Cours Génie Logiciel

*Analyse d’enregistrements vidéo pour l’étude cinématique de mouvements*

Bastien Burri

Danny Kunz

Martin Künzi

Mathieu Rosser

Équipe 4

2 avril 2014

| Version | Date | Auteurs | Statut | Description de changements |
| --- | --- | --- | --- | --- |
| 0.1 | 2014-03-30 | Équipe 4 | Dev | Intégration des documents réalisés |
| 0.2 | 2014-03-31 | Équipe 4 | Dev | Diagrammes de classes et diagrammes dynamiques |
| 0.3 | 2014-04-01 | Équipe 4 | Dev | Tests fonctionnels |
| 1.0 | 2014-04-02 | Équipe 4 | Final | Spécifications définitives |

Table des matières

[2 Introduction 3](#_Toc384299068)

[2.1 Planning 5](#_Toc384299069)

[3 Spécifications des besoins 6](#_Toc384299070)

[3.1 Besoins utilisateurs 6](#_Toc384299071)

[3.1.1 Acteurs 6](#_Toc384299072)

[3.1.2 Cas d’utilisations 6](#_Toc384299073)

[3.2 Scénarios 7](#_Toc384299074)

[3.2.1 UC 1.1 – Choisir une vidéo 7](#_Toc384299075)

[3.2.2 UC 1.2 – Saisir les paramètres de l’application 8](#_Toc384299076)

[3.2.3 UC 1.3 – Lancer l’analyse de la vidéo pour calculs MRU 9](#_Toc384299077)

[3.3 Exigences fonctionnelles et non fonctionnelles 11](#_Toc384299078)

[3.4 Maquette 12](#_Toc384299079)

[4 Spécifications techniques 14](#_Toc384299080)

[4.1 Analyse du domaine métier 14](#_Toc384299081)

[4.2 Diagramme de classe complet 17](#_Toc384299082)

[4.3 Dynamique de l’application 18](#_Toc384299083)

[4.3.1 UC 1.1 – Choisir une vidéo 18](#_Toc384299084)

[4.3.2 UC 1.2 – Saisir les paramètres de l’application 19](#_Toc384299085)

[4.3.3 UC 1.3 – Lancer l’analyse de la vidéo pour calculs MRU 21](#_Toc384299086)

[4.3.4 Etats de lecture d’une vidéo de JPanelVideo 23](#_Toc384299087)

[4.4 Architecture de déploiement 24](#_Toc384299088)

[4.5 Choix des librairies externes 25](#_Toc384299089)

[4.6 Tests fonctionnels 26](#_Toc384299090)

[5 Conclusion 28](#_Toc384299091)

# Introduction

Ce document présente les spécifications des besoins et spécifications techniques du projet « *Analyse d’enregistrements vidéo pour l’étude cinématique de mouvements*».

L’objectif du projet est d’étudier la cinématique des mouvements d’un mobile avec l’analyse d’une vidéo filmée latéralement, dans laquelle l’objet se déplace en ligne droite. Un seul objet se déplace dans la vidéo. L’application sera utilisée pour expliquer la cinématique des mouvements aux étudiants de première année du cours de physique.

L’utilisateur spécifie à l’application les bornes de départ et d’arrivée (A, B) du mobile dans la vidéo. Dans le cas d’une vidéo bien cadrée, ces bornes correspondent aux extrémités du champ de vision de la caméra. La zone A-B de mesure doit ensuite être séparée en un certain nombre N de temps intermédiaires, au choix de l’utilisateur. Pour ce faire, l’utilisateur a deux possibilités :

* Il spécifie la distance réelle entre A et B, et alors les N bornes intermédiaires seront placées de manière équidistante, sans tenir compte de l’effet de perspective de la caméra ;
* Il spécifie la distance réelle entre la caméra et le point central à angle droit du plan filmé ainsi que l’angle de vision de la caméra, et alors les N bornes intermédiaires seront placés en tenant compte de l’effet de perspectives de la caméra. Les distances intermédiaires ne seront pas toujours les mêmes à cause de l’angle de vision.

Une fois lancé, le programme détecte lorsque le mobile passe la borne A et lance un chronomètre. Il mémorise les temps de passage aux bornes intermédiaires. Lorsque l’objet arrive à la borne B, le chronomètre s’arrête. Le logiciel donne ensuite les résultats importants suivants :

* La vitesse et l’accélération moyennes du mobile pour le parcours A-B ;
* La vitesse et l’accélération instantanées du mobile pour chaque borne intermédiaire.

Les objectifs suivants seront réalisés dans le cadre du projet P2 Java :

* Analyser une vidéo d’un mobile en connaissant la distance entre les bornes de départ et d’arrivée, sans tenir compte des perspectives de la caméra ;
* Analyser une vidéo d’un mobile en connaissant la distance entre la caméra et le centre du plan filmé, en tenant compte des perspectives de la caméra ;
* Calculer la vitesse et l’accélération moyennes, ainsi que les vitesses et accélérations instantanées aux points intermédiaires ;
* Représenter graphiquement l’accélération et la vitesse instantanée des différents temps intermédiaires ;
* Trouver le(s) point(s) où la vitesse instantanée est égale à la vitesse moyenne.

Les deux modes d’analyse de la vidéo pour déterminer les vitesses et accélérations du mobile aux bornes intermédiaires sont les suivants :

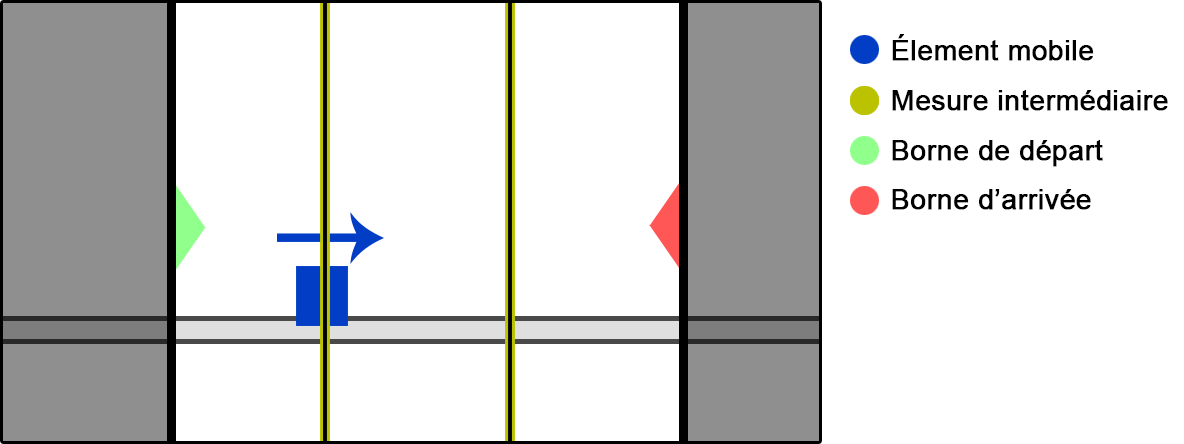


Figure 1. Mode d'analyse ne tenant pas compte de la perspective : les bornes intermédiaires sont équidistantes

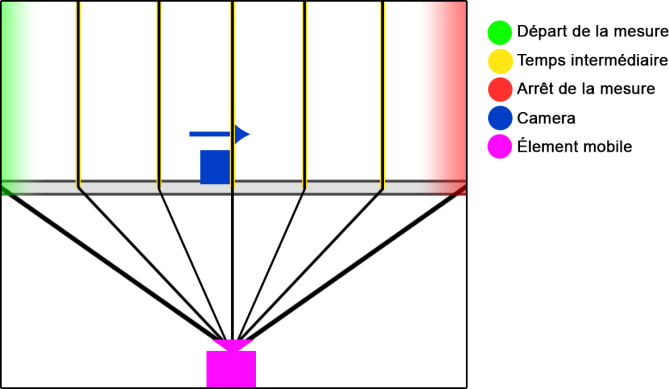


Figure 2. Mode d'analyse tenant compte de la perspective de la caméra : l’espacement entre les bornes intermédiaires dépend de l’angle de vision de la caméra

Dans la suite de ce document, le terme MRU est utilisé pour parler de la cinématique des mouvements. MRU signifie mouvements rectilignes uniformes ou mouvements rectilignes uniformément accélérés.

