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. In this part, we first learn the cross correlation in 1 dimension and its applications to signal processing. Then we extend it to 2 dimensions and familiar its usage in image processing.

After these basic concepts and application, we consider the efficiency of computer cross correlation. We find it is really slow as the signals and images become larger, then we apply a new method as an alternative to spatial method, which is spectral cross correlation with a Fourier transform and inverse Fourier transform. We find this method achieve better performance in scale data.

1.1 Normalized 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 of the above function is that the value of r is somewhat arbitrary because of the variant amplitude of f and g. One way around this is to normalize 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. First find the max value's position of the cross correlation vector computed from two signals. Then find corresponding positions of the two signals and compute the offset of the signal vector. If we know the sample rate and propagation speed, we can compute offset time and distance between the two sensors as follow:

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

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

In this task, we take two signal files and compute the cross correlation between them. Figure show the result of the cross correlation.

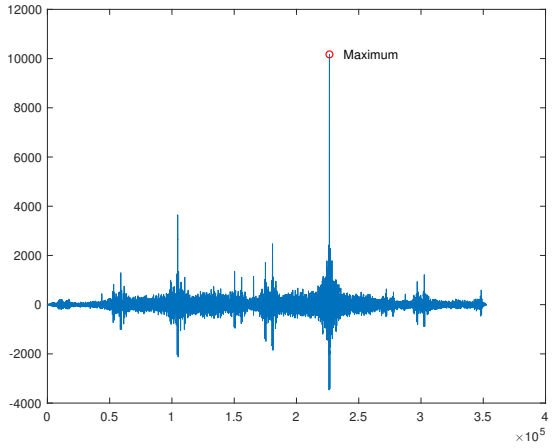


Figure 1.1: Cross correlation of two signals.

1.3 Normalised Spatial Cross Correlation in 2d

The cros correlation can be extended to two-dimensional matrix. Consider two matrices, t (template) and A (search region), The matrix A will 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. Normalized 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 Image Alignment

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

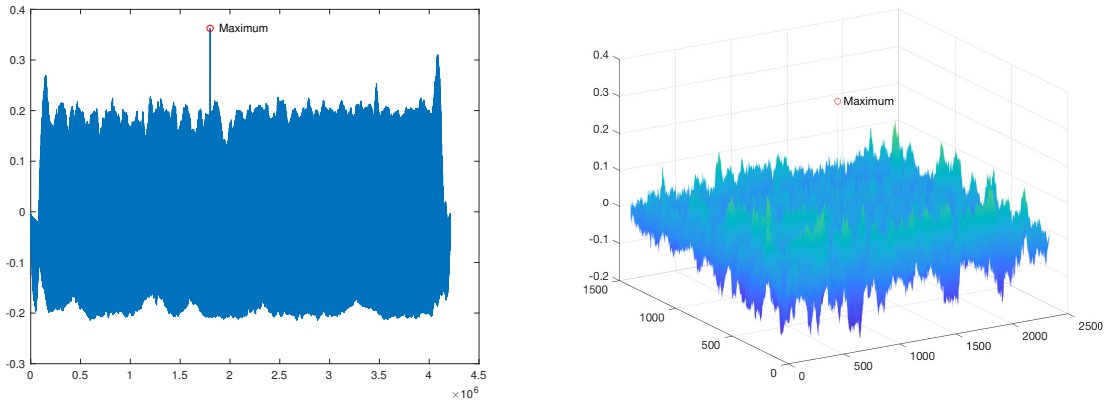
this task, we get two images, one of them is a section of the other. We can easily find where the section of the image fits in the whole through cross-correlation.

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

(a) Rocket Man (section). **(b)** Maze (search region).

Figure 1.2: Image alignment.

The visualization of cross correlation is shown in Figure 1.3a and Figure 1.3b . 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 1.4.



(a) 2D visualization **(b)** 3D visualization

Figure 1.3: Cross correlation visualization

1.5 Spectral Cross Correlation

Cross correlation can also be done in the spectral domain by completing a Fourier transform, multiplying signals, and doing an inverse Fourier transform.

In this task, we use a fast Fourier transform (FFT) algorithm to convert a signal from its original domain to a representation in the frequency domain and inverse fast Fourier transform (IFFT) vice versa. It is an advanced algorithm of the discrete Fourier transform (DFT), And it manages to reduce the complexity of computing the DFT from $O(n^2)$, which arises if one simply applies the definition of DFT, to $O(n \log n)$.

Figure 1.4: Image Alignment result.

Fast Fourier transform is a very useful and powerful algorithm in computing convolution and cross correlation. It can be shown that the discrete convolution of signal u and v as defined by,

$$(u * v)(\tau) = \sum_{m=1}^{N} u(m)v(\tau - m)$$

can also be expressed in terms of Fourier transform

$$(u * v)(\tau) = \mathcal{F}^{-1}\{\mathcal{F}(u) \cdot \mathcal{F}(v)\}$$

where $\mathcal{F}(u)$ is the Fourier transform of u, $\mathcal{F}(v)$ is the Fourier transform of v, and $\mathcal{F}^{-1}$ is the inverse Fourier transform.

