An Image Filter for Counting Pixel Neighbors

Release 0.01

Robert Tamburo[1]

May 24, 2011

[1]robert.tamburo@gmail.com

Abstract

This papers describes an image filter that counts the number of neighbors a pixel has storing that value at the pixel's image index. Functionality is provided to 1) adjust the neighborhood size, 2) count only those pixels within a specific value range, and 3) only record a neighbor count for specific pixels of interest.

This paper is accompanied with source code for the filter and test, test images and parameters, and expected output images.

Contents

1 Implementation of Algorithm

`CountNeighborsImageFilter` counts the number of neighbors a pixel has and stores the count at the pixel's image index. The entire is image is iterated over and at each pixel a neighborhood, centered at the current pixel, is constructed. Pixels within the neighborhood are counted if the user-defined criteria is satisfied. The count does not include the pixel of interest. The size of the neighborhood can be set with the function `SetRadius()`, which has a default setting of 1 to contain all fully connected pixels. Several options for count inclusion criteria are available:

- `SetCountAtValue()`: Counts all pixels equivalent to specified value.

- `SetCountAboveValue()`: Counts all pixels above a specified value.
- `SetCountBelowValue()`: Counts all pixels between a specified value.
- `SetCountBetweenValue()`: Counts all pixels between two specified values.
- `SetCountNonZero()`: Counts all non-zero pixels (default).

`SetValueOfInterest()` allows the user to only count neighbors for pixels of a specific value, which is disabled by default

2 Example Usage

This filter requires setting an input image `SetInput()`, the radius of the neighborhood with `SetRadius()`, the counting strategy, and the option to count neighbors for specific pixels. Example usage is shown below.

```
typedef itk::CountNeighborsImageFilter<ImageType, ImageType> FilterType;
FilterType::Pointer filter = FilterType::New();
filter->SetInput(inputImage);
ImageType::SizeType radius;
radius.Fill(1);
filter->SetRadius(radius); // set radius of neighborhood
filter->SetCountNonZero(1); // set counting strategy
filter->SetValueOfInterest(255); // set pixel of interest
filter->Update();  // execute filter
```

`itkCountNeighborsImageFilterTest.cxx` contains test examples with 4 different sets of parameters for an input image (Fig. 1). The input image contains hand drawn patterns of value 127, 195, and 255.

The first set of parameters counts all non-zero pixels within a neighborhood of radius one and centered around pixels with a value of 255 (Fig. 2).

The second set of parameters counts all pixels above 254 within a neighborhood of radius one and centered around any pixel (Fig. 3).

The third set of parameters counts all pixels between 126 and 196 within a neighborhood of radius two and centered around any pixel (Fig. 4).

The fourth set of parameters counts only pixels with a value of 255 within a neighborhood of radius one and centered around pixels with a value of 127 (Fig. 5).

3 Software Used

This filter was developed on a Windows 7 64-bit computer. It has been successfully tested with ITK version 3.18.0, MinGW version 5.1.6, and CMake version 2.8.2 (Windows binary).

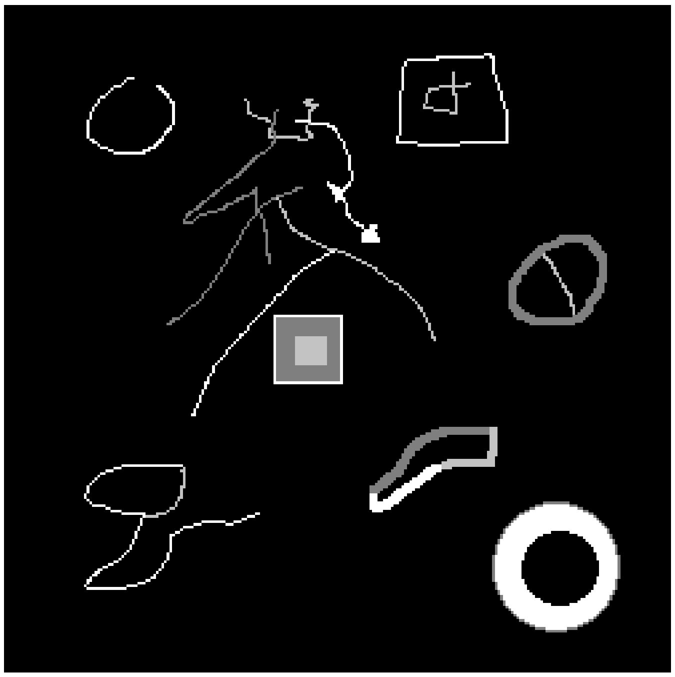

Figure 1: The input image `input.png` for this example.

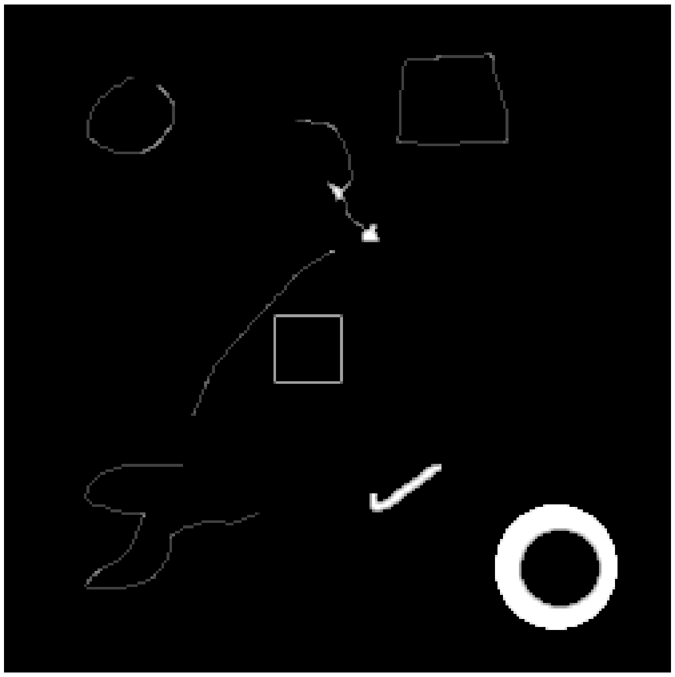

Figure 2: The output image produced with the first set of parameters. Intensities scaled in a viewer for visualization purposes

