Project no.:	**IST-FP6-FET-16276-2**
Project full title:	**Learning to emulate perception action cycles in a driving school scenario**
Project Acronym:	**DRIVSCO**
Deliverable no:	**D3.3**
Title of the deliverable:	**Demonstration of the micro-chip implementation**

Date of Delivery:	04.08. 2009	
Organization name of lead contractor for this deliverable:	UGR	
Author(s):	F. Barranco, M. Tomassi, S. Granados, E. Ros (UGR)	
Participant(s):	UGR (revised by UGE)	
Work package contributing to the deliverable:	WP3	
Nature:	D,R, PU	
Version:	1.0	
Total number of pages:	28	
Start date of project:	1 Feb. 2006	**Duration:** 42 **months**

Project Co-funded by the European Commission		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other program participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Revision Notes: This demo will be shown in the final project review as a stand-alone vision-on-chip demonstrator. Deliverable cross-revised by UGE.
Delay justification: No delay

Summary

This deliverable describes the on-chip vision system demo. We have built a PC-FPGA interface (open rtvision) for facilitating the demo building. Beyond the demo itself this interface may be interesting for other developers and therefore has been released as open software and is available at: http://code.google.com/p/open-rtvision/. In appendix I, we attach the user's manual of this interface tool with some processing examples of the vision-on-chip demo. An illustrative video (of how to use this interfacing tool) can be found at http://code.google.com/p/open-rtvision/.

The purpose of the vision-on-chip demo is to illustrate how the different processing engines (optical flow, disparity and local contrast descriptors) can be programmed in the FPGA and work in real time. We can see working in real-time these different set-ups or illustrative examples:

- Motion processing on-chip. Multi-scale phase-based optical flow engine.
- Stereo processing on-chip. Multi-scale phase-based disparity extraction engine.
- Local contrast descriptors on-chip. Energy, phase and orientation extraction on-chip.
- Low level vision-system on-chip. All the processing engines working in parallel on the same chip.
- Condensation module on-chip. We include the condensation module on-chip with motion and disparity estimation extractors to illustrate how it saves considerable communication bandwidth keeping the low level representation maps. In order to allow a quality comparison we have also built a software module performing real-time decondensation.

Content

1. Introduction

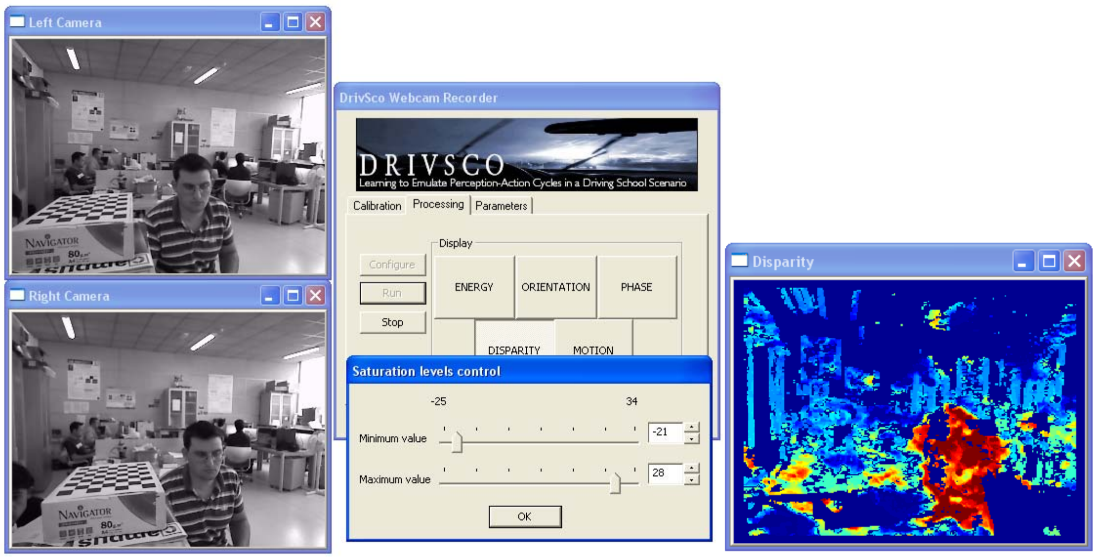
This deliverable briefly describes the demonstrator of the low level vision on-chip. This demonstrator includes different chip configurations. The FPGA device is programmed with different processing engines. In this way, specific purpose datapaths are built on-chip for the different vision modalities. The demo allows configuring the chip several times to try out the different processing engines developed in the project. Furthermore, the chip can be configured to include all the processing engines (optical flow, disparity and local contrast descriptors) to work in parallel on the same chip (as described in D1.3).

We have built a PC-FPGA interface to dynamically change the configuration of the chip and evaluate different processing engines. This PC-FPGA interface is called open rtvision and has been released as open software (http://code.google.com/p/open-rtvision/) because it can be of interest for other FPGA developers. A short user's documentation about how to set-up and use this interface is included as Appendix I of this deliverable. An illustrative video (on how to use this interfacing tool and configuring different processing engines on-chip) can be found at http://code.google.com/p/open-rtvision/.

Beyond the demo itself the interface has been developed including the possibility of processing real-time video captured from on-line cameras or stored video sequences from hard disks. This allows the use this software also to evaluate the accuracy of the different processing engines using benchmark images and sequences. Furthermore it allows the efficient processing of previously acquired sequences such as DRIVSCO driving sequences. The application allows storing the obtained results and evaluating also the computing speed while the processing is done on-line to evaluate its real-time capability and stability in terms of latency and computing speed.

2. Different processing engines on-chip

The following screen snapshots show the visualization of the different system processing engines. In Fig. 1 we show an example of the disparity for a lab scene; the saturation level control is available for controlling the colormap application visualization. In Fig. 2 we show the result of the disparity results using a well-known set of images (Tsukuba images).



Minimum value -21

Maximum value 28

Fig. 1. Example of disparity processing. The saturation level control is available for change the colormap application visualization. The colormap in the case of the disparity shows closer objects with warm colors and farther ones with cold colors.

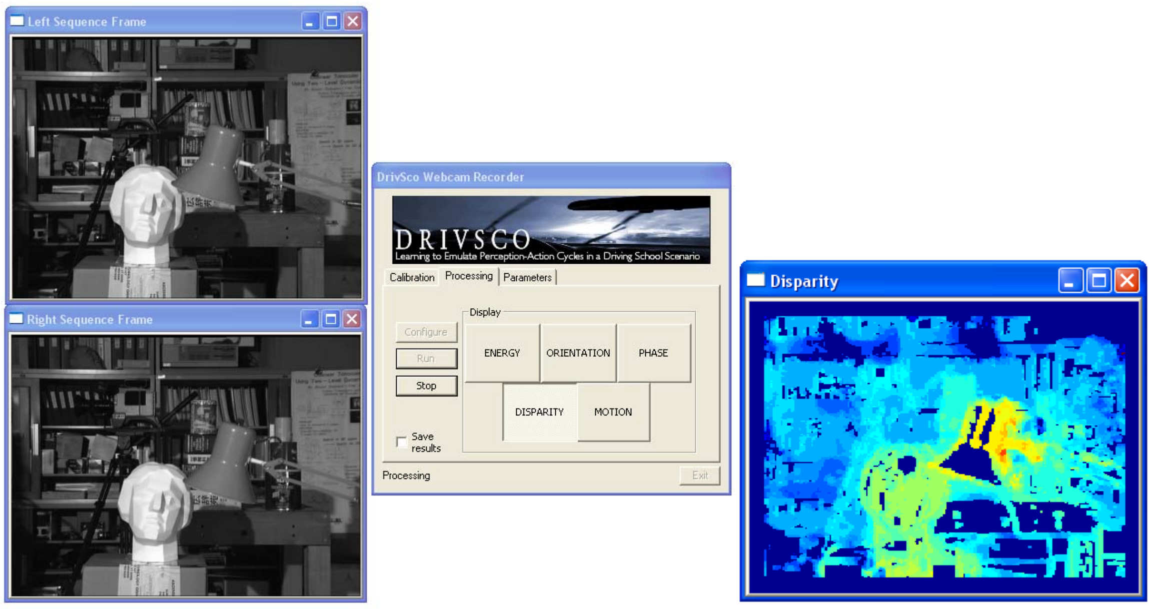


Fig. 2. Example of disparity processing on the well-known Tsukuba images.

In Fig. 3 we show an example of the local features computation using the left camera as the input source. In the different windows we show the Energy, the Orientation and the Phase results.

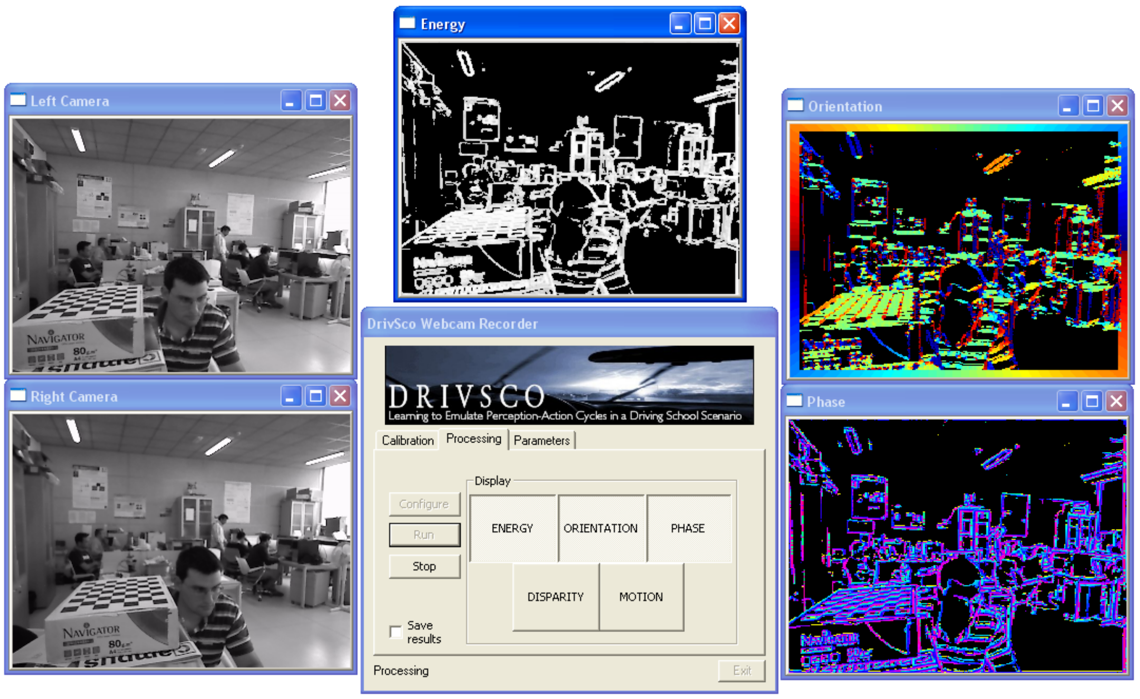


Fig. 3. Example of use of the application showing the local features computation results. The windows show the Energy, the Orientation and the Phase results (clockwise order beginning with the Energy window, above the main application window).

Fig. 4 shows the results of the optical-flow computation. The optical-flow is computed for the Left camera frames while the man in the scene is standing up.