As for cross correlation, it can be calculated through summation of a product.

$$(u * v)(\tau) = \sum_{m=1}^{N} u^*(m)v(\tau + m)$$

or using FFTs.

$$(u * v)(\tau) = \mathcal{F}^{-1}\{(\mathcal{F}(u))^* \cdot \mathcal{F}(v)\}$$

Where the $*$ refers to the complex conjugate.

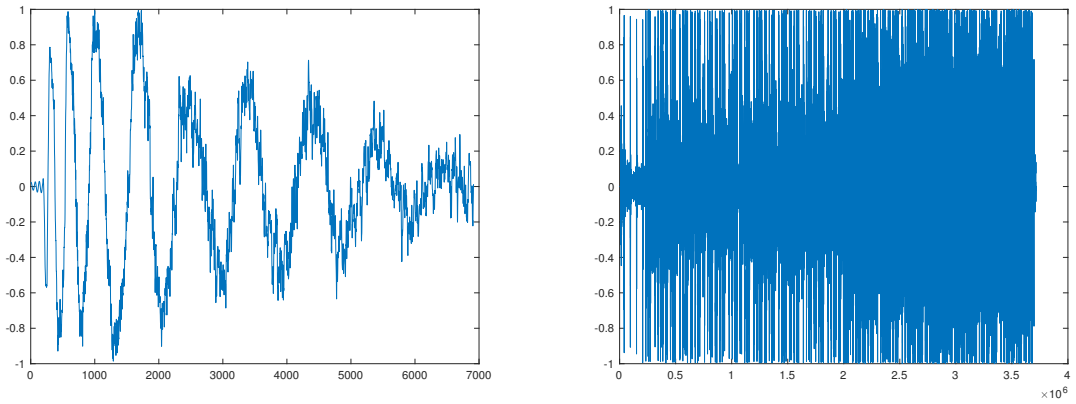
Table 1.1 compare the run time of two different method of task in section 2. From which we can find that spectral method using Fourier transform achieves better performance.

method	Spatial	Spectral
time(s)	4.6741	0.1103

Table 1.1: Run time compare of spatial and spectral methods.

1.6 Pattern Finder

In this task, I pick a piece of music and cut a small piece of it (a drum) through listening. Figure 1.5 shows two piece of waveforms separately. And 1.6 shows all occurrence of the element, because it is not a normalized correlation picture, all positions with the y-coordinates larger than 1000 is the appearance of the drum. We can also find that others background noise have effect of our calculate but not serious.



