Clear Cut: Algorithm Design

September 30, 2018

Contents

1 Input data

The input data is an image of arbitrary size, $M \times N$, where M is the horizontal dimension and N is the vertical dimension of the image. Most images have three channels (RGB = "Red-Green-Blue"), this means an array with three $M \times N$ arrays. Images with exif data are checked and modified so that the image is imported in the intended orientation, e.g. in case the image is a photo taken from a phone camera.

1.1 Image size reduction

To improve the efficiency of the edge detection algorithm, the image size is reduced to that the average image dimension is less than 500 pixels in length. The final image size to be thrown into the algorithm is $M_{\text{eff}} \times N_{\text{eff}}$, where

$$\frac{M_{\text{eff}} N_{\text{eff}}}{2} < 500 \text{ pxl}. \tag{1}$$

The image is currently reduced slowly by max pooling. This relies on calculating the smallest kernel size and repeating the max pooling process until the condition in Equation 1 is satisfied. This can be implemented more effectively by calculating the smallest kernel that would satisfy the condition in Equation 1 after just one implementation of max pooling.

As it currently exists, the process consists of determining the smallest number that divides the height (width) of the image to return an integer. If there is no such factor, i.e. the height (width) is a prime number, the image is cropped by removing the single pixel row (column) at the M_{eff}^{th} (N_{eff}^{th}) index. By definition this would result in that image dimension being divisible by 2, which is then the number of pixels of the smallest kernel in that dimension. Throughout the image reduction process, the values (M_{eff}, N_{eff}, M_{kernel}, N_{kernel}) are stored in a Python dictionary to keep a record of the image reduction history.

2 Edge detection procedure

2.1 The gradient image

In order to determine the edges of an image, we must choose a criteria to distinguishing a single pixel located at (i_0, j_0) as being part of "an edge". We would refer to such pixels as "edge pixels" or "non-edge pixels" in future. This is achieved by mathematically comparing its value to those of its surrounding pixels. The simplest method to adopt is to consider only the pixels neighbouring pixels (i, j), i.e. any combination of pixels that satisfy

$$i_0 - 1 \leq i \leq i_0 + 1, \text{ and } j_0 - 1 \leq j \leq j_0 + 1. \tag{2}$$

A more advanced technique could extend the range of neighbouring pixels. For a pixel not located on the perimeter of the image, it would have 8 neighbouring pixels.

2.2 Dealing with channels

Each of the RGB channels have the edge detection algorithm applied to them.

DevOps	Best practices and administration of the Atlassian Suite, e.g. Jira, Confluence
Non-functional Testing	Performance testing of a variety of applications (e.g. HPE LoadRunner, JMeter)
Automation/Scripting	Developed two software solutions using Java SE 8 and Excel's VBA macro
API Programming	Utilising RESTful services from online documentation within scripts
Deep Learning	Using Google's TensorFlow framework (python) for predictive analytics