DPM Demo Kit

User's Manual

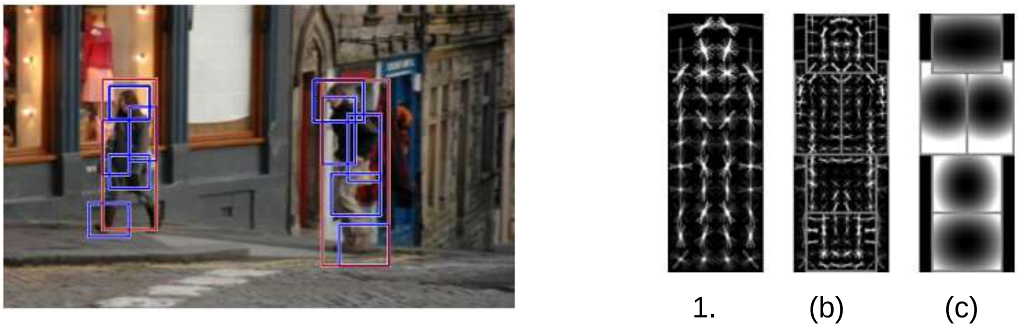
Version: 1.0.1

dpm_dk_um_1_0_1.doc

1 INTRODUCTION

Deformable Parts Models (DPM), first proposed by Felzenszwalb et al. (see REF[1]), identify objects in images using a star-structured part-based model, defined by a root filter plus a set of parts filters and associated deformation models.
The score of a star model at a particular position and scale within an image is the score of the root filter at the given location plus the sum over parts of the maximum, over placements of that part, of the part filter score on its location, minus a deformation cost measuring the deviation of the part from its ideal location relative to the root (see

Figure 1-1).
Both root and part filter scores are defined by the dot product between a filter (a set of weights) and a sub-window of a feature pyramid computed from the input image.



1. (b) (c)

Figure 1-1: on the left, detections obtained with a model. On the right, representation of the DPM model: a coarse root filter (a), several higher resolution filters (b) and a spatial model for the location of each part relative to the root (c). The filters specify weights for histogram of oriented gradients features. Their visualization show the positive weights at different orientations. The visualization of the spatial models reflects the “cost” of placing the center of a part at different locations relative to the root.

Deformable parts models lead to efficient object detectors that achieve state of the art results on the PASCAL and INRIA person datasets. Figure 1-2 shows a few examples of object models.

Figure 1-2: examples of object models. From left, person model, bike and car model.

eVS has implemented a DPM solution around Xilinx® Zynq™-7000 All Programmable System-on-Chip (SoC) that combines an industry standard dual-core ARM® Cortex™-A9 MPCore™ processing system (PS) with Xilinx 28-nm programmable logic (PL) on a single monolithic device.
In the following text we refer to the eVS Deformable Parts Model implementation as 'DPM' and introduce the kit available for its evaluation.

2 INSTALLATION

The DPM demo kit is delivered as an archive that you should deflate into a directory of your choosing. **Please note that it will not work with folders that have spaces or – character in their path.** The demo kit includes reference design files for DPM demo as specified in the following sections**.**
All deliverables are fully compatible with Xilinx development tools – Xilinx Design Suite 14.7 including Xilinx ISE, Xilinx Platform Studio (XPS), System Generator for DSP and the SDK.
Kit deliverables include pre-compiled hardware SoC reference designs prepared for Xilinx Platform Studio (XPS) design suite, and complete software DPM demo application, which can be customized and upgraded to closely fit to specific requirements.

2.1 Hardware deliverables

- Configuration bitstream file for the programmable logic (with time bomb after 30 minutes)

2.2 Software

- 3rd party core standalone (bare-metal) drivers
- DPM core standalone (bare-metal) drivers
- Zynq FSBL sources and the Xilinx SDK project – custom version for standalone applications
- Demo application sources
- 3rd party precompiled utility libraries

