M1 E3A

Stage de recherche

Prise en main du capteur C-Track

PIERRE-ALEXANDRE PEYRONNET

9 JUILLET 2022

Table des matières

1 Introduction **2**

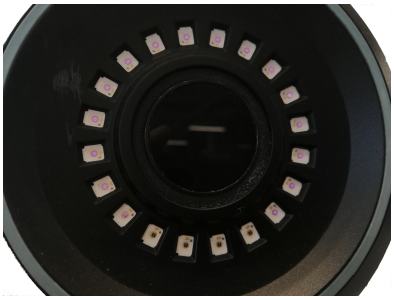
2 Installation **2**
2.1 Installation physique 2
2.2 Installation logicielle 5

3 Utilisation **7**
3.1 Calibrage 7
3.2 VXtrack 8
3.3 API 10

1 Introduction

Le capteur C-Track est un capteur permettant de déterminer la position d'un objet dans l'espace, à l'aide de mires réfléchissantes (figure 1a). Ce capteur émet des rayons infrarouges qui seront réfléchis par les mires, deux caméras permettent alors de déterminer la position des mires dans le référentiel du C-Track (figure 1b).

(a) Mires



(b) LEDs infrarouge et caméra

FIGURE 1 – Acteurs du positionnement via C-Track

2 Installation

L'installation du capteur C-Track se fait en deux étapes : une installation physique qui consiste au câblage des différents éléments, et une installation "logicielle" qui correspond à l'installation du logiciel et au paramétrage du système.

2.1 Installation physique

Le C-Track se trouve normalement dans sa malle (figure 2), où l'on retrouve aussi les CD ou clé USB d'installation de VXelements, la barre de calibrage, un tournevis, des câbles et d'autres objets (figure 3).

FIGURE 2 – Malle du C-Track fermée

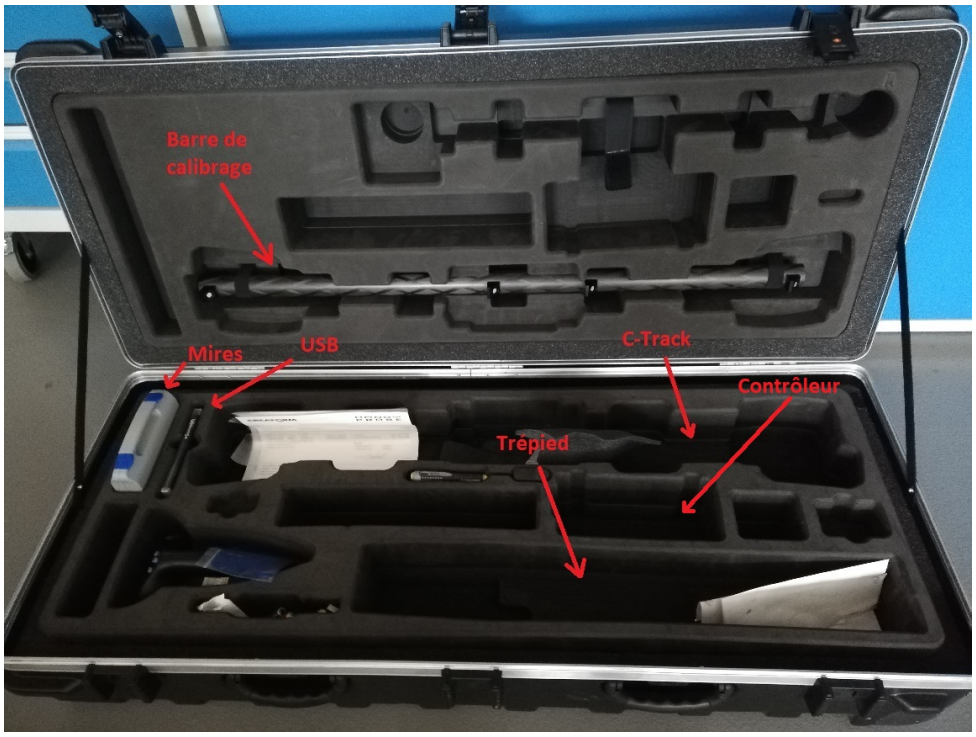


FIGURE 3 – Contenu de la malle du C-Track

Pour installer physiquement le C-Track, il faut d'abord le positionner à l'endroit souhaité, on choisira donc plutôt une position qui met le capteur en face des mires. Pour cela, on a à notre disposition un trépied muni d'un niveau à bulle, permettant de placer le capteur où on le souhaite, à la hauteur que l'on veut et le plus droit possible (figure 4).

FIGURE 4 – C-Track positionné

Ceci fait, il faut câbler le C-Track à son contrôleur (figure 5). Le contrôleur est le

FIGURE 5 – Contrôleur C-Track

centre du système, c'est lui qui fait la liaison entre le réseau électrique, votre ordinateur et le capteur. C'est donc à partir de ce dernier que tous les câbles vont être connectés (figure 6), on retrouve :

- Un câble d'alimentation « Réseau électrique ⟶ Contrôleur » ;
- Un câble d'alimentation « Contrôleur ⟶ C-Track » ;
- Un câble Ethernet « Contrôleur ⟷ PC » ;
- Deux câbles FireWire « Contrôleur ⟷ C-Track » ;

Les câbles d'alimentation vont évidement servir à fournir la puissance aux appareils, le câble Ethernet permet la communication entre votre ordinateur et le contrôleur, et les FireWire permettent la communication entre le contrôleur et le C-Track.

