EEE F348 FPGA BASED SYSTEM DESIGN LAB

A REPORT
ON

BLUR DETECTION USING FPGA

BY

Name of the Student	ID Number
Vinayak Tulsyan	***2020AAPS0442H***
Riya Bansal	***2020AAPS1333H***
Yogya Chawla	***2020AAPS1776H***

Prepared on completion of the
FPGA LABORATORY
EEE F348
AT

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

ACKNOWLEDGEMENTS

We are expressing our sincere gratitude to all those who have contributed to successfully completing this FPGA lab course and project.

First and foremost, I would like to thank our lab instructors for providing us with the necessary guidance and support throughout the lab sessions. Their expertise in FPGA technology and willingness to answer our queries have enhanced our understanding of the subject matter.

I am also indebted to my fellow lab mates, who have provided valuable insights and suggestions during our lab sessions. Their constructive feedback and collaborative approach have greatly enriched my learning experience.

Additionally, I would like to acknowledge the support provided by the lab technicians, who have ensured that the lab equipment and software were functioning optimally. Their timely assistance in resolving technical issues has been invaluable.

Lastly, I would like to thank BITS Pilani Hyderabad Campus for allowing us to participate in this lab and for equipping us with the necessary resources to carry out our experiments.
Thank you all for your contributions and support toward the successful completion of this course.

With the help and contribution of the people mentioned above, this project was a success.

ABSTRACT

This report presents an approach for detecting image blur using Field Programmable Gate Arrays (FPGAs). Image blur is a common issue affecting image quality and making extracting useful information from images difficult. The proposed approach utilises FPGA-based hardware acceleration to improve the speed and accuracy of blur detection. The approach is based on the analysis of the edge and frequency information of an image, and it uses a set of features that are computed on an FPGA-based system to detect blur accurately. The proposed method is suitable for real-time applications, including surveillance, autonomous vehicles, and robotics, where blur detection is critical for accurate object recognition and tracking. The FPGA-based blur detection approach offers a promising solution for high-performance image processing systems that require fast and reliable blur detection capabilities. To achieve this, we use the PYNQ board and Jupyter Notebook.

INDEX

Topics	Page No.
Acknowledgments	1
Abstract	2
Introduction	4
Objective	4
Proposed Idea	4
Requirements	5
Literature Survey	5
Implementation	6
Blur Detection	8
Limitations	10
References	10

INTRODUCTION

Blur detection plays a vital role in different fields like photography, medicine etc. It is necessary to detect blur images so that the information is not misinterpreted and thus, images can be sorted if they are blurred on not. We have used Laplacian method to achieve this, which is further explained in the report. The Xilinx® PYNQ board proved to be of immense help in executing the Python code.

The acronym PYNQ stands for "Python Productivity for Zynq." It is an open-source project developed by Xilinx that makes it easier to construct integrated systems using Zynq All Programmable Systems on Chips (APSoCs).

PYNQ-Z2 is an FPGA-based development platform member of the ZYNQ XC7Z020 FPGA family. This platform was developed expressly to enable PYNQ. It enables designers to construct more powerful systems by letting them take advantage of the ARM programming and advanced software offered by ZYNQ.

In addition, SoCs may be processed in Python by utilizing jupyter notebook, and the code can be designed and implemented directly in PYNQ-Z2. Tool libraries are the entry point for programmable scripts, and program libraries, much like programmed APIs, are imported and programmed similarly.

PROPOSED IDEA

Initially, the plan was to make a system that could detect the blur in the image that might be caused due to problems with either the camera lens, low frame rate, or perhaps a fast-moving image recorder. We did successfully write a code that could run on the Pynq Board to make this happen, but due to limitations in the laboratory inventory, we had to shift to a program that sorts images as blurred or sharp.

OBJECTIVE

To detect if the image is blurred using FPGA, specifically PYNQ Z2. It should be able to show us if the image is blurred or not blurred within the given threshold, which can be changed as desired.

Requirement Specification

- PYNQ Z2 Board
- PMOD camera (not available in the inventory list)
- Jupyter notebook
- SD card, Ethernet cable, and other peripherals.

Literature Survey

Laplacian filter

Enhancing edges and pinpointing areas of fast intensity change, the Laplacian or Laplacian of Gaussian (LoG) operator is a mathematical operator used in image processing. The Laplacian of a picture is a derivative of the intensity function in the second order. A kernel resembling the Laplacian operator is convolved with the image to arrive at this value. The resultant picture emphasizes sharp transitions in brightness, such as at corners and edges. A laplacian filter can be more sensitive and can help detect finer blurs. Image segmentation, feature extraction, and object recognition are all pre-processing steps that benefit greatly from using the Laplacian of an image to determine the borders and boundaries of objects in the picture.