(a) drum waveform **(b)** music waveform

Figure 1.5: Two oscillographs

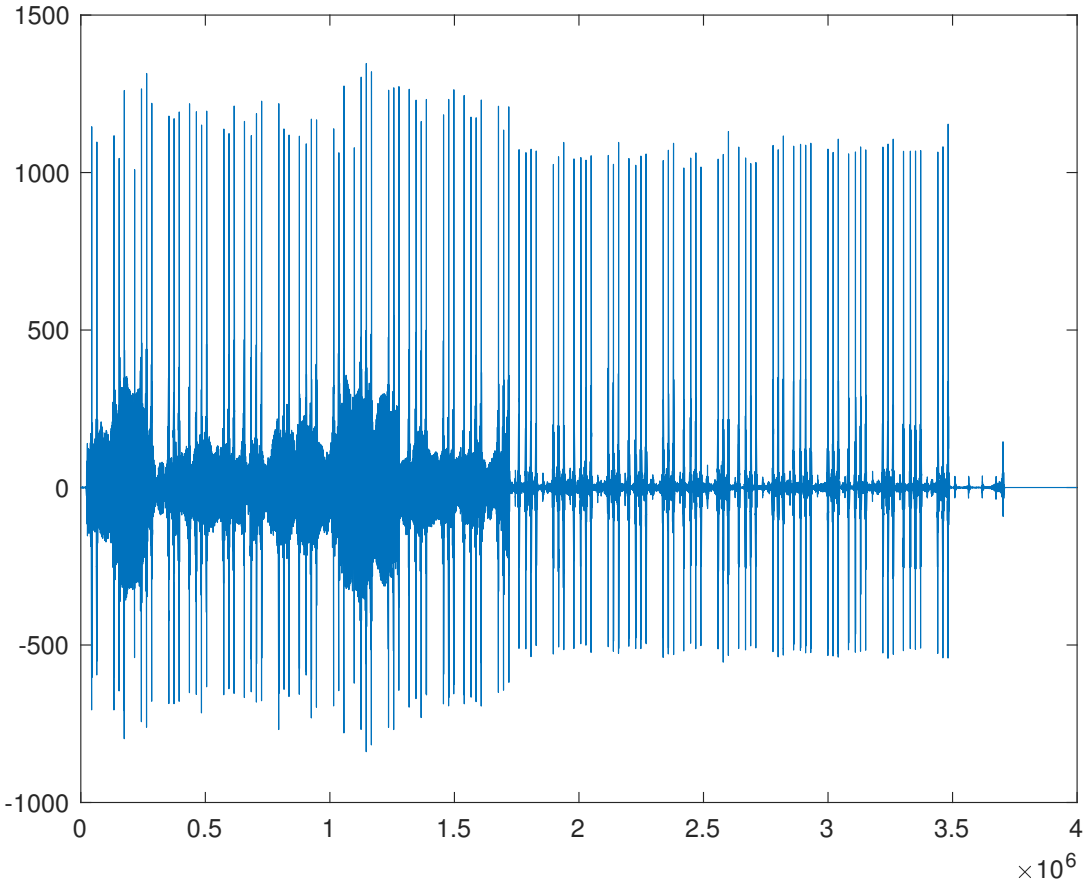


Figure 1.6: Two oscllographs

1.7 Cross correlation in 3 dimension

This part is just some thinking of me, not really theme related.

From the above research, we know that 1d cross correlation can be used to process acoustical signal and 2d cross correlation is very useful for image processing. Then, how about 3d cross correlation? If 3d cross correlation can be build, we can easily know the space structure similarity of two object, theoretically. This task may have its value in research in microcosmos like molecular physics and microbiology. Consider we have one cell with particular disease as specimen but we do not know the reason of its pathogenicity. At the same time, we have several cells, some of which have the disease. If we can find spatial cross correlation between the specimen and other cell, and same diseased cell have the similar space structure, we can distinguish the healthy cell and diseases.

Appendix A

Matlab code

A.1 Part 1

A.1.1 Normalised Spatial Cross Correlation in 1d

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author:   Haonan Li                                                    %
% Purpose:  Compute the normalised cross correlation                     %
%           vector of two vectors of the same size                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function vec_c = normalised_spatial_correlation_1d(vec_a, vec_b)

len = length(vec_a);
vec_a = vec_a - mean(vec_a);
vec_b = vec_b - mean(vec_b);
% compute sigma
sigma = sqrt(sum(vec_a.^2)+sum(vec_b.^2))
% add 0 in the head and tail of vec_b
vec_b = [zeros(1, len-1), vec_b, zeros(1, len-1)];

% compute vec_c
vec_c = zeros(1, 2*len-1);
for i = 1:(2*len-1)
    % nor_a, nor_b are the normalization factors
    vec_c(i) = sum(vec_a .* (vec_b(1,i:i+len-1)))/sigma;
end
```

A.1.2 Signal Offset

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author:   Haonan Li                                                    %
% Purpose:  Given two signal file, these signals have                    %
%           come from the same the same sourcce and just                 %
%           offset by some time, find the offset time                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function offset = signal_offset_checker(f_1, f_2)

% read data from file
```

```
file_1 = importdata(f_1);
file_2 = importdata(f_2);
sig_1 = file_1.data';
sig_2 = file_2.data';
SAMPLE_RATE = 44100;

% compute cross correlation of two signal
tic;
% special method
cross_cor = spatial_correlation_1d(sig_1, sig_2)';
% spectral method
% cross_cor = spectral_correlation_function(sig_1, sig_2)';
run_time = toc

% find the position of max cross coorelation value,
% then compute the offset.
[max_value, max_pos] = max(abs(cross_cor));
offset = abs(length(sig_1) - max_pos)

%compute offset time and sensor distance
offset_time = offset / SAMPLE_RATE
distance = 333 * offset_time
```

A.1.3 Normalised Spatial Cross Correlation in 2d

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author:   Haonan Li                                                   %
% Purpose:  Receive two matrix, t(template) and A(search                %
%           region) returns normalized cross-correlation                %
%           of these two matrix, A larger than t                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function mat_r = normalised_spatial_correlation_2d(mat_t, mat_A)

% if input is two image
% pic_a = imread(mat_t)
% pic_b = imread(mat_A)
% mat_t = mean(pic_a, 3)
% mat_A = mean(pic_b, 3)

% init mat_r with 0
size_t = size(mat_t);
size_A = size(mat_A);
% I don't think this size is best, but I have no idea to deal with
% out-of-bounds errors at the image edge.
mat_r = zeros(size_A(1)-size_t(1)+1, size_A(2)-size_t(2)+1);
size_r = size(mat_r);