## D:\Users\mathieu.rosser\Documents\Dropbox\ProjetP2Java\2-Spécifications\planification.PNGPlanning

Figure 3. Planification prévisionnelle du projet. Pour les détails, par exemple les responsables des tâches, voir la forge du projet : <http://isic.lan/projects/13-14inf-p2-equipe4/issues/gantt>

Les tâches décrites ci-dessus sont issues des spécifications présentées dans la suite de ce document.

# 

# Spécifications des besoins

Ce chapitre est consacré aux spécifications des besoins du projet. Ces spécifications décrivent quels seront les fonctionnalités et services offerts par l’application. Pour commencer, les cas d’utilisations du système sont présentés. Chaque cas d’utilisation est ensuite détaillé sous forme de scénario. Pour finir, la maquette permet de représenter visuellement l’ensemble des fonctionnalités attendues.

## Besoins utilisateurs

### Acteurs

Le seul acteur présent dans le système est l’utilisateur, qui peut être n’importe qui s’intéressant à analyser la vitesse et l’accélération d’un objet sur une vidéo. Le rôle joué par un professeur ou un étudiant est donc le même dans le cadre de ce projet.

### Cas d’utilisations

Les fonctionnalités offertes sont séparées en trois cas d’utilisations principaux.

|  |  |
| --- | --- |
| **ID** | UC 1.1 |
| **Nom** | Choisir une vidéo |
| **Description** | L’utilisateur utilise un menu de sélection de vidéo pour définir la vidéo à analyser.  Le menu de sélection permet de parcourir le système de fichiers de l’utilisateur et de choisir la vidéo de son choix.  La vidéo choisie est chargée dans l’application. |

|  |  |
| --- | --- |
| **ID** | UC 1.2 |
| **Nom** | Saisir les paramètres de l’application |
| **Description** | L’utilisateur définit les paramètres de l’application pour l’analyse qu’il souhaite effectuer. Il spécifie le mode dans lequel il veut analyser la vidéo puis il définit aussi les différentes mesures demandées par le programme selon le mode.  Le mode peut être sans perspective ou avec perspective de la caméra. |

|  |  |
| --- | --- |
| **ID** | UC 1.3 |
| **Nom** | Lancer l’analyse de la vidéo pour calculs MRU |
| **Description** | L’utilisateur clique sur le bouton de démarrage de l’analyse de la vidéo. Le programme lance l’analyse puis calcule les vitesses et les accélérations sur les différents points de mesure. L’utilisateur peut ensuite consulter ces résultats sur la page de résultats qui lui est présentée. |

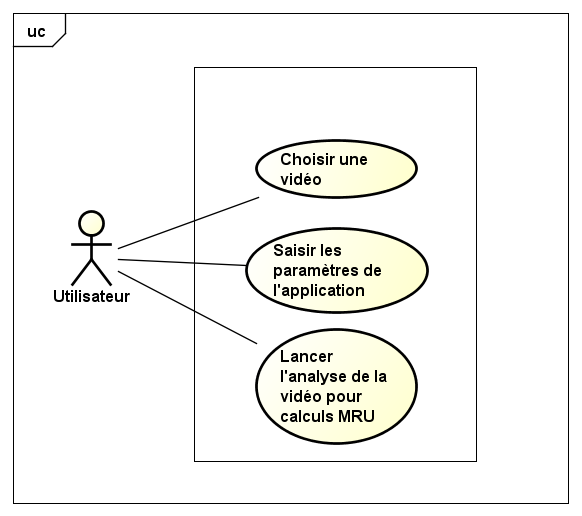


Figure 4. Diagramme des cas d'utilisations du système

## Scénarios

Chaque cas présenté au chapitre 3.1.2 est ici détaillé sous forme de suites d’actions ou scénarios.

### UC 1.1 – Choisir une vidéo

#### Description textuelle

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | UC 1.1 | **Nom** | Choisir une vidéo | | |
| **Date** | 2014-04-01 | **Auteur** | Équipe 4 | **Priorité** | Haut |
| **Description et objectifs** | Le but de ce cas d’utilisation est d’ouvrir le sélecteur de fichier sur l’ordinateur pour choisir une vidéo à analyser, qui est ensuite affichée par l’application. | | | | |
| **Acteurs** | Utilisateur | | | | |
| **Déclencheur** | Clic sur le bouton de chargement d’une vidéo via la barre de menu. | | | | |
| **Scénario nominal** | 1 : Cliquer sur le bouton de chargement d’une vidéo.  2 : Sélection d’une vidéo compatible.  3 : Affichage de la vidéo sur le programme. | | | | |
| **Scénarios alternatifs** | 3.1 : Détection d’un problème d’affichage de la vidéo  3.2 : Affichage d’un message à l’utilisateur | | | | |
| **Préconditions** | Aucune | | | | |
| **Post-Conditions** | La vidéo s’affiche sur l’application | | | | |
| **Exigences** | Fonctionnelles : FR1 – voir chapitre 3.3 | | | | |

#### Diagramme de séquence système

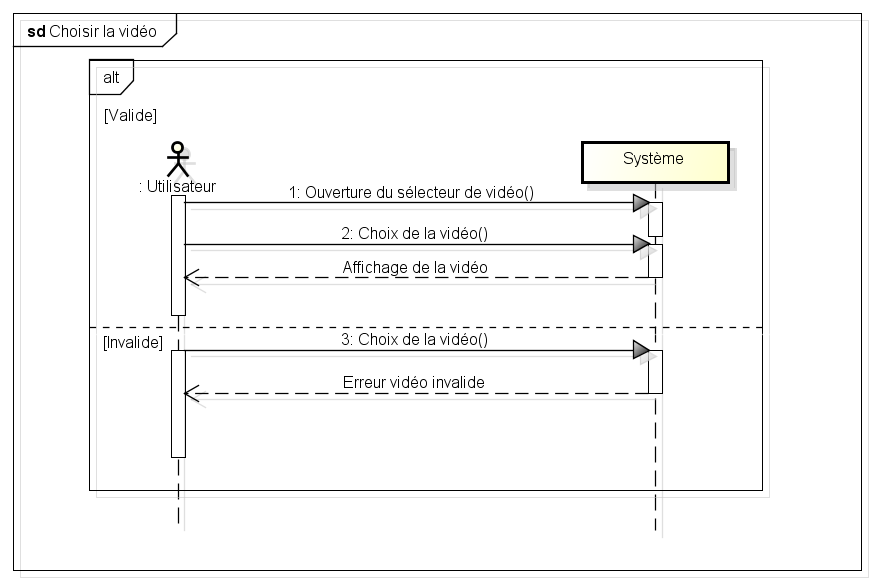


Figure 5. Diagramme de séquence système pour UC 1.1

### UC 1.2 – Saisir les paramètres de l’application

#### Description textuelle

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | UC 1.2 | **Nom** | Saisir les paramètres de l’application | | |
| **Date** | 2014-04-01 | **Auteur** | Équipe 4 | **Priorité** | Haut |
| **Description et objectifs** | Le but de ce cas d’utilisation est de saisir tous les paramètres nécessaires pour pouvoir effectuer l’analyse vidéo. Ce cas permet de spécifier le mode d’analyse entre « avec perspective » et « sans perspective ». | | | | |
| **Acteurs** | Utilisateur | | | | |
| **Déclencheur** | Souhait de spécifier les paramètres d’analyse ou d’analyser la vidéo | | | | |
| **Scénario nominal** | 1 : Choix d’un mode d’analyse  2 : Déplacement des bornes de départ et d’arrivée  3 : Définition de la distance départ/arrivée  4 : Définition du nombre de points intermédiaires de mesures | | | | |
| **Scénarios alternatifs** | 3.1.1 : Définition de la distance caméra/plan  3.1.2 : Définition de l’angle de vue de la caméra  3.2.1 : Valeurs incorrectes | | | | |
| **Préconditions** | UC 1.1 – Une vidéo doit être sélectionnée | | | | |
| **Post-Conditions** | La vidéo est chargée et les paramètres sont définis.  Sur la vidéo, le départ, l’arrivée et les intermédiaires sont dessinés. | | | | |
| **Exigences** | Fonctionnelles : FR2 – voir chapitre 3.3 | | | | |

