Eclipse Vision System Plugin

Diallo Algassimou

30 mars 2012

Table des matières

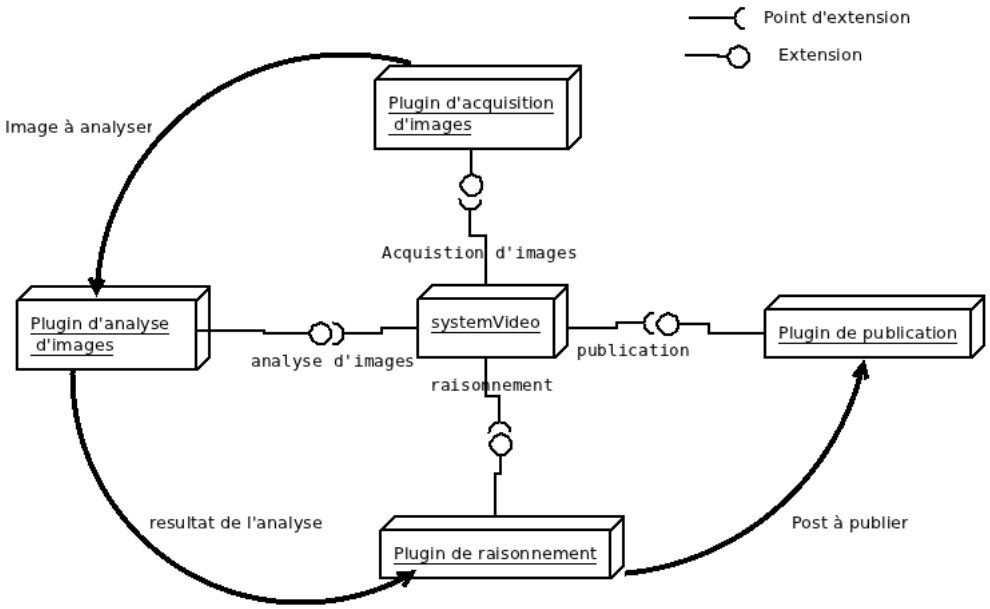

FIGURE 1 – Architecture

Présentation générale

1 Introduction

Eclipse Vision System permet de détecter des faits à partir d'un flux d'images et de créer des posts sur un réseau social. L'origine du flux d'images, les traitements apportés aux images ainsi que les faits recherchés dans celles-ci et les posts publiés sur un réseau social sont autant de points de variations qui sont pris en compte par Eclipse Vision System.

2 Architecture

Eclipse Vision System definit deux plugins pour des points d'extensions d'eclipse :

- actionSets (pour lancer le système) systemVideo
- PreferencePages (Permet d'avoir une page de preferences pour notre système) PreferencePages

systemVideo est le plugin centrale de notre système. Il fournit quatres points d'extensions qui permettent de capturer toutes les variations possibles dans l'utilisation du système. Ces points d'extensions sont les suivants :

- *imageAcquisitionExt* : point d'extension pour les plugins d'acquisition (capture) d'images ;
- *imageAnalysisExt* : point d'extension pour les plugins d'analyse d'images ;
- *imageReasonigExt* : point d'extension pour les plugins de traitement d'images ou le raisonnement avant publication ;
- *imagePublicationExt* :point d'extension pour les plugins de publication de posts.

Les plugins suivants ont été dévéloppés et peuvent être utilisés :

- **imageAcquisitionCamera** : plugin de capture d'une image à partir d'une webcam ;
- **imageAcquisitionVideo** : plugin de capture d'une image à partir d'une video ;
- **imageAnalysis** : plugin d'analyse d'une image (reconnait des visages dans une image) ;
- **imageReasoningSimple** : Publie si l'image contient plus de deux visages ;
- **imagePublishBlogger** : publie le message "Nous avons plus de deux visages" sur un google blog.

Tutorial

3 PagePreference

Pour chacun des points d'extensions l'utilisateur peut choisir le plugin qu'il veut utiliser parmi tous ceux disponibles, et aussi il peut renseigner l'adresse d'un serveur proxy et le port de celui-ci.

4 exécution

Dans cette exécution nous allons considérer que les options ont été fixées comme le montre la figure 2.

4.1 imageAcquisitionVideo

Ce plugin va capturer les images apartir d'une video pour cela il va demander le chemin de la video (figure 3).

