Eclipse Vision System Plugin

Diallo Algassimou

30 mars 2012

Table des matières

1 Présentation générale

Eclipse Vision System est un plugin est un simple plugin de vision capable de détecter des faits importants à partir d'images extraites d'une vidéo ou d'une webcam et de poster ces images sur un réseau social. Un scénario d'Eclipse Vision System peut se présenter ainsi :

- Capture d'images à partir d'une vidéo ou d'une caméra ;
- Analyse de l'image ;
- Traitement de l'image : détection d'effets particuliers sur l'image ;
- Publication de l'image.

Eclipse Vision System est donc architecturé autour d'un plugin central que nous appelerons par la suite *visionSystem*. *visionSystem* est le plugin de base fournissant plusieurs points d'extensions sur lesquels viendront se greffer les plugins effectuant les traitements enumérés ci-dessus.

2 Documentation

La réalisation du projet passe par le développement de plusieurs plugins que nous présenterons individuellement. On trouvera dans les sources les codes et commentaires associés pour chaque plugin au besoin. VisonSystem est le plugin central. Ce plugin fournit quatre points d'extensions spécifiques sur lesquels peuvent se greffer différents plugins, un poit d'extension par type de plugin.

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

Les plugins suivants ont été dévéloppés et peuvent être utilisés :

- **imageAcquisitionCamera** : plugin d'acquisition (capture) d'une image à partir d'une video ;
- **imageAcquisitionVideo** : plugin d'acquisition (capture) d'une image à partir d'une webcam ;
- **imageAnalysis** : plugin d'analyse d'une image ;
- **imageReasoning** : plugin de traitement de l'image avant publication ;
- **imagePublication** : plugin de publication d'images.

2.1 Mode d'utilisation

Une fois l'application lancée sous eclipse, l'utilisateur a la possibilité de renseigner un certain nombre de paramètres. pour se faire :

- dans la nouvelle fenêtre eclipse qui s'ouvre, cliquer sur *window* ;
- ensuite dans le menu qui s'affiche, cliquer sur *Preferences* ;
- renseigner vos paramètres puis sur cliquer sur *Apply* et *Ok*.

2.1.1 Illustration

inclure ici quelques captures d'écrans

2.2 Utilisation avancée

Cette section présente plus en détails les différents points d'extension et l'architecture globale du systéme. Les utilisateurs expérimentés peuvent définir des extensions pour les points d'extension qu'offre VisonSystem.

2.2.1 Points d'extension et interfaces

Les différents points d'extensions(*imageAcquisitionExt, imageAnalysisExt, imageReasonigExt, imagePublicationExt*) présentés ci dessus doivent implémenter respectivement chacun les interfaces suivantes :

- IImageAcuquisition :

```
public void run () ;
public void setIImageAnalysis (IImageAnalysis analyse);
```

- IImageAnalysis :

```
public void analyse (BufferedImage img);
public void setIImageResoning (IImageReasoning imgR);
```

- IImageReasoning :

```
public void reasonnig (List<Object> o) ;
public void addIImagePublish (IImagePublication imgP);
```

- IImagePublication :

```
void publish (String title , String content) ;
```

2.2.2 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 IImageAnalysis 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

3 Conception

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

4