**EE569 Introduction to Digital Image Processing**

**Homework Report #2**

**Name:** Boyang Xiao  **USC ID:** 3326730274  **Email:** boyangxi@usc.edu

# Problem 1: Edge Detection

## Part a: Sobel Edge Detection

### Abstract and Motivation

Edge detection is a very basic but important topic in Image processing and Computer Version. To find out the part where the illumination massively varies, techniques of Edge Detection can detect the edges in images, which is very obvious for human’s vision system but not for the computer. Therefore, techniques of Edge Detection can provide higher-level information about the images to support higher-level techniques such as Feature detection and Semantic segmentation.

In this part, one of the most intuitive methods of Edge Detection, the Sobel Edge Detection, will be introduced and implemented.

### Approaches and Procedures

To find out the areas where the brightness level of pixels varies a lot, Sobel Edge Detection firstly calculated the gradient of pixel values in both directions of X and Y (vertically and horizontally). We can use some specialized filter to get calculate the x-gradient and y-gradient. A simple pair of filters is shown as Figure 1. below.

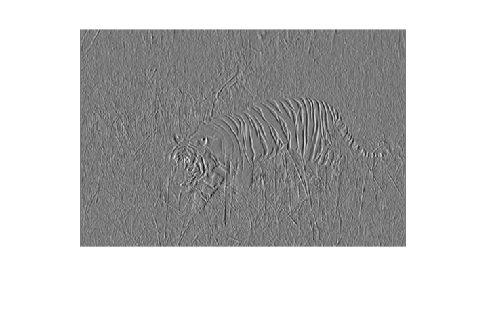
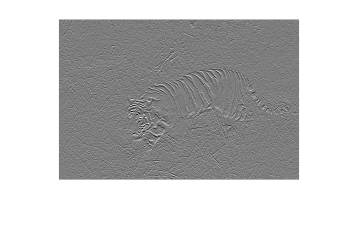


Figure 1: (a) The filter for x-gradient (b)the filter for y-gradient

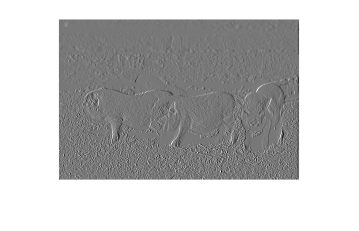
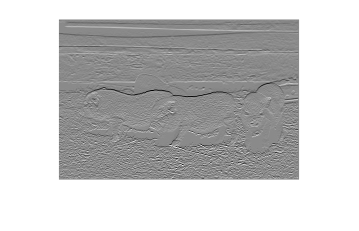
Then the geometric mean of each pixel’s x-gradient and y-gradient is calculated and we call it the map of gradient **magnitude**. If we want to get a binary edge map, which means the edge pixels are assigned a “255” value and the non-edge pixels are assigned a “0” value, we can choose a threshold to binarize the gradient magnitude map and get an image only contains black and white pixels, where black pixels stand for non-edge areas and white pixels stand for edge areas.

### Experimental Results

**Step 1:** The results of x-gradient map and y-gradient maps for Tiger.raw and Pig.raw are shown in Figure 2. below.

(a) x-gradient map for Tiger.raw (b) y-gradient map for Tiger.raw

(c) x-gradient map for Pig.raw (d) y-gradient map for Pig.raw

Figure 2: x-gradient and y-gradient maps for Tiger.raw and Pig.raw

**Step 2:** The results of magnitude maps of Tiger.raw and Pig.raw are shown in Figure 3 below.

(a) Magnitude map for Tiger.raw (b) Magnitude map for Pig.raw

Figure 3: Magnitude maps for Tiger.raw (a) and Pig.raw (b)

**Step 3:** The binary edge maps of Tiger.raw and Pig.raw are shown in Figure 4 below. The percentage threshold for Tiger.raw is set as 60.9% and the percentage threshold for Pig.raw is set as 59%.

(a) Edge map for Tiger.raw (b) Edge map for Pig.raw

Figure 4: Binarized edge maps for Tiger.raw (a) and Pig.raw (b)

Although the basic edges are detected by Sobel Edge Detection, but the results are not that good. There are a lot of false alarms, such as the skin patterns of the tiger and the grassy ground around the pigs. Also, there are a lot of missed detection, such as the pigs’ backs and the tiger’s face and tail. When tuning the threshold values, it’s very difficult to optimize the trade-off. If I want to remove more redundant details, some of the edges are also removed, and vice versa. It’s fair enough to say that Sobel Edge Detection is the simplest method and that it can only obtain very limited results.