#### Diagramme de séquence système

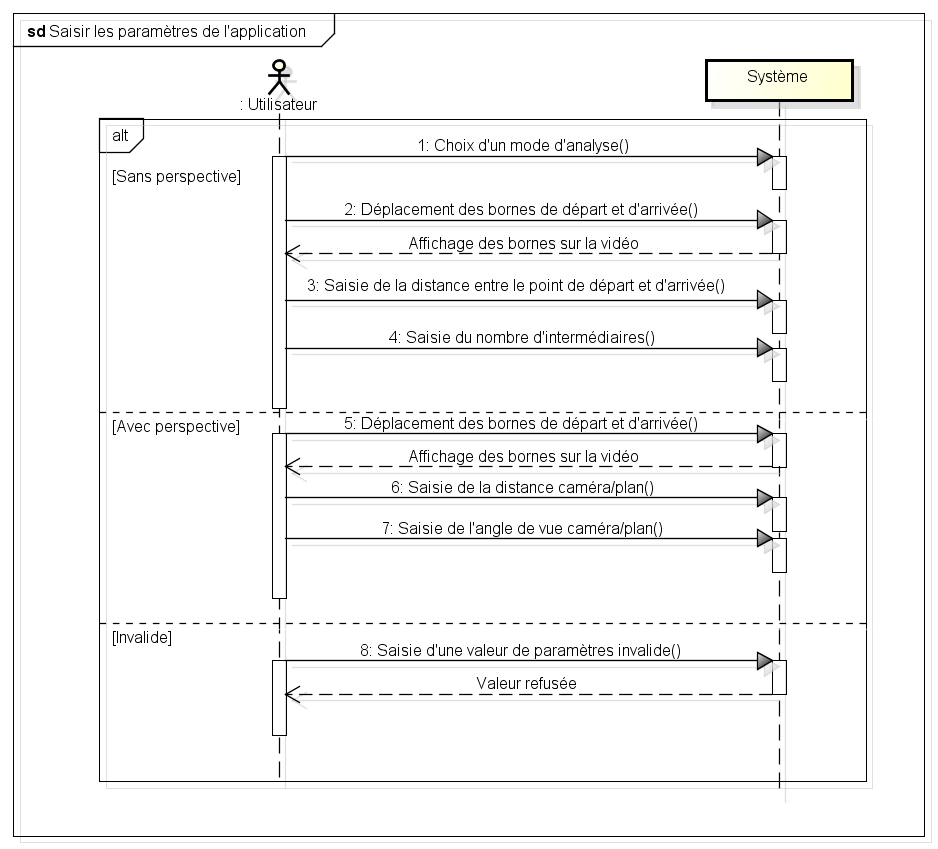


Figure 6. Diagramme de séquence système pour UC 1.2

### UC 1.3 – Lancer l’analyse de la vidéo pour calculs MRU

#### Description textuelle

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | UC 1.3 | **Nom** | Lancer l’analyse de la vidéo pour calculs MRU | | |
| **Date** | 2014-04-01 | **Auteur** | Équipe 4 | **Priorité** | Haut |
| **Description et objectifs** | Le but de ce cas d’utilisation est de lancer l’analyse de la vidéo sélectionnée avec les paramètres spécifiés. Les résultats des calculs MRU sont ensuite présentés à l’utilisateur dans une fenêtre de résultats. | | | | |
| **Acteurs** | Utilisateur | | | | |
| **Déclencheur** | Clic sur le bouton d’analyse de la vidéo pour calculer les données MRU du mobile. | | | | |
| **Scénario nominal** | 1 : Clique sur le bouton de lancement de l’analyse : la vidéo démarre et est analysée  2 : Ouverture de la fenêtre des résultats contenant mesures, résultats et graphiques | | | | |
| **Scénarios alternatifs** | 1.1 : Paramètres mal configurés – signalé à l’utilisateur 2.1 : Erreur d’analyse : aucun mobile détecté – signalé à l’utilisateur | | | | |
| **Préconditions** | UC 1.1, UC 1.2 – Vidéo définie et paramètres spécifiés | | | | |
| **Post-Conditions** | L’utilisateur voit la vidéo se lancer, l’analyse s’effectue (dessin sur la vidéo) et la fenêtre de résultats et graphiques s’ouvre à la fin du processus d’analyse. | | | | |
| **Exigences** | Fonctionnelles : FR3, FR4  Non fonctionnelles : NFR1, NFR2, NFR3  Voir chapitre 3.3 | | | | |

#### Diagramme de séquence système

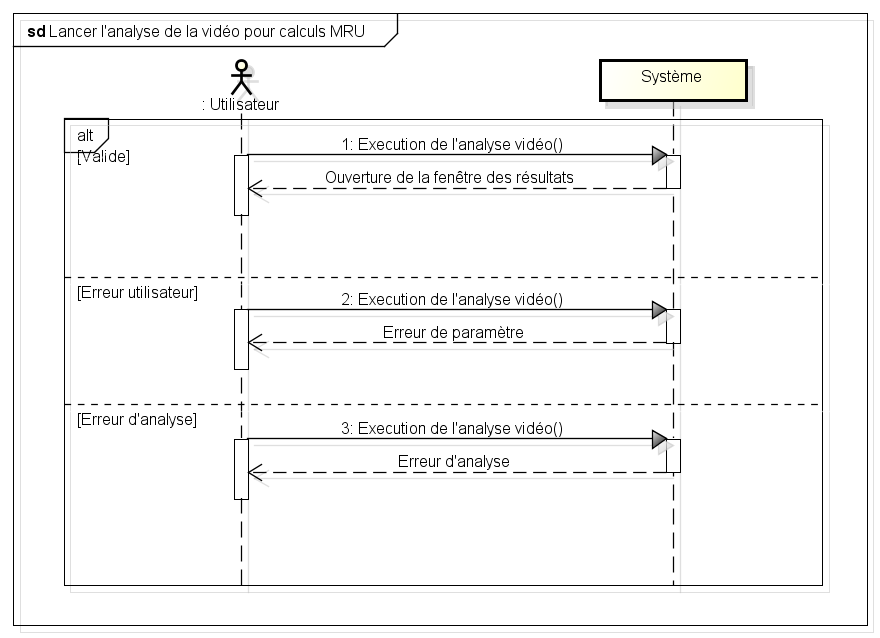


Figure 7. Diagramme de séquence système pour UC 1.3

## Exigences fonctionnelles et non fonctionnelles

Les exigences fonctionnelles fournissent des détails d’implémentation sur les fonctionnalités de l’application. Les exigences non fonctionnelles représentent les facteurs extérieurs imposés à l’application devant être pris en compte lors du développement.

|  |  |  |
| --- | --- | --- |
| ID Exigence | Type | Description |
| FR1 | Fonctionnelle | Lire les formats vidéo courants, en évitant le plus possible de devoir recourir à une conversion de la vidéo par l’utilisateur avant de pouvoir analyser la vidéo. |
| FR2 | Fonctionnelle | Pouvoir paramétrer l’application pour effectuer une analyse en mode « sans perspective » puis pouvoir changer ces paramètres en mode « avec perspective » pour réaliser une nouvelle analyse. La nouvelle analyse pourra être utilisée pour comparer les résultats entre les deux modes. |
| FR3 | Fonctionnelle | Pouvoir analyser des vidéos avec des objets rapides ou lents. L’analyse doit pouvoir se faire avec des vidéos de résolution différentes, mise à l’échelle automatiquement par l’application. |
| FR4 | Fonctionnelle | Afficher automatiquement les résultats et les graphiques à la fin de l’analyse des mouvements du mobile dans la vidéo. |
| NFR1 | Performance | Traiter les images pour détecter un objet dans des délais raisonnables sans fausser les résultats des calculs de mouvements. |
| NFR2 | Performance | Assurer un traitement rapide des calculs des vitesses et accélérations à la fin de l’analyse vidéo. |
| NFR3 | Ergonomie | Ne pas paralyser l’IHM lors de l’analyse de la vidéo et garantir la réactivité de l’interface en général. |

## Maquette

À partir des éléments précédents de spécification des besoins, la maquette de l’application a été réalisée. L’application sera séparée en deux fenêtres :

* Fenêtre principale permettant de paramétrer l’application et d’analyser une vidéo ;
* Fenêtre de résultats présentant les vitesses et accélérations calculées à partir de l’analyse.

