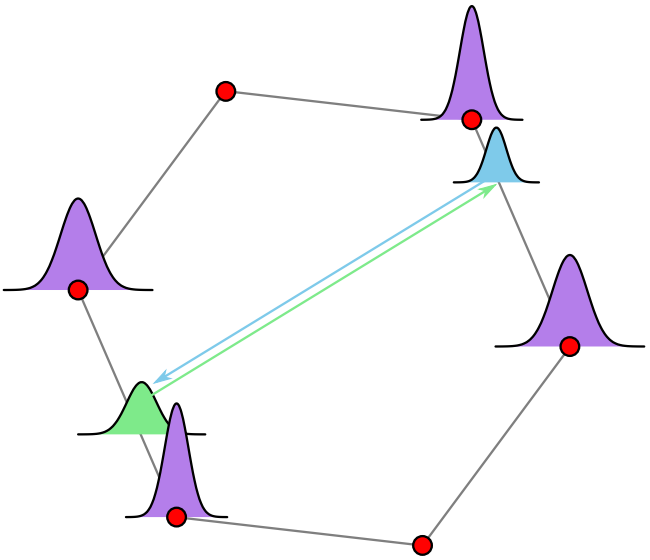

THE UNIVERSITY OF MELBOURNE

Final Report

The Art of Scientific Computing: Stereo Vision

Author:
Haonan Li

Subject Handbook code:
COMP-90072

The University of Melbourne

10 April 2018

Subject co-ordinators: A/Prof. Roger Rassool & Kevin

Contents

Chapter 1

Cross Correlations

Cross correlation is powerful (and very simple) statistical tool for computing the degree to which two signals are correlated (or similar). and also for computing lags.

1.1 Normalised Spatial Cross Correlation in 1d

In signal processing, cross-correlation is a measure of similarity of two series as a function of the displacement of one relative to the other. As we all know, signal could be continuous and discrete. In this project, we only focus discrete signal.

For discrete functions f and g, the cross-correlation is defined as:

$$r = \frac{1}{N}\sum_{i=1}^{i=N}(f(i)-\bar{f})(g(i)-\bar{g})$$

The problem at the moment is that the value of r is somewhat arbitrary because of the varient amplitude of signal. One way around this is to normalise signal with the root-mean-quare. Normalized cross correlation is typically done by subtracting the mean and dividing by the standard deviation. As:

$$r = \frac{1}{N}\sum_{i=1}^{i=N}\frac{(f(i)-\bar{f})(g(i)-\bar{g})}{\sigma_f\sigma_g}$$

where

$$\sigma_f = \sqrt{\frac{1}{N}\sum_{i=1}^{i=N}(f(i)-\bar{f})^2} \qquad \sigma_g = \sqrt{\frac{1}{N}\sum_{i=1}^{i=N}(g(i)-\bar{g})^2}$$

Matlab code of normalized 1d cross correlation is in appendix.

1.2 Signal Offset

If there are two signals(denote by vector) come from the same source and are just offset by some time. We can easily use cross correlation find the offset time and distance between two sensors. Just find the max value's posision of the cross correlation vector compute from two signal. Then find corresponding positions in the signal and compute the offset of the signal vector. If we know the rate and propagation speed, we can compute offset time and distance between the two sensors as follow:

$$\text{offset time} = \frac{\text{offset}}{\text{sample rate}}$$

$$\text{distance} = \text{offset time} * \text{propagation speed}$$

1.3 Normalised Spatial Cross Correlation in 2d

The cros correlation can be extended to two-dimensional matrix. Condiser two matrices, t (template) and A (search region), The matrix A willl always larger than the matrix t. We can use two nested for-loops to "leg" t over A, compute for each "lag" the cross-correlation. Normolized cross correlation of two matrices defines as:

$$R(lag_x, lag_y) = \frac{\sum_{x,y}[A(x,y) - \overline{A_{lag_x,lag_y}})][t(x - lag_x, y - lag_y) - \bar{t}]}{\{\sum_{x,y}[A(x,y) - \overline{A_{lag_x,lag_y}})]^2 \sum_{x,y}[t(x - lag_x, y - lag_y) - \bar{t}]^2\}^{0.5}}$$

Where $\bar{t}$ is the mean of t, $\overline{A_{lag_x,lag_y}}$ is the mean of A in the region under t.
Matlab code of normalized 2d cross correlation is in appendix.

1.4 Find patterns

Images are just a matrix of pixel values in most image processing. It is naturally think of computing cross correlation between two images to get more relation informations of them. In this task, we get two images, one of them is a section of the other. We can easily find where the section of the image fits in the whole through cross-correlation.

Here, we have a section (rocket man) show in Figure 1.1 and a search region (maze) as Figure 1.2.

Figure 1.1: Rocket Man (section)

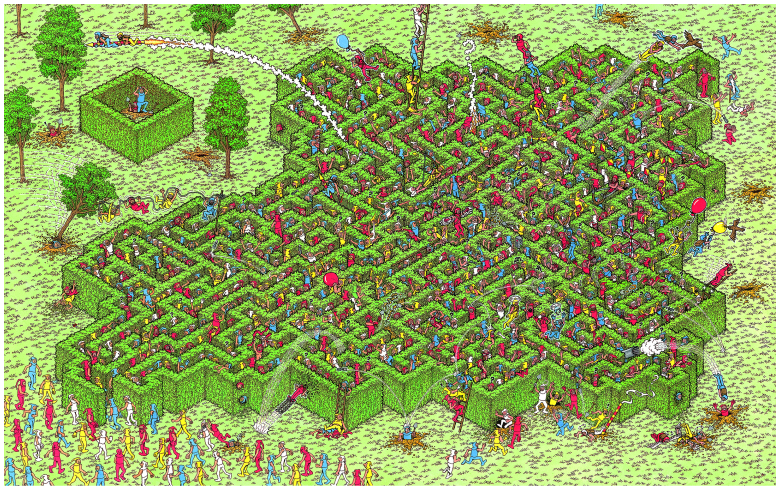

Figure 1.2: Maze (search region).