### Non-programming Questions

All of the questions HAVE been answered above.

## Part b: Canny Edge Detector

### Abstract and Motivation

From the Edge Detection results above, the Sobel Edge Detector and some other very traditional edge detectors can detect pixels in areas of brightness discontinuities. But these when marking out all of these pixels, the whole contours are discrete. However, the real contours, which are actually the real meaningful information about the image, are continuous and connected to each other. Therefore, a more advanced detector, the Canny Edge Detector, who consider the connectivity and continuity of the contours, will be introduced in this part.

### Approaches and procedures

The procedures of Canny Edge Detector is divided into three steps:

* + Gaussian filtering.
  + Non-maximum Suppression.
  + Hysteresis Thresholding.

**Step 1: Gaussian Filtering:** This step is as same as the traditional edge detectors. It applies a Gaussian filter to the image and get the magnitude map to remove the noise and to mark the areas where the brightness massively changes. Both the gradient magnitude values and gradient direction values (be rounded to simply 4 directions) for each pixel will be recorded.

形状

描述已自动生成

Figure 5: Non-maximum Suppression Operation

**Step 2: Non-maximum Suppression:** When getting the gradient magnitude and gradient directions for each pixel as mentioned above, the very next step is to apply a non-maximum suppression to every pixel. For every pixel, the gradient magnitude value will be compared to its most adjacent two neighbors, as shown in Figure 5. And the two neighbors are picked according to this central pixel’s gradient direction value. For example, if the gradient direction of pixel (x, y) is 135, the neighboring pixels are picked from its upper-left corner and right-down corner, the pixel and .

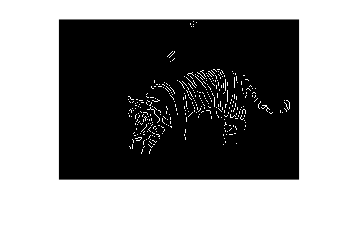
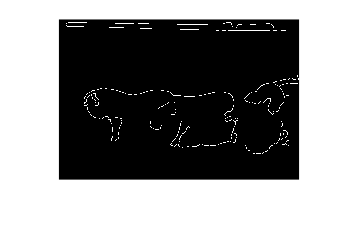
After selecting the neighbors, the Non-maximum Suppression operation will compare these three pixels’ gradient magnitude values to preserve the maximum one and suppress the other lower ones. The algorithm is as below:

After application of non-maximum suppression, remaining edge pixels provide a more accurate representation of real edges in an image.

**Step 3: Hysteresis Thresholding:** Even the Non-maximum Suppression can provide an accurate representation of contours of the image, there will still be some unsatisfying parts caused by noises and color variation. To get a more accurate contour of the image, the Canny Edge Detector uses double threshold to detect the real edges and the weak edges. If the pixel is above the higher threshold, the pixel value gradient magnitude is preserved. And if the pixel is between the upper bound threshold and the lower bound threshold, the pixel is representing a weak edge. The pixels below the lower bound threshold will all be suppressed. Whether to preserve the weak edges will be decided by their connectivity with other strong edges.

### Experimental results

After applying the Canny Edge Detector on MATLAB using function edge(…) with parameter “canny” in the Image processing toolbox, the detection results are shown as Figure 6 below. The two thresholds [Lower bound, Upper bound] set for Tiger.raw are **[0.16, 0.42]**, and the two thresholds set for Pig.raw are **[0.11, 0.51]**.

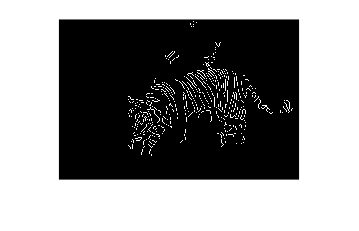
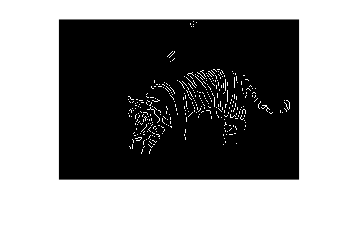
(a) Canny detection for Tiger.raw (b) Canny detection for Pig.raw

Figure 6: Canny Edge Detector results for Tiger.raw (a) and Pig.raw (b)

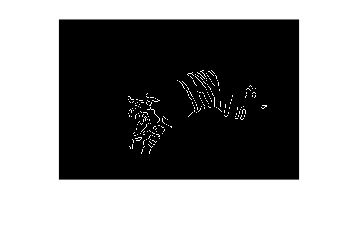
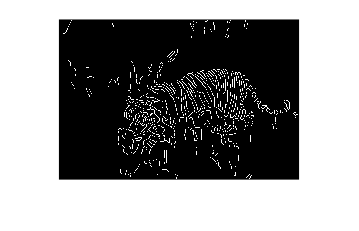
From the results above, Canny Edge Detector gets very clear and continuous contours for both the images instead of some discrete scatter points of the edge pixels. After selecting the two threshold values distinctively for each image, the noises and color variation parts are effectively removed and the true edges of the images are preserved well.

