Assigment 4 Addendum

Department of Electrical and Computer Engineering
Ryerson University

ELE 882 – Winter 2014

1 Oilify Effect

The "oilify" effect is a slight modification to the Oilify filter[1] in GIMP. The effect takes an image and makes it look very similar to an oil painting. The way it does this is somewhat roundabout and is based on the local image statistics. An example of the effect is shown in Figure 1. The meaning of the parameters will be clear by the end of the discussion.

1.1 IMPORTANT

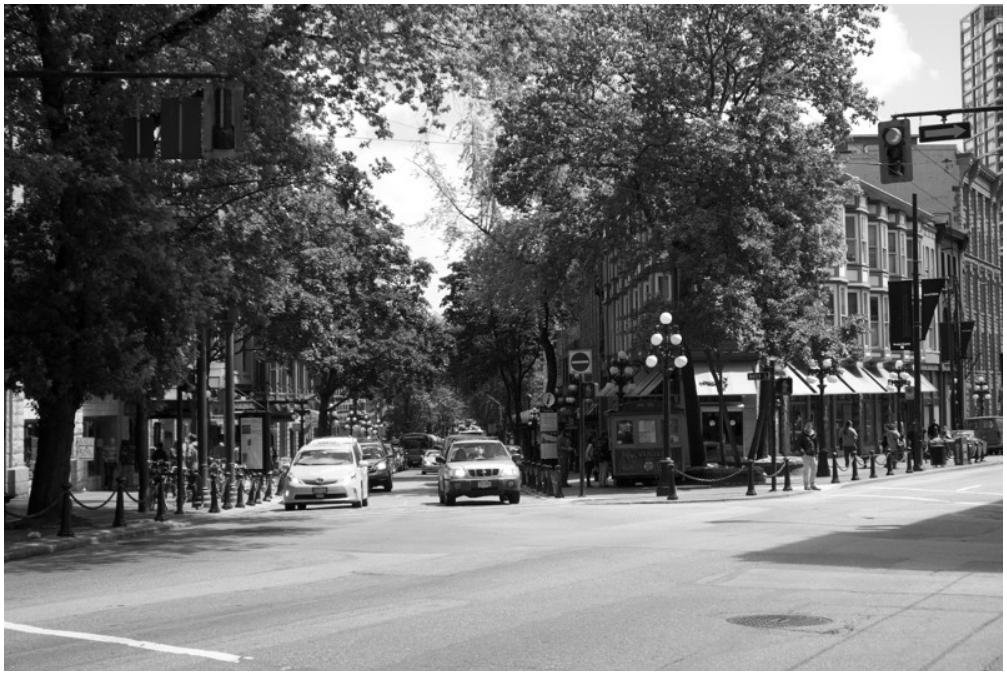
Before the method is described, a lot of what is in this document was from reverse engineering the plugin's source code[2]. Other examples of how the oilify effect works can be easily found online. Examining other implementations can often be useful in understanding how a particular algorithm works. **However, this does not mean you can copy the work verbatim!** Aside from the obvious plagiarism and copyright issues, simply copying and pasting code is *never* a good idea. There are always hidden assumptions behind how some piece of code works and if you are not making those same assumptions, you will be in for some nasty bugs. It is much better to understand how something works and then move on from there.

1.2 Algorithm

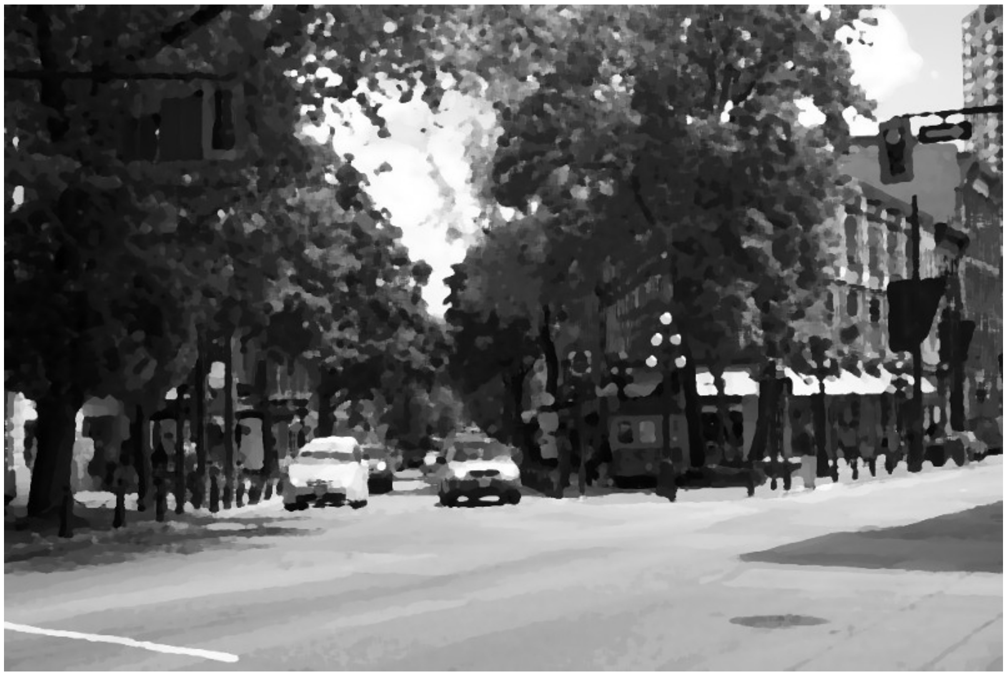
Before explaining the algorithm, first, let $h[n]$ be the *local* N-bin intensity histogram around the pixel at (x, y). The values for $h[n]$ are collected from an $R \times R$ neighbourhood around the pixel. Finally, let $acc[n]$ be a length N array, originally all zeros, that is used to store the accumulated (summed) pixel values. The $acc[n]$ array has a one-to-one-mapping with the histogram bins.

