Detection of a scaled and resized sub-image in a parent image

Bartlett, Rosie
rosiegbartlett@gmail.com

January 2021

Abstract

This paper is an outline of several proposed methods to detect a non-linearly scaled sub-image inside a parent image. I will discuss the different aspects of each method, comparing the efficiency and accuracy of each method for differently sized images.

1 Proposed method

1.1 Fourier Series Approximation

The proposed method to detect a sub-image inside its parent image uses fast Fourier transforms to approximate each row of the image as a Fourier series, and then compare the original values of the approximation at individual points, along with several derivatives of the appproximation. This ammounts to checking that the colour of the images at the chosen points is the same, and that the context of the points are the same. For the points used, the top left corner of the sub-image will have it's C matrix calulated, which I will refer to as C_0, which will be used a benchmark for all other points until a match is found. The C matrix refers to Equation 1.1, where $f_n(t)$ is the Fourier approximation of the array of values representing the n^{th} row of the image, and m is the upper limit of derivatives chosen; which will be explored further.

$$C = \begin{bmatrix} f_n(t) \\ {f_n}'(t) \\ {f_n}''(t) \\ \vdots \\ {f_n}^m(t) \end{bmatrix} \tag{1.1}$$

Once C_0 has been calulated, the code will work from the top left corner of the parent image in the following manner:

1. Calulate the Fourier coefficients matrix, F, for the current row of pixels
2. Calulate the coefficients of the derivatives of the approximation, upto and including the m^{th} derivative
3. For each pixel in the current parent image row, calulating the C matrix for each, using the appropriate value of t
4. The current C is then compared to C_0, and if the requirements are met then the code assumes a point to have been found on the parent image that correlates with the upper left point in the sub-image. These requirements are detailed below.

For the code to assume a point to have been found to correlate with the upper left corner of the sub-image, it first checks divides C_0 by C elementwise, resulting in R_0, which is saved. It then calulates the mean of the elements of R_0, giving $\bar{R}_0$. These are then used to calulate the mean squared error of the matrix, as described in Equation 1.2

$$\frac{1}{m+1}\sum_{n=0}^{m}\left(R_0[n]-\bar{R_0}\right)^2 \tag{1.2}$$

If the result of this is less than some constant d, the threshold error, then the requirements are said to have been met and this the top left corner of the sub-image is assumed to have been found. For further notation, this pixel will be refered to as p_0.

Once this point has been found, the code then goes back to the sub-image to calulate the C matrix for the top right pixel, C_1. This is then compared with pixels to the right of p_0. If a matching pixel is found in the method described above, replacing C_0 and R_0 with C_1 and R_1 respectively, a further check is required. The value of R_0 is checked aginst R_1 to ensure that they are similar, by calulating the mean squared error between the two matrices and checking to make sure this is below a specifiec threshold value. If this extra condition is met, then the point at which these conditions are met is assumed to be the top right corner of the sub-image; this point is p_1.

The search for p_1 is only performed along the same row as the row p_0 is on since the image has not been rotated. If p_1 is not found on the same row as p_0, then p_0 is disregarded and the code begins the search for p_0 from the previous position of p_0 until it is found again.

Once both p_0 and p_1 are located, the code must then locate p_2 and p_3, the bottom left and right pixels of the image respectively. To do this, the parent image is rotated 90º and the Fourier coefficients are calulated for the new rows that p_0 and p_1 lie on. The code then goes along the row p_0 is on until the coditions for p_1 are met for a point on the row. The code then checks the position on the row with p_1 on with the same offset as the located p_2 to check that the value is as expected. If it is then the poition of the sub-image is assumed to have been found and is returned. If this is not met, then the code disregards the location of p_2 and continues searching for an appropriate p_2 for which the coditions for p_3 are met.