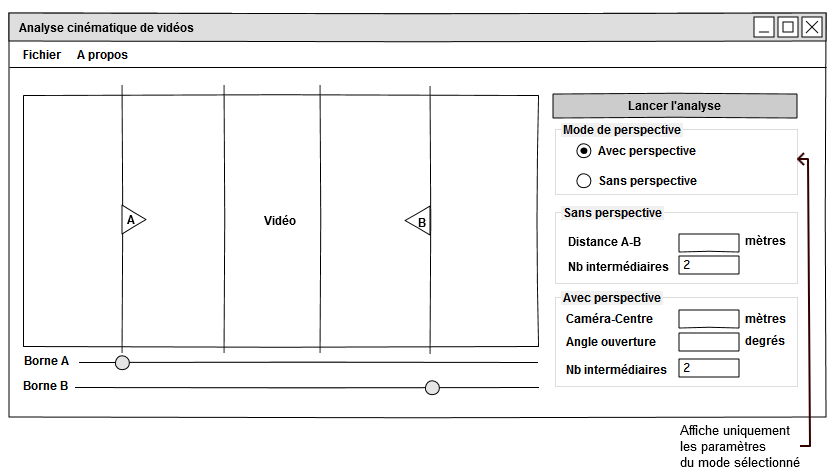


Figure 8. Maquette de la fenêtre principale permettant d'analyser une vidéo

La fenêtre principale offrira les possibilités suivantes :

* Charger une vidéo, à partir du menu Fichier ;
* Spécifier les bornes de départ et d’arrivée (A-B) du mobile dont on cherche à connaître la vitesse et l’accélération sur la distance séparant ces deux bornes ;
* Spécifier le nombre de bornes intermédiaires entre les bornes de départ et d’arrivée (A-B) ;
* Analyser la vidéo en mode « Avec perspective », en tenant compte de la perspective de la caméra.  
  Dans ce mode, l’utilisateur doit spécifier la distance caméra-centre du plan vidéo ainsi que l’angle de vision de la caméra ;
* Analyser la vidéo en mode « Sans perspective », sans tenir compte de la perspective de la caméra.  
  Dans ce mode, l’utilisateur doit spécifier la distance entre les bornes de départ et d’arrivée (A-B).

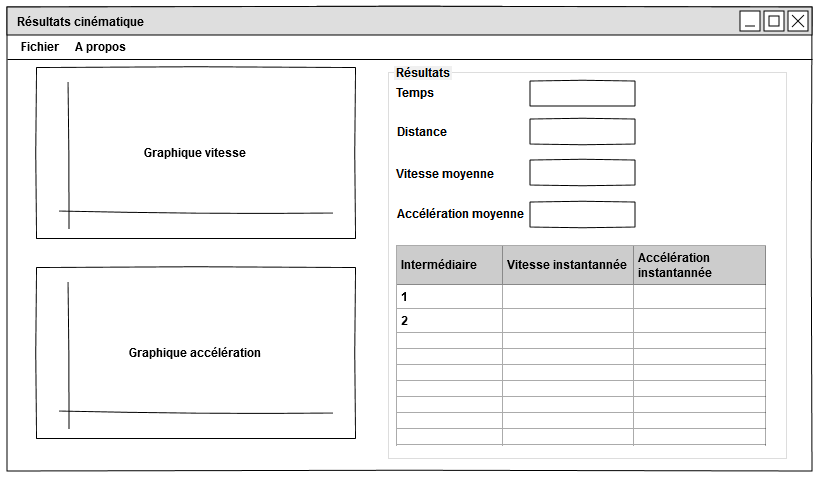


Figure 9. Maquette de la fenêtre de résultats

La fenêtre de résultats affiche l’ensemble des résultats calculés : vitesse moyenne, accélération moyenne, vitesses et accélérations instantanées aux points intermédiaires. Les graphiques représentent visuellement les résultats obtenus pour l’ensemble de l’analyse de la cinématique des mouvements.

# Spécifications techniques

Ce chapitre est consacré aux spécifications techniques. Ces spécifications décrivent comment vont être réalisés les besoins exprimés au chapitre 3.

Premièrement, le domaine métier est analysé dans le but de déterminer les classes entités (classes métiers). Ensuite, le diagramme de classe complet est présenté en ajoutant les classes dialogues et les contrôleurs. À partir du diagramme de classe, la dynamique de l’application est modélisée pour chaque cas d’utilisation. Les choix des librairies et les tests fonctionnels complètent ces spécifications techniques.

## Analyse du domaine métier

L’analyse du domaine métier demande premièrement de discrétiser le modèle physique de notre application. L’objectif est de passer d’un modèle physique où la différence de temps entre deux mesures réelles tant vers zéro (limite mathématique) à un modèle discret où la mesure de temps est effectué selon un delta temps.

Pour notre application, la mesure du temps s’effectue aux bornes de passages – départ, intermédiaires et arrivée. Ces bornes sont toutes à une position connue par calcul. La distance entre deux bornes consécutives est aussi connue. L’application des formules des lois du mouvement (vitesse, accélération) est donc possible en mesurant le temps système à chaque borne.

**Le modèle physique discrétisé des lois de la vitesse et de l’accélération est le suivant :**

Chaque borne a une position connue, obtenue par calcul

Chaque borne a un temps de passage connu avec le temps système, mesuré lors de l’analyse

1. Borne de départ :
2. Borne intermédiaire 1 :
3. Borne intermédiaire 2 :

1. Jusqu’à la borne d’arrivée :

À la borne de départ, on ne peut rien calculer. Au premier intermédiaire, on peut calculer la vitesse. À partir du deuxième point intermédiaire, on peut calculer l’accélération.

**On distingue deux cas d’application de ce modèle discrétisé :**

* Les calculs ne tenant pas compte de la perspective de la caméra ;
* Les calculs tenant compte de la perspective de la caméra.

Pour les calculs ne tenant pas compte de la perspective, la distance A-B entre le départ et l’arrivée est connue. Les intermédiaires sont donc placés à équidistance les uns des autres. Avec intermédiaires, on a intervalles. Il est donc possible d’appliquer directement notre modèle discrétisé, car les distances sont calculables avec les intervalles d’intermédiaires et les temps sont mesurés :

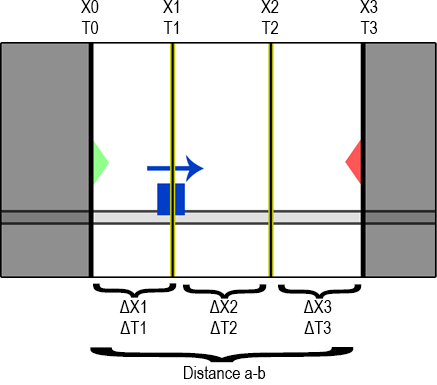


Figure 10. Mesures du temps en mode « sans perspective », après avoir calculé les positions X selon la formule

Pour les calculs tenant compte de la perspective de la caméra, il est nécessaire de déterminer la distance entre les bornes de départ et d’arrivée (A-B). Les deux mesures réelles fournies par l’utilisateur sont :

* Distance entre la caméra et le centre du plan filmé à angle droit ;
* Angle de vision de la caméra.