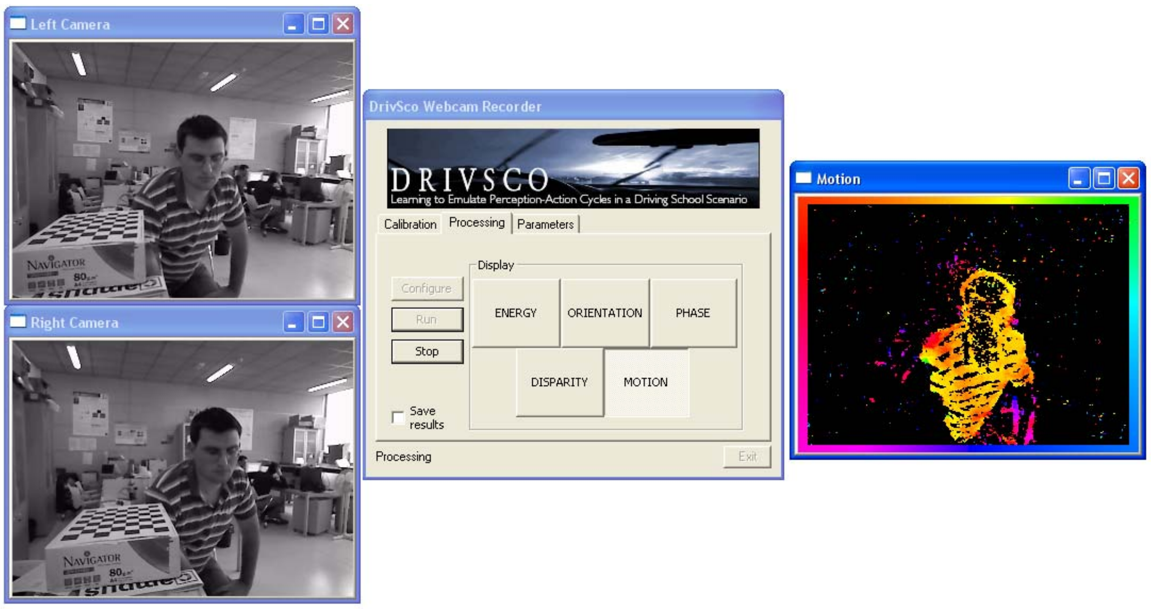


Fig. 4. Example of use of the application showing the optical-flow computation results. The input data is the frame from the Left Camera.

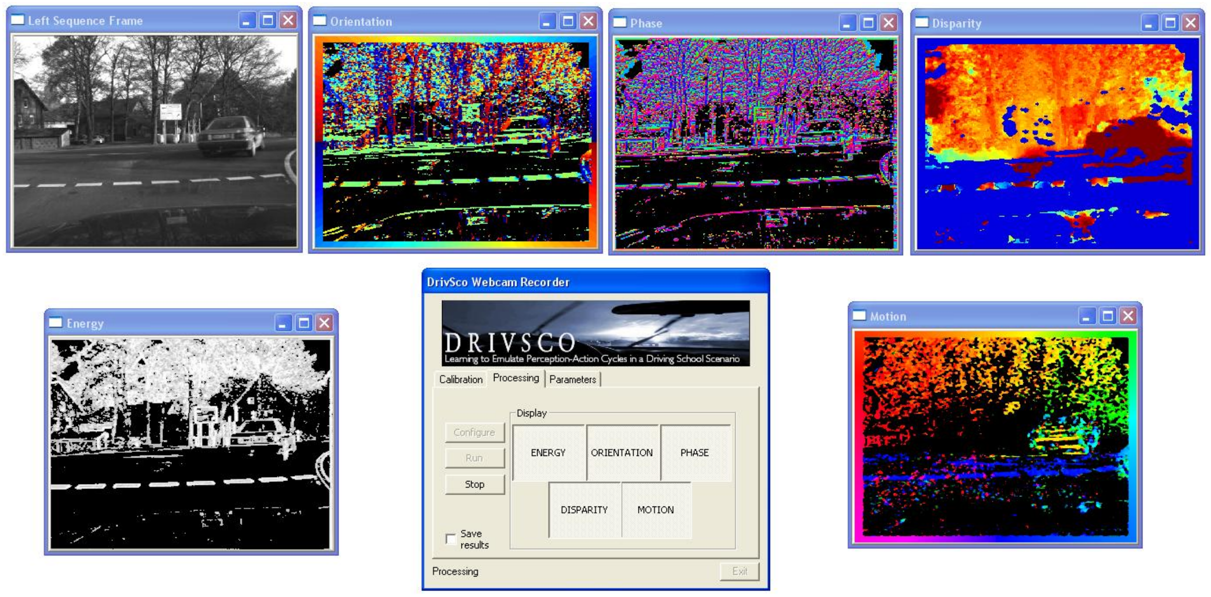


Fig 5. PC-FPGA interface. Efficient drivers, communication software and real-time visualization and monitoring are included in this interface.

Finally, Fig. 5 shows a snapshot of the screen when all the processing engines are running in parallel on the same chip. A DRIVSCO driving sequence is processed. The different low level vision cues are shown on different windows.

3. Condensation: Demonstration of the micro-chip implementation

In order to illustrate the hardware condensation core functionality, we have implemented and integrated a demonstrator in the software platform developed for the project (open rtvision). We have produced hardware chip configurations including the disparity, the optical flow and a core for all the features together, adding in all the cases the cores for the local features (in the examples we have used energy as relevance indicators), therefore we need the values of the energy for the condensation processing. In all the cases, a module for the condensation process is included, too.

We enumerate the new functionalities for this demonstrator including condensation:

- Presentation of the results for the original feature, and for and the de-condensated map (in order to allow a comparison with the original one). For the sake of simplicity, all the examples are shown for the disparity.
- Addition of the RP (relevant points) of the original feature: they can be added from the software to the hardware core using a lookup table that is loaded in the device memory; they can be included directly from another processing element in the final displaying as we will explain in the following.
- Presentation of the original feature and the condensated information fulfilling the DRIVSCO requirements (images of 512x512 pixels and the system working at least at 25 fps). Usually, the application should work using only the condensated feature, but we also show the original information in order to analyze the differences between them and to compare the small quantity of data we use with the condensation processing.
- Displaying of the data bandwidth of the original information and the bandwidth using the condensated data.

In the previous list the functionalities are shown as four different modules, however, all of them are integrated in the same platform.

In the first case, the user activates the condensation when the visual feature (the disparity in this example) is enabled. When the user clicks on the “Disparity” button it enables three new windows: the window for the original disparity (read from memory because in this case the hardware core produces the original feature and the condensated feature data), the window of the condensated feature and the window of the de-condensated feature. The de-condesation is shown to allow comparison between it and the original feature. In Fig. 6 we show these windows (their captions identify the displayed information) and the application main dialog. It is important to compare the volume of data of the condensated feature, it represents (in this example) about a 20% of the original one. The original information can be reconstructed (de-condensated information) with that small amount of data and without significant differences between them (the MSE between hardware generated cues and decondensated ones is about 1.26).

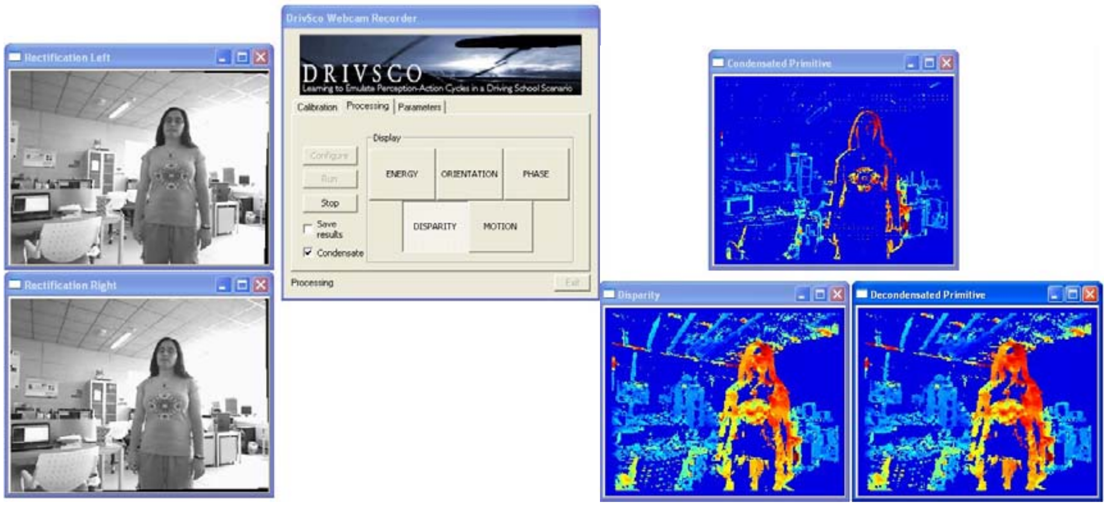


Fig. 6. (Left) the rectified input images. (Center) the application main dialog window with the condensation enabled. (Right) the three windows of the condensation processing: the condensated information (top), the original hardware-generated primitive (bottom-left) and the decodensated data (bottom-right).

The condensated information shows the values of the hexagonal grid and the RPs (relevant points). On the other hand, the de-condensated information is an interpolation of the condensated one. The NaN values of the original information are used as a mask for simplifying the visual comparison between the de-condensation and the original feature. If needed, the denser de-condensated information can be shown.
For the implementation described previously three new modules are added to the platform: one of them for reading the grid and RP data from the FPGA memory, another one for displaying the condensated information and the last one for displaying the de-condensated information. As previously mentioned, the new modules are integrated in the platform for the whole system.

In the demonstrator it is included the functionality of focusing the condensation on a specific area, as we present in Fig. 7, passing a lookup table to the hardware core as a new parameter. We can see a square region with the points highlighted (marked) as RP. The points of this area are selected by the software writing their addresses in a reserved device memory area. This functionality is useful if it is necessary the extraction of the original information from a determined area or region (without condensation). For example, if we condensate the optical flow, the RP software mask can be the area of an IMO processed in a different module. In this case, the estimation of the optical flow for the IMO region would be processed more accurately (without interpolating the feature data), providing the complete information of the estimation in this area.