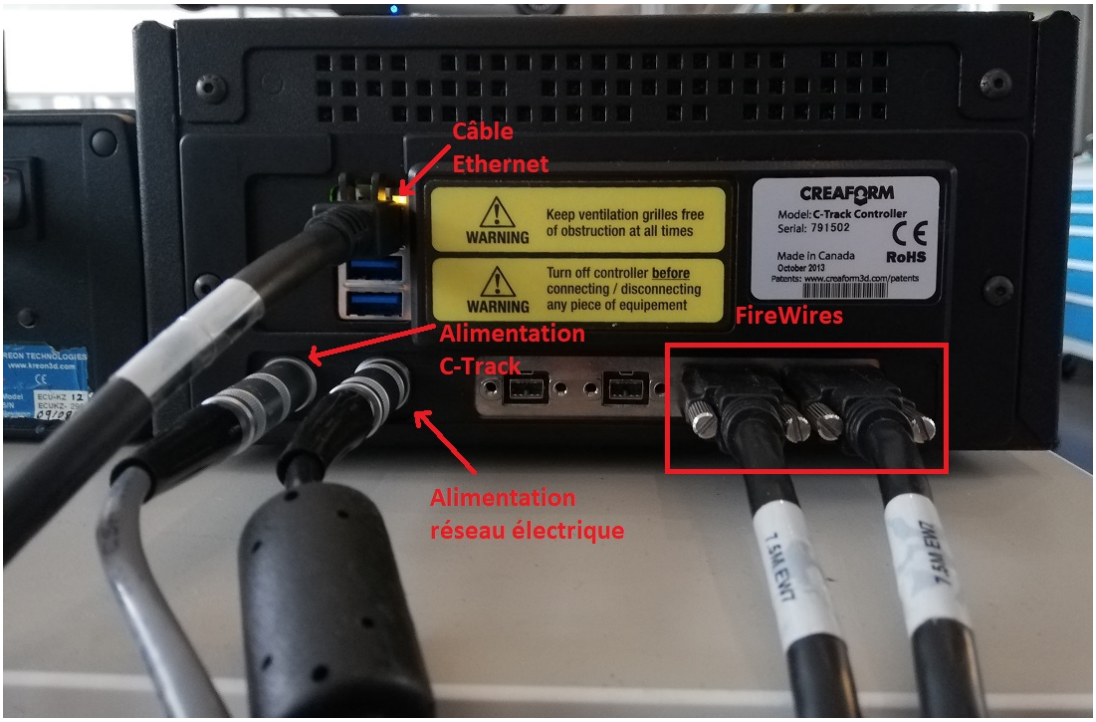


FIGURE 6 – Câblage du contrôleur

On peut désormais allumer le contrôleur à l'aide du bouton « Démarrer », il va alors attendre pendant quelque minutes que le C-Track soit échauffé (figure 7), puis il sera utilisable.

FIGURE 7 – Démarrage du contrôleur

2.2 Installation logicielle

Après l'installation physique, il faut procéder à l'installation des logiciels et à la configuration de votre ordinateur.

Dans cette partie, on supposera que le logiciel VXelements est déjà installé sur votre ordinateur et que la licence est valide. Si ce n'est pas le cas et que l'installation pose problème, référez vous à la documentation précédente de Matthias Bordron ou à une personne compétente.

Il faut ensuite paramétrer le réseau associé à la connexion Ethernet, afin que le PC puisse retrouver le contrôleur. Pour cela, faites la commande « Windows + R » et entrez « ncpa.cpl » (figure 8), ceci ouvrira la page des connexions réseau du PC (Wi-Fi et Ethernet). Il faut ensuite aller dans les propriétés de la connexion Ethernet correspondante au

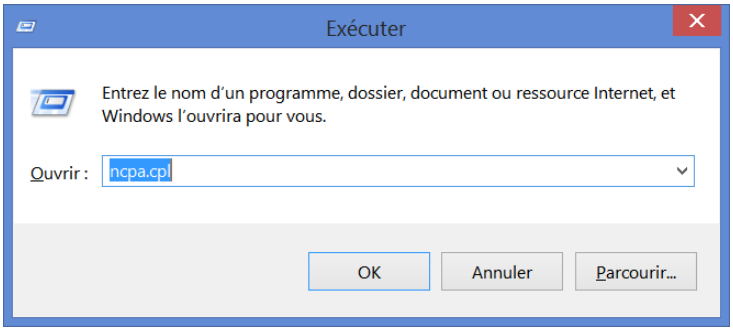


FIGURE 8 – Fenêtre « Exécuter »

contrôleur, puis dans les propriétés de l'élément « Protocole Internet Version 4 (TCP/IPv4) ». Dans l'onglet « Général » on doit alors sélectionner le bouton radio « Utiliser l'adresse

IP suivante : », puis entrer « 200.200.200.1 » en adresse IP, et « 255.255.255.0 » en masque de sous-réseau (figure 9).

