**STATIC IMAGE PROCESSING USING PYNQ Z1**

**ABSTRACT:**

This document provides a detailed guide for the Pynq-Z1 Image Processing Project, an interactive system developed using the PYNQ-Z1 board. The project processes a static image with various filters (Grayscale, Threshold, Blurred, Edge Detection, and Sharpened) and displays the results on an HDMI monitor, allowing users to switch between filters dynamically. It serves as a foundation for image analysis applications, such as quality control in manufacturing.

**INTRODUCTION:**

The Pynq-Z1 Image Processing Project leverages the Xilinx PYNQ-Z1 board to perform real-time image processing and visualization. Utilizing Python and OpenCV, the project applies multiple image filters to a source image and outputs the results via an HDMI-connected monitor. This documentation outlines the setup, usage, code structure, and potential enhancements for the project.

**SYSTEM REQUIREMENTS:**

**Hardware:** PYNQ-Z1 board, HDMI monitor, HDMI cable, 12V power supply.

**Software:** PYNQ image installed on a microSD card, Jupyter notebook environment.

**Libraries:** Python with PIL, NumPy, Matplotlib, and OpenCV installed.

**Bitstream:** Compatible `base.bit` file for the PYNQ-Z1 overlay.

**SETUP INSTRUCTIONS:**

1. Insert the microSD card with the PYNQ image into the PYNQ-Z1 board.
2. Connect the HDMI cable from the PYNQ-Z1 to the monitor and power on the board.
3. Access the Jupyter notebook server via the PYNQ-Z1’s IP address in a web browser.
4. Upload the source image (`SourceImage.jpg`) to `/home/xilinx/jupyter\_notebooks/Image\_Processing/`.
5. Ensure the `base.bit` file is available in the working directory.

**WORKING:**

The PYNQ-Z1 Image Processing Project code operates by loading a 640x480 image in CELL 01, converting it from RGB to BGR for OpenCV compatibility, and displaying it in the Jupyter notebook. CELL 02 initializes the PYNQ-Z1 overlay and configures the HDMI output at 640x480 with 24-bit depth. CELL 03 applies filters—Grayscale, Threshold, Edge Detection, Blur, and Sharpening—using OpenCV, converts single-channel images to 3-channel BGR, and displays them in subplots. CELL 04 runs an interactive loop, prompting the user to select a filter (1-5) or exit (6), then writes the chosen 3-channel BGR image to the HDMI monitor, ensuring shape and dtype compatibility with a 1-second delay for updates.

**ISSUES FACED:**

1. **No Display After Initialization**

* **Description**: After running the code, no display appeared on the HDMI monitor despite successful initialization.
* **Solution**: Power cycled the PYNQ-Z1 board and restarted the Jupyter kernel. This reset the FPGA and cleared any stale states, allowing the HDMI output to initialize correctly.

1. **Grayscale Image Not Displaying on HDMI**

* **Description**: The grayscale image failed to display on the HDMI monitor, while other filters worked, due to a shape mismatch.
* **Solution**: Converted the single-channel grayscale image (gray) to a 3-channel BGR image using cv2.cvtColor(gray, cv2.COLOR\_GRAY2BGR) to match the 3-channel frame required by hdmi\_out.newframe().

1. **Shape/Dtype Mismatch Errors During Frame Writing**

* **Description**: Errors occurred when writing frames to the HDMI output, indicating shape or data type inconsistencies.
* **Solution**: Ensured all images were resized to 640x480 (matching VideoMode) and converted to compatible NumPy arrays with the correct dtype (e.g., np.uint8). Verified shapes with frame.shape and filtered\_np.shape checks.

1. **No Display After Reopening and Rerunning Notebook**

* **Description**: After closing the notebook, shutting it down from the running tab, and reopening it to rerun the cells, no display appeared on the HDMI monitor despite the code executing without errors.
* **Solution**: Power cycled the PYNQ-Z1 board and restarted the Jupyter kernel before rerunning all cells (CELL 01 for image loading, CELL 02 for HDMI initialization, CELL 03 for filter application, and CELL 04 for interactive display), ensuring the FPGA and HDMI pipeline reinitialized correctly to restore functionality.

**CONCLUSION:**

The Pynq-Z1 Image Processing Project demonstrates a practical application of embedded image processing. It provides a flexible platform for experimenting with image filters and can be extended for industrial or educational purposes.