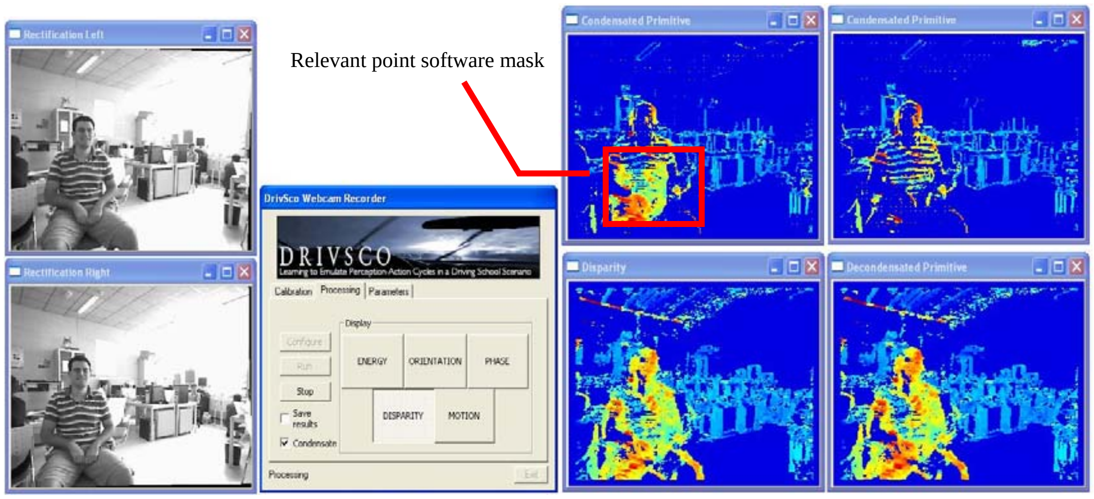
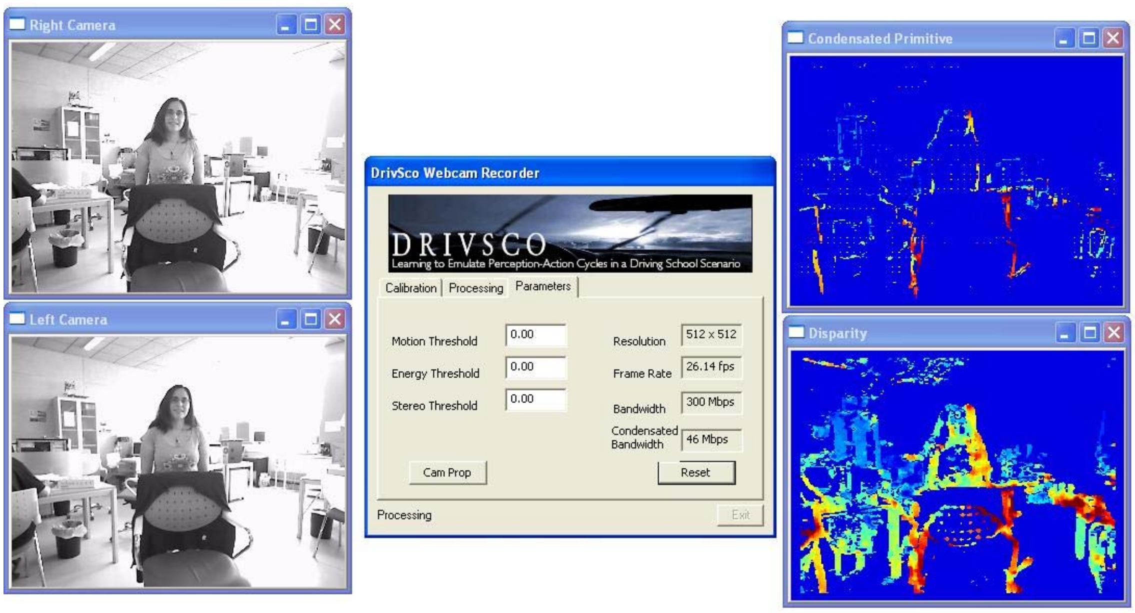


Fig. 7. (Left) the rectified input images. (Center) the application main dialog window with the condensation enabled. (Right) the condensation processing windows. At top-right, we show the condensated information (right) and the condensation mode including the RP mask (highlighted square area) from the software. In the left-top image we see higher density in this square region that has been marked as relevant (no condensation is done in this region to preserve original representation). In the bottom images (decondensated maps) no difference can be seen between both approaches.

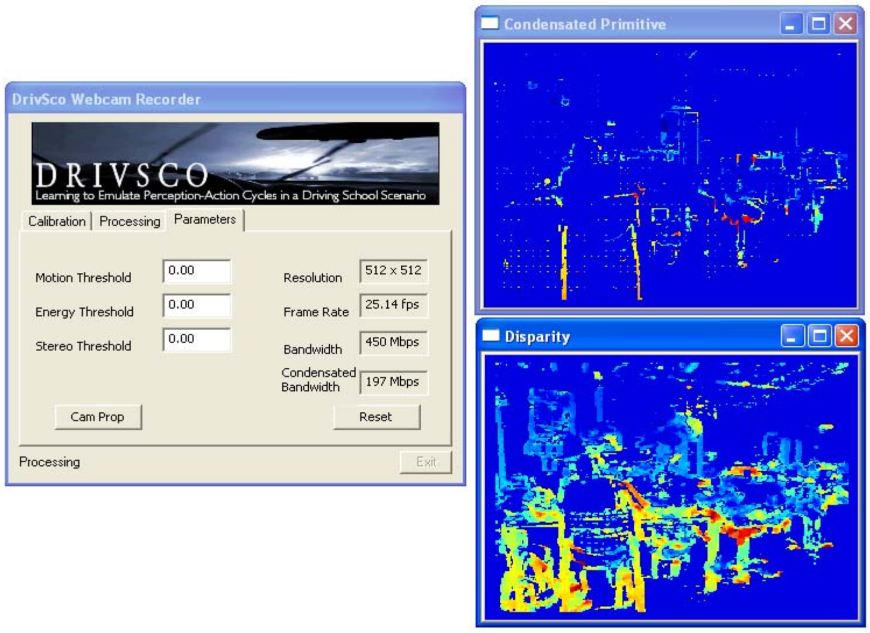
The possibility of adding relevant points from other modules or processing stages allows integrating cues (reliable cues) from other modules such as higher processing stages or specific modules such as SIFT primitives or SURF. All these points or backpropagated primitives can be naturally added to the relevant point list of the condesated representation.



Motion Threshold 0.00
Energy Threshold 0.00
Stereo Threshold 0.00
Resolution 512 x 512
Frame Rate 26.14 fps
Bandwidth 300 Mbps
Condensated Bandwidth 46 Mbps

Fig.8. (Left) rectified input images. (Center) main application dialog including working speed estimators such as computing Frame Rate, bandwidth and bandwidth required by the condensated maps. (Right) Top the condensated map and Bottom the original map. The frame rate, as we can see in the parameters tab of the application main dialog is of 26 fps (fulfilling the DRIVSCO specification).

The next case illustrates the DRIVSCO requirement fulfilling. In the main application dialog is included a checkbox for enabling or disabling the de-condensation processing. The de-condensated information and the original feature information are not necessary for the DRIVSCO application since only condensated maps are transferred, but in this case, we display the original feature and the condensation maps in order to check the volume of information we manage with the condensation processing. Fig. 8 shows the condensation and the original information together with the frame rate we can reach (more than 25 fps with a resolution of 512x512).


DrivSco Webcam Recorder

DRIVSCO
Learning to Emulate Perception-Action Cycles in a Driving School Scenario

Calibration | Processing | Parameters

Motion Threshold 0.00
Energy Threshold 0.00
Stereo Threshold 0.00
Resolution 512 x 512
Frame Rate 25.14 fps
Bandwidth 450 Mbps
Condensated Bandwidth 197 Mbps

Cam Prop
Reset
Processing
Exit

Fig. 9. The disparity results (original hardware-generated ones vs. condensated representation). The parameter tab of the application shows the bandwidth for the original scheme without condensation (450 Mbps) and the bandwidth using the condensation modules, in this case is about 197 Mbps. It is important to take into account that the local features in this case, are not condensation feasible features therefore the new bandwidths are 300 Mbps in the first case (without condensation) and about 47 Mbps in the second case (condensated maps), corresponding to about 84% of data bandwidth reduction.

Finally, in the parameter tab of the main dialog window of the application we can see two data bandwidth boxes. The first one shows the bandwidth of the application taking into account the total volume of data of the read. Summarizing, for local features we have 3 x ImSize (energy, orientation and phase of 8 bits); for disparity 2 x ImSize (data of 16 bits); for optical flow we have 4 x ImSize (velocity of x and velocity of y, data of 16 bits).

DataBandwidth = (3 + 2 + 4) x ImSize x FrameRate x 8 = 450 Mbps

The condensation cannot be applied to the local features. The bandwidth subtracting the data from the local features is:

DataBandwidth = (2 + 4) x ImSize x FrameRate x 8 = 300 Mbps

The second one is the used bandwidth with the condensation. In this case, for local features we have 3 x ImSize (energy, orientation and phase); for disparity 2 x 1/gridSize x ImSize + (4 x RPdata) (data of 16 bits); for optical flow we have 2 x (2 x 1/gridSize x ImSize + (4 x RPdata)) (velocity of x and velocity of y, data of 16 bits).

DataBandwidth = ((3 + 6/gridSize) x ImSize + 12 x RPdata) x FrameRate x 8

As in the previous case, removing the local features result from the data bandwith calculation we obtain:

DataBandwidth = (6/gridSize x ImSize + 12 x RPdata) x FrameRate x 8

The gridSize depends on the parameters and the volume of RP depends on the energy; therefore this bandwidth is dynamic. In our experiments we use a gridSize of 5x5. The calculation of this bandwidth is shown in the Fig. 9. As we can see, the condensation ratio with is about 0.167.

4. Summary

In this deliverable we describe briefly the demonstrator of the vision on chip. We include how different vision modalities can be configured on the same chip (all the ones developed in WP1). Furthermore we also include the condensation module developed in WP3. We illustrate how this condensation module can be used to effectively reduce the communication bandwidth (condensation ratio in terms of bandwidth is 0.167 in the illustrative example of Fig. 9).
Furthermore, we illustrate how the condensation module can embed also attention functionalities since it can receive areas or points of interest from other modules (marked as Relevant points). In the demo we have used this for preserving the original representation maps at these relevant points but this could also be used for fusing reliable features (coming from higher processing stages) with low-level extracted features. We are investigating this mechanism in the framework of a joint paper between UGR and SDU.

Appendix I: Open rtvision. User's documentation

A.1. DRIVSCO Platform

The software presented in this document has been developed by the group of the University of Granada, in the framework of the EU Project DRIVSCO (***Learning to emulate perception-action cycles in a driving school scenario***).

We have implemented a hardware/software platform to work as an interface between on-line cameras and FPGA boards. It provides the input images and shows the results of the different hardware processing engines. The whole system consists of a co-processing FPGA board and a host computer connected through the PCI Express interface.

The application has been implemented using the IDE Microsoft Visual Studio .NET and the OpenCV and the Intel IPP library. The input datastream to the software can be a pair of cameras or two sequences from stored files. The application captures two frames (one of each camera) and stores them into the memory of the co-processing board. The image data are processed by the device which implements the local features, the disparity or the optical flow estimation processing and writes the results in its memory. Then, the application loads these results from the board memory and post-processes them for a proper visualization.

The application also saves in hard disk the results, either from the FPGA memory or from the output of the module that implements the post-processing for the visualization (e.g. the colormap application, noise reduction...). This result saving capability facilitates the benchmarking of different hardware processing engines using widely used benchmark sequences or images as input streams. This saving capability is also useful for comparing results of co-processing system vs computer processing (software implementations). Furthermore, the system generates information about the frame rate of the featured processing. This can be used to evaluate the real-time processing capability, the on-chip computing performance and data-throughput stability.

The system allows us to configure all the parameters related with the processing or even, with the input data: real resolution, interpolated resolution, frame rate of the input datastream, colormap of the input, thresholds for the different processing, hardware latency, etc.

A.2. Software requirements

The IDE used for the development of the application was Visual Studio .Net 2003 and 2005, therefore the configuration requirements are referred to this tool although general configuration settings are given.

We consider that the Xirca driver has already been installed in the system and the webcams drivers too. Xirca (www.sevensols.com) is the FPGA prototyping platform used in this project. We also need the installation of the OpenCV, OpenMP and IPP libraries.

The OpenMP API supports multi-platform shared-memory parallel programming. It can be enabled directly from Visual Studio .Net 2005 and subsequent Visual Studio .Net IDEs.

The Intel(R) Integrated Performance Primitives (Intel(R) IPP) is a library of multi-core-ready and optimized software functions.

Using the software with the Pulnix Cameras, it is necessary the installation of the DALSA Coreco frame-grabber installation and the Pulnix and DALSA Coreco programs too.

The following DALSA Coreco programs are especially important: CamExpert (for testing or checking the different options, camera configuration settings, camera properties, supported cameras ...) and the Firmware Update tool of X64-CL Express Device Driver.

Accupixel Camera is a Pulnix application very useful for setting the camera properties.

The FPGA device is configured using the JTAG protocol or directly through its PLX. If we use the JTAG configuration method it is necessary the installation of the XILINX ISE software, specially the iMPACT software.