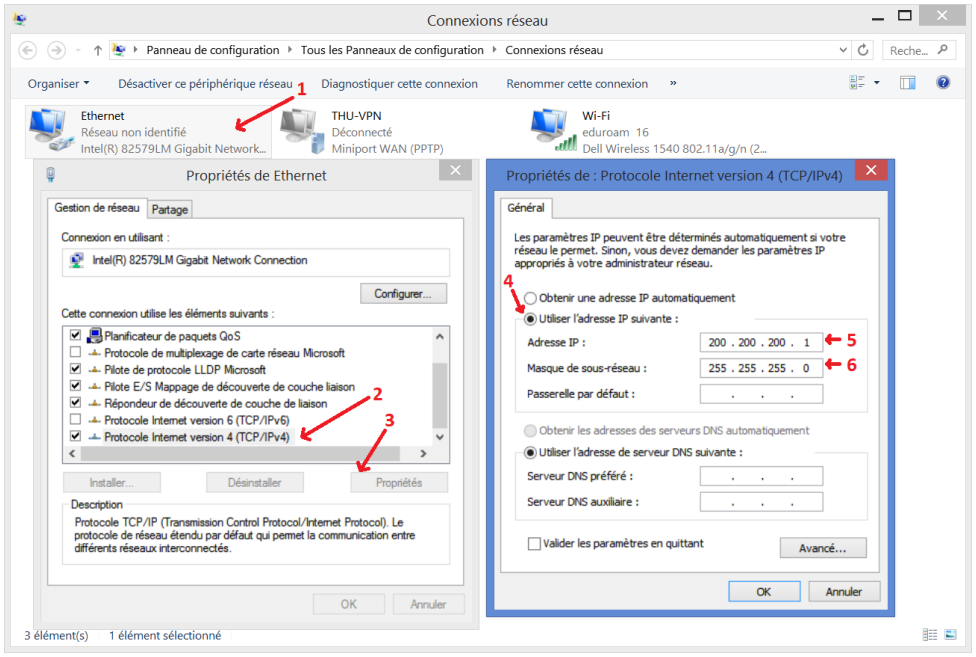


FIGURE 9 – Changement de l'adresse IPv4

Afin de vérifier la bonne connexion entre les deux éléments, on peut essayer d'envoyer des paquets de données au contrôleur. Il faut alors ouvrir un terminal (« Windows + R » et entrez « cmd »), puis écrire la commande « ping » suivie de l'adresse IP du contrôleur, à savoir « 200.200.200.2 ». Si vous recevez des paquets, la connexion fonctionne (figure 10).

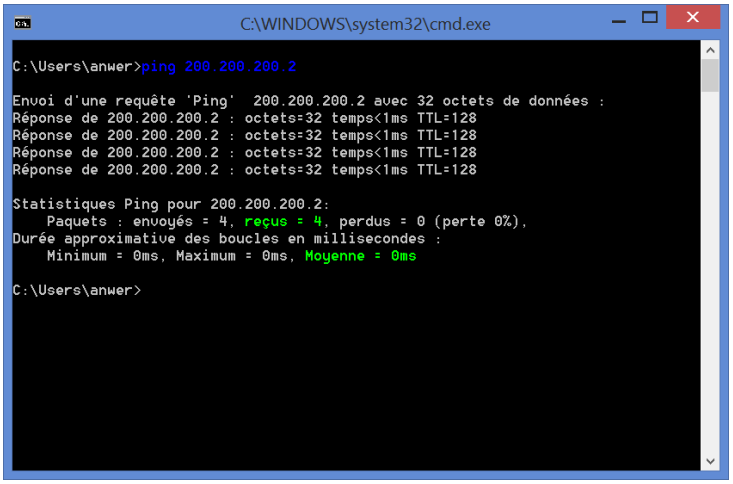


FIGURE 10 – Requête ping au contrôleur

3 Utilisation

L'utilisation du C-Track se fait de deux façon : avec le logiciel VXelements (figure 11) qui propose une interface graphique permettant d'observer les mires dans le référentiel du C-Track, et avec une API contenant des bibliothèques utilisables en C#, C++ ou Visual Basic. Cette dernière permet d'utiliser les fonctionnalités de VXelements dans un programme que l'on aura soi-même écrit.

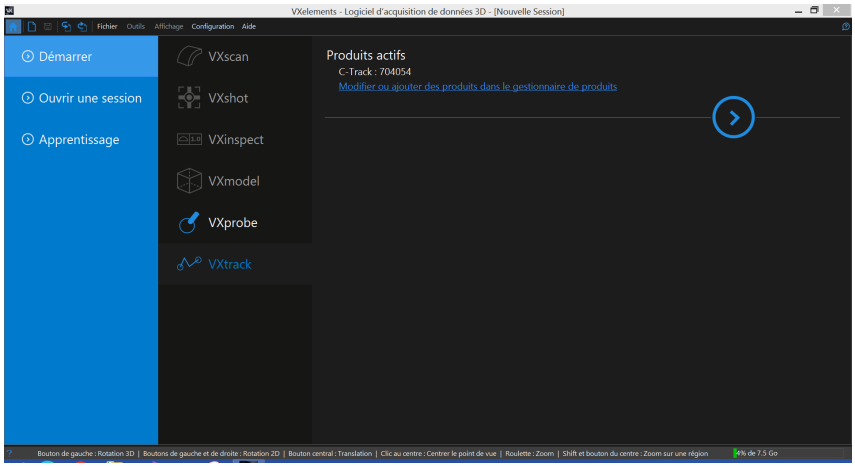


FIGURE 11 – Menu principale de VXelements

3.1 Calibrage

Avant de passer à l'utilisation à proprement parler, il faut calibrer le C-Track. Cela se fait environ toutes les deux semaines via le logiciel VXelements, il faut démarrer « VXtrack », cliquer sur l'onglet « configuration », « C-Track » et « Étalonnage » (figure 12). Il faut ensuite choisir le volume dans lequel on souhaite travailler, ainsi que la barre

