ViSEvAl software Description

CONTENTS

DESCRIPTION

The software ViSEvAl (ViVualisation and EvAluation) provides a GUI interface to visualise results of video processing algorithms (such as detection of object of interest, tracking or event recognition). Moreover this software can compute metrics to evaluate specific tasks (such as detection, classification, tracking or event recognition).
The software is composed of two binaries (ViSEvAlGUI and ViSEvAlEvaluation), and several plugins. The users can add their own plugins to define a new metric for instance.

INSTALLATION

OS: Linux (tested on Fedora 12)
gcc 4.4.4

Dependencies:
Two libraries are mandatory: **QT4**(for GUI facilities and plugin facilities), **poco** library (http://pocoproject.org/) (for automatic xml parser)
Optional: **FFMpeg** (only use in the plugin to load .ASF video)

1. go in the ViSeAval directory (call **SoftwareDirectory** in the next)
2. launch the script **./install.sh**. The script will create all the makefile needed by the application and the plugins, and will compile all the code. If all is ok, you will find the executables in **SoftwareDirectory/bin/appli** directory.
3. Type the bash command:
   setenv LD_LIBRARY_PATH $SoftwareDirectory/lib:/usr/local/lib$LD_LIBRARY_PATH
   (to tell to the applicatin where is the **ViSEvAlLib**, and the optional libs for **ffmpeg**)
4. go in the directory **bin/appli**
5. run **ViSEvAlGUI** for the GUI tool or run **ViSEvAlEvaluation** for the command line tool

ViSEvAlGUI In the menu: File -> Open..., select the desired .conf file

ViSEvAlEvaluation file.conf result.res [0-1] [0-1]
file.conf the desired configuration file
result.res the file where the results while be wrote
[0-1] optional value 0: the results are printed for each frame, 1: only the global results are printed
[0-1] the evaluation of the detection (XML1) and of the fusion (XML2) is only done on the common frames.

DELIVERY CONTENTS

FILES LIST

./bin:
/bin/appli the directory contains the tools binaries
./bin/plugins the directories contain the .so of the different plugins
./bin/plugins/CDistance
./bin/plugins/CEventMetric
./bin/plugins/CFrameMetric
./bin/plugins/CLoadingVideoInterface
./bin/plugins/CObjectFilter
./bin/plugins/CTemporalMetric

./doc: the directory contains the documentation
ShortDocumentation.doc

./include:

./include/appli the directory contains the header files of the application

./lib: the directory contains the ViSEvAl library

./scripts: the directory contains several useful tools
asf2jpeg binary which extracts jpeg images from an asf video
makeVideoFile.sh create the .vid file
splitxml1-2-3file.sh split a xml1, xml2 or xml3 file in several tiny

./src:

./src/appli the directory contains the .cpp files of the application (remove them if only the executable is shared)
./src/appli/icones contains the icones for the tool bar of the GUI
./src/plugins contains the header and source files of all the plugins
./src/plugins/CDistance
./src/plugins/CEventMetric
./src/plugins/CFrameMetric
./src/plugins/CLoadingVideoInterface
./src/plugins/CObjectFilter
./src/plugins/CTemporalMetric
./xsd: contains the .xsd file to describe xml format of the different files
camera.xsd *format for the camera files*
datta.xsd
format for the XML1-2-3 and timestamp files

DOCUMENTATION

DEFINITION OF THE .CONF FILE

An example of a .conf file can be found in the ./bin/appli directory: example.conf
The user can write a .conf file to configure the tools (or use the **save** function of the GUI):

- the extension of the file is **.conf**.
- The comments begin by **#**
- The scheme of the attribute is **keyword "value"**

The different attributes are:

- **SequenceLoadMethod** to define which method is used to load the videos. The possible values are defined by the plugins. With the default plugins the values can be
  - **"JPEG-Images"** to load a **.vid** file
  - **"ASF-Videos"** to load a **.asf** file
- **NeedChangeImageReferential** to use or not a change of referential of the detected data (xml1, xml2) to be compatible with the ground truth data. The possible values are:
  - **"Yes"**
  - **"No"**
- **UseStaticFilter** to filter or not the detected objects: for static object (same 2D center and same 2D size), only the first occurrence is kept. The possible values are:
  - **"Yes"**
  - **"No"**
- **SequenceLocation** to indicate where is the video file (either **.vid** or **.asf**) and which camera is associated. The format of the value is:
  - **"CameraID:absolute_path_of_the_file"**
- **TrackingResult** to indicate where is the file of the tracking result (**.XML1.xml**) and which camera is associated. The format of the value is:
  - **"CameraID:absolute_path_of_the_file"**
- **FusionResult** to indicate where is the file of the data fusion result (**.XML2.xml**). The format of the value is:
  - **"absolute_path_of_the_file"**
- **EventResult** to indicate where is the file of the event recognition (**.XML3.xml**). The format of the value is:
  - **"absolute_path_of_the_file"**
- **GroundTruth** to indicate where is the file of the ground truth (**.xgtf**) and which camera is associated. The format of the value is:
  - **"CameraID:absolute_path_of_the_file"**
- **[XMLSchemeData]** to indicate where is the xml schema (**.xsd**) associated to xml1, xml2, xml3 files to check the correct format of the xml files. Optional value. The format of the value is:
  - **"absolute_path_of_the_file"**
- **[XMLSchemeCamera]** to indicate where is the xml schema (**.xsd**) associated to the camerafiles to check the correct format of the xml files. Optional value. The format of the value is:
  - **"absolute_path_of_the_file"**
- **TimeStampCamera** to indicate where is the timestamp file (either **..txt**) and which camera is associated. The format of the value is:
  - **"CameraID:absolute_path_of_the_file"**

- **XMLCamera** to indicate where is the xml camera file (either ..**xml**) and which camera is associated. The format of the value is:
  - **"CameraID:absolute_path_of_the_file"**
- **MetricFrame** to indicate which frame metric to use. The format of the value is:
  - **"(Mono,Fusion)Metric_name:Metric_user_name:Distance_name:Distance_threshold:Filter_name[:Filter_parameter_name1:Filter_parameter_value1....]".** Values between **(value1,value)** have to be chosen: **Mono** to evaluate on XML1, and **Fusion** to evaluate n XML2. Values between (**[value]**) are optional.
- **MetricTemporal** to indicate which frame metric to use. The format of the value is:
  - **"(Mono,Fusion)Metric_name:Metric_user_name:Distance_name:Distance_threshold:Filter_name[:Filter_parameter_name1:Filter_parameter_value1....]".** Values between **(value1,value)** have to be chosen: **Mono** to evaluate on XML1, and **Fusion** to evaluate n XML2. Values between (**[value]**) are optional.
- **MetricEvent** to indicate which frame metric to use. The format of the value is:
  - "**Metric_name:Metric_user_name[:Distance_parameter_name1:Distance_parameter_value1....]**". Values between (**[value]**) are optional.

DEFAULT PLUGINS

When one of the both tool is launched, the available plugins are displayed. With a default configuration these plugins are:

```
-----------------------------------
Loading video interfaces:

ASF-Videos
JPEG-Images

-----------------------------------
Loading distance:

3DBertozzi
3DDiceCoefficient
3DOverlapping
Bertozzi
DiceCoefficient
Overlapping
-----------------------------------
Loading object filter:

CloseTo         with parameters X, Y and Threshold
FarFrom         with parameters X, Y and Threshold
Identity
-----------------------------------
Loading frame metric:

M1.1
M1.2
M2.1
M2.2
M3.1
```

Loading temporal metric:
M3.2
M3.4
M3.5

Loading event metric:
M4.1
M4.2 with parameter Duration
M4.3 with parameter Frame
M4.4 with parameter Duration

THE DIFFERENT SCRIPTS

To help the user in managing the different input of the evaluation tools, several scripts are provided:

- **asf2jpeg**
  This script based on ffmpeg extract the jpeg images from an asf file, and save in a file the timestamps of all the images. This last file is used in particular for synchronising the cameras and the different detection files (xml1, xml2, xml3).
  Usage : **asf2jpeg asf_file_name timestamp_file_name**
  Extracts in the current directory the jpeg images of the **asf_file_name** and writes the timestamps in t**imestamp_file_name**
  Needs that **convert** function (ImageMagick) is installed.