2.3 Configuration data

- Configuration files to be located on SD card for system customizing

2.4 Directory structure

Directory		Purpose
`hardware`		
	`drivers`	Standalone (bare-metal) drivers for DPM core and 3rd party cores
	`sw-services`	xyl_oslib 3rd party OS abstraction library for Xilinx Xilkernel embedded kernel – use in standalone (nonOS) applications and other libraries
`releaseProj`		
	`FPGA`	This directory contains all binary FPGA files for DPM demo
	`SD`	This directory contains all initialization files for DPM demo

		system. Copy them to SD card.
	makeBin	Utility script for creating boot.bin file and executables
software		
	apps	Demo application source code
	SDK_workspace	Xilinx SDK workspace folder for building bare-metal applications
	utilities	Matlab script to translate a DPM model in Matlab into a txt file loadable by DPM demo

3 Third Party IP cores

The DPM demo kit reference design integrates some companion cores from the Xylon's logicBRICKS IP core library. These IP cores are optimized for Xilinx FPGA and Zynq-7000 All Programmable SoC were chosen to shorten development time and provide a stable platform to evaluate the DPM.
For a full hardware evaluation of DPM you may want to change and recompiling the reference design. To make this possible, it is necessary to buy an evaluation license for the Xylon's logicBRICKS cores.

3.1 Logicbricks IP cores used in this design

3.1.1 logiCVC-ML Compact Multilayer Video Controller

The logiCVC-ML IP core is an advanced display graphics controller for LCD and CRT displays, which enables an easy video and graphics integration into embedded systems with Xilinx Zynq-7000 All Programmable SoC and FPGAs. This IP core provides flexible display control, it also includes a level of hardware acceleration: alpha blending, panning, buffering of multiple frames, etc.
More info: http://www.logicbricks.com/Products/logiCVC-ML.aspx

3.1.2 logiVIEW Perspective Transformation and Lens Correction Image Processor

The logiVIEW Perspective Transformation and Lens Correction Image Processor IP core for video and imaging applications using single or multiple camera imagers removes fish eye distortions caused by extreme wide-angle field of view (FOV) lenses, makes perspective corrections to the captured video and stitches multiple corrected video inputs in a resulting single image for the LCD display. In the demo reference design logiVIEW is used as scaler, to provide DPM core the pyramid of images for processing.
More info: http://www.logicbricks.com/Products/logiVIEW.aspx

3.1.3 logiWIN Versatile Video Input

The logiWIN IP core accepts a streaming video input, decodes it and converts into the RGB format. The input video can be real-time scaled, de-interlaced, cropped and positioned on the video display. It integrates high-quality anti-aliasing algorithm that guarantees high picture quality without visible artifacts.
More info: http://www.logicbricks.com/Products/logiWIN.aspx