However, the choice between any other types of filters depends on the specific application and the characteristics of the images being analyzed. For example, a Sobel filter which is the first-order derivative (gradient-based filter)is used to detect the edges of the images. The local variance is calculated using a 2x2 or 3x3 matrix window that detects the amount of blur. In this project, we chose to go with the Laplacian filter because of its more accurate nature.

The Variance of an Image

Mathematically, the variance of an image can be calculated by first finding the mean intensity value of all the pixels in the image and then computing the average of the squared differences between each pixel's intensity value and the mean. This measures how far the pixel values are spread out from the mean. The variance of an image is often used as a measure of image quality and can be used to indicate the amount of noise in an image. Images with low variance

have a more uniform intensity distribution and lesser noise.

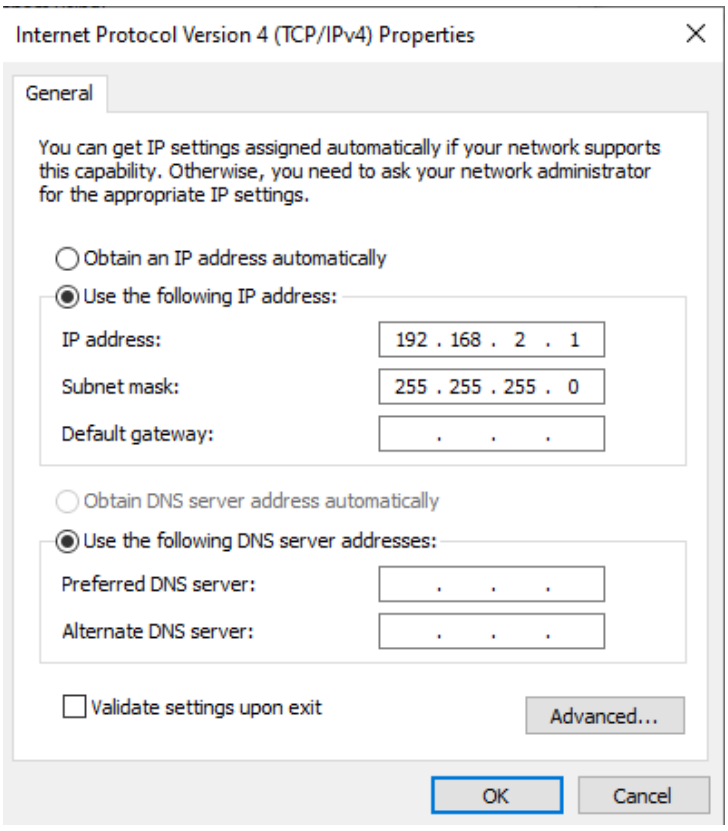
They may appear flat or featureless, while images with high variance have more excellent contrast and appear more detailed. In addition to its use as a quality metric, the variance of an image can also be used as a feature for image analysis tasks such as image classification, segmentation, and object recognition.

In a nutshell, the variance is the amount of detail in an image, and “blurriness” is just that loss of that detail, unsharp edges, and overall centrally-focussed distribution of the image. Hence, if given a reference value, or “threshold”, we can tell whether the image is blurry or sharp.

IMPLEMENTATION

- Setting up the Pynq board using the ethernet cable and USB B cable.
- Formatting of the SD card for using the iso file for the Pynq board.
- Connect the board to the computer by assigning a static IP address 192.168.2.xx to the computer using IP v4 internet protocol. Then type “ipconfig” on the command prompt.

Fig 1. IP configuration



- After the lights start to blink and 4 green lights show up, the board is ready to be configured.
- Open Jupyter Notebook using the IP address 192.168.2.99:9090/tree
- We then load the code and test images for execution on Jupyter.