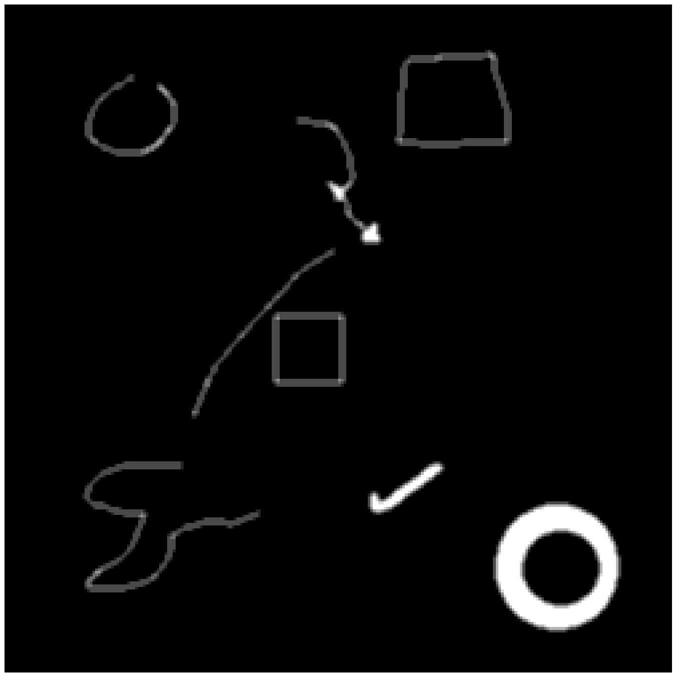

Figure 3: The output image produced with the second set of parameters. Intensities scaled in a viewer for visualization purposes

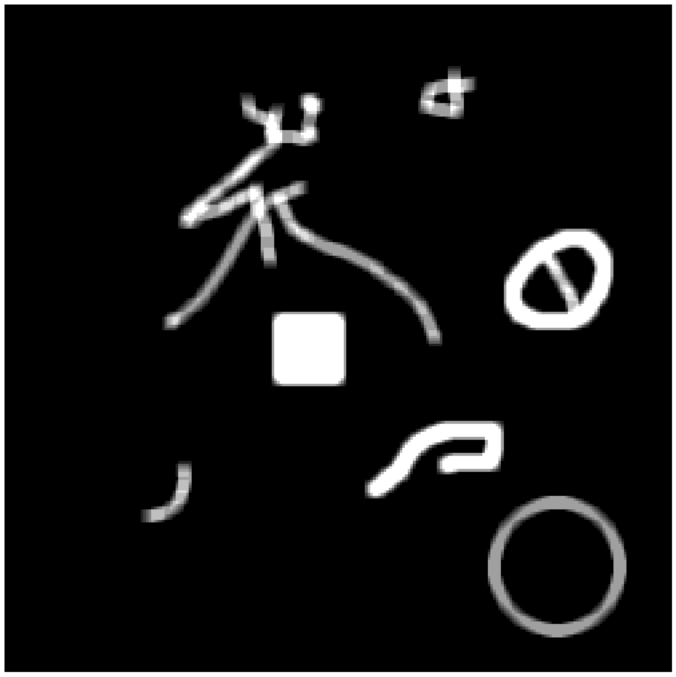

Figure 4: The output image produced with the third set of parameters. Intensities scaled in a viewer for visualization purposes

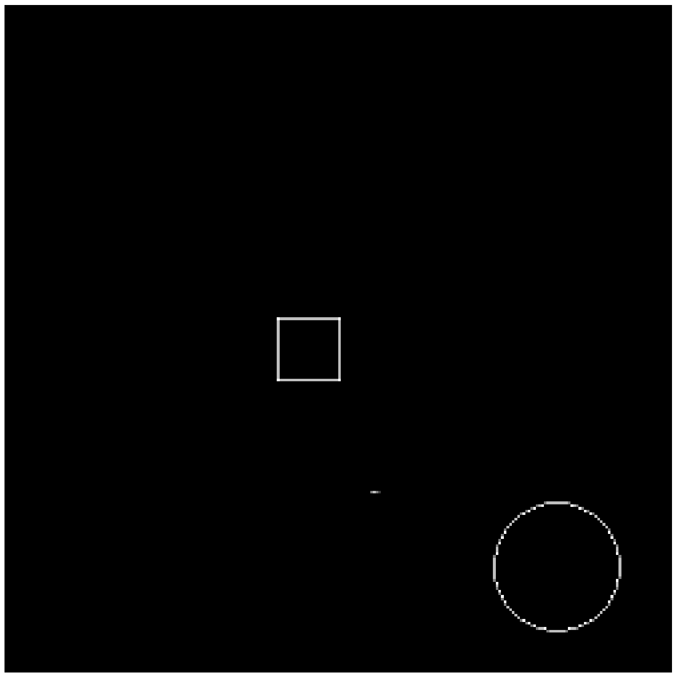

Figure 5: The output image produced with the fourth set of parameters. Intensities scaled in a viewer for visualization purposes