4 DPM reference design

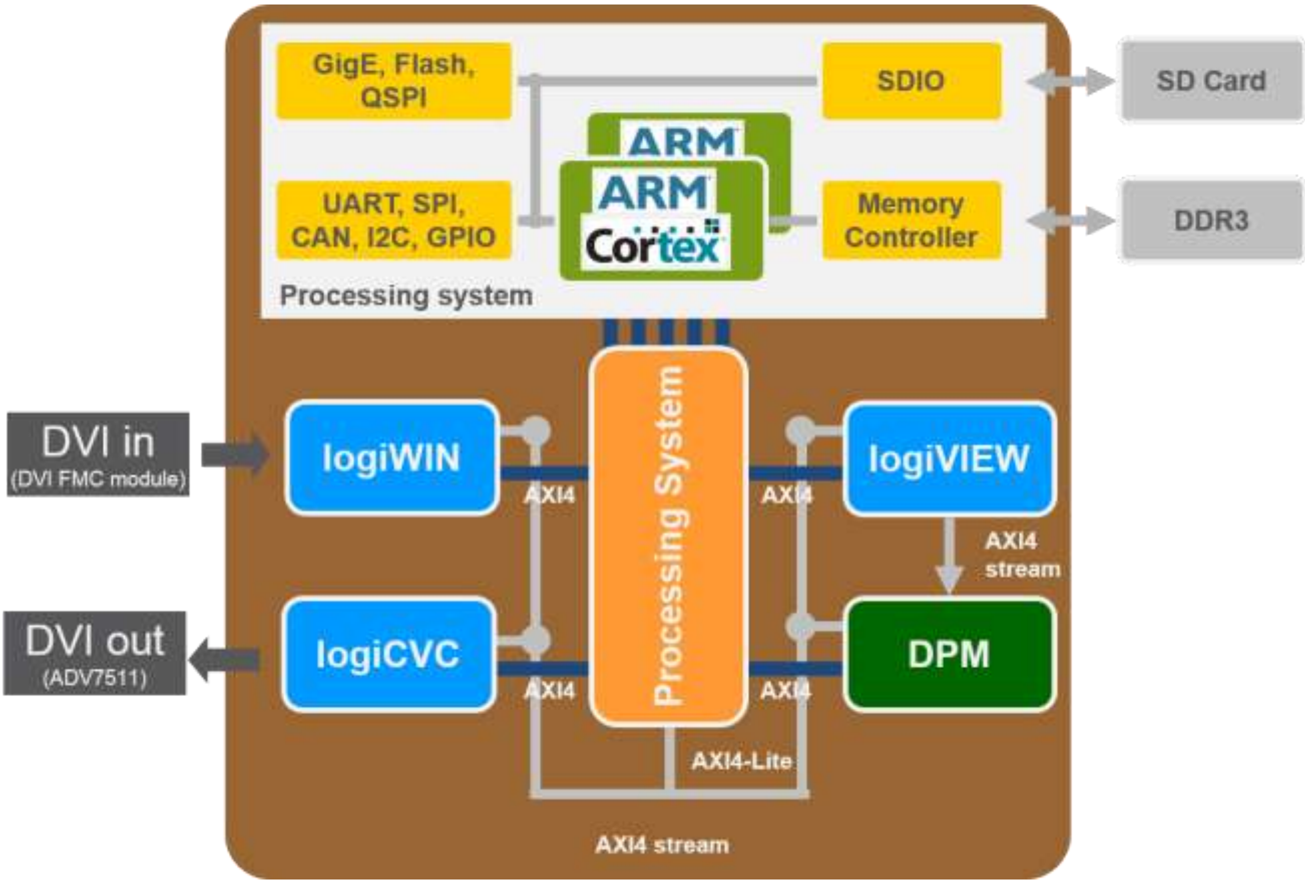


Figure 4-1: DPM reference design. In blue the 3rd party IP cores.

5 QUICK START

The purpose of this chapter is to help users quickly setup DPM demo kit. Please follow the instructions:

- **Assemble and wire the hardware required for DPM demo kit.** Your DPM demo kit is designed for a specific hardware setup, see section 5.1 for the required items.
- **Start-up your DPM demo kit.** Your DPM demo kit requires to some preparation steps to initialize your system, you will need to:
  - Format SD card as FAT32 if not already so
  - Copy the content of directory releaseProj/SP to the root of SD card
  - Plug the SD card on the ZC706
  - Set switches and jumpers for SD boot. See section 5.2
  - Power up the system
- **Customize your design to fit your system requirements.** After you have verified your system runs properly, you need to generate parameters for system customization. See section 5.3. All the parameters that need to be changed are on the SD card on the ZC706 board. You will need to:
  - Set DPM demo configuration parameters
- **Apply newly generated files.** Files are saved to SD card and will be applied on next startup.

5.1 Required hardware

Along with the demo kit content, before you start the initialization process, please make sure you have all the items listed below:

- **For operations** (target board)
  - 1x Xilinx Zynq-7000 SoC ZC706 Development Board with XC7Z045 FFG900-2
  - 1x Avnet® DVI I/O FMC daughter card
  - 1x SD card
  - power supply and cabling

  See Figure 5-1 for the FMC daughter card mounting and relevant ports to use for demoing DPM.