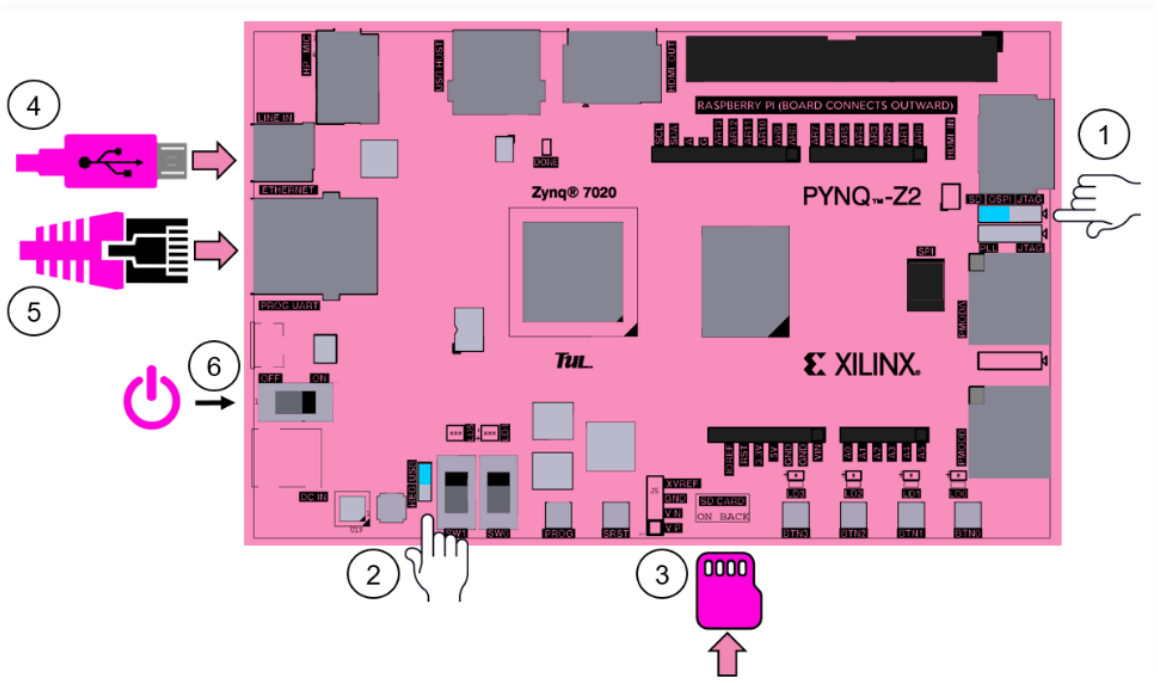


Fig 2. Pynq Board overview

- We then imported built-in Python modules like:
    1. “os” which helps the Python OpenCV code to interact with the processor’s operating system to change, write and read its directories.
    2. “matplotlib” is a graphing tool of python that will help us plot the processed images, which was later brought in as a solution due to problems mentioned in later sections
    3. “cv2”, that is the OpenCV library in python that helps in processing, filtering images.
    4. “numpy” is the tool in Python that helps in dealing with multi-dimensional arrays mathematically.

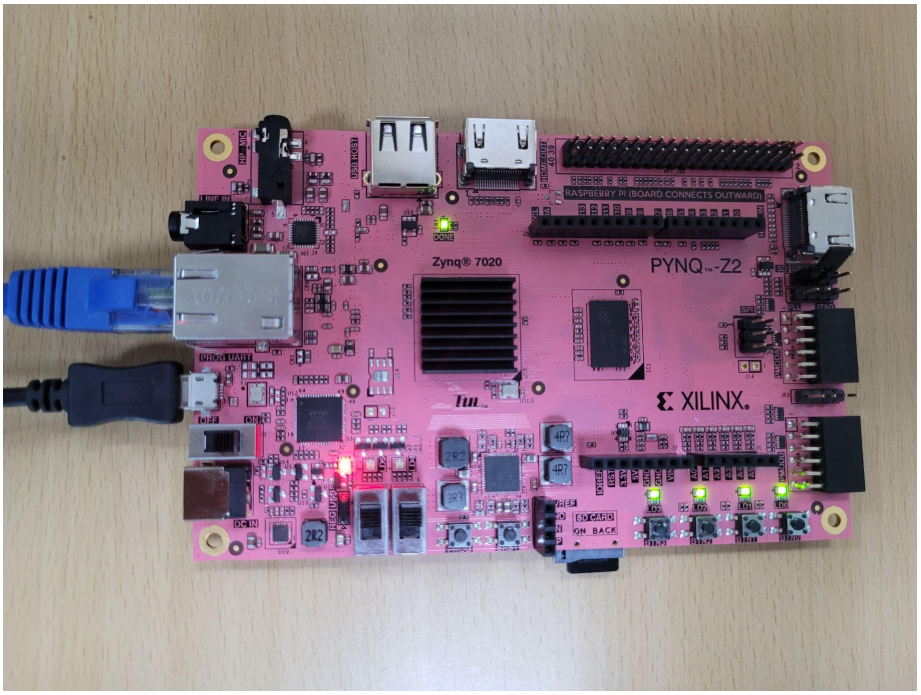


Figure 3. Pynq Z2 board available in our laboratory

Blur Detection

To test our code, we loaded 2 images into the program together, one naturally blurred and the other, normal. We coded into our program to calculate the variance of the image using the laplacian().var() method and based on whether it falls below or above a given threshold, the image is deemed “blurry” or “sharp” respectively. The code finally processes the image to give images tagged as blurred or otherwise. We had initially planned to write a program that can process video-stream in real-time to tell whether or not it is blurred. This can han have immense applications, for example, a surveillance camera lens can get covered in dust or can get wet due to rain outside and can become unreliable. But, if this method is applied to the real-time captured video stream, the problems with the camera lens can be rectified with immediate effect.