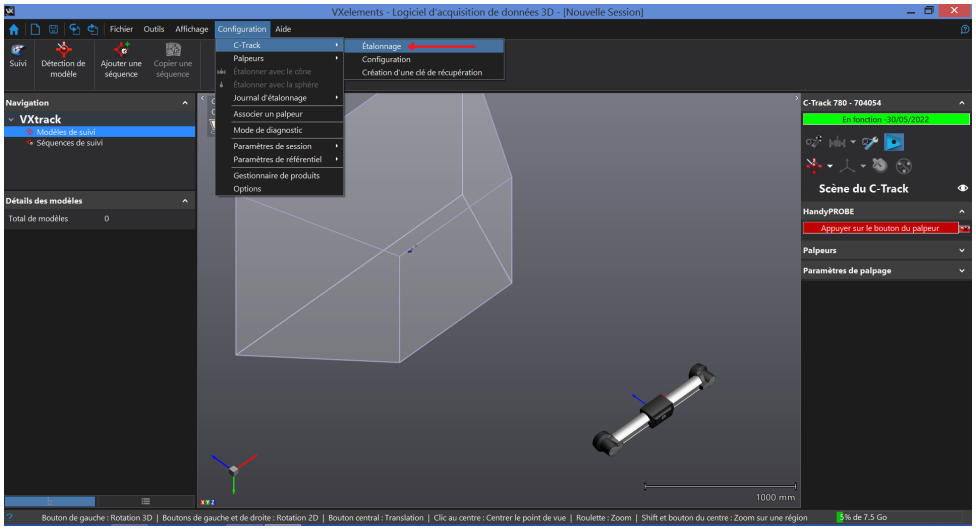
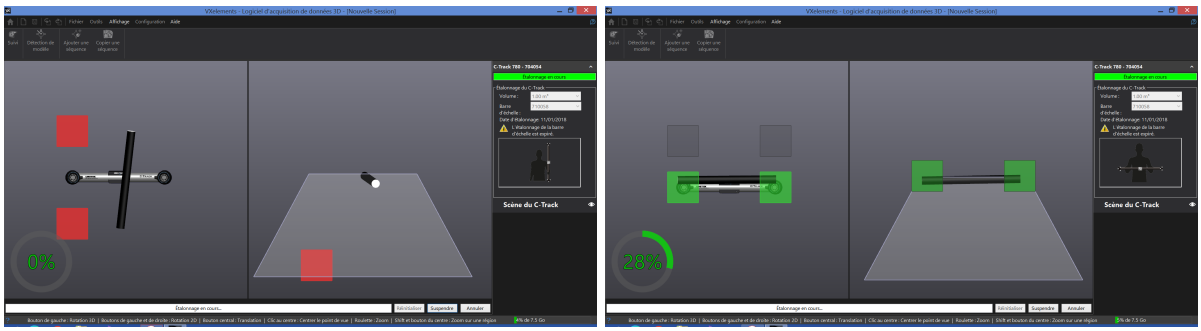


FIGURE 12 – Ouverture du menu d'étalonnage

d'étalonnage, normalement « 710058 » dans notre cas. On doit ensuite placer nous-même cette barre à différents points de l'espace, indiqués par l'application, jusqu'à atteindre une progression de 100 %, le capteur sera alors calibré (figures 13).



(a) Calibrage en attente (b) Cible atteinte

FIGURE 13 – Calibrage du C-Track

3.2 VXtrack

Toujours dans le logiciel VXelements, le calibrage nous permet désormais d'observer les mires dans l'espace. Pour cela, il suffit de démarrer VXtrack, puis on peut se lancer directement à la recherche des mires, mais la vue initialement proposée n'est pas forcément pratique. Il vaut mieux alors cliquer sur l'icône « Configuration du C-Track », qui va afficher les C-Track, ainsi que le volume de travail. Ceci permet aussi de faire apparaître des boutons permettant de sélectionner différents angles de vue, notamment la « Vue isométrique », qui permet d'avoir une bonne représentation de l'espace de travail (figure 14). Ceci fait, vous êtes maintenant capable d'observer correctement les mires détectées