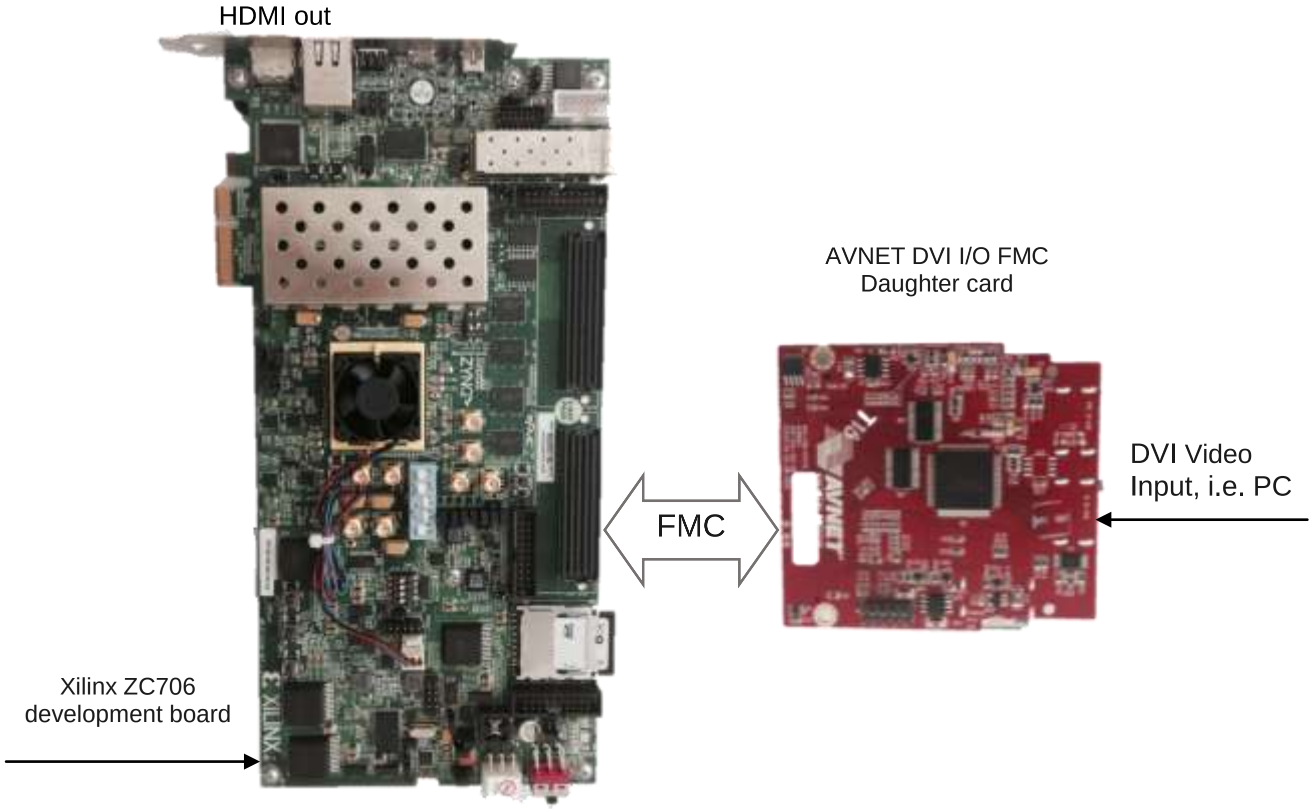


Figure 5-1: DPM demo kit based on Xilinx ZC706 board

- **For initialization** (target board)
  - Display that supports 640x480 resolution (DVI)
  - PC or laptop with HDMI or DVI input
  - Cable for connecting HDMI input on the FMC daughter card with the PC/laptop (UDMI to HDMI cable or HDMI to DVI cable)
  - Cable for connecting HDMI output on the ZC706 with the display (HDMI to DVI cable)

- **For setup**
  - PC or laptop with USB port and USB-UART driver from Silincon Labs installed (Silicon Labs USB-UART drivers)
  - USB Type-A to USB Mini-B cable

5.2 Board setup

The board is initialized with files stored on SD card, copied from releaseProj/SD and specific switch positions detailed in the following:

- Set SW 16 to have the following sequence: 1 – down, 2 – down, 3 – up, 4 – up, 5 – down (with up position closest to the number and down is position farther from the number
- Set SW4 to have the following sequence: 1 – off, 2 – on.

