Automatic Fingerprint Recognition

Benjamin May & Edward Wastell

March 3, 2014

Abstract

Stuff happened.

1 Introduction

Fingerprints have been used to identify people since the 19^{th} century and have been used in criminal investigations since about that time. More recently fingerprints have been used as biometric markers used in boarder control, library stock control and computer and building access control systems. The need for a robust automatic fingerprint recognition system is obvious.

Most fingerprint recognition systems in use are based on the idea of identifying minutiae (points where a ridge ends or joins with another ridge) — in this article a system that uses the greylevel gradient to find minutiae is discussed.

2 Theory and Implementation

2.1 Point Normal Direction

The point normal function works by taking four mutually adjacent points (*i.e.* a 2×2 array of pixels) and fitting a plane to them. If the greylevel at the pixel (x_k, y_k) is denoted h_k and the level from the fitted plane is denoted p_k then the plane fitting part of the function can be seen as minimising the following expression:

$$\min_{n_1,n_2,c} \sum_k |h_k - p_k|^2 \qquad (1)$$

Which in matrix form can be expresed as:

$$\left| \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{pmatrix} - \begin{pmatrix} -x_1 & -y_1 & 1 \\ -x_2 & -y_2 & 1 \\ -x_3 & -y_3 & 1 \\ -x_4 & -y_4 & 1 \end{pmatrix} \begin{pmatrix} n_1 \\ n_2 \\ c \end{pmatrix} \right|^2 \qquad (2)$$

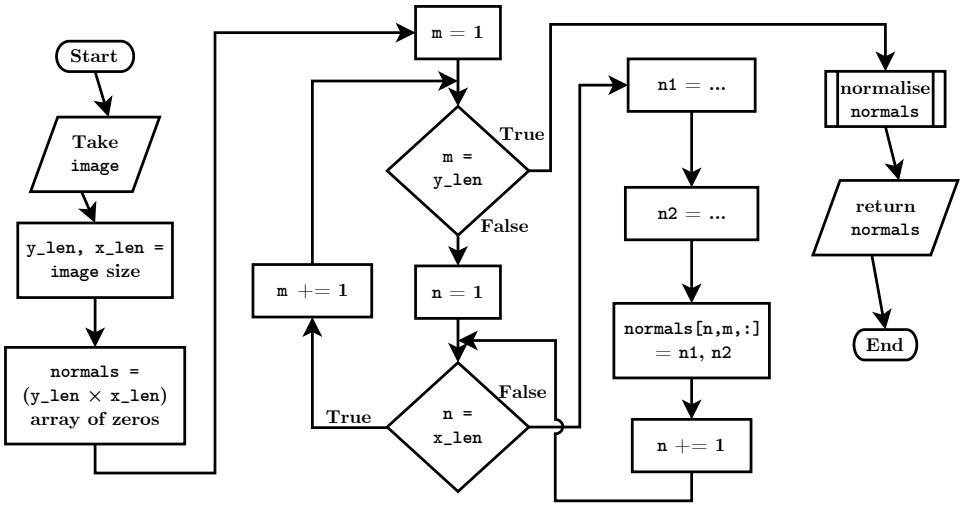

Figure 1: Flow chart of the PND function as implemented.

Where n_1, n_2 and c are the x, y and z components of the surface normal respectively. This is a simple least-squares minimisation and after some rearranging we obtain the following expressions for the optimum surface normal:

$$\begin{aligned} n_1 &= \frac{-h_1 + h_2 + h_3 - h_4}{4} \\ n_2 &= \frac{-h_1 - h_2 + h_3 + h_4}{4} \\ c &= \frac{h_1 + h_2 + h_3 + h_4}{4} \end{aligned} \tag{3}$$

Since we are only concerned with the components in the x, y–plane the function implemented here doesn't bother calculating c (figure 1). Once all 2×2 neighbourhoods have been fitted the function returns n_1 and n_2 then terminates.

As the NPD function needs a four points to fit a plane to the decision has to be made regarding how edges are handled. One possibility is to 'wrap' the edges of the image around, effectivly forming a torus — this was not implemented here because it would mean that for two edges the values of n_1 and n_2 would depend on greylevels from the other two edges. The second possibility (implemented here) is to make the output array smaller by 1 pixel in both dimensions, so if the original image is $n \times m$ pixels the array of normals is $(n-1) \times (m-1)$. From figure 1 it can be seen that the PND function implemented here loses the top and right hand pixels.

2.2 Averaged Tangent Direction

The ATD function calcualtes the x, y–plane tangent that best fits with the surface normals generated by the PND function in a given $n \times n$ neighbourhood. This has the effect of smoothing out noise but, as with analogous smoothing operations, can

obscure features with a size comprable to that of the kernel chosen. Here a default kernel of size 9×9 is chosen, which should balance computation time and over smoothing against noise correction.

3 Experimental Procedure

4 Results & Analysis

5 Conclusion