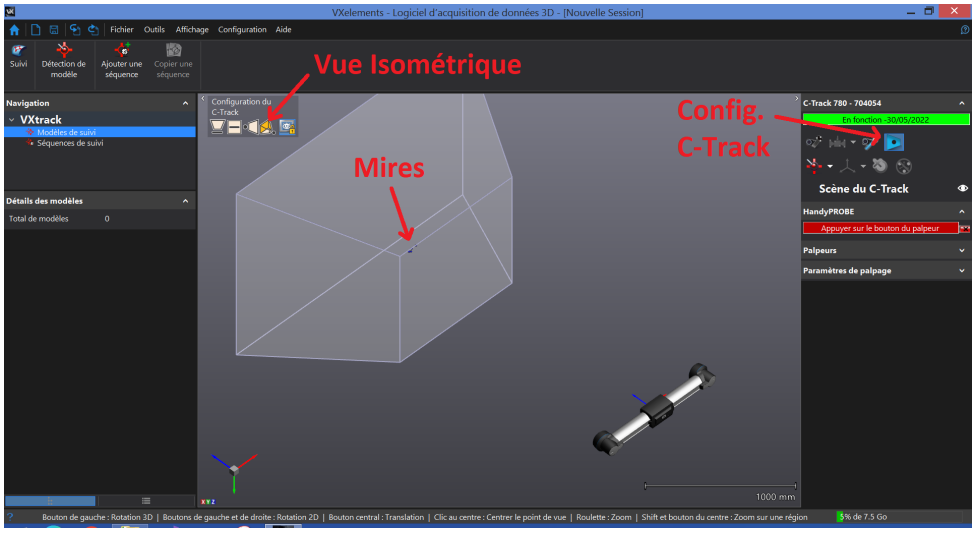


FIGURE 14 – Passage en vue isométrique

par le C-Track.

On peut ensuite démarrer une séquence de suivi, afin d'enregistrer les coordonnées des mires au cours du temps, puis éventuellement les enregistrer dans un fichier CSV. Il

suffit de cliquer sur le bouton « Suivi » situé en haut à gauche de l'écran pour démarrer une séquence, on peut alors observer un tableau de coordonnées se remplir au cours du temps (figure 15). Une fois la séquence arrêtée, on peut l'exporter en cliquant droit dessus

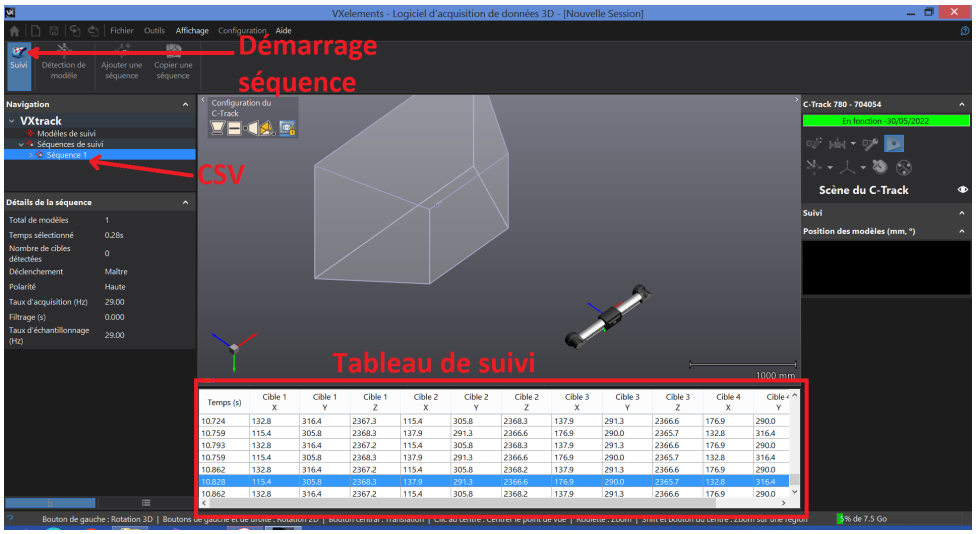


FIGURE 15 – Séquence de suivi des mires détectées

puis en choisissant « Exporter la séquence au format CSV ».

Plutôt qu'avoir les coordonnées des mires, on peut créer un « Modèle de suivi » qui est en fait un objet constitué de plusieurs mires. Cliquez sur « Détection de modèle », puis sélectionnez les mires de votre objet en cliquant dessus, elles deviendront rouges, puis acceptez la sélection (figure 16). Désormais, lorsque vous cliquerez sur « Suivi »,

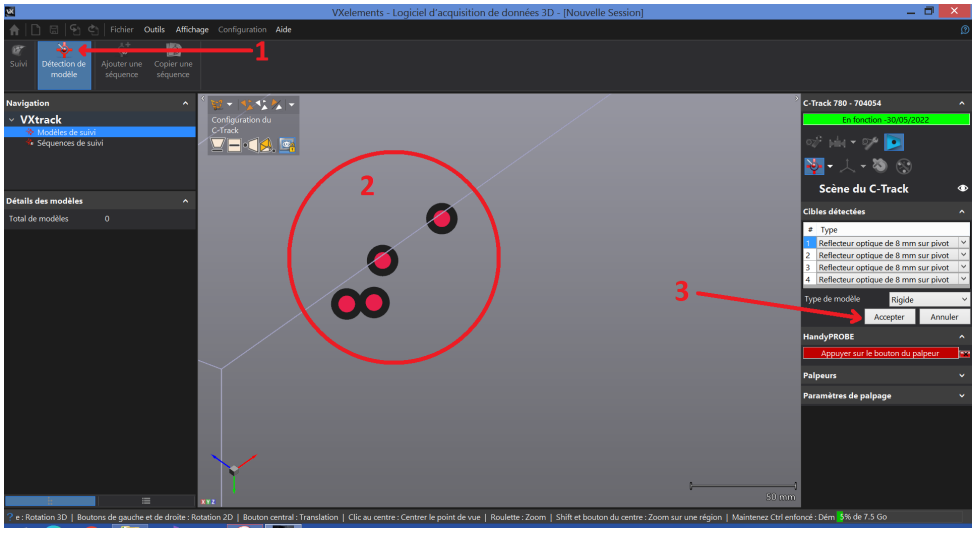


FIGURE 16 – Création d'un modèle

vous observerez les coordonnées de l'objet dans le référentiel du C-Track. Vous pourrais d'ailleurs choisir parmi trois vues, en cliquant droit sur le graphique apparu : graphique (figure 17), tabulaire et projection.