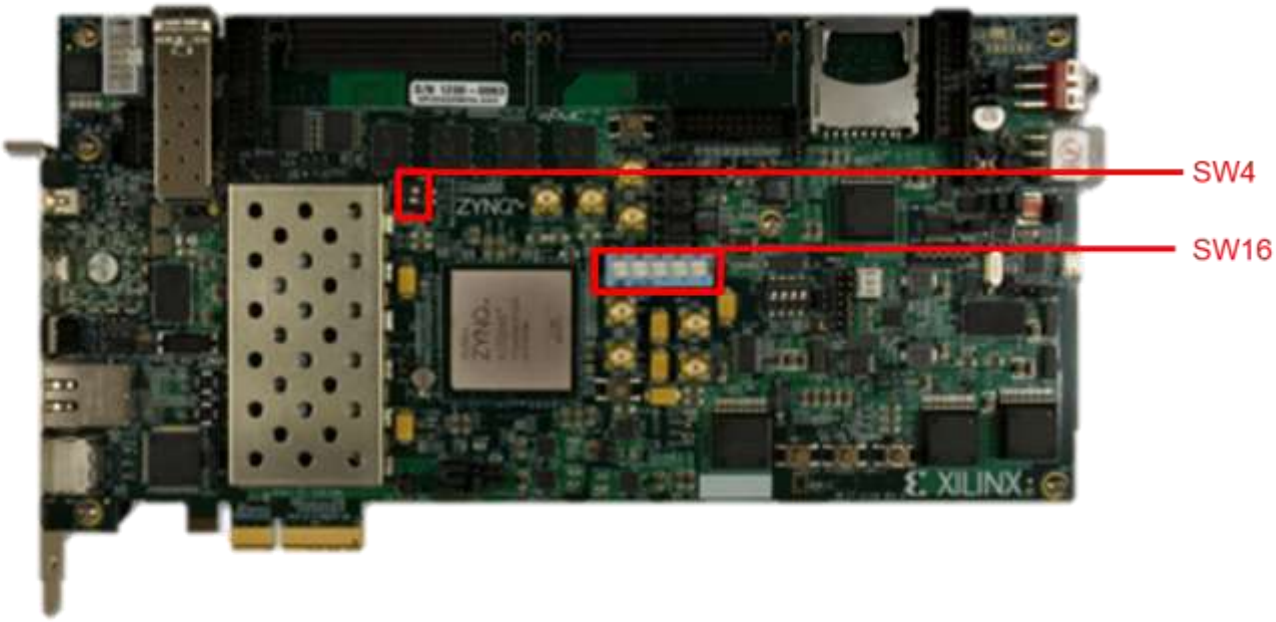


Figure 5-2: ZC706 board: front view

Power-up the board. If programming is done successfully, after start-up there should be an image on the screen. If there is no image on the output display, check for error or success messages on serial interface. Make sure all system parts match the ones from Section 5.1 and that the initialization process is done as instructed above.

5.3 Demo setup

DPM demo parameters are stored in DATA directory:

- **CONFIG.txt** that contains detection and post-processing parameters
- **DPMdata.txt** that contains the DPM model
- **PYRAMID.txt** that contains the scales/ROIs sequence

All files are loaded during software initialization.
CONFIG.txt and PYRAMID.txt can be read and modified though the UART-based DPM Detection System menu (Figure 5-3). This menu can be accessed by pressing any key from the keyboard when the demo is running.
The file DPMdata.txt contains all information about DPM model: components, parts, anchors, filters and so on. It can be generated automatically using the Matlab script in releaseProj/utilities directory **print_model.m**.