En utilisant les formules de trigonométrie induite par l’angle droit entre la caméra et le plan filmé, on obtient les formules suivantes :

Si la distance entre le départ et l’arrivée n’est pas égale à la largeur du plan filmé, il faudra modifier l’angle de vision. Cette modification est faisable en connaissant la borne de départ et d’arrivée. Ensuite l’angle de vision correspond à l’addition des angles entre la caméra et ces deux bornes.

Dans la formule ci-dessus, certains ont un indice négatif. Celui-ci devra être corrigé avec un *offset* (décalage) pour être dans l’intervalle.

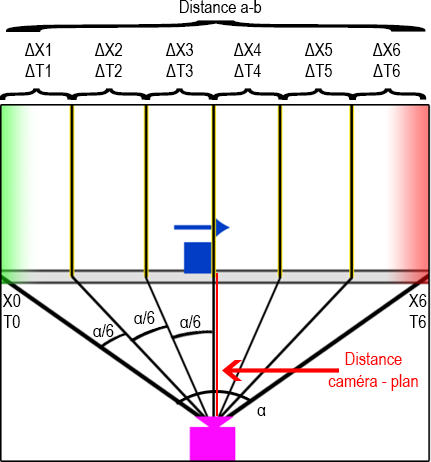


Figure 11. Mesures du temps en mode « avec perspective », après avoir calculé les positions X selon la formule

Au niveau des unités, le mètre est utilisé au niveau physique. Pour l’analyse de la vidéo, les distances entre les intermédiaires seront en pixels. Le calcul et le stockage d’une échelle de conversion pixel vers mètre est donc nécessaire. Cette échelle sera calculée lorsque l’utilisateur spécifie les distances réelles en mètres.

Pour gérer ces données du domaine métier, nous utilisons six classes entités présentées dans le diagramme de classe complet au chapitre 4.2 Diagramme de classe complet.

* **MRUData** : classe qui gère les distances/données saisies par l’utilisateur, ainsi que les correspondances entre le monde physique (unité physique en mètre) et le monde des pixels (unité logique en pixel) ;
* **MRUIntermediates** : classe de base qui permet de calculer les intermédiaires de deux façons différentes au travers du mécanisme d’héritage
  + **MRUIntermediatesDistance** : intermédiaires sans tenir compte de la perspective ;
  + **MRUIntermediatesCamera** : intermédiaires en tenant compte de la perspective ;
* **MRUIntermediate** : classe permettant de stocker les intermédiaires, leurs positions en mètres et en pixels, ainsi que les résultats des calculs (accélération, vitesse) ;
* **MRUCalculation** : classe effectuant les calculs d’accélération et de vitesse pour chaque borne intermédiaire.

Le diagramme de classe complet présenté à la page suivante s’articule autour des deux IHM et les classes métiers. Un contrôleur s’occupe de relier les données aux IHM en gérant la logique de l’application.

## D:\Users\mathieu.rosser\Documents\Dropbox\ProjetP2Java\2-Spécifications\Diagramme classe\Diagramme de classe.pngDiagramme de classe complet

Figure 12. Diagramme de classe avec les classes IHM, contrôle et entités

## Dynamique de l’application

Les diagrammes de communication et les diagrammes de séquences détaillées de ce chapitre décrivent le comportement de notre application. Chaque cas d’utilisation est détaillé sous forme d’interaction entre les objets issus du diagramme de classe.