- **makeVideoFile.sh**
  This script makes the **.vid** file needed by the GUI tool.
  Usage: **makeVideoFile.sh directory filename**
  Writes in **filename.vid** the name of the jpeg files in **directory**. Adds automatically the extension **.vid** at the end of **filename**.
- splitxml1-2file.sh
  The xml1, xml2 files are often very heavy. To gain time, this script is proposed to split in several files the bigger ones.
  Usage: **splitxml1-2-3file.sh file_name num_frame proto_name type**
  Splits the xml1/xml2/xml3 file **file_name** in several files by using the desired number of frame **num_frame** by file. The new files follow the prototype name **proto_name,** according to the **type** of the file: **1** for xml1 one, **2** for xml2 one and **3** for xml3.
  Be careful: do not use a number in the proto_name.

THE .XSD FILE

The tools take as input several xml files (for the tracking, for the data fusion, for the detected event, for the camera, for the ground truth,...).
To save time in development and in updating, we have proposed to use xml scheme. The .xsd files are generated with an example of each xml files thanks to the **Trang** tool:
http://www.thaiopensource.com/relaxng/trang.html

Two files are generated:

- camera.xsd for the xml schema of the camera file
- data.xsd for the xml scema of xml1, xml2, xml3 and timestamp files

Note: the xsd files have to be manually checked, in aprticular to verify the type of the values (replace token by string, etc...)

Then the **xsdcxx** function (yum install **xsd** if necessary and probably **poco** library (http://pocoproject.org/)) is used to generate the associated .hxx and .cxx files:

- xsdcxx cxx-tree --generate-serialization --root-element Camera camera.xsd
- xsdcxx cxx-tree --generate-serialization --root-element Cofriend data.xsd

The .xsd file mus be put inside xsd directory, the .hxx file must be put inside include/appli directory and the .cxx files must be put inside src/appli directory

The GUI is composed of several view, the main view, the multi-camera view and the parameter window.

