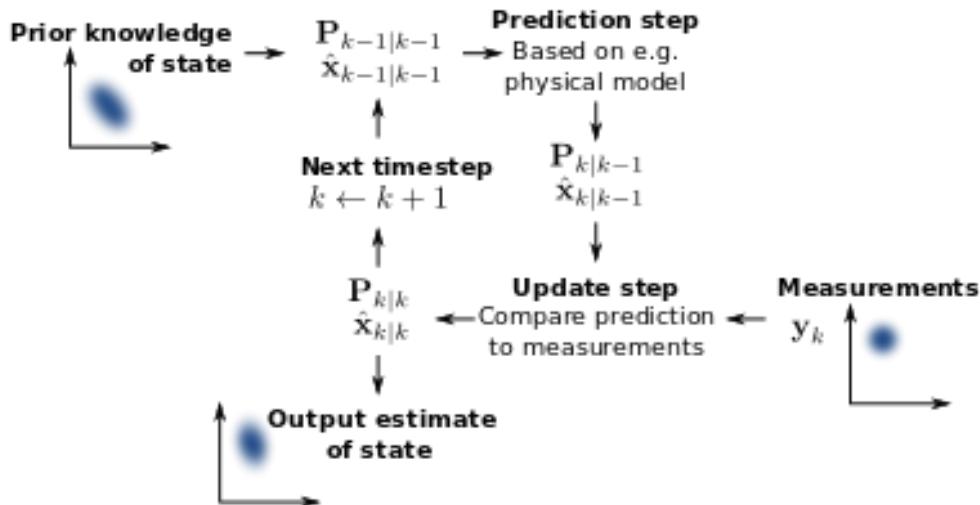


Figure 43: Schéma explicatif du filtre de Kalman [7].