### Discussions

In this section, the threshold values selection and its influence will be discussed. The interface for these two thresholds in MATLAB function edge(…) is defined as a parameter vector filled in a two-value vector [Lower bound, Upper bound]. The Lower bound value should not be bigger than the Upper bound value.

(a) [0.05, 0.42] (b) [0.16, 0.42] (optimized)

(c) [0.16, 0.59] (b) [0.16, 0.25]

Figure 7: Canny Edge Detector results with different threshold parameters [Lower bound, Upper bound]

In Figure 7, several threshold values are used in the Canny Edge Detector and the results are displayed. If the lower bound value is too low, as shown in Figure 7(a), the whole detection results will not be greatly influenced compared to Figure 7(b) which can be the optimized setting for this specific image, but some parts of “false edges” will be counted into the contours, like the areas above the tiger’s back. If the upper bound is too large, as shown in Figure 7(c), on the other side, some of the “real edges” will be missed, such as the tiger’s chest and tail. And if the upper bound is too small, as shown in Figure 7(d), many of the noises and color variations will be counted into the contours, such as the grassy ground texture, since the Upper bound threshold is the main filter to remove these disturbances.

In conclusion, the two threshold values in Canny Edge Detector can both influence the final results in their own ways. The Upper bound value can determine how much of the noises and color variation disturbances are removed, while the lower bound value can determine how much of the “weak edges” are preserved. For each different scenario and each different image, these two parameters should be selected accordingly and carefully.

### Non-programming questions

Q1 and Q2 are answered in section 1.2.2.

Q3 is answered in section 1.2.4.

## Part c: Structured Edge

### Abstract and motivation

All the methods introduced above is assumption based, which means that we assume the features of edges ahead and find out the areas that match those features. However, this kind of methods can make a lot of mistakes since many edges are not only depending on the brightness variation but also depending on the humans’ understanding. Therefore, the learning-based algorithms are introduced to deal with this situation. The most famous learning-based edge detection methods are Sketch Token (ST) and Structured Edge (SE). The SE detector is the enhanced version of the SE detector, and it will be introduced and tested in this part.

### Approaches and procedures

The whole procedure for SE detection is as shown in Figure 8. Each step will be specified below.

* **Step 1. Sampling:** To train the model, a large number of image samples should be collected. The sample images should contain clear edges areas and also non-edge areas. After obtaining the sample images, the images are partitioned into same-sized patches (e.g. 35\*35 pxls patches). And all the patches are labeled to match a certain type of edges. This is the preparation for the training data set.



Figure 8: Flowchart for SE detection procedure

* **Step 2. Feature Extraction:** Since every layer of decision tree is separated according to different features, the features of samples should be extracted before training. Each sample patch can be converted to different color spaces, like CIE-LUV space, and it can also be converted to magnitude map and oriented magnitude maps. Each pixel of these different maps can be a feature for a single sample patch. Besides, the sample patches can also be down-sample to smaller size patches and the variation level of each pair of pixels can also be a feature. After extracting all these features, each single patch can have more than 20,000 features.
* **Step 3. Training:** The main part of the SE detector is the Random Forest, which is made up of a certain number of decision tree. Every decision tree is a binary tree and the whole sample patches are separated into two groups level by level according to the selected features and the corresponding thresholds. Every leaf node of each decision tree should contain all the sample patches that contain the same kind of edges. The decision tree numbers and depths are setup ahead of time and part of samples are used to validate the accuracy of the training process. After the validation accuracy is high enough, the training process is finished.
* **Step 4.** **Edge Prediction:** After the training, the model can be used to detect the edge. When obtaining a new image, the image is firstly be portioned into patches that have the same size as the training data set. Then every patch is input to the random forest model and each decision tree can obtain a classification result, which stands for a certain type of edge. The probability of each type of edge can be calculated from the trees’ results and the highest probability is the final results for this single patch. After all the patches get a probability result, the whole probability edge map is obtained.
* **Step 5.** **Binarization:** In some cases, the probability edge map might not be needed but the binary edge map is needed instead. We can set a threshold to classify the edge pixels and non-edge pixels. If the pixel is an edge, the pixel is assigned a 255 value. Otherwise, the pixel is set a 0 value.