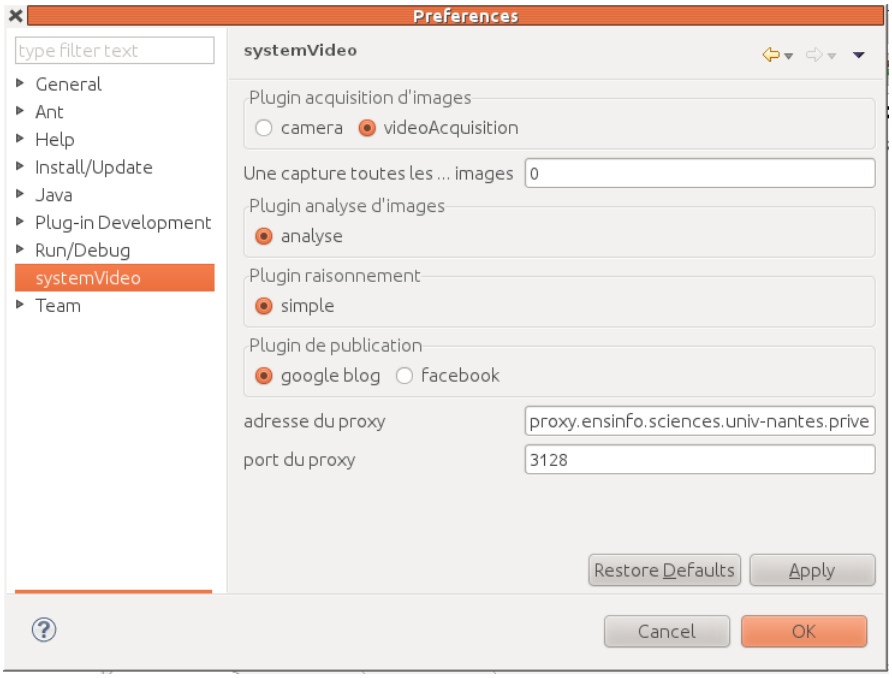

FIGURE 2 – Page de préferences

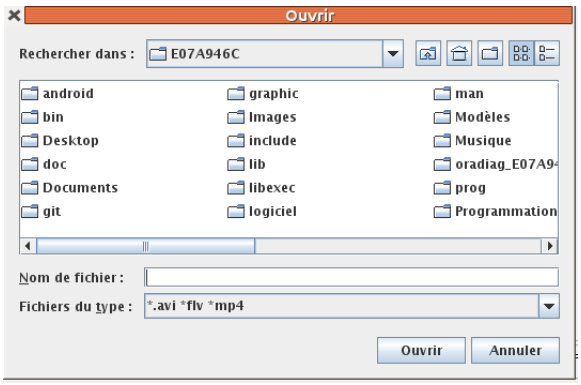

FIGURE 3 – Chemin du fichier video

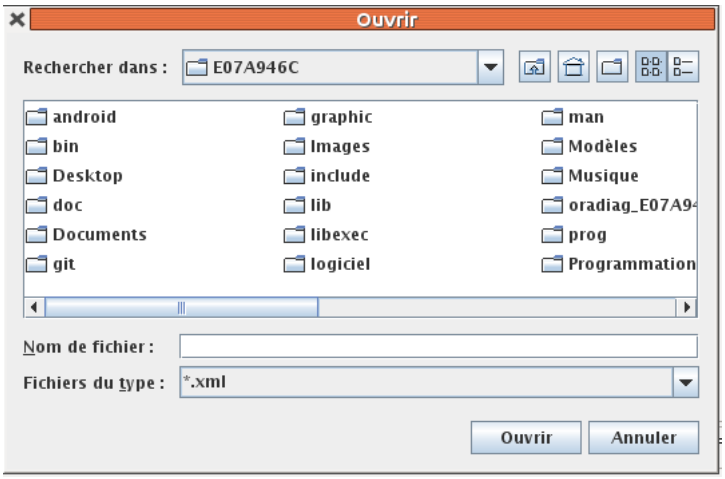

FIGURE 4 – fichier définissant un visage

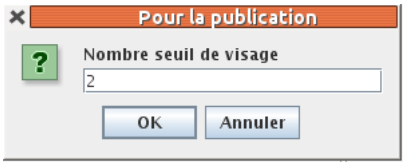

FIGURE 5 – Le nombre seuil de visages

4.2 imageAnalysis

Ce plugin analyse une image à la recherche de visages. Pour cela il a besoin d'un fichier permettant de definir ce qu'est un visage. Ce fichier est fourni par default avec les sources de openCV il suffit de renseigner le chemin du fichier (haarcascade_frontalface_alt.xml) dans la fenêtre qui s'ouvre (voir figure 5).