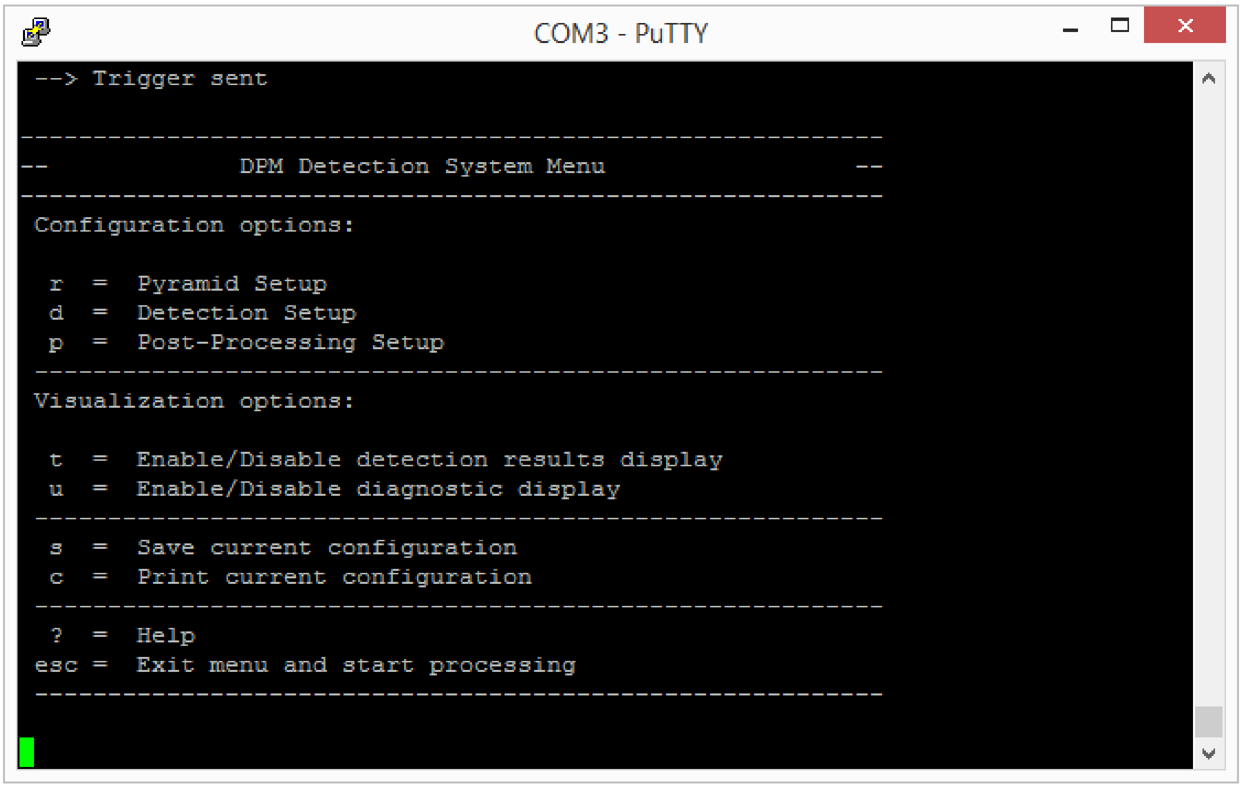


Figure 5-3 Main DPM demo configuration menu

5.3.1 Pyramid setup

If you want to change the pyramid of scales, press ‘r’ on the DPM Detection System menu to go to the Pyramid setup menu, then follow this procedure:

1. Press ‘o’ to set the number of octaves. Please note that this number refers to the part filter scales only. The root filter scale is automatically estimated inside the DPM core. Default value is 3.
2. Press ‘i’ to set the number of intervals. Default value is 10.
3. Press ‘r’ to set a ROI. You can specify the percentage of image to eliminate from processing. This will be applied to all scaled images. Default value is 0%.
4. Press ‘x’ to create the pyramid.
5. If you are satisfied, press ‘s’ to save the new pyramid of scales on file. This operation will overwrite the file PYRAMID.txt in DATA folder in the SD card.

The new or updated file PYRAMID.txt will be loaded at the next boot.