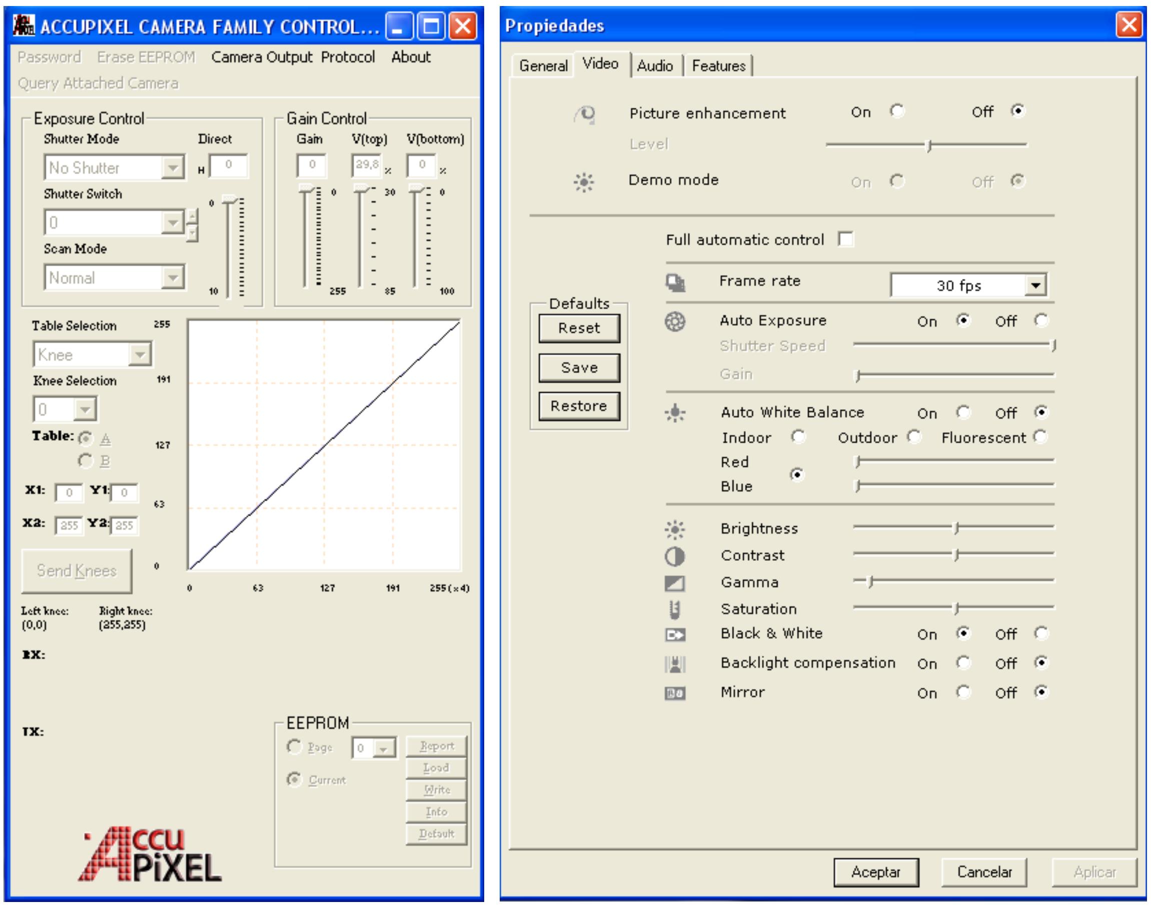
The installation of the software only consists of an installer file.

All the tests for the application taking into account the processing requirements were featured in the following computer:

The processor is an Intel(R) Core(TM) 2 Quad CPU Q6600 at 2.40GHz, with 4096 MB of DDR2 RAM memory at 333 MHz. The OS is Microsoft Windows XP Home Edition Service Pack 2.

A.3. Camera configuration parameters

The camera properties are configured either using the ini file or the program installed with the camera toolkit package for this target. The parameters configured with the last option depend on the specific cameras. Some examples of these programs are shown in the following screen captures (Fig. A.1)



Exposure Control
Shutter Mode: No Shutter
Direct: H 0
Shutter Switch: 0
Scan Mode: Normal

Gain Control
Gain: 0
V(top): 29,8 %
V(bottom): 0 %

Table Selection: Knee
Knee Selection: 0
Table: [✓ A] B
X1: 0 Y1: 0
X2: 255 Y2: 255
Left knee: (0,0) Right knee: (255,255)
RX:
TX:

EEPROM
Page 0 [✓ Current]

Picture enhancement: On [✓ Off]
Level
Demo mode: On [✓ Off]
Full automatic control ☐
Frame rate: 30 fps
Auto Exposure: [✓ On] Off
Shutter Speed
Gain
Auto White Balance: On [✓ Off]
Indoor Outdoor Fluorescent
Red
Blue
Brightness
Contrast
Gamma
Saturation
Black & White: [✓ On] Off
Backlight compensation: On [✓ Off]
Mirror: On [✓ Off]

Fig. A.1. Camera property configuration applications. On the left it is shown the program for the configuration of the Pulnix Cameras, with properties as the exposure control, the gain control, etc. On the right, there is an example of a configuration program for webcams, with parameters such as the frame rate, the exposure time, white balance, brightness, etc.

A.3.1. Webcams application

We manage a configuration file to control the parameters of the data stream (frame flow) that our application receives. The file is located in the installation folder of the application, and it is called **XircaV4config.ini**. The list of parameters is detailed in the next subsection.

A.3.1.1. List of parameters

GLOBAL PARAMETERS

- *NCAMERAS*: Number of cameras for the system
- *WIDTH* and *HEIGHT*: Size in pixels (must be a valid resolution for the webcam model)
- *WIDTH_R* and *HEIGHT_R*: Real size for the resolution to work with
- *FPS*: Frames per second (must be a valid frame rate for the webcam model)
- *METHOD*: Method for the camera linking interface. It can be:
    - *CVCAM*: 1
    - *CVCAM_RESIZE*: 2
    - *CVCAP*: 3
- *WINDOW_W* and *WINDOW_H*: Real size for the camera windows in the application (if *WINWIDTH* does not match with *WIDTH* or *WINHEIGHT* does not match with the *HEIGHT*, then application needs to interpolate)
- *VERBOSE*: Display all the options to configure cameras: 1 for TRUE and 0 for FALSE
- *COLORMAP*: Colormap for the camera image visualization: 1 for GRAY256 and 2 for RGB24
- *SAVING_RESULTS*: Format for the saved results: 0 if we want to save just the results from the device, 1 if we want to save the results post-processed for visualization and 2 if we need both of them

FPGA PROCESSING PARAMETERS

Local Features:

- *STEREO_LOCALFEATURES*: Number of image inputs for the local features processing
- *ETHRESHOLD_LOCALFEATURES*: Threshold for local features processing
- *LATENCY_LOCALFEATURES*: Latency for the hardware of local features processing

Disparity:

- *STEREO_DISPARITY*: Number of image inputs for disparity processing (always 2)
- *ETHRESHOLD_DISPARITY*: Threshold for the disparity processing
- *LATENCY_DISPARITY*: Latency for the hardware disparity processing
- *NSCALES_DISPARITY*: Number of scales for multiscale disparity processing

Motion:

- *DIV_THRESHOLD_MOTION*: Division threshold for motion processing
- *ETHRESHOLD_MOTION*: Threshold for motion processing
- *LATENCY_MOTION*: Latency for the hardware motion processing
- *NSCALES_MOTION*: Number of scales for multiscale motion processing

- *MOTION_VALUE_THRESHOLD*: Threshold value for motion post processing

A.3.2. Pulnix Camera application

With the Pulnix Cameras, we use other configuration files. We also manage a configuration file to control the parameters of the frame flow received by our application. This file is located in the installation folder of the application, and it is called **XircaV4SaperaParams.ini**. The list of parameters is the same presented in the previous section.

Furthermore, we use a configuration file for the application referred to the parameters of the ccf files of the camera. You can check the structure or find more information about these files in the Pulnix and Dalsa Coreco user's manuals. This file is located in the installation folder and it is called **DrivscoRecorder.ini**

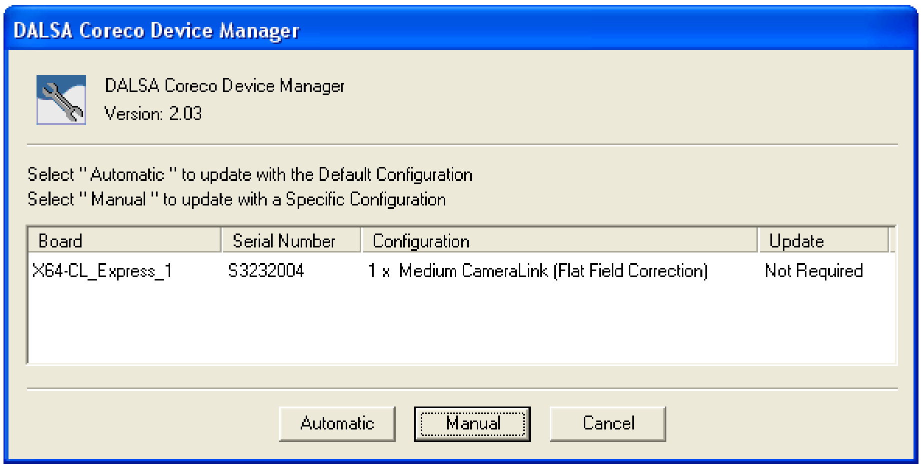
We have another file, **TM-1400.ini** which is the file with the configuration parameters for the Accupixel application. The details of its structure can be found in the Pulnix user's manuals.

In the installation folder of the Pulnix Camera Platform it can be found two ccf files. These are the configuration files for the Pulnix Cameras, establishing a Master-Slave mode between the two cameras. This is necessary for the synchronization between the two cameras, i.e.it is very important that the right and the left camera have to capture the scene at the same time (above all in optical flow processing). More information about this files and their structure is located in the Pulnix and Sapera user's manuals.

Sometimes, the application is launched and Pulnix cameras or Sapera software are not instanced. In this case, we solve this problem using the Dalsa Coreco ***Firmware Update*** tool.

With the following steps the firmware is reset and the camera acquisition can begin again.

1. Select a mode for the update of the frame grabber, using the Device Manager. Select 'Manual' to update the device with a specific configuration (Fig. A.2).



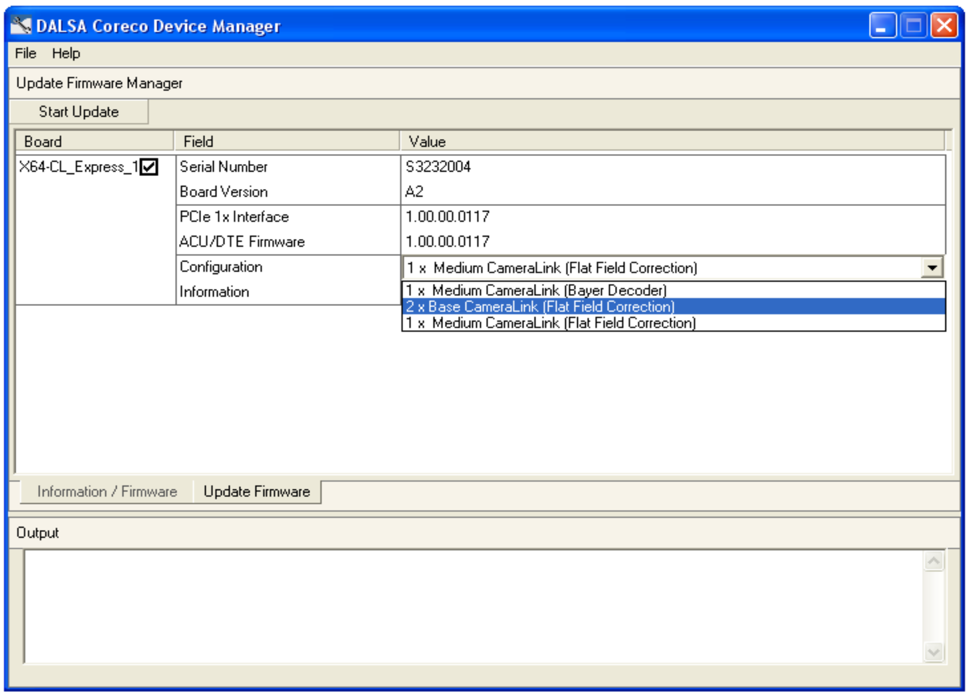
DALSA Coreco Device Manager
Version: 2.03

Select " Automatic " to update with the Default Configuration
Select " Manual " to update with a Specific Configuration

Board	Serial Number	Configuration	Update
X64-CL_Express_1	S3232004	1 x Medium CameraLink (Flat Field Correction)	Not Required

Fig. A.2. Dalsa Coreco Device Manager, initial dialog.