### UC 1.1 – Choisir une vidéo

#### Diagramme de collaboration

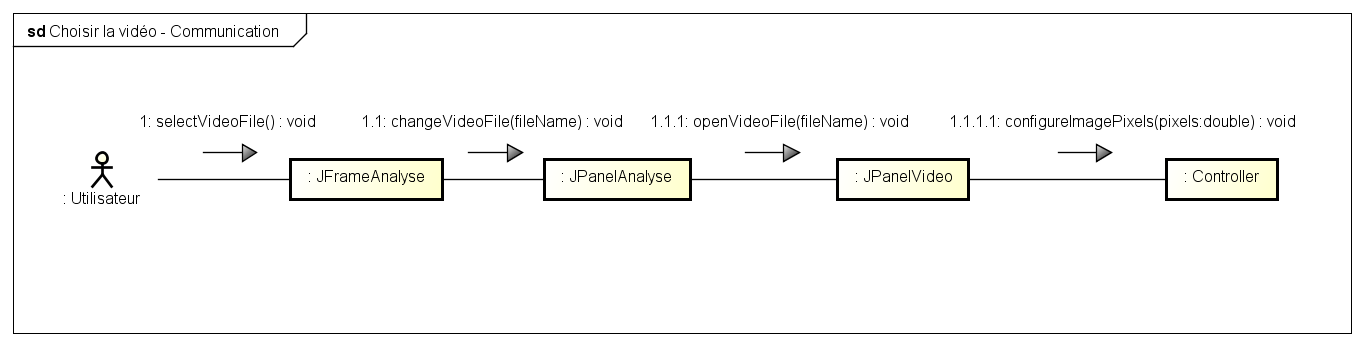


Figure 13. Diagramme de collaboration pour UC 1.1

#### Diagramme de séquence détaillée

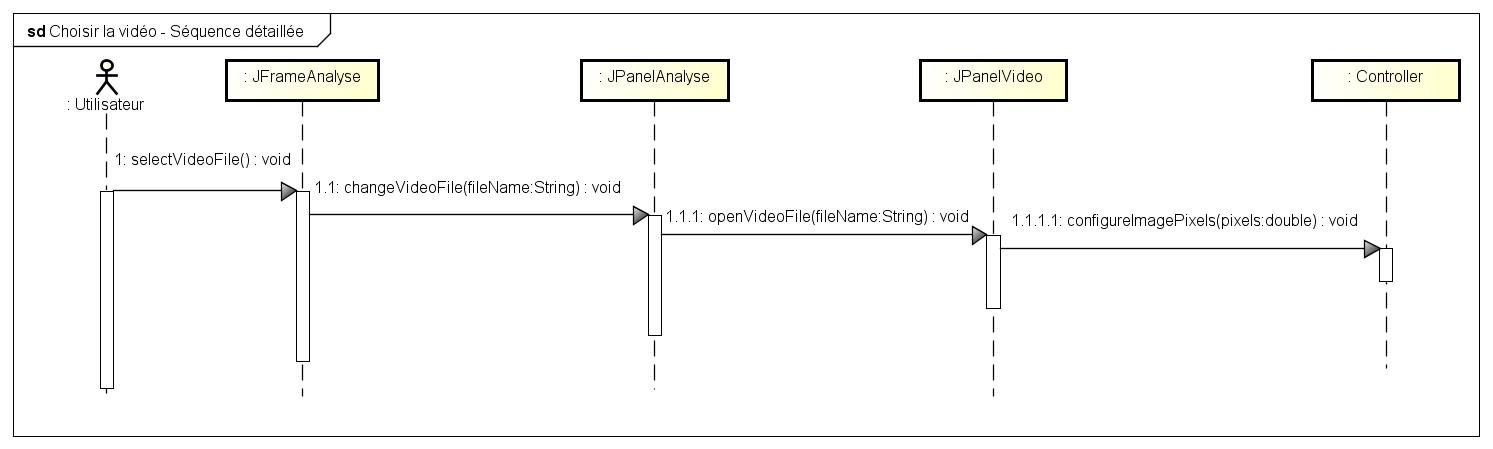


Figure 14. Diagramme de séquence détaillée pour UC 1.1

### UC 1.2 – Saisir les paramètres de l’application

#### Diagramme de collaboration

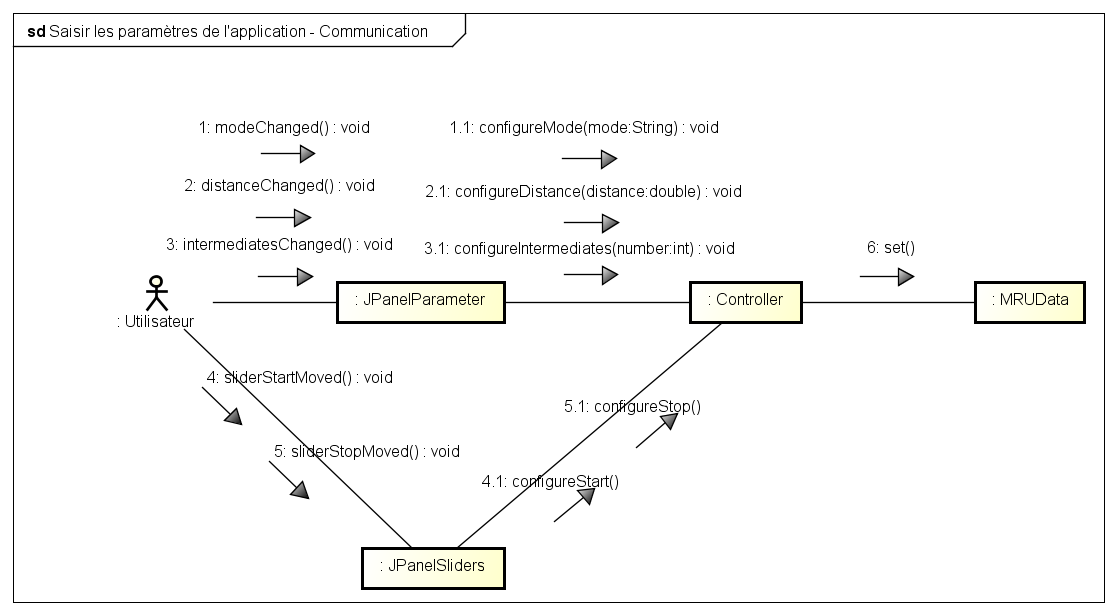


Figure 15. Diagramme de communication pour UC 1.2

#### Diagramme de séquence détaillée

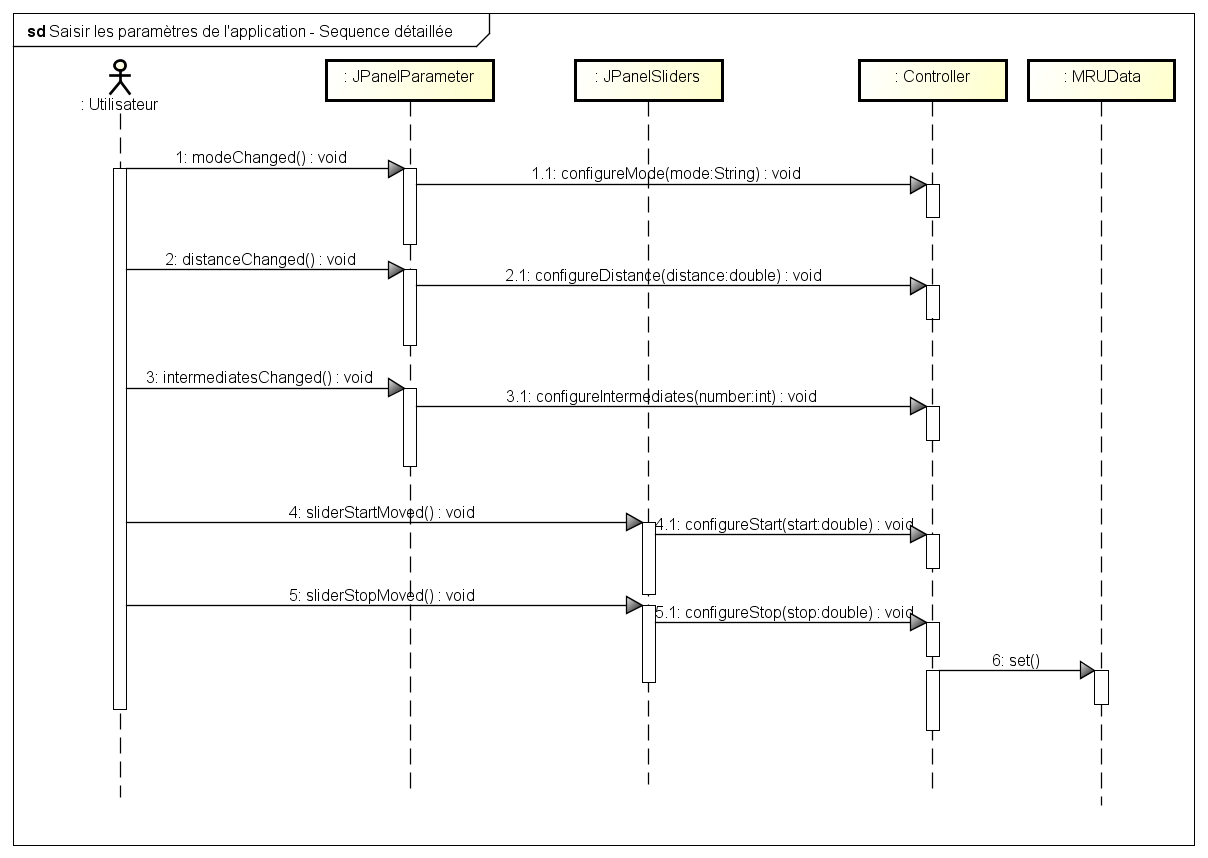


Figure 16. Diagramme de séquence détaillée pour UC 1.2

### UC 1.3 – Lancer l’analyse de la vidéo pour calculs MRU

#### Diagramme de collaboration

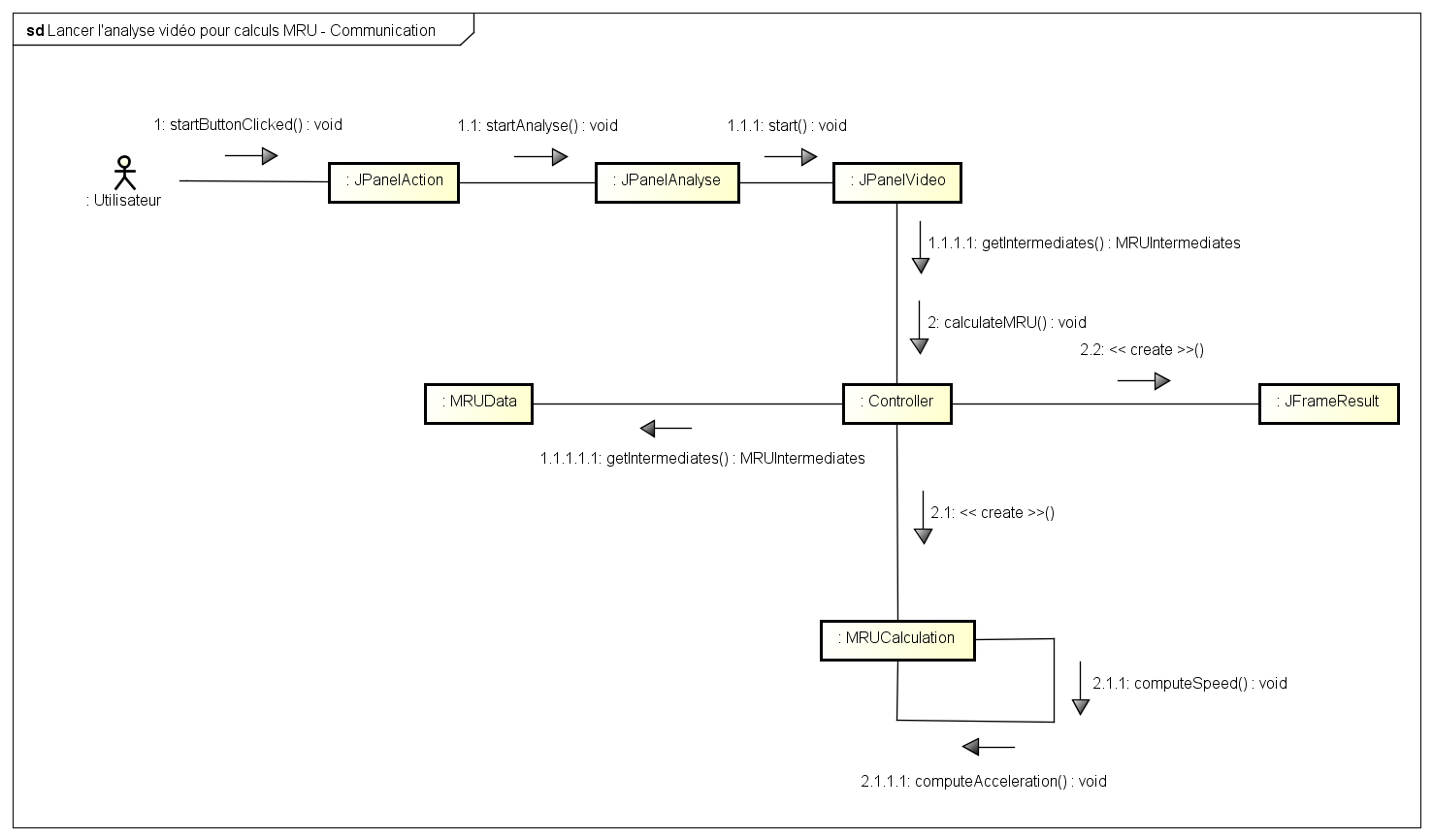


Figure 17. Diagramme de collaboration pour UC 1.3

#### D:\Users\mathieu.rosser\Documents\Dropbox\ProjetP2Java\2-Spécifications\Diagramme classe\Lancer l'analyse vidéo pour calculs MRU.pngDiagramme de séquence détaillée

Figure 18. Diagramme de séquence détaillée pour UC 1.3

### Etats de lecture d’une vidéo de JPanelVideo

Le diagramme d’états ci-dessous représente les différents états possibles d’une vidéo chargée dans le panel d’affichage et traitement de la vidéo.

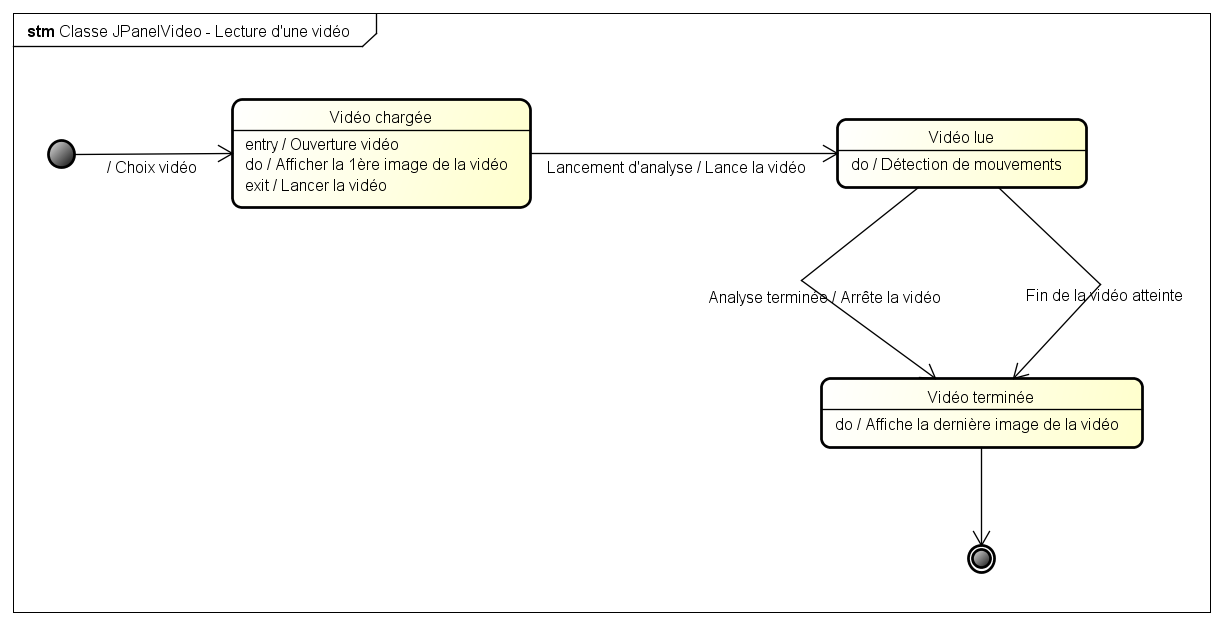


Figure 19. Diagramme des états de lecture d'une vidéo dans la classe JPanelVideo

## Architecture de déploiement

Le déploiement de l’application se fera sur des PC de l’école ou des PC personnels. Comme l’application est codée en Java, le terminal sur lequel l’application sera exécutée requiert l’installation d’une machine virtuelle Java si elle n’est pas déjà installée.

L’application sera distribuée sous forme de classes Java compilées et zippées dans des librairies JAR (Java Archive).

Deux packages de déploiement seront distribués :

* Package contenant uniquement les librairies JAR du projet, pouvant être exécutées avec une JVM (Java Virtual Machine) déjà installée sur un PC ;
* Package contenant les librairies JAR du projet et une installation du JRE (Java Runtime Environment) contenant une machine virtuelle Java, pour un PC qui ne serait pas encore équipé de Java.