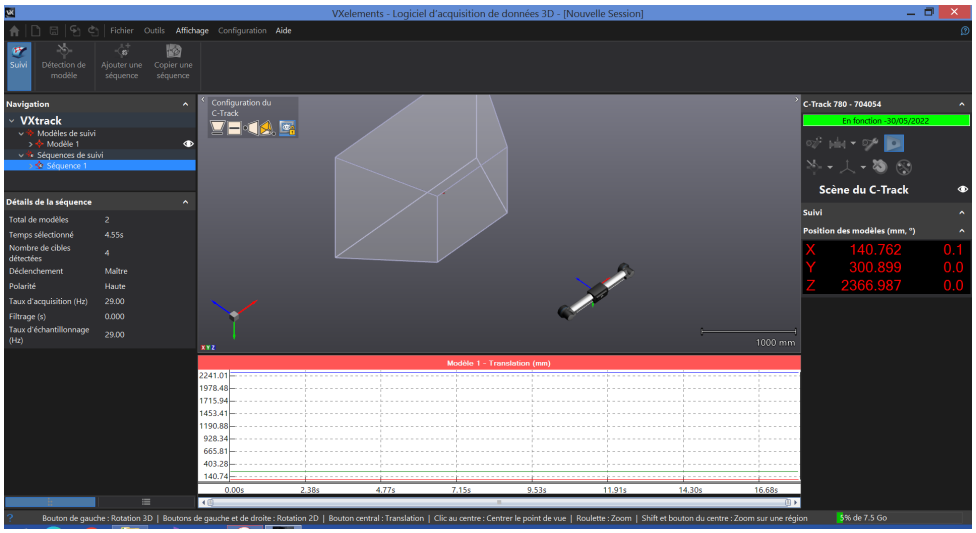


FIGURE 17 – Séquence de suivi d'un modèle

3.3 API

Enfin, pour utiliser les fonctionnalités de VXelements au travers de votre propre programme, vous pouvez intégrer l'API contenu dans le fichier « VXelementsAPI.dll » à un projet C++, C# ou Visual Basic. Cette API utilise .NET Assembly et C++/CLI, ce qui conditionne son utilisation. Une solution simple est d'utiliser l'IDE Visual Studio, ce qui permettra d'utiliser les trois langages et d'installer toutes les composantes nécessaires au bon fonctionnement de l'API. Dans le logiciel « Visual Studio Installer », vous aurez trois composantes à installer : « Développement en .NET Desktop », « Développement Desktop en C++ » et la composante individuelle « Prise en charge de C++/CLI » (figures 18 et 19).