% mat_t_ is a mat of mat_t minus mean(mat_t)
mat_t_ = mat_t - mean(mat_t);

for i = 1:size_r(1)
    for j = 1:size_r(2)
        % compute mat_r(i,j)
        % first get the part of mat_A covered by the mat_t
        mat_A_under = mat_A(i:(i+size_t(1)-1), j:(j+size_t(2)-1));
        mat_A_under_ = mat_A_under - mean(mat_A_under);
```

```
        % compute numerator and denominator of the cross correlation
        numerator = sum(sum(mat_t_ .* mat_A_under_));
        denominator = sqrt(sum(sum(mat_t_.^2)) * sum(sum(mat_A_under_.^2)));
        mat_r(i,j) = numerator/denominator;
    end
end

% figure
figure,surf(mat_r);figure(gcf)
```

A.1.4 Image Alignment

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author:   Haonan Li                                                   %
% Purpose:  Give two pictures, one is a part of another  %
%           find the corresponding position and mark it  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function pic_R = find_the_rocket_man(pic_t, pic_A)

% read images
pic_t = imread(pic_t);
pic_A = imread(pic_A);
mat_t = mean(pic_t, 3);
mat_A = mean(pic_A, 3);
size_t = size(mat_t);

% compute cross correlation
tic;
cross_corr = normalised_spatial_correlation_2d(mat_t,mat_A);
run_time = toc

% max cross correlation position
[pos_y pos_x] = find(cross_corr == max(max(cross_corr)))

% save marked image
imshow(pic_A);
hold on;
marker = plot(pic_A(1,1), 'p');
marker.XData = pos_x+size_t(2)/2;
marker.YData = pos_y+size_t(1)/2;
marker.MarkerSize = 30;
marker.Color = 'r';
marker.MarkerFaceColor = 'r';
saveas(gcf, 're','png');
```

A.1.5 Spectral Cross Correlation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author:   Haonan Li                                                   %
% Purpose:  Compute the spectral cross correlation       %
%           vector of two vectors of the same size       %
%           using Fourier transform.                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
7
8  function vec_c = spectral_correlation_1d(vec_a, vec_b)
9
10 f_a = fft(vec_a);
11 f_b = fft(vec_b);
12 conj_f_a = conj(f_a);
13 mid_res = conj_f_a .* f_b;
14 vec_c = ifft(mid_res);
15
16
17 % Can not deal with N not the power of 2
18 function res_fft = my_fft(vec)
19
20 vec_size = size(vec);
21 N = vec_size(2)
22 c = zeros(1,N);
23 % indexing computation
24 j1 = 0;
25 for i = 1 : N
26     if i < j1 + 1
27         tmp = vec(j1 + 1);
28         vec(j1 + 1) = vec(i);
29         vec(i) = tmp;
30     end
31     k = N / 2;
32     while k ≤ j1
33         j1 = j1 - k;
34         k = k / 2;
35     end
36     j1 = j1 + k;
37 end
38
39 % Butterfly computing
40 dig = 0;
41 k = N;
42 while k > 1
43     dig = dig + 1;
44     k = k / 2;
45 end
46
47 n = N / 2;
48 for m = 1 : dig
49     dist = 2 ^ (m - 1);
50     idx = 1;
51     for i = 1 : n
52         idx1 = idx;
53         for j1 = 1 : N / (2 * n)
54             r = (idx - 1) * 2 ^ (dig - m);
55             coef = exp(j * (-2 * pi * r / N));
56             tmp                 = vec(idx);
57             vec(idx)          = tmp + vec(idx + dist) * coef;
58             vec(idx + dist)   = tmp - vec(idx + dist) * coef;
59             idx = idx + 1;
60         end
61         idx = idx1 + 2 * dist;
62     end
63     n = n / 2;
64 end
65 res_fft = vec;
```

A.1.6 Pattern Finder

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author:   Haonan Li                                                    %
% Purpose:  Compute the spectral cross correlation                       %
%           vector of two vectors of the same size                       %
%           using Fourier transform.                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function res = pattern_finder(wavfile,patfile)

[y1,Fs] = audioread(wavfile);
x1 = y1(:,1);

[y2,Fs] = audioread(wavfile);
x2 = y2(:,1);

corr_vec = spectral_correlation_1d(x1,x2)


%{
X_f=fft(x);
figure (1)
subplot(2,1,1);
plot(x);
xlabel('time')
title('time domain')
subplot(2,1,2);
plot(abs(X_f));
xlabel('frequency')
title('frequency domian')

X1_f=X_f;
X1_f(10:90) = 0;

X2_f=X_f;
X2_f(1:90)=0;
X2_f(96:end)=0;

x1_reconstruc = ifft(X1_f);
x2_reconstruc = ifft(X2_f);

figure (2)
plot(real(x1_reconstruc));
title('Reconsturctured x1')

figure (3)
plot(real(x2_reconstruc));
title('Reconsturctured x2')
%}
```

Bibliography