The visualizetion of cross correlation is shown in Figure 1.3 and Figure 1.4 . The maximum of the cross-correlation corresponds to the estimated location of the section. Which means the most similar position of the section and search region. We mark it with a red star in the whole image. The result shows in Figure 1.5.

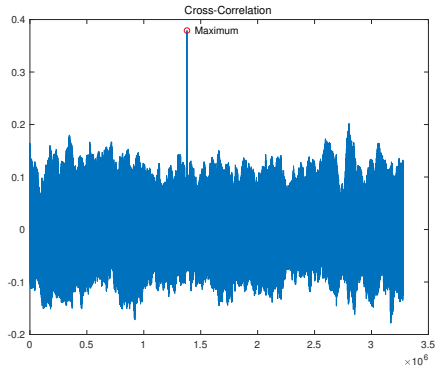

Figure 1.3: Cross correlation 2D visualization

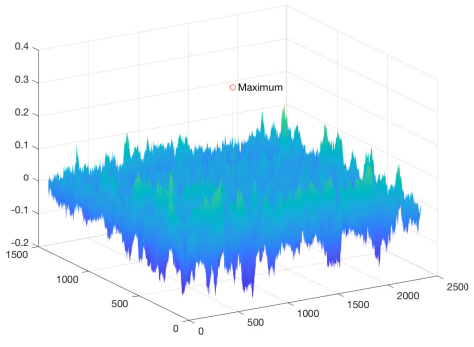

Figure 1.4: Cross correlation 3D visualization

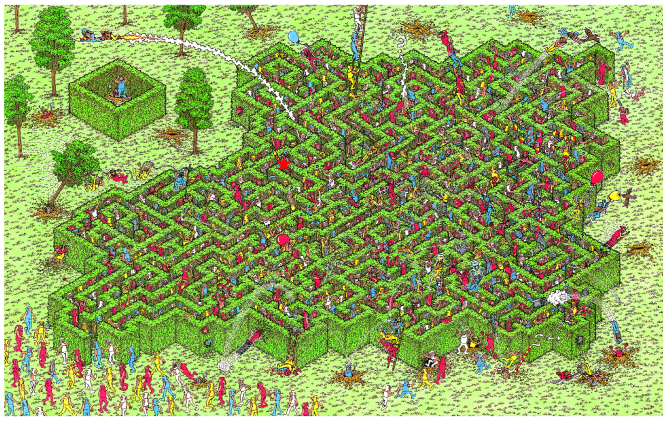

Figure 1.5: Align result.

Appendix A

Program flowchart

A flowchart is a conceptual aid which highlights input and output that a program should have, as well as the consequences of any decisions that it must make. Sometimes, these will include *pseudocode*, which mimics some of the more common coding structures in a universal way. (For instance, you might say 'loop over all array addresses'.)

This particular flowchart indicates a program that reads three numbers from the terminal, and then produces the largest of the numbers back to the terminal. The colour scheme is not mandatory, but helps the reader to know what is going on at a glance.

This flowchart was drawn with the free 'Lucidcharts' extension to Google Docs, from which the result was exported as an `svg` file, and any unnecessary features were then cropped out with Inkscape. Choose any method you wish to create your own flowchart!

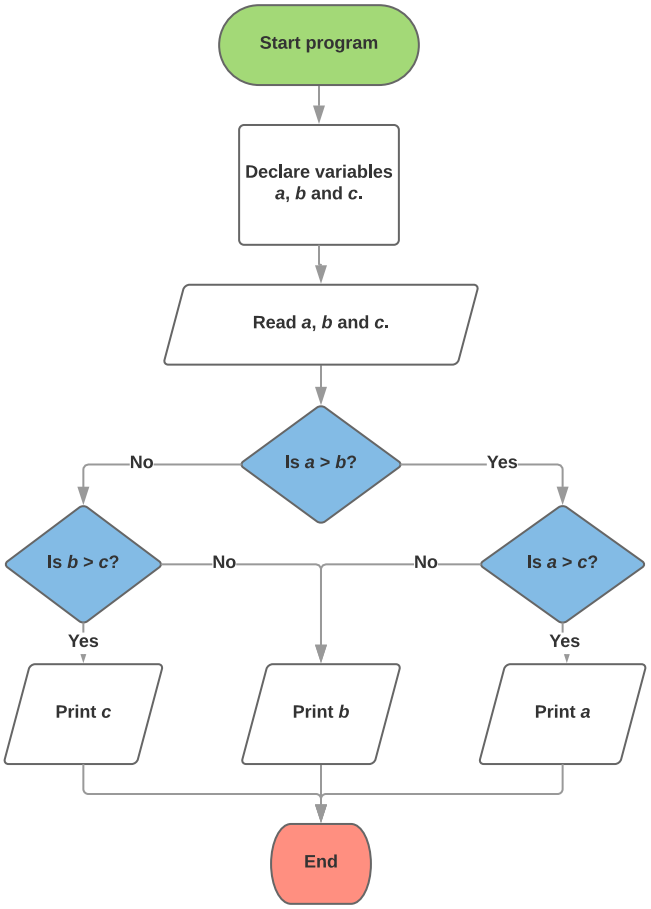

Figure A.1: A flowchart for the determination of the largest of three user-selected numbers.

Bibliography

[1] M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions.* Dover Publishing Inc. New York, 1970.

[2] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in Fortran (Cambridge).* Cambridge Univ. Press, 1992.