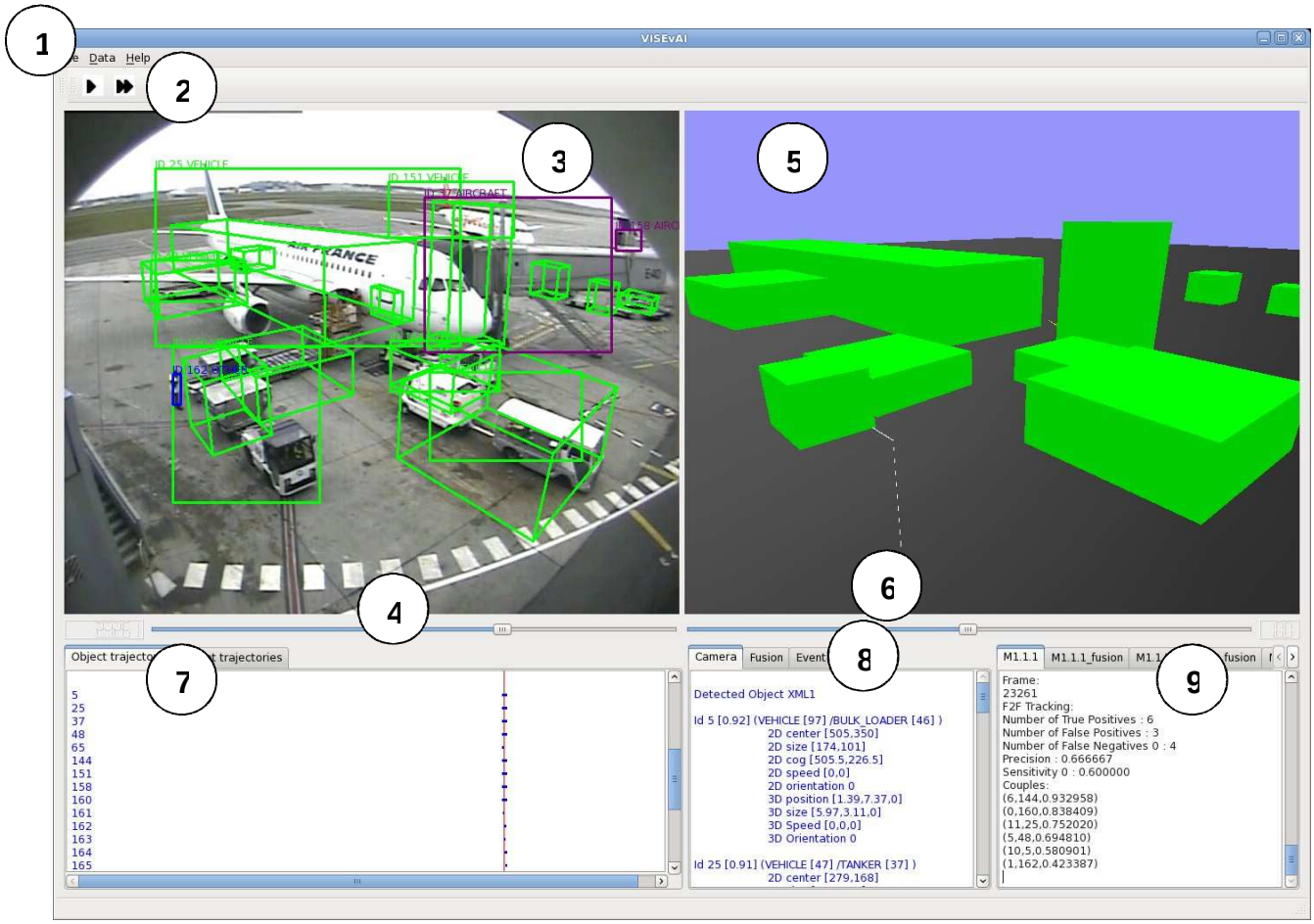

Figure 1 Main view of the GUI

Figure 2 Multi-camera view

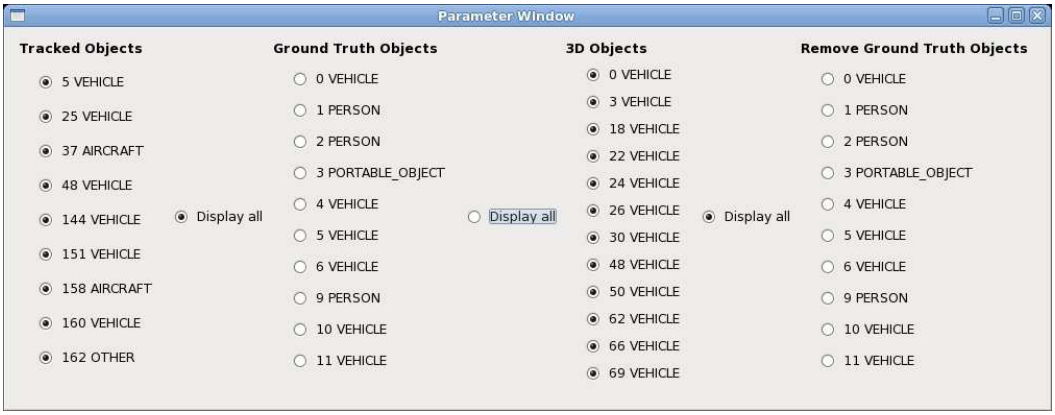

Figure 3 Parameter window

DESCRIPTION OF THE MAIN VIEW

1. Menu
    - **a. File**
        - i. **Open …**
          Open a configuration file which contains the path of the different data files, and parameters for the different metrics. The extension of this file is .conf.
        - **ii. Save …**
          Save a configuration file which contains the path of the different data files, and parameters for the metrics. The extension of this file is .conf
        - **iii. Exit**
          Close the application
    - **b. Data**
        - **i. Open Video**
          Load a file containing the path of the file describing the video. The loading method is asked and the identifier of the camera is chosen.
        - **ii. Open F2F tracking xml file (xml1)**
          Load the xml file containing the results of the frame to frame algorithm. The identifier of the camera is asked. The extension of the file is XML1.xml
        - **iii. Open fusion tracking xml file (xml1)**
          Load the xml file containing the results of the fusion algorithm. The extension of the file is XML2.xml
        - **iv. Open event xml file (xml3)**
          Load the xml file containing the results of the event detection algorithm. The extension of the file is XML3.xml
        - **v. Open ground truth file**
          Load the xml file created with the Viper software. The extension is .xgtf
        - **vi. Choose camera ID**
          Choose the identifier of the camera to display.

