Project 1 - Edge Detection

Henrique Machado Goncalves - RA 169621
h169621@dac.unicamp.br
Lucas Fernandes André - RA 182495
l182495@dac.unicamp.br
Tiago Petená - RA 187700
t187700@dac.unicamp.br

***Abstract*—Edge detection is an important early vision task. Edges serve as inputs for several applications. In this project, we re-interpreted the edge detection task as a classification problem, as proposed by Martin, Fowlkes, and Malik and others long ago.**

I. Introduction

The work in this project can be divided can be summarized as follows:We created a Filter Bank containing six different orientations, with 3 scales and based on that we define the odd and even filters based on the first and second derivatives of a Gaussian filter

II. Filter Bank

Before creating any filter, we must first select the Scales that are going to be used, and why we chose them to be this way: $\sigma = \{0.5, 0.75, 0.9\}$, we took into account how many pixels where left inside the σ for each image and arrived at the scale values after fine-tuning. To create the filter bank, we first used the approximation of the oriented energy that is given by:

$$OE_{\theta,\sigma} = (I * f^e{}_{\theta}, \sigma)^2 + (I * f^o_{\theta,\sigma})^2$$

This generated 18 filters, since we have 6 different θ and 3 different σ Afterwards, we defined the Gaussian for each of the 3 different scales based on:

$$G_\sigma = \frac{1}{2\pi\sigma^2} * e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

And then, we used the Sobel filter to be approximation of the first derivative which makes our Odd Filters, assuming A is the image:

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * A \quad and \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$

$$G = \sqrt{G_x^2 + G_y^2}$$

Where * denotes the 2 dimensional convolution operation And we used Laplacian for our second derivative and even filters

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

. In order to make the filters based on orientation of θ, we used the following formula

$$cos(\theta)A_x + sin(\theta)A_y = A_\theta$$

Where A is the Filter before being flipped using θ.

thus, creating 36 other filters and at last, we created a center surrounded filter for each scale based on:

$$C_\sigma = G(\sigma) - G(2\sigma)$$

thus making 57 different filters for our bank

III. Edge Classification

The main focus of the project is the Contour Problem, which is considered a binary classification problem, in order to solve it, we attacked the problem using Machine Learning and by using ”The Berkeley Segmentation Dataset and Benchmark” for the train and test images. The selector chosen was the Stochastic Gradient Descent Classifier from scikit-learn.

A. Features

In our edge classification problem, there are several ways to attack the problem, therefore we devised 4 sets of features based on our Filter Bank, which are as follows:

- Create the filters based on the oriented energy per scale, plus the center-surrounded one.
- Create the filters based on the filter bank.
- Create the filters based on the even filters, plus the center-surrounded one.
- Create the filters based on the odd filters, plus the center-surrounded one.

Therefore we end up with 4 algorithms that are trained in different ways so that we are able to choose the best set of features that give the best Edge Detection.

IV. Experiment

The results of the SGD classifier based on the set of feature are as follows: From that, we arrive at the average score that

Feature	Feature 1	Feature 2	Feature 3	Feature 4
Score	0.900	0.921	0.913	0.931

TABLE I
Table 1: Classifiers based on scores

is 0.916

A. Convolutions

The Convolution function that was created by the group as specified on the project specification can be found here in the src directory. It is a very simple but important function for the work of this project.

V. Conclusion

After analyzing the score from each of the different set of features, we can conclude that the best SGD Classifier is the one that unites the odd filters with the center-surrounded ones. Using a lot of flipped Sobel Filters with different scales.

Furthermore, we can analyse that the feature set composed mainly of Sobel Filters is the better at edge detection form our dataset, which was expected from our readings on the web.That happens because the Sobel detects edges are where the gradient magnitude is high.This makes it more sensitive to diagonal edge than horizontal and vertical edges and by using the Sobel filters in different angles, we were able to have sets of features that were more sensitive in multiple diagonals, hence why the results are better.

The reason why the Laplacian filters weren't as good is because they are very noise sensitive because it detects edges based on significant differences in the second derivatives even thought it is used in more sophisticated edge detection methods.