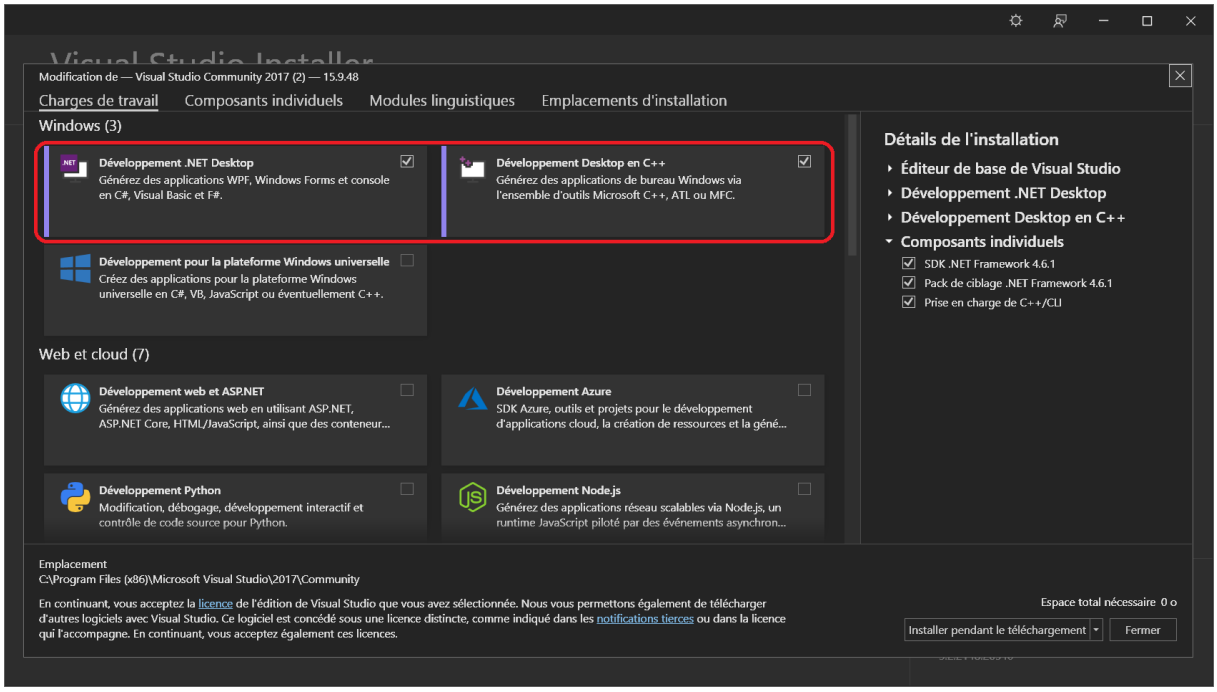


FIGURE 18 – Composantes C#, VB et C++

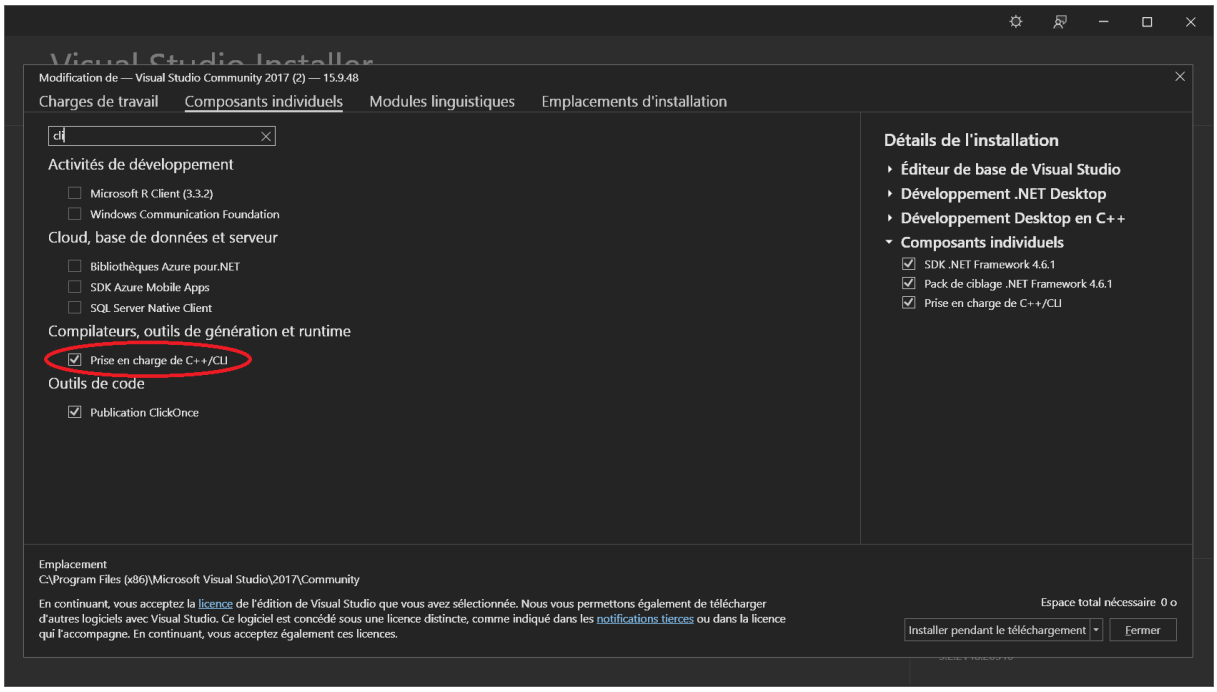


FIGURE 19 – Composantes C++/CLI

Ceci fait, dans le cas du C++, vous pouvez maintenant créer un projet de la catégorie « Application console CLR » (figure 20), il ne restera plus qu'à intégrer l'API.

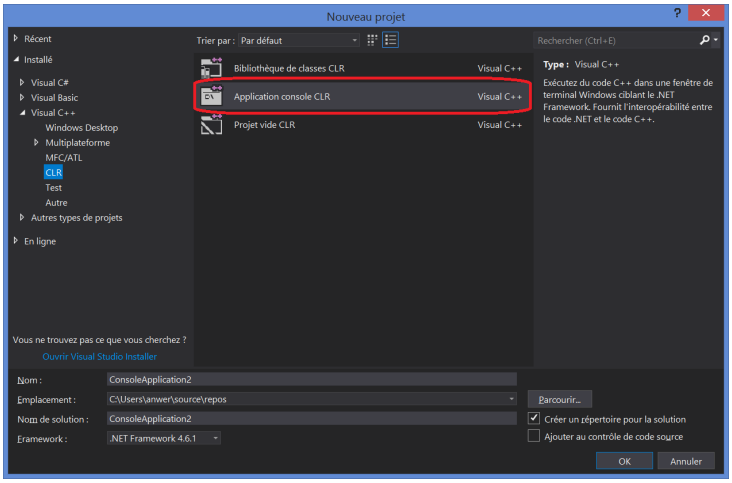


FIGURE 20 – Création d'un projet

Placez ensuite le fichier VXelementsAPI.dll dans le dossier du projet, puis sur Visual Studio dans l'explorateur de solution, cliquez droit sur Référence et ajoutez une référence (figure 21). Vous pourrez sélectionner l'API en cliquant sur « Parcourir ».