vii. Choose parameters for metrics

Open a new window which allows the user to change the parameters of the metrics: distances, filters and thresholds.

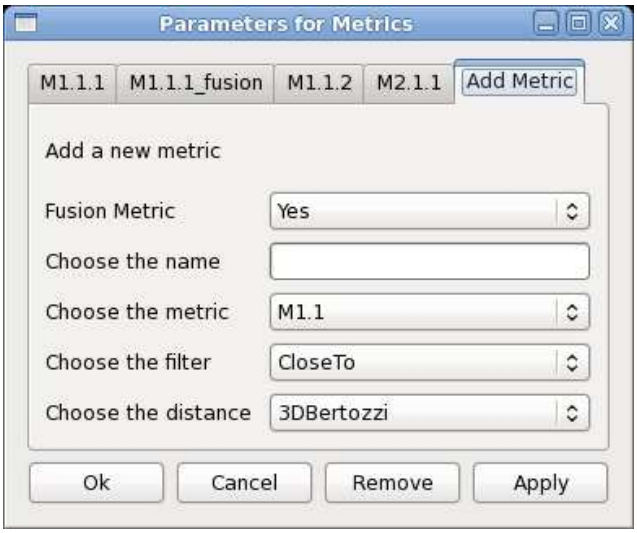

Figure 4 Window to change parameters of a metric or create one

This figure shows the different parameters available for adding a metric. First the user choose if the metric will be used with fusion data or not (xml2 file or not). The the metric is chosen and a name is given. Then the filter and the distance are chosen.

The user can also modify existing metrics by selecting the corresponding tab. The distance and its threshold can be changed.
The parameters of the filter can also be updated.

The window has also four buttons:

- **Apply**: apply the changes of the user concerning the metrics
- **Ok**: close the window and apply the changes of the user concerning the metrics
- **Cancel**: close the window without changing nothing
- **Remove**: remove the current metric, an advertising window is displayed to confirm the decision.

viii. Open camera

Load the file containing the calibration of a camera.

ix. Open timestamp

Load the file containing the timestamp of a camera.

c. Help

i. About

2. Tool bar

a. Play

Play the video frame by frame

b. Play faster

Multiply by 2 the play speed. The frames are displayed every 2 frames and so on.

3. Real Video

Display the real images of the video. The number of the displayed frame is written on the bottom left of the image. Bounding boxes of the detected objects are also displayed with their ID and the type provided by the classification.

4. Real video slider

The slider allows the user to easily displace in the video sequence.

5. 3D visualization

Display 3D boxes of the detected objects (blue: unknown, red: person, green: vehicle, dark magenta: plane, more the color is darkest, better is the confidence value of the detection). The 3D view is active and the user can move the camera with the mouse (left button: rotate the scene, middle button: translate the scene in the camera axis, right button: translate the scene in the perpendicular axis with the camera one).

Some shortcut exist to place the camera (0: to have a top view of the scene, 5: to emulate the view point of the camera 5). The cameras are automatically made by using the corresponding calibration file.

A simple context is also displayed, constituted of the different lines on the ground of the apron.

To help visualization shortcut **b** allow a blender view and **w** allows a wireframe view.