Figure 20. Schéma de l'architecture de déploiement

## Choix des librairies externes

Ce chapitre décrit les choix des librairies effectués durant la phase des spécifications techniques. Différents points chauds ont été mis en évidence et demandent d’utiliser des librairies :

* Acquisition d’une vidéo avec Java pour l’afficher dans une IHM ;
* Traitement d’une vidéo avec Java pour détecter des mouvements d’un objet filmé ;
* Affichage de graphiques avec Java.

La lecture d’une vidéo en Java n’est pas nativement intégrée au langage, à la différence d’un framework comme Qt. Comme nous avons besoin d’effectuer du traitement d’image en plus de l’acquisition vidéo, nous avons donc cherché les librairies capables de lire des vidéos et de les traiter en Java :

* **JMF – Java Media Framework**

Librairie officielle Java, fournie par Oracle, pour lire des fichiers multimédias.

Cette librairie a un gros inconvénient : la dernière version publiée date de 2003. Nous l’avons testée, mais nous n’avons pas pu lire les derniers formats vidéo tels que le MP4 ou le MKV. Pour lire ces types de vidéos, il faut ajouter différents plug-ins ou convertir la vidéo auparavant.

**La librairie n’a pas été retenue**.

|  |
| --- |
| <http://www.oracle.com/technetwork/java/javase/tech/index-jsp-140239.html> |

* **OpenCV, JavaCV et FFMPeg**

Open source Computer Vision est une librairie dédiée intégralement aux traitements d’image. La librairie est codée en C++, mais il existe un wrapper Java : JavaCV. En utilisant les librairies OpenCV et JavaCV, il est possible d’utiliser les fonctionnalités de lecture et de traitement vidéo.

**Les tests fonctionnels de JavaCV se sont révélés concluants et cette librairie a été retenue**. Les vidéos utilisées durant les tests ont pu être lues avec les fonctionnalités offertes par JavaCV. Ce wrapper permet d’intégrer facilement la librairie FFMpeg, qui permet de lire différents types et formats de vidéos grâce à son système de codecs. Ces librairies sont documentées et éprouvées.