Ceci fait, vous aurez accès aux `namespace` permettant d'utiliser les fonctions de VXelements. Pour commencer, il faudra indiquer dans les 1re lignes de votre programme les directive et instruction présentés dans le code 1. Dans la fonction `int main()`, la première chose à faire est ensuite d'appeler la fonction `ApiManager::Connect()`, qui va enclencher une connexion entre le programme et VXelements. Attention, cette étape ne

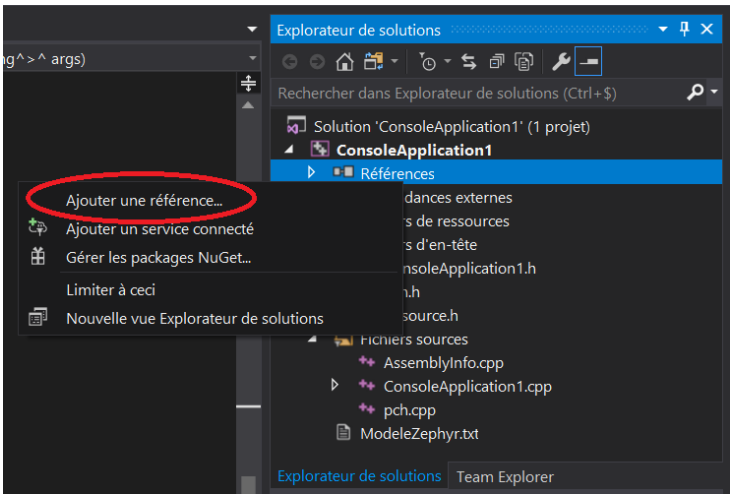


FIGURE 21 – Ajout de l'API au projet

```
#include <msclr/auto_gcroot.h>

using namespace VXelementsAPI;
```

CODE 1 – Appel de l'API

fonctionnera pas si une tâche nommé « VXelementsApiImplementation » est en cours d'exécution, il faudra la fermer à chaque fois que l'on exécute notre programme.

On peut ensuite déclarer des objets associés à VXelements (code 2). Le 1er objet

```
msclr::auto_gcroot<IVXelements^> mVXelements;
msclr::auto_gcroot<VXtrack::IVXtrack^> mVXtrack;
msclr::auto_gcroot<VXtrack::ITrackingSequence^> pSequence;

mVXelements = ApiManager::VXelements;
mVXtrack =  mVXelements->VXtrack;
```

CODE 2 – Déclaration des variables

contiendra l'application VXelements en cours d'exécution, on y récupère ensuite les fonctionnalités de VXtrack. On peut alors au choix détecter un modèle à la main, ou importer un modèle pré-enregistré. Pour la détection à la main, il faut utiliser la méthode `mVXtrack->DetectModel()`, il faudra alors créer un modèle comme on l'a vu dans la partie précédente. Lors de cette étape, il faut temporiser le programme de sorte à ne pas lever une exceptions via les lignes suivantes, pour cela on peut utiliser le code 3, accompagné de `using namespace System`. Dans le cas d'une importation, il suffit d'intégrer le fichier du modèle dans la solution et d'exécuter la méthode `mVXtrack->ImportModel(char *filename)`. Le code 4 résume cette étape.

On peut maintenant lancer un suivi, et transmettre la séquence courante à notre objet `pSequence` (code 5).

```
while (mVXtrack->TrackingEntities->Length == 0)
    Threading::Thread::Sleep(1);
```

CODE 3 – Temporisation lors de la recherche de modèle

```
//1. Détection à la main
mVXtrack->DetectModel();
while (mVXtrack->TrackingEntities->Length == 0)
    Threading::Thread::Sleep(1);

//2. Importation
mVXtrack->ImportModel("ModeleZephyr.txt");
```

CODE 4 – Chargement d'un modèle suivant les deux méthodes

```
mVXtrack->StartTracking();
cli::array<VXtrack::ITrackingEntities^>^ entities = mVXtrack->TrackingEntities;
pSequence = mVXtrack->CurrentTrackingSequence;
```

CODE 5 – Démarrage d'une séquence de suivi

Ceci permet de lancer une séquence, puis de récupérer un objet associé aux modèles initialisés et à la séquence courante. On peut alors exécuter une boucle infini dans laquelle on va récupérer les coordonnées des mires et du Zephyr.

Dans le cas où le tableau `entities` est non vide, on peut récupérer le modèle du Zephyr (situé très probablement dans la 1re et unique case du tableau) et son vecteur de pose (position et orientation), puis on peut récupérer les cibles associées à ce modèles (code 6). On peut ainsi accéder à la position des mires ou à leur normale (via une boucle

```
VXtrack::ItrackingEntity^ zephyrEntity = entities[0];

cli::array<Types::IPositioningTargets^>^ targets;
Types::IPose3d^ pPose;

targets = pSequence->GetLastPositioningTargets(zephyrEntity);
pPose = pSequence->GetLastPose(zephyrEntity);
```

CODE 6 – Chargement des mires et de la pose du modèle

`for` par exemple) à l'aide des paramètres `Center` et `Normal`. Pour la pose du modèle, on utilise les paramètres `Translation` pour la position et `Rotation` pour l'orientation.