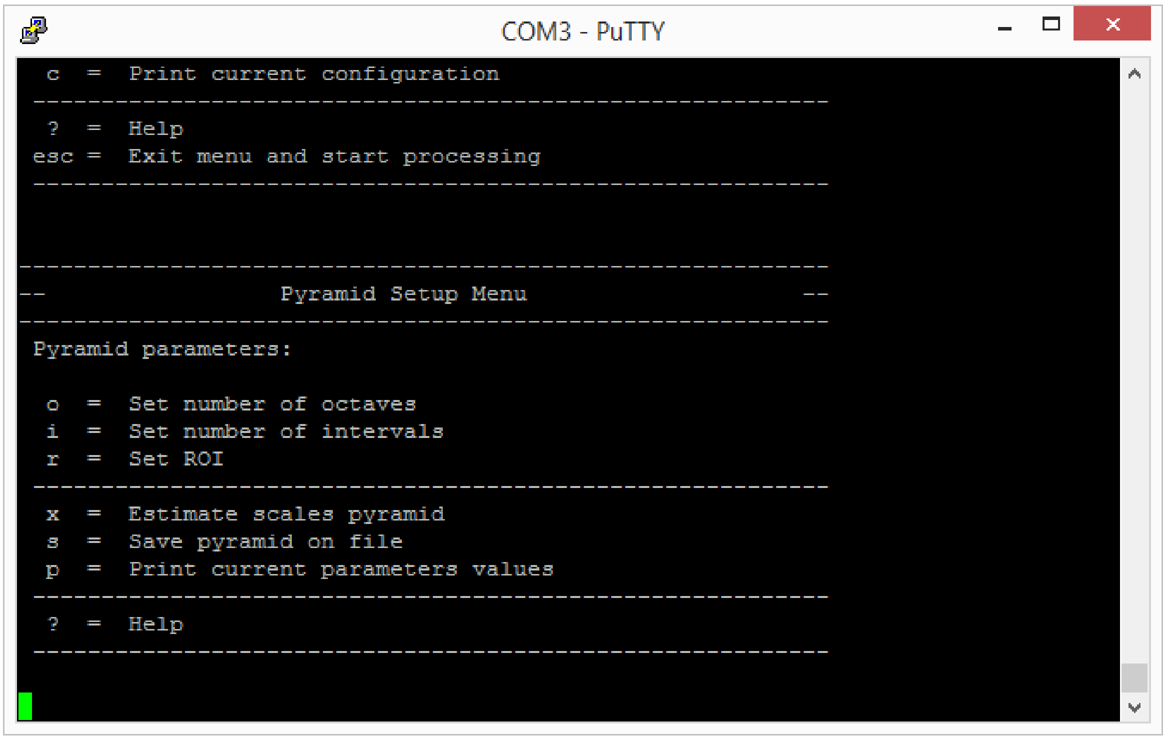


Figure 5-4: Pyramid setup menu

5.3.2 Detection setup

Press ‘d’ to enter Detection Setup menu. This menu is used to set the cutoff parameter, i.e .the threshold that decides positive/not positive. The default value is -0.5.
Press <esc> in the main menu to exit DPM Detection System menu. DPM updates with the new setting.

5.3.3 Post-processing Setup

Press ‘p’ to enter Post-processing Setup menu. This menu contains the parameters for the post-processing phase (Figure 5-5). The purpose of the post-processing is reducing the number of false positives. It groups together the detections corresponding to the same object at different scales based on a similarity measure.
The similarity value represents a threshold on the similarity between two detection windows, based on position and area. This is used by the grouping algorithm to decide whether two windows belong to the same group. A higher value means being less restrictive, i.e. the final groups will be composed by windows that are more distant (in area and position). The default value is 0.3.
The other parameter is the minimum number of elements in a group. A pedestrian is supposed to be detected at multiple scales and neighboring positions. The default value is 2.
Press <esc> in the main menu to exit UART-based DPM Detection System menu. DPM updates with the new settings.