After all the steps above, the edges of an image should be extracted and the results are fully based on samples and training.

### Experimental results

The SE edge detection results are shown as Figure 9 below. The parameters for detection model is set as below. The functions of each parameter has been specified below, and the parameters set below can obtain a high accuracy while the speed of calculation is also preserved.

model.opts.multiscale=0; % use multiscale input image to obtain accuary

model.opts.sharpen=2; % sharpen the output results

model.opts.nTreesEval=4; % for top speed set nTreesEval=1

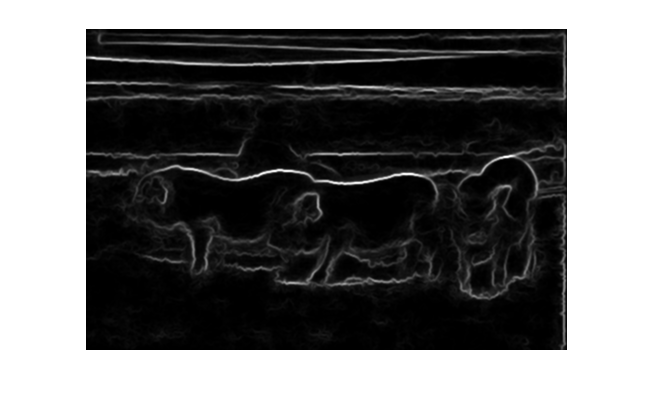
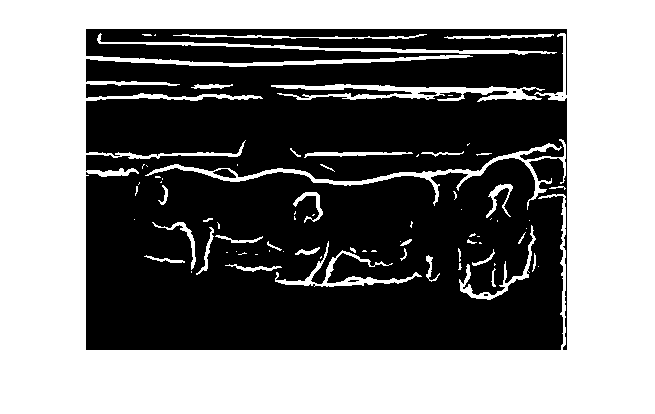
model.opts.nThreads=4; % max number threads for evaluation

model.opts.nms=0; % set to true to enable non-maximum suppression

Besides, the threshold value of both image is set as 18% of the max value to obtain the binarized edge map.

(a) Probability edge map for Tiger.raw (b) Binary edge map for Tiger.raw

(c) Probability edge map for Pig.raw (d) Binary edge map for Pig.raw

Figure 9: SE detection results for Tiger.raw and Pig.raw

As the results shown in Figure 9, the SE detector can obtain a very good result. The edges of both images are detected accurately while the textures and grassy ground are removed perfectly. Since, the SE detector is training-based, it can handle well with the weak edges and the false alarm parts.

### Non-programming questions

**Q1: Please digest the SE detection algorithm. Summarize it with a flow chart and explain it in your own words.**

**A1:** This question has been answered in section 1.3.2 Approaches and procedures

**Q2: Explain the process of decision tree construction and the principle of the RF classifier.**

**A2:** Random Forest (RF) model is a typical kind of ensemble training, which means it use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. In RF model, it uses multiple decision trees to do the classification and then obtain the results synthesizing the results from all the decision trees. If the classification is discrete, the RF model uses the mode from all the results. And if the classification is regression, the RF model uses mean value of all the results to calculate the final result.

As for the decision tree, every decision tree is a binary tree. The root node of the decision tree contain all the samples, and with every classification, the whole simples from the root node are separated into two groups, which contains different sample that have different features according to the features selected for this level. As the decision tree grows deeper and larger, the classification is finer and finer and finally the whole samples are classified into the number of classes as we expected. Since every tree can get different results as the selected features and growing orders are all different, the RF model uses the mode of the results from all the trees and it preserve the accuracy and prevent the overfitting flaw.

Mode specifications of RF model used in SE detector are in section 1.3.2 Approaches and procedures.