2. The next step is the selection of a configuration, in our case we have a stereo system therefore the most appropriate option is “*2x Base CameraLink (Flat Field Correction)*”. The Pulnix Cameras are connected with the frame grabber using the CameraLink protocol.

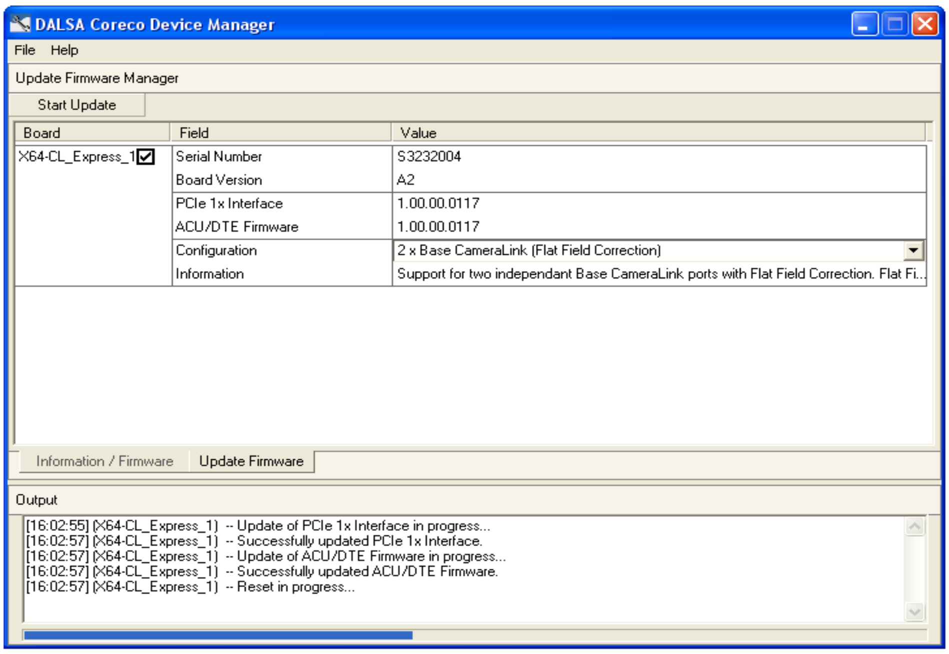


Board	Field	Value
X64-CL_Express_1 ☑	Serial Number	S3232004
	Board Version	A2
	PCIe 1x Interface	1.00.00.0117
	ACU/DTE Firmware	1.00.00.0117
	Configuration	1 x Medium CameraLink (Flat Field Correction)
	Information	

```
1 x Medium CameraLink (Bayer Decoder)
2 x Base CameraLink (Flat Field Correction)
1 x Medium CameraLink (Flat Field Correction)
```

Fig. A.3. Dalsa Coreco Device Manager: Configuration selection

3. Finally, click on “Start Update” and the device is updated and reset. Then the systems will be ready for the image acquisition.



Board	Field	Value
X64-CL_Express_1 ☑	Serial Number	S3232004
	Board Version	A2
	PCIe 1x Interface	1.00.00.0117
	ACU/DTE Firmware	1.00.00.0117
	Configuration	2 x Base CameraLink (Flat Field Correction)
	Information	Support for two independant Base CameraLink ports with Flat Field Correction. Flat Fi...

```
[16:02:55] [X64-CL_Express_1] -- Update of PCIe 1x Interface in progress...
[16:02:57] [X64-CL_Express_1] -- Successfully updated PCIe 1x Interface.
[16:02:57] [X64-CL_Express_1] -- Update of ACU/DTE Firmware in progress...
[16:02:57] [X64-CL_Express_1] -- Successfully updated ACU/DTE Firmware.
[16:02:57] [X64-CL_Express_1] -- Reset in progress...
```

Fig. A.4. DALSA Coreco Device Manager: Updating and reseting the device

A.4. Application setup

The setup package consists only of an installer file. The software is licensed with a GNU LGPL and therefore we provide the application with the overall source code. The installation process is:

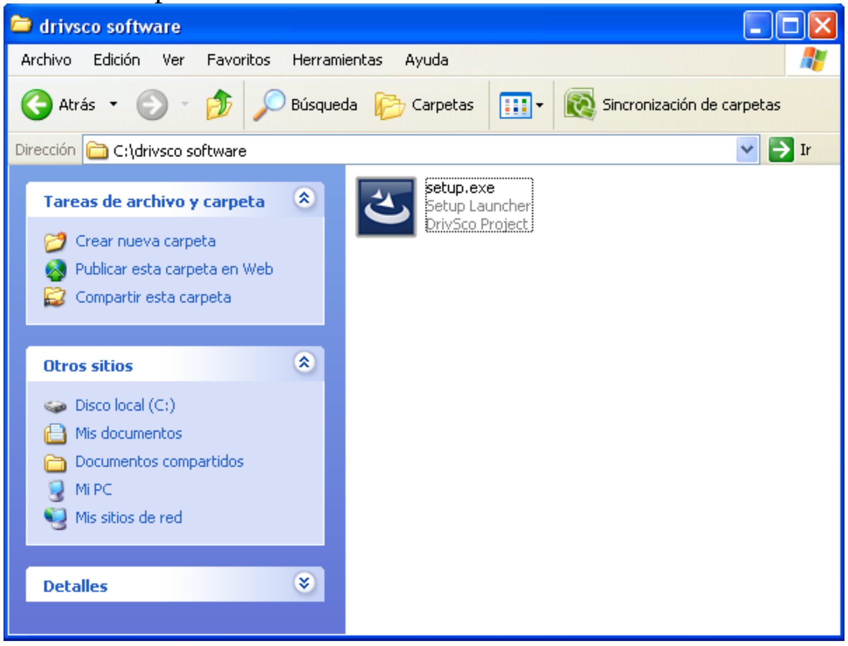
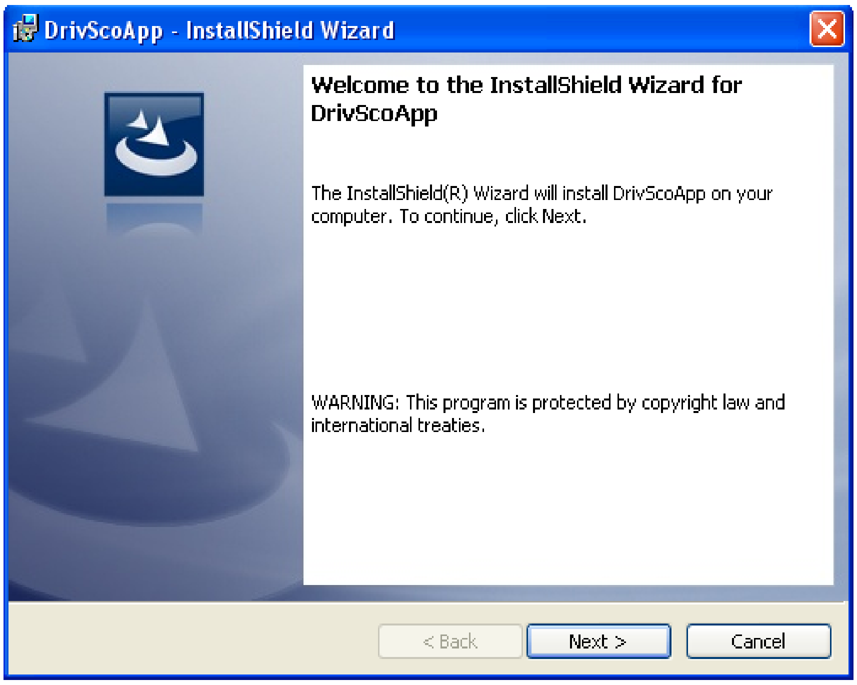


Fig. A.5. DrivSco software installer

1. Download the application and double-click on the setup icon.



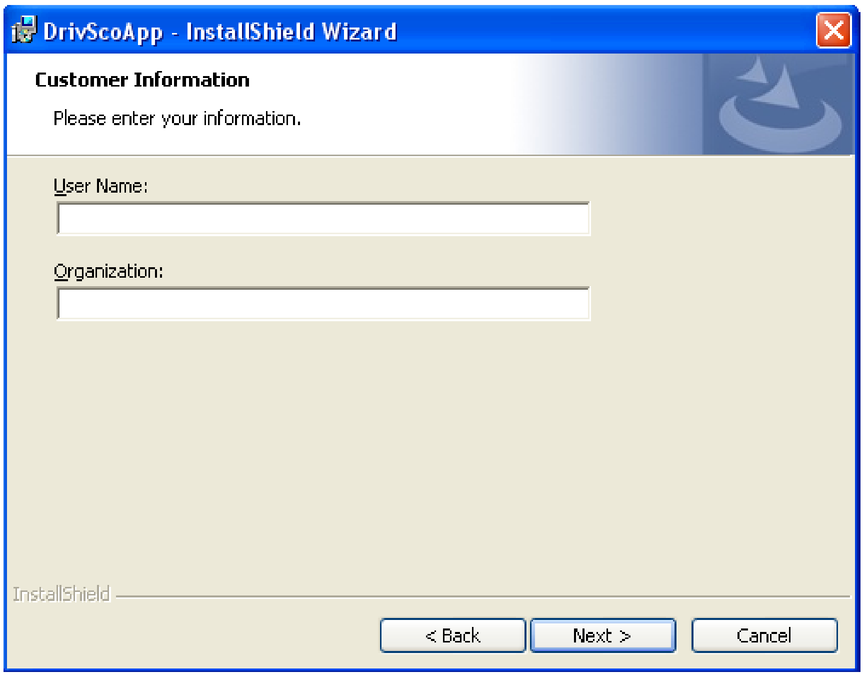
Welcome to the InstallShield Wizard for DrivScoApp

The InstallShield(R) Wizard will install DrivScoApp on your computer. To continue, click Next.

WARNING: This program is protected by copyright law and international treaties.