4.3 imageReasoning

Ce plugin compte le nombre d'images trouvées par le plugin précedent et post un billet sur le blog si le nombre de visage depasse un certains nombre (Ce nombre est renseigné par l'utilisateur comme le montre la figure **??**).

4.4 imagePublishBlogger

Ce plugin permet la publication de post sur un blog google. Pour cela il demande en parmatres les identifiants de connexion ainsi le nom de l'auteur du post (voir figure 6).

FIGURE 6 – identifiants de connexion compte google

Utilisation avancée

Comme le montre la figure 1 systemVide définit quatre points d'extensions. Pour ecrire un plugin pour systemVideo on choisit l'un de ces points d'extensions et on definit une classe qui implemente l'interface du point d'extension.

5 Points d'extension et interfaces

5.1 acquistion d'image

L'interface de ce point d'extension est la suivante :

```
package interfaces;

public interface IImageAcquisition {
        public void run () ;
        public void init () ;
        public void setIImageAnalysis (IImageAnalysis analyse);
}
```

méthod run

méthod init

méthod setImageanalysis

5.2 analyse d'image

```
package interfaces;

import java.awt.image.BufferedImage;
```

```
public interface IImageAnalysis {
        public void analyse (BufferedImage img);
        public void init ();
        public void setIImageResoning (IImageReasoning imgR);
}
```

5.3 raisonnement

```
package interfaces;

import java.util.List;

public interface IImageReasoning {
        public void reasonnig (List<Object> o) ;
        void init () ;
        public void addIImagePublish (IImagePublish imgP);
}
```

5.4 publication de post

```
package interfaces;

public interface IImagePublish {
        void publish (String title , String content) ;
        void init () ;
        void setProxyHost (String proxyHost);
        void setProxyPort (String proxyPort);
}
```

5.4.1 Architecture globale

1. **VisonSystem** : VisonSystem est le plugin de central. Ce plugin fournit quatre points d'extensions spécifiques pour tout le processus de traitement et définit les interfaces que devront implémenter les plugins clients ; A savoir *IImageAcuquisition*, *IImageAnalysis*, *IImageReasoning*, *IImagePublication*.
2. **imageAcquisitionCamera** : plugin d'acquisition (capture) d'une image à partir d'une video.
3. **imageAcquisitionVideo** : plugin d'acquisition (capture) d'une image à partir d'une webcam.
   Ces deux plugins définissent chacun une extension pour le point d'extension prévu à cet effet par visionSystem. La bibliothèque utilisée ici

pour la capture est la bibliothèque **xuggler**. Chacun de ces plugins possèdent dans sa classe d'implémentation un objet de type IImage-Analysis qui est ue interface définissant une méthode d'analyse. Le mode d'opération de ces deux plugins est le même : capture une image , instancie un objet de type IImageAnalysis et délègue l'analyse de cette image à l'objet IImageAnalysis. ...

4. **imageAnalysis** : plugin d'analyse d'une image. L'analyse de l'image quand à elle a été faite avec le bibliothèque **OpenCv**. Ce plugin définit lui aussi un obejt de IImageReasoning. Ce plugin récoit l'image à traiter de l'un des deux plugins précédents. L'analyse de l'image est effectué puis l'image est aisni délégué au plugin imageReasonig qui lui se charge du traitement final avant la publication. ...
5. **imageReasoning** : Ce plugin possède en son sein une liste d'image à publier dans laquelle il range au fur et à mesure les images qui devront être publiées. ...
6. **imagePublication** : Plugin de publication d'image, recoit les images à publier du plugin imageReasoning puis procède à la publication de celle ci.

   Comme on peut le constater les différents plugins présentés s'appuient sur des bibliothèques tierces. De ce fait nous créons des plugins qui regroupent ces bibliothèques qui deviennent ainsi des "repository" de bibliothèques tierces. Ceci est une approche parmi tant d'autre. l'avantage de cette approche est qu'elle évite une duplication des archives des bibliothèques dans tous les plugins utilisant ces bibliothèques. Si un plugin utilise une librairie, il suffira juste d'établir une dépendance avec le plugin (repository) contenant cette librairie. Ceci nous emmène donc à créer les deux plugins suivants :
7. **javacvPlugin** : repository de plugin pour javaCv
8. **xugglerPlugin** : repository de plugin pour xuggler

6 Conception

Le processus de traitement de l'image suis une séquence bien définie. Le

7