Numerical Recipes

Report 1

Synchronization of lines in an Image

The hand-in deadline for this report is:

17th November 2017

Aim

The aim of this project is to synchronize the horizontal position of the lines in an image that have been corrupted in the digitisation process.

Introduction

A (simplified) video signal consists the analogue data for each line separated by a negative pulse as shown schematically below in figure **??**. The negative pulses signify the "start

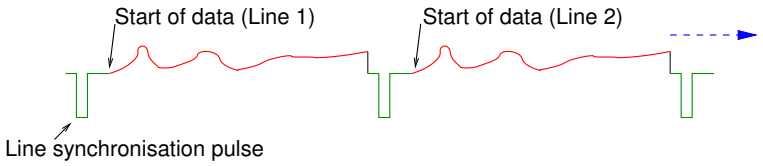

Figure 1: Schematic layout of a video signal with a synchronistion pulse at the start of each line of image data.

of line". The data for each line consists of an analogue voltage signal that starts at an, ideally, fixed time after each start of line pulse. So to digitise a video signal we have to detect the negative pulse and then start sampling the data at a fixed time later. We then sample the line data at regular time intervals so taking N sample along the line. In this case the time between sample gives the effective Δx for the final sampled image.

This process has two common problems, these being a) failure to correctly detect the line synchronization pulse, b) the time interval between the synchronization pulse and the start of the line data is subject to random variations with random initial noise. Both of these problems result is the horizontal displacement of some of the line of the digitised image which severely degrades the image, a typical image is shown in figure **??**.

Method

In order to correct these images your have to

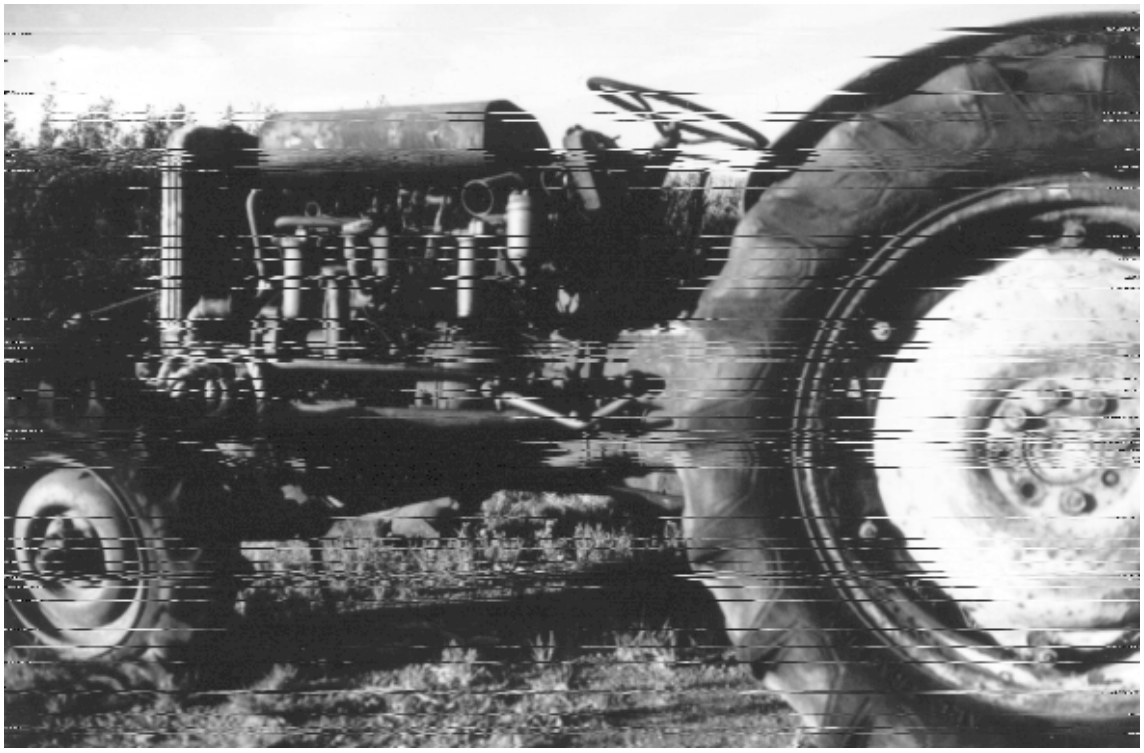

Figure 2: Typical image with random shifted lines.

1. Find the shift between adjacent lines; for simplicity you can assume that the FIRST line is not shifted, and adjacent lines in an image are "very similar".
2. Move the shifted lines back into place.
3. Consider what to do with lost image data at the ends of the lines.

To set about this you need a way of detecting the relative shift between two "very similar" lines. To do this you can cross-correlate pairs of lines. This is done by taking the Fourier transform of each, conjugating one of them, and multiplying them together. If you now take the inverse Fourier transform of this cross-correlation product then:

- If the lines are not shifted relative to one and other, the peak will be at zero.
- If they are relatively shifted then the peak will be at the value corresponding to the shift.

Supplied Images

There are 4 supplied images, with every more desynchronised lines, located at

```
~wjh/Course/NumRec/images/desync1.pgm
~wjh/Course/NumRec/images/desync2.pgm
```

~wjh/Course/NumRec/images/desync3.pgm
~wjh/Course/NumRec/images/desync4.pgm

In each image both the number of corrupted lines *and* the length of the horizontal corruption is random. The degree of corruption is different in the three test scenes with `desync1` being the least corrupted and `desync4` the most.

These images are in `pgm` binary format. This is a binary image format with a short header giving the image size, followed by the data with one `byte` per pixel. There is a short PYTHON program in figure **??** to read an image into a two-dimensional `np` array and then display it using `matplotlib`. There are supplied routines to read and write these images. They can also be read by `gimp`.

```
from scipy import misc      # Import misc
import matplotlib.pyplot as plt

def main():
    #         Get the filename as string
    fn = str(raw_input("File : "))

    #         Read file it np array
    im = misc.imread(fn)

    #         Display with grayscale colour map
    plt.imshow(im,cmap=plt.cm.gray)

     #         Show the image
     plt.show()

main()        # Run the program
```

Figure 3: Python demonstration program to read and display an image.

Tasks

To devise an algorithm to correct images corrupted in this way and write a program to perform this correction. You should test your program on all four images.

Report

You should submit a report. The basic style of the report should be that of

1. Introduction describing the context and the problem.

2. Description of algorithms and methods employed.
3. Presentation of Results (from all test images)
4. Summary and conclusions
5. Appendix which **must** contain all code written by you. All code must be well documented in the code itself.

The report, excluding the code should not exceed 8 pages. The report is to be written so that it would be understandable by a scientist who has no knowledge of the course. Therefore it must describe the context, problem, methods applied and results in a self-standing way, which does not rely upon any other knowledge of the problem. The report should be submitted on LEARN as a pdf file. Report marking criteria include:

1. Structure of report
2. Quality of communication to reader with no prior knowledge.
3. Quality and completeness of presentation of results.
4. Accuracy/correctness of algorithms and results
5. Quality of code design
6. Depth to which the problem has been addressed and comparisons made as asked for.
7. Quality of conclusions.

This project and report must be your own work, and must contain a declaration to this effect as detailed in your *Course Booklet*.