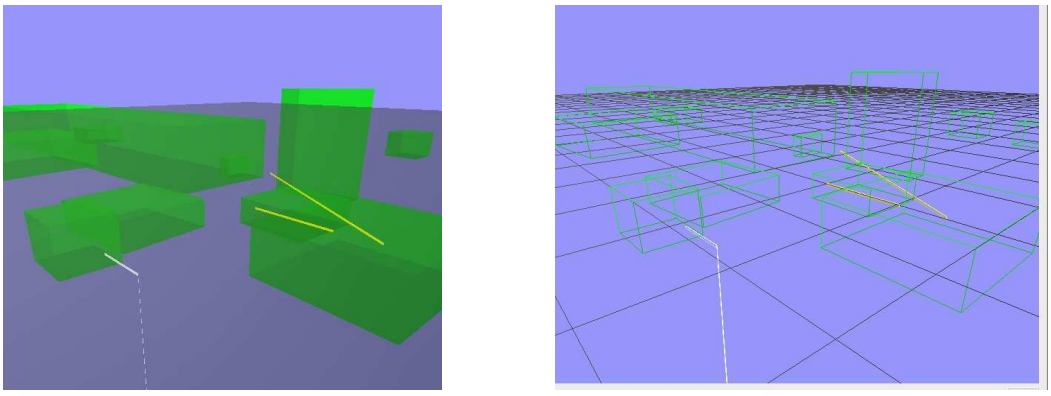

Figure 5 Blend and wireframe views of a 3D scene

6. Scale slider

This slider allows the user to change the scale of the images and the interface. The user can zoom from 1% up to 200%.

7. Objects and events trajectories

Display for the whole sequence the temporal description of the detected object and recognized events in different tabs.

For each object the id is displayed on the right of the window and a line shows when an object is occurred in the sequence. The dark magenta lines display objects detected with data fusion, blue lines display detected object in frame to frame, and green lines display ground truth objects.

For each recognized event the ID and the name is displayed on the right of the window. The dark magenta lines display recognized events and green lines display ground truth events.

A red line models which frame is displayed.

8. Objects and events description

Display in text mode information about the detected object in the current frame (Id, confidence, type and associated probability, subtype and associated probability, 3D position, 3D size, orientation and speed). The fusionned data are displayed in dark magenta, the data detected with frame to frame algorithms are displayed in dark blue, and the ground truth data are displayed in green.

Information about the events occurring in the current frame is also displayed.

In the same way the detected events are displayed in magenta, and the ground-truth events are displayed in green. A tab is used for detection, fusion and event.

9. Metrics

Different information about the metrics are displayed in the window: the name of the metrics, the true positive rates, the false positive rates, the false negative rates, the precision, the sensitivity and the couple (ground truth-detection made).

MULTI-CAMERA VIEW

When several video files are opened, a sample of all the view is displayed in the multi-views window.

PARAMETER VIEW

The user can easily changes the displaying of the object in the different parts of the interface. The first column (Detected Objects) shows the detected objects (frame to frame) displayed in the current frame, by selecting/un-selecting an object, represented by its identifier and its type, the corresponding bounding-box is displayed/un-displayed on the current image.

The same behaviour happens for the data fusion objects, the 3D object appears/disappears in the 3D view.

The bounding boxes of the ground-truth objects can also be displayed on the current image.

The display all button allows the user to display or un-display all the objects of one category.

The tools can be find in the directory **./scripts.**

- **asf2jpeg**

  This binary is based on ffmpeg and extracts the jpeg images from an asf file.
  Usage: **asf2jpeg** <ASFfilename>
  Extracts in the current directory the jpeg images of the ASFfilename
  Needs that convert function (ImageMagick) is installed.
  In testing, be careful.

- **makeVideoFile.sh**

  Usage: **makeVideoFile.sh** <directory> <filename>
  Writes in filename the name of the jpeg files in directory. Adds automatically the extension .vid at the end of filename. The jpeg file must be named 000….000XXXX.jpg or 000….000XXXX.jpeg

- **splitxml1-2-3file.sh**

  Usage: **splitxml1-2-3file.sh** file_name num_frame proto_name type
  Splits the xml1 file file_name in several files by using the desired number of frame num_frame by file. The new files follow the prototype name proto_name according to the type of the xml file (1, 2 or 3)
  The application takes about 3 minutes to load a 1.5 Go size file.
  Be careful: do not use a number in the proto_name.