Au niveau du traitement d’images, les fonctionnalités nécessaires pour détecter un mouvement sont présentes dans la librairie. Les tests d’implémentation effectués avec cette librairie sont détaillés au chapitre 4.6 Tests fonctionnels.

Nous avons aussi testé la librairie open source **MarvinFramework**, qui s’appuie sur JavaCV avec un système de plug-ins réalisant nombres de traitements d’images prédéfinis. La librairie n’a pas été retenue en supplément d’OpenCV, car sa documentation et ses sources ne sont pas à jour. Nous avons aussi rencontré des problèmes techniques avec des vidéos hautes résolutions.

|  |  |
| --- | --- |
| <https://code.google.com/p/javacv/> | <http://opencv.org/> |
| <http://www.ffmpeg.org/> | <http://marvinproject.sourceforge.net/> |
|  | |

**Pour réaliser des graphiques en Java, nous avons choisi la librairie JFreeChart**. Cette librairie, étudiée au cours de Java, est éprouvée et documentée.

|  |
| --- |
| <http://www.jfree.org/jfreechart/> |

## Tests fonctionnels

Les tests fonctionnels réalisés ont permis d’implémenter différents points chauds identifiés, à savoir l’acquisition, l’affichage et le traitement de la vidéo. Ils ont permis de tester les librairies choisies.

L’acquisition de vidéo a été réalisée avec les classes utilitaires de **JavaCV**, qui permettent de lire une vidéo en utilisant les techniques d’**OpenCV** ou de **FFMpeg**. L’utilisation de la librairie FFMpeg est à préférer pour lire le maximum de vidéos grâce aux codecs intégrés à la librairie.

Une fois la vidéo lue, il reste à la traiter pour détecter les mouvements de l’objet filmé. **Pour réaliser cette détection de mouvements, nous calculons les différences entre deux images séparées par un temps T.**

Le principe de détection de mouvement est le suivant :

1. Capture de l’image 1 au temps
2. Capture de l’image 2 au temps

dépend de la vitesse d’analyse de la vidéo, en général 1 / 24, qui correspond à la période entre deux images pour une vidéo à 24 images par seconde.

1. Calcul de l’image de détection de mouvement basé sur la différence entre les deux images

Cette image est noire là où il n’y a pas de mouvements. Elle est grise aux endroits où les pixels ont changé entre les deux images.

Cette image montre les différences entre l’image 2 et l’image 1. L’objet se déplaçant, il y aura des différences à son ancienne position sur l’image 1 et à sa nouvelle position sur l’image 2.

1. Calcul de la valeur absolue des différences

L’absolu des différences permet de ne garder que les différences liées au mouvement de l’objet dans l’image 2, sans tenir compte des différences de l’objet à son ancienne position sur l’image 1.

1. Appliquer un seuil (*threshold*) pour différencier l’objet en mouvement des bruits parasites

Le résultat de l’opération de seuil est une image noire et blanche. Les objets en mouvements sont en blanc. Le noir correspond aux zones sans mouvement. Le paramètre de seuil doit être affiné pour éviter d’avoir trop de bruit avec une vidéo instable.

1. Trouver les contours des objets identifiables dans l’image des différences

Un seul objet étant mobile dans les vidéos à analyser, les paramètres de la recherche de contours sont à affiner pour trouver le plus grand objet mobile.

Avec l’objet mobile détecté, il est possible de déterminer s’il passe une position intermédiaire sur l’axe X (positions en X). Au moment où il passe un intermédiaire, il faut alors stocker son temps de passage, calculé comme la différence entre le temps système actuel et le temps système au moment du passage de la borne de départ.

Pour réaliser la détection de mouvements décrite à la page précédente, nous utilisons les fonctionnalités d’OpenCV. Le code permettant de voir si l’objet passe des bornes intermédiaires a été implémenté durant la phase de tests. Pour dessiner sur une image, on peut utiliser les méthodes d’OpenCV ou surcharger les méthodes de dessin des composants Swing de Java.

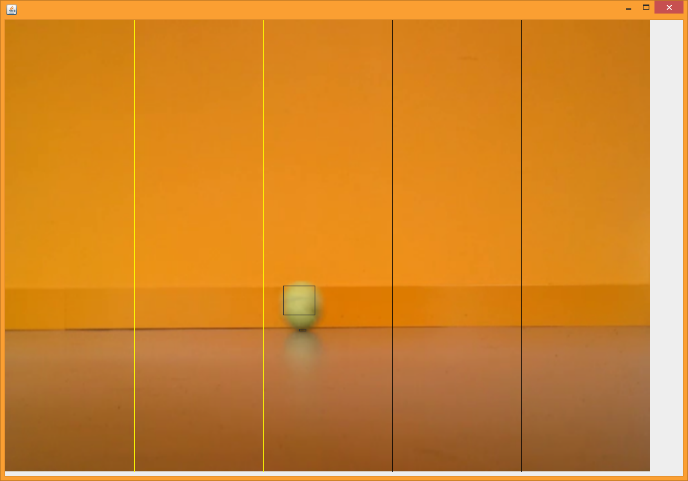


Figure 21. Visuel d’une vidéo montrant la détection d’une balle en mouvement. Les passages de la balle aux points intermédiaires sont détectés.

|  |  |
| --- | --- |
| D:\Users\mathieu.rosser\Documents\Dropbox\ProjetP2Java\2-Spécifications\Video\video_diff.PNG | D:\Mathieu\Mes documents\Dropbox\ProjetP2Java\2-Spécifications\Video\video_diff_seuil.PNG |

Figure 22.

À gauche :

Résultat de la différence absolue entre deux images consécutives. On voit un léger bruit de mouvement le long du chemin emprunté par la balle.

À droite :

Résultat de l'application du seuil à l'image des différences absolues, ce qui a supprimé les bruits de mouvements

# Conclusion

Comme il a été expliqué précédemment, le but du projet est d’étudier la cinématique des mouvements d’un élément unique se déplaçant en ligne droite dans une vidéo filmée en vue latérale.

L’utilisateur disposera de deux possibilités pour entrer les paramètres nécessaires aux mesures :

* Il spécifiera la distance réelle entre les bornes de départ et d’arrivée, ce qui placera les bornes intermédiaires à distances égales les une des autres par l’utilisation de la classe MRUIntermediatesDistance ;
* Il spécifiera la distance entre la caméra et l’objet ainsi que l’angle d’ouverture de la caméra, ce qui placera les bornes intermédiaires à des différences d’angles égales par l’utilisation de la classe MRUIntermediatesCamera.

Pour analyser une vidéo, le programme fera appel à la librairie JavaCV afin de lire une vidéo et détecter l’objet en mouvement en déterminant la différence entre deux images consécutives.

Lorsque l’objet passera par les points intermédiaires, le temps de passage sera stocké dans un objet MRUIntermediate. À la fin des mesures sur la vidéo, le programme utilisera la classe MRUCalculations afin de calculer, pour chaque point intermédiaire, la vitesse et l’accélération.

Une fois ces calculs effectués, les résultats seront affichés à l’utilisateur. Ces résultats présentent les graphiques d’évolution de la vitesse et de l’accélération avec l’utilisation de la librairie JFreeChart. Enfin, puisque l’ensemble des résultats a été stocké, le programme vérifiera si, à un ou plusieurs points, la vitesse et/ou l’accélération sont égales à la vitesse et/ou l’accélération moyenne.

Les tests fonctionnels ont été effectués sur des libraires de traitement vidéo et de gestion des graphiques. JavaCV, wrapper Java pour OpenCV et FFMpeg, ainsi que JFreeChart ont été retenues. Ces librairies offrent des outils efficaces et documentés sans avoir à compliquer le projet.

Le projet tel qu’il est conçu sera à même de répondre aux objectifs qui lui ont été soumis, et disposera d’une évolutivité qui permettra de perfectionner certains points ou d’implémenter l’étude d’autres formes de mouvements, comme une trajectoire parabolique par exemple.