Fig. A.6. Welcome page of the installation process

2. Click on ‘Next’ and accept the license agreement (GNU LESSER GENERAL PUBLIC LICENSE)


DrivScoApp - InstallShield Wizard

Customer Information

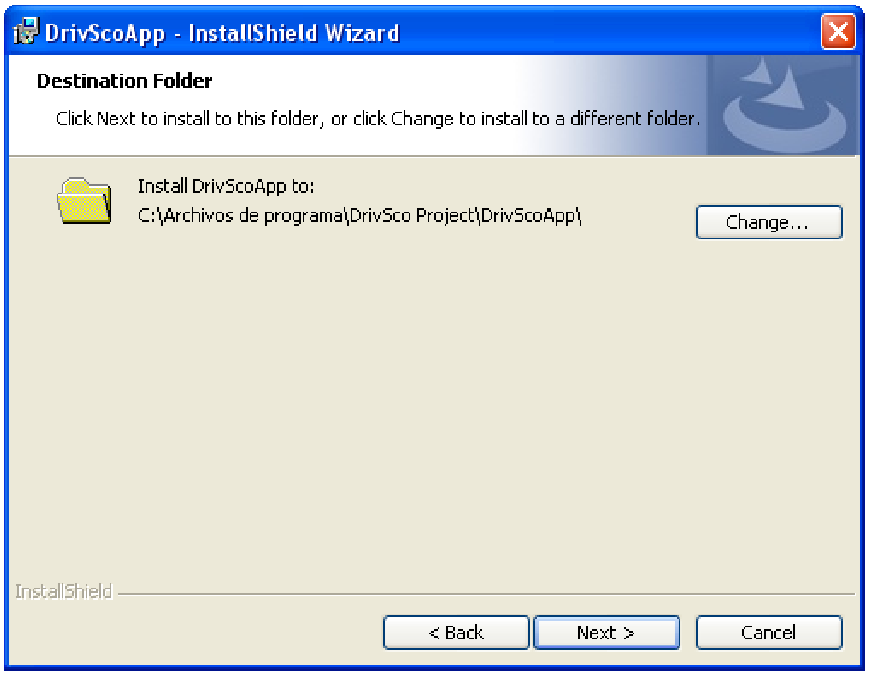
Please enter your information.

User Name:

Organization:

Fig. A.7. Customer information form

3. Click on 'Next' and complete the customer information form. Click on 'Next'.


DrivScoApp - InstallShield Wizard

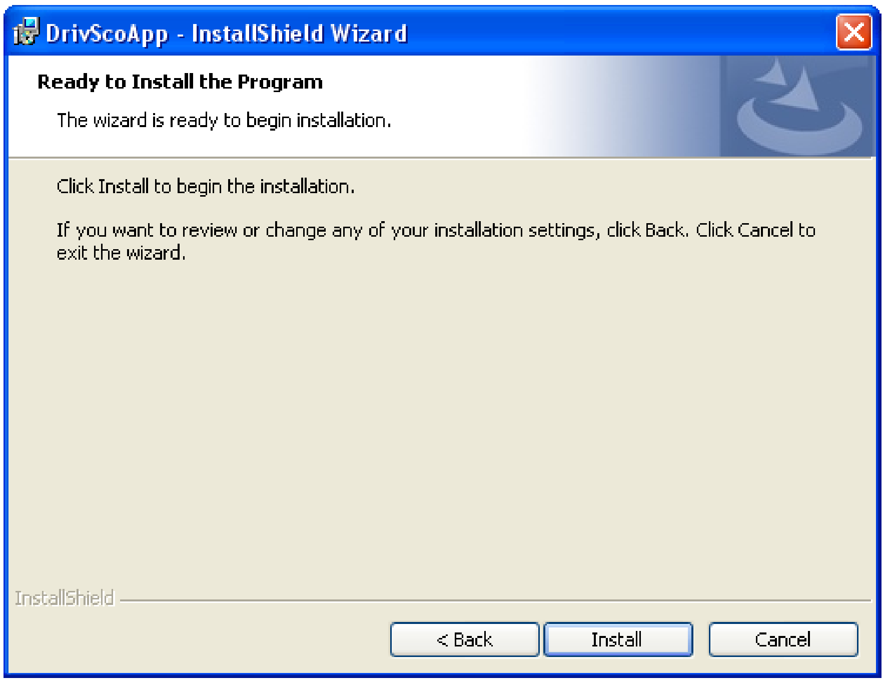
Destination Folder

Click Next to install to this folder, or click Change to install to a different folder.

Install DrivScoApp to:
C:\Archivos de programa\DrivSco Project\DrivScoApp\

Fig. A.8. DrivSco platform destination folder

4. Choose the destination folder and click on ‘Next’. Then, install the application and click on ‘Finish’.


DrivScoApp - InstallShield Wizard

Ready to Install the Program

The wizard is ready to begin installation.

Click Install to begin the installation.

If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

Fig. A.9. DrivSco platform ready to begin installation

A.5. Using the DRIVSCO Platform

The following chapter guides the user along the platform, explaining the different options. The first dialog is shown in Fig. A.10. In this dialog we can select the input source for the application. It can be two video files (select FILE option) or the cameras (select CAM option). It also allows us deciding if we want to do the rectification process to the input data or not.

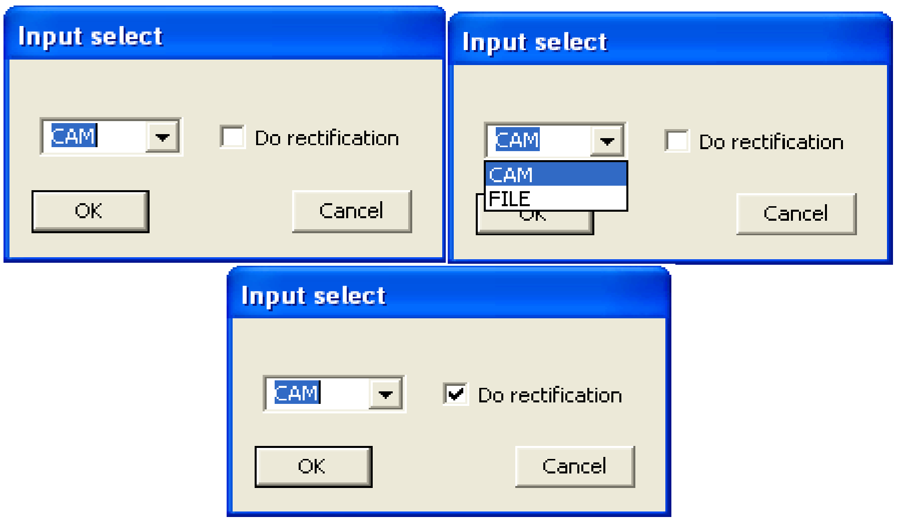


Fig. A.10. Initialization platform dialog. This dialog allows selecting the input source: the cameras or the sequence files. It also allows selecting the rectification process to the input data, as it can be seen in the third window.

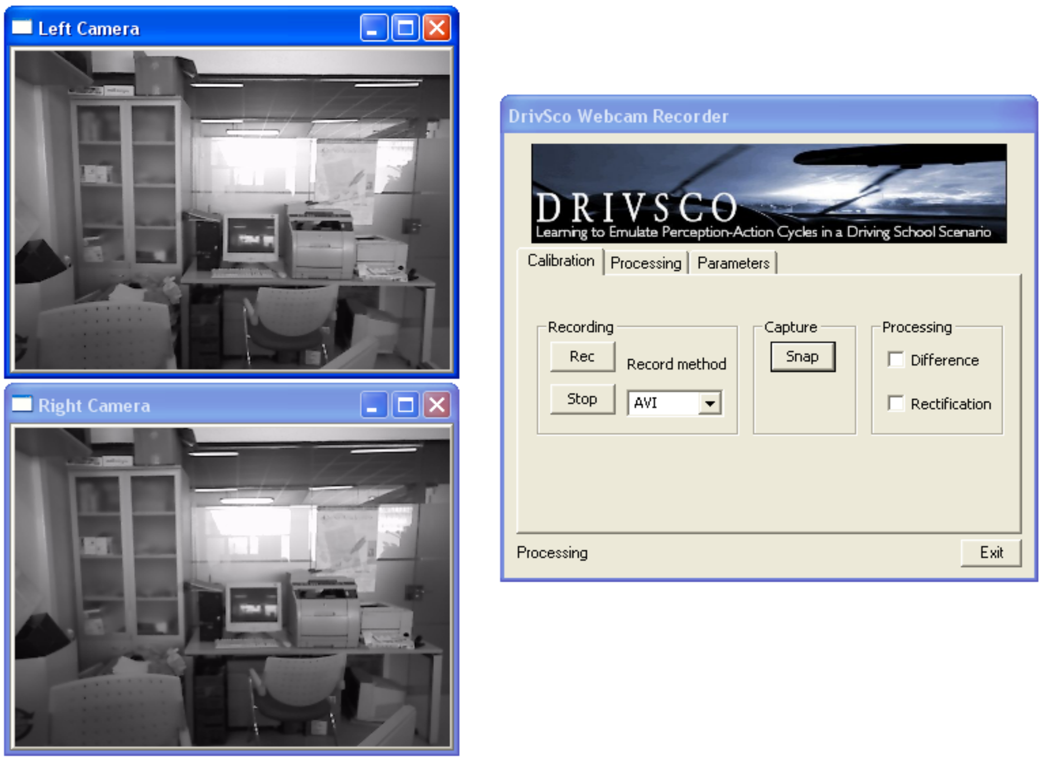


Fig. A.11. Application launched. On the right we see the main program dialog and on the left, the input image for the left and the right camera.

Then, the application is launched and we see the capture of Fig. A.11.

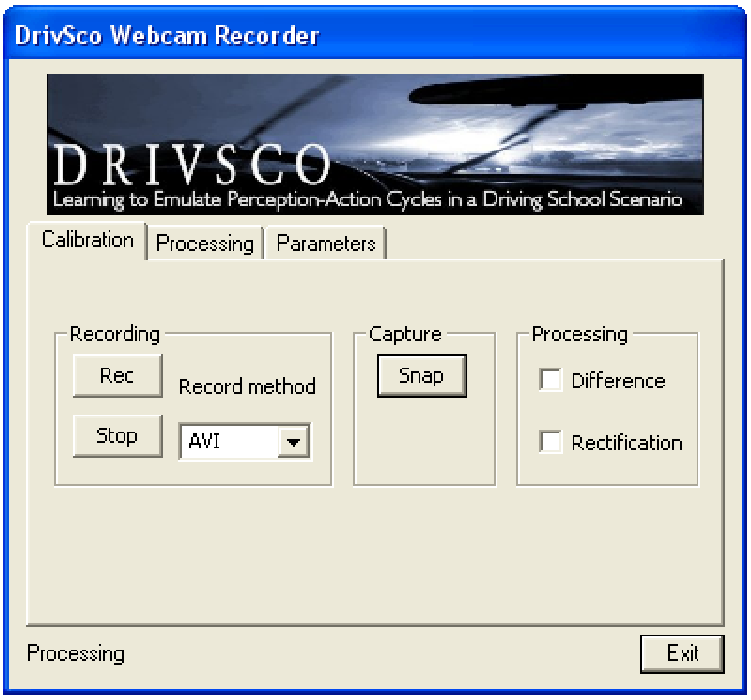


Fig. A.12. Calibration Tab. In this tab the input data can be saved (to an .avi file or as a .pgm sequence of files), or frames can be captured individually for rectification processing and, it can be shown the difference between the left and the right frames or done the rectification for each one of them.

In the main dialog the processing is organized in three different tabs: the calibration tab, the processing tab and the parameters tab.

A.5.1. Rectification and recording tab

The calibration tab is the main one. It can save the input data (from the cameras or from the sequece files selected previously) to a video file (.avi) or to a sequence of image files (.pgm). This could be useful to process sequence files in a controlled scenario (see Fig. A.13).

In this tab is also possible to capture snapshots individually. It can be useful for calibration processing.

The last processing possible here is the difference between the input left and right frames by clicking the ‘Difference’ check box. Rectification can be done only if in the previous dialog the ‘Do rectification’ check box was clicked.

The rectification is very significant for the disparity processing. With our application it can select the LUT files with the values of the calibration for the rectification processing.