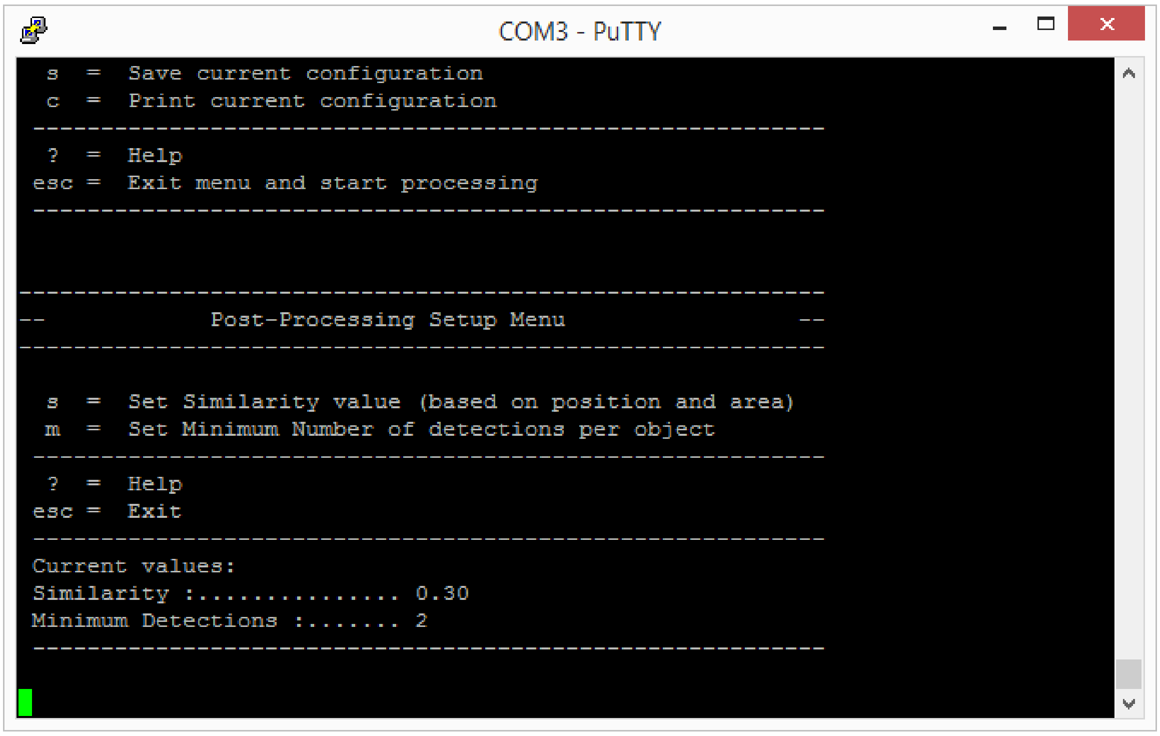


Figure 5-5: Post-processing setup menu

5.3.4 Object detection output

The system output, with both detection and post-processing, is displayed on screen with a box around each object found.
From DPM Detection System menu, you can decide to visualize the results of detection only or the results after post processing by ‘t’.
The number on the bottom left of the display image is the frame rate of DPM core (PL partition only).

6 CUSTIMIZING SOFTWARE

DPM software application is delivered in source code to allow software customizing. In the following text instructions on how to setup environment for software customizing are given.

6.1 Setting up SDK workspace

There are a few steps required to set up SDK workspace:

1. Open SDK, select workspace: software/SDK_workspace
2. In SDK go to: Xilinx Tools → Repositories → New, and add hardware directory
3. In SDK go to Project and exclude Build automatically (optional but recommended)
4. In SDK go to: File → Import → General → Existing projects to workspace
5. In Select a root directory select software/SDK_workspace, select all projects and click Finish
6. In case Build automatically option has been disabled build manually all of the imported applications

6.2 Applying customized files

If the software application is changed, Zynq boot files must be regenerated. Scripts that simplify that process are located in releaseProj/makeBin directory.
This directory contains:

- **bl.elf** – second stage bootloader executable used for loading DPM software applications from the SD card
- **fsbl.elf** – Zynq FSBL
- **copy_files.bat** – batch script used to copy files needed for DPM SD card preparation
- **stitch_images.bat** – batch script to generate DPM bootable binary
- **stitch.bif** – file list for the boot.bin generation (used by bootgen tool)

Using helper scripts and other files described above for creating the Zynq boot files, two files are generated:

- **boot.bin** – Zynq bootable file. It contains the fsbl, DPM FPGA configurations and bootloader
- **dpm.elf** – software application executable

6.2.1 Updating Zynq configuration files process

If any user-built FPGA and/or software application binaries exist within SDK_workspace the script use those files automatically, otherwise default files from releaseProj directory will be used.
Steps for updating configuration files:

1. Run Xilinx shell (SDK → Xilinx Tools → Launch shell)
2. Position to releaseProj/makeBin path
3. Run copy_files,bat to copy files
4. Run stitch_images.bat to create bootable files

Steps 3 and 4 will generate boot.bin file in releaseProj/SD directory and copy updated application executable files (if found in the workspace) in releaseProj/SD directory. To apply these newly generated files to your system copy complete SD directory to SD card and follow instructions from Section 5.2.

7 REFERENCES

Table 1 List of references

Reference	**Description**
REF[1]	Felzenszwalb, P. F. (2010). Object Detection with Discriminatively Trained Part Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Revision History

Version	Date	Author	Approved by	Note
1.0.0	11/07/2014	MF	RM	Initial release
1.0.1	11/19/2015	MM		Only reference hw platform for demo kit in the UM.