The entire oilify algorithm is presented as pseudocode in Algorithm 1. The algorithm is an example of a non-linear, adaptive filter where the filtering kernel changes depending on the local image content. Here, it uses the local histogram to decide which pixel value is most dominant and then calculates the output value as a weighted sum of all values with respect to the most occurring one.

[1] `http://docs.gimp.org/2.8/en/plug-in-oilify.html`
[2] `https://git.gnome.org/browse/gimp/tree/plug-ins/common/oilify.c?h=gimp-2-8`



(a) Original Image



(b) “Oilified” Image

Figure 1: An image that has been oilified using the following parameters: $N = 16$, $R = 3$ and $\gamma = 10$.

How exactly does it work? Consider the second loop (lines $11-15$). If translated into a mathematical expression then it is

$$I'(x,y) = \frac{1}{K}\sum_{i=0}^{N-1}\left(\frac{h[i]}{h_{max}}\right)^{\gamma}\frac{acc[i]}{h[i]}, \tag{1}$$

where K is the normalization constant

$$K = \sum_{i=0}^{N-1}\left(\frac{h[i]}{h_{max}}\right)^{\gamma}. \tag{2}$$

Recall that a histogram *counts* the number of times something occurred. So, if a pixel with value i occurred M times, then the value of $h[i] = M$. Therefore, inside of the accumulator array, $acc[i] = Mi$ because every time that value is encounter it is added into the appropriate bin (line 6). Dividing $acc[i]$ by $h[i]$ then removes that scaling factor of M and provides us with the original value of i. If $N = 256$ then we don't need to do this but if $N < 256$ then we have to because each histogram bin now maps to *multiple* possible intensity values. This allows us to create an average intensity for each bin.

Therefore, reading (1) we can see that each possible pixel value in the histogram is weighted according to which value occurred most often. The purpose of the exponent is to control how much influence the strong value has. If γ is large then it forces the weights for all values *except* for the most common one to be close to zero. The output colour will most likely be the most occurring value. Conversely, if γ is small it then forces the less common values to have more weight. In this case, the output value will most likely be a mixture (average) of all the possible intensity values.

One final point on finding the correct histogram bin. Before, we would assume that a histogram has 256 bins so there was a nice, one-to-one mapping between the histogram bins and the intensity values. Now, though, we are *quantizing* the intensity values so that there fewer possible values. Because we are quantizing uniformly, all we really need to do is to scale the input intensity and then round to the nearest integer. The easiest was to do this is to just divide the current intensity value by the largest possible value[3]. This provides us with a value between 0 and 1. We then multiply by the largest possible bin index, $N-1$, and cast to an integer value (mathematically equivalent to a floor operation). This is exactly what is being done on line 4.

[3] 255 for an `unsigned char` image. This is unnecessary for a `double` image because the maximum value is already 1.

Algorithm 1 Oilify Algorithm

```
procedure $\textsc{Oilify}(I, x, y, N, R, \gamma)$
    for $x_i = x - R$ to $x + R$ do
        for $y_i = y - R$ to $y + R$ do
            $i \leftarrow \lfloor (I(x_i, y_i)/I_{max}) \cdot (N - 1) \rfloor$        ▷ Histogram bin index.
            $h[i] \leftarrow h[i] + 1$
            $acc[i] \leftarrow acc[i] + I(x_i, y_i)$
        end for
    end for
    $h_{max} \leftarrow \textsc{Max}(h)$        ▷ Maximum value in the histogram.
    $A, B \leftarrow 0$        ▷ Initialize $A$ and $B$ and set to zero.
    for $i = 0$ to $N - 1$ do
        $w \leftarrow (h[i]/h_{max})^{\gamma}$
        $B \leftarrow B + w$
        $A \leftarrow A + w \cdot (acc[i]/h[i])$
    end for
    return $A/B$
end procedure
```