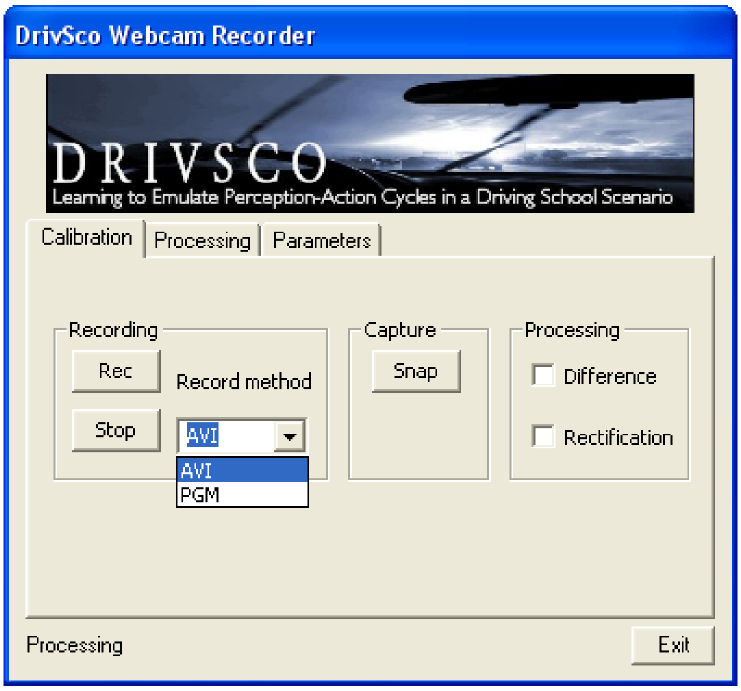


Fig. A.13. The input data can be save to an .avi video file or to a sequence of .pgm files

A.5.2. Interface with device board tab

The processing tab (Fig. A.14) is the interface between the co-processing device board and the host computer connected by the PCI Express interface. The 'Configure' button allows us to store the bitstream file in the board. It is done using the JTAG protocol or by the PLX. If the bitstream will be stored using the JTAG protocol it is necessary the installation of the XILINX ISE software, particularly the iMPACT software. On the other hand, to store the bitstream file throughout the PLX no new software is necessary (see Fig. A.15 and Fig. A.16).

Errors due to the JTAG configuration are shown in the DOS console to fix them. More information about the iMPACT XILINX software is provided in its user's manuals.

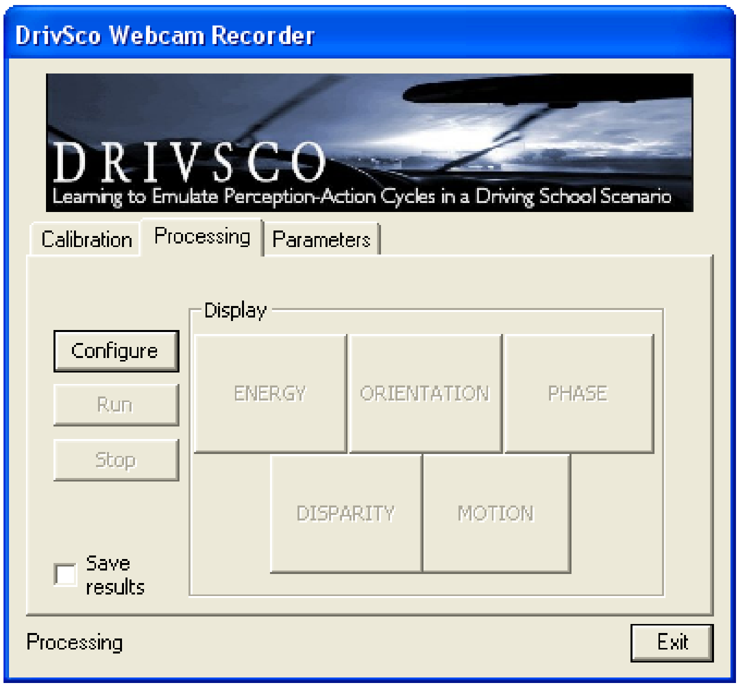
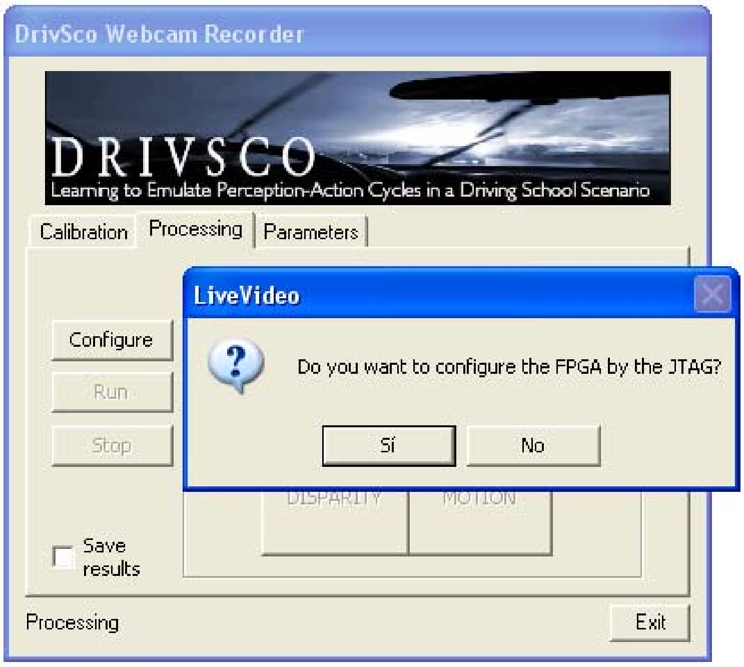
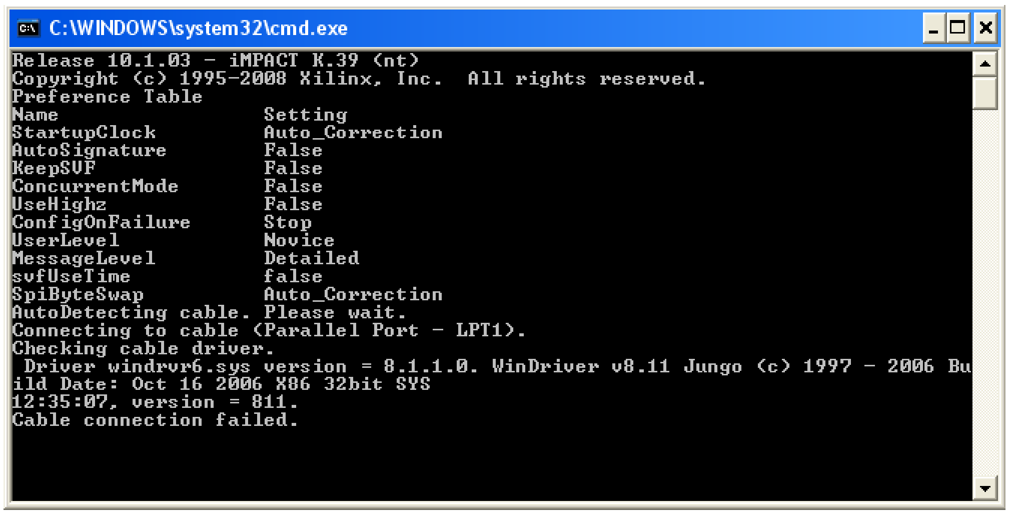


Fig. A.14. Processing Tab. It is the interface between the device board and the host computer. In this tab you select the processing which is going to be shown and stop this processing. The results can be saved.



Do you want to configure the FPGA by the JTAG?

Fig. A.15. Example of use of the application. After clicking 'Configure' button, you can decide if the device is going to be programmed by the JTAG or throughout its PLX.



```
Release 10.1.03 - iMPACT K.39 (nt)
Copyright (c) 1995-2008 Xilinx, Inc.  All rights reserved.
Preference Table
Name                    Setting
StartupClock            Auto_Correction
AutoSignature           False
KeepSVF                 False
ConcurrentMode          False
UseHighz                False
ConfigOnFailure         Stop
UserLevel               Novice
MessageLevel            Detailed
svfUseTime              false
SpiByteSwap             Auto_Correction
AutoDetecting cable. Please wait.
Connecting to cable (Parallel Port - LPT1).
Checking cable driver.
 Driver windrvr6.sys version = 8.1.1.0. WinDriver v8.11 Jungo (c) 1997 - 2006 Bu
ild Date: Oct 16 2006 X86 32bit SYS
12:35:07, version = 811.
Cable connection failed.
```

Fig. A.16. Capture of the application calling to the iMPACT XILINX program for programming the device board by the JTAG. The application is looking automatically for a cable connection.

Once the bitstream file is loaded, you can select the processing that will be shown and then click on ‘Run’. To stop the processing click on ‘Stop’.

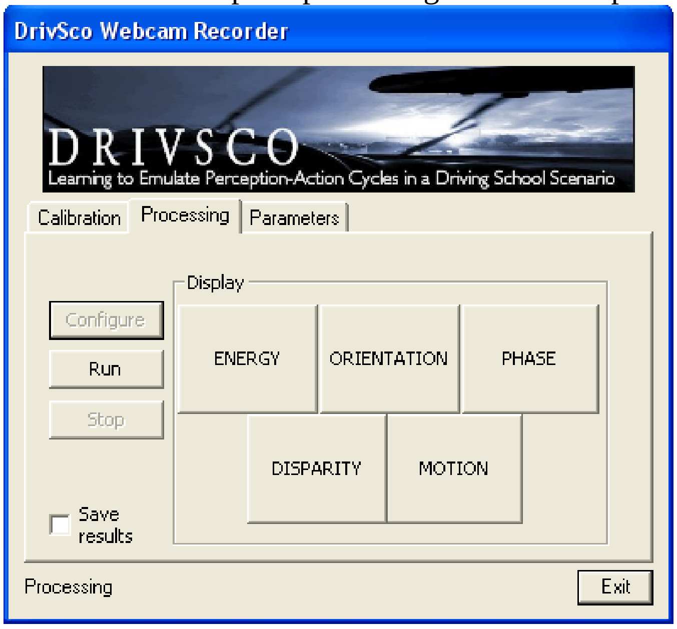


Fig. A.17. State of the application after the configuration of the device board

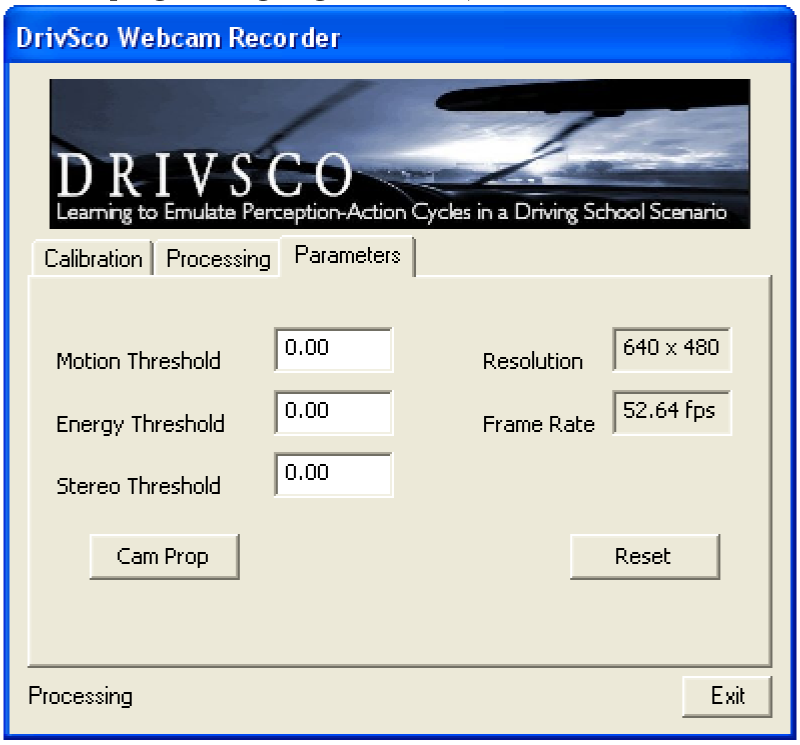
Furthermore, you can save the processing results clicking the ‘Save results’ checkbox. According to the parameters configuration of the .ini file, you will save the results from the device board or save the post-processed results for visualization.