We have made a code that can loop through a number of images to sort which ones are smudged and which are not. The one shown below is just a prototype that goes through just two images at a time.

Working code for the project:

```
#importing all libraries and tools needed
import cv2
import numpy as np
import matplotlib.pyplot as plt
import os

# Define path to directory containing the images
dir_path = "/home/xilinx/jupyter_notebooks/P2_G5"
font = cv2.FONT_HERSHEY_SIMPLEX
threshold = 500
#defining threshold

# Loop through all files in the directory

for filename in os.listdir(dir_path):
    # Check if the file is an image (JPEG or JPG or
PNG)
     if filename.endswith(".jpg") or
filename.endswith(".jpeg") or
filename.endswith(".png"):

          # Load the image
          img_path = os.path.join(dir_path, filename)
          img = plt.imread(img_path)
          variance = cv2.Laplacian(img,
cv2.CV_64F).var()

#classifying image as blur or sharp
          if variance < threshold:
               cv2.putText(img,'Blur Image', (0, 50),
font, 0.5, (255,255, 69), 2)
          else:
               cv2.putText(img,'Sharp Image', (0, 50),
font, 0.5, (255,255, 69), 2)

        # Display the images one by one
     plt.imshow(img)
     plt.title(filename)
     plt.show()
```

Then, the subplots gave the following results :

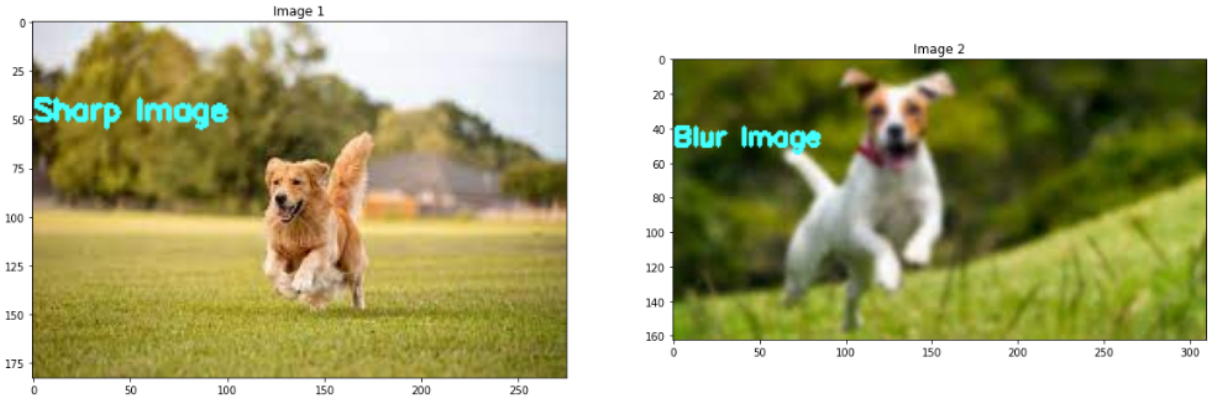


Figure 5. Result of the program executed

LIMITATIONS

- We had to resort to matplotlib since the Pynq board does not provide GUI to display the processed images in a separate window. A tool, GTK, needed at the backend to show these images had to be installed separately, and we did not have the means to do that.
- We could not find the password to our board to connect it via SSH(Secure SHell); hence we worked entirely on the Jupyter Notebook itself.
- We did not have access to a PMOD Web Cam in our lab inventory and hence had to shift to blur detection in images from real-time blur detection.

REFERENCES

- ***https://youtu.be/iW7oGRXX_dY***
- ***M. I. AlAli, K. M. Mhaidat and I. A. Aljarrah, "Implementing image processing algorithms in FPGA hardware," 2013 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), Amman, Jordan, 2013, pp. 1-5, doi: 10.1109/AEECT.2013.6716446.***
- ***https://www.geeksforgeeks.org/detect-an-object-with-opencv-python/***

- ***https://towardsdatascience.com/yolo-object-detection-with-opencv-and-python-21e50ac599e9***
- ***https://pynq.readthedocs.io/en/v2.3/appendix.html#assign-your-computer-a-static-ip***
- ***https://pynq.readthedocs.io/en/v2.3/getting_started.html#change-the-hostname***
- ***https://pyimagesearch.com/2020/06/15/opencv-fast-fourier-transform-fft-for-blur-detection-in-images-and-video-streams/***