A.5.3. Parameters tab

The parameters tab shows (Fig. A.18) data about the processing like the resolution used for the processing (configured by the ini files) or the frame rate reached (in frames per seconds). The ‘Reset’ button allows us to begin the frame rate processing again.

The editable tags are threshold for future releases but they are not currently in use.

The ‘Cam Prop’ button calls the Camera Configuration Program set up automatically (in the case of the webcams) or by the user (in the case of Pulnix Cameras he decides the program is going to execute).


DrivSco Webcam Recorder

Motion Threshold 0.00

Energy Threshold 0.00

Stereo Threshold 0.00

Resolution 640 x 480

Frame Rate 52.64 fps

Processing

DRIVSCO
Learning to Emulate Perception-Action Cycles in a Driving School Scenario
Calibration
Processing
Parameters
Cam Prop
Reset
Exit

Fig. A.18. Parameters tab. It shows information about the processing like the resolution or the frame rate. It also gives us the possibility to call to a Configuration Program for the camera parameters.

A.5.4. Processing examples

The following screen snapshots show the visualization of the different system processing engines.

In the Fig. A.19 we show an example of the disparity of the real-world with the saturation level control available for controlling the colormap application. In Fig. A.20 we show the result of the disparity results using a well-known set of images (Tsukuba images).

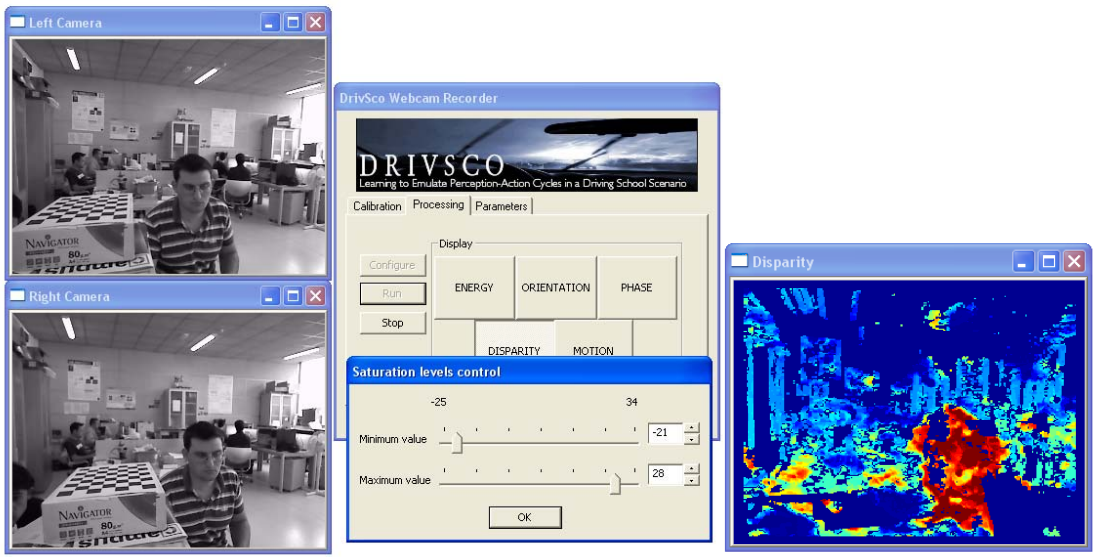


Fig. A.19. Example of disparity processing. The saturation level control is available for change the colormap application. The colormap in the case of the disparity shows nearest objects with hot colors and furthest ones with cold colors (the name of the colormap is jet).

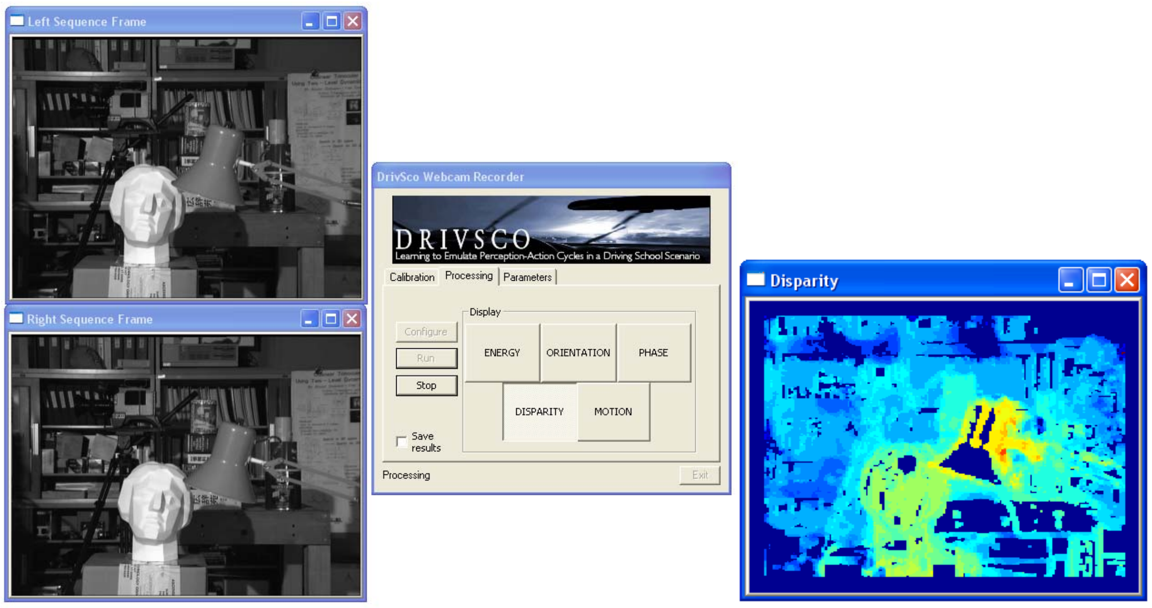


Fig. A.20. Example of disparity processing using a well-known image, the Tsukuba images.

In Fig. A.21 we show an example of the local features computation using the left camera as the input source. In the different windows we show the Energy, the Orientation and the Phase results.

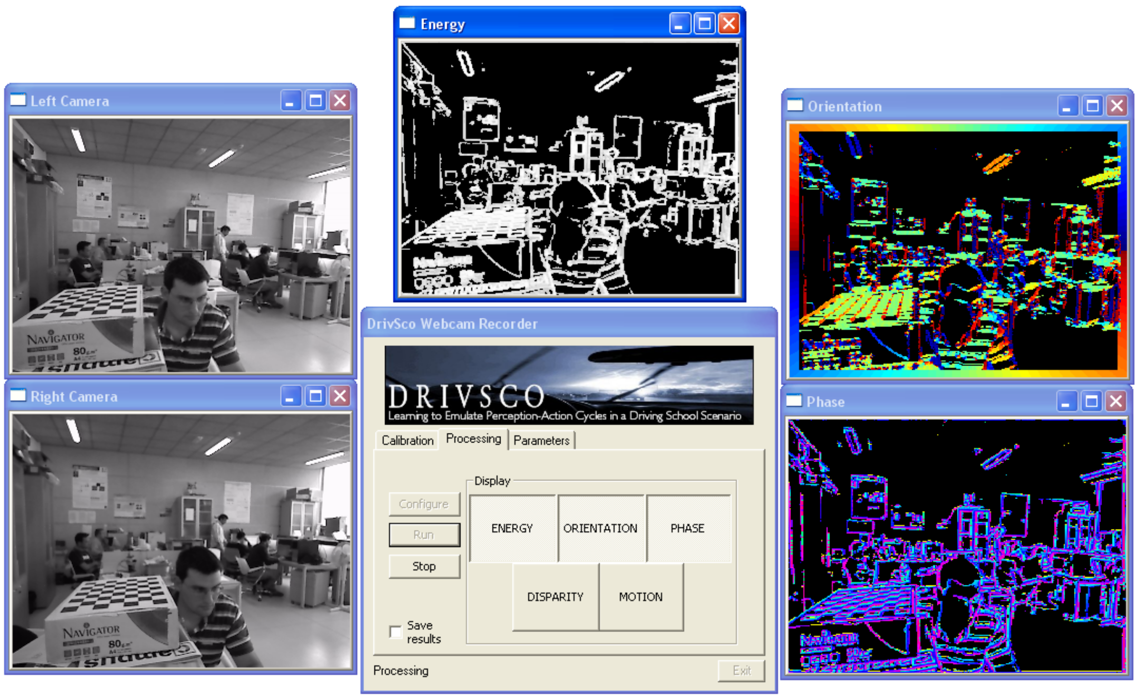


Fig. A.21. Example of use of the application showing the local features computation results. The windows show the Energy, the Orientation and the Phase results (clockwise order beginning with the Energy window, above the main application window).

Finally, Fig. A.22 shows the results of the optical-flow computation. The optical-flow is computed for the Left camera frames and the man in the scene is standing up.

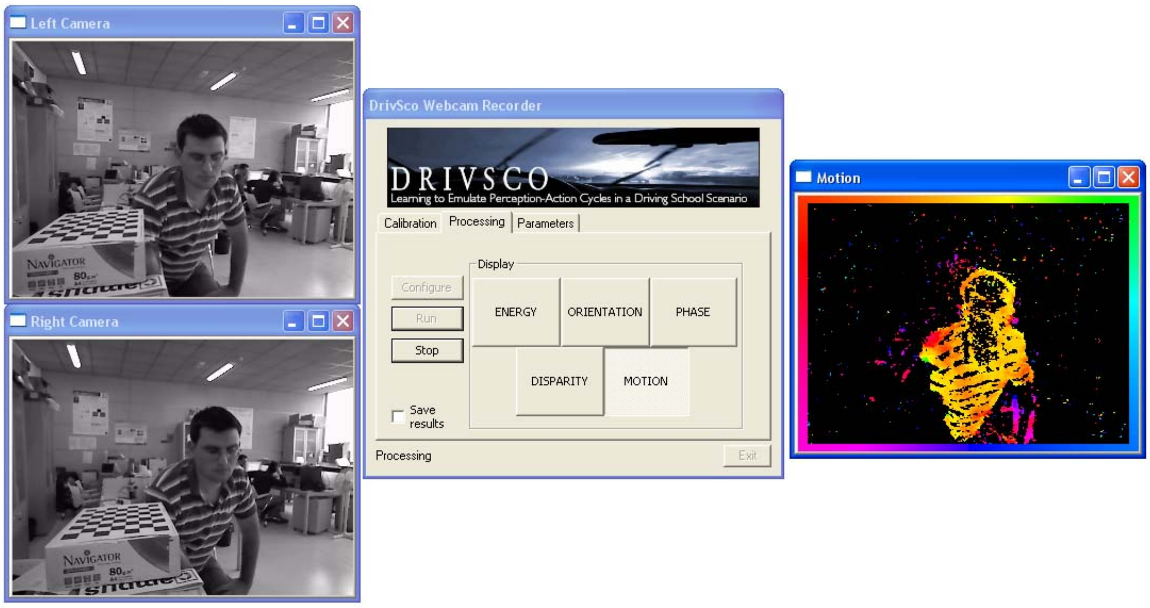


Fig. A.22. Example of use of the application showing the optical-flow computation results. The input data is the frame from the Left Camera.