Abstract

Recently, a wide range of applications of image segmentation are arisen, from automatic driving to medical diagnosis. Bottom-up and top-down are two segmentation approaches. The bottom-up approaches are based on analyzing the discontinuities in image features (e.g. texture, color and brightness) and the top-down approaches are guided by object category knowledges. The objective of this project is to explore ways to combine the top-down approach with a bottom-up approach to achieve better performance that cannot be obtained by either the bottom-up or top-down segmentation alone. To achieve this aim, one existing top-down segmentation approach called SegNet and one bottom-up segmentation approach named *gPb* were implemented through this project. Moreover, one bottom-up approach using region-growing and canny were designed. After implementing these approaches alone, some combining methods and final combined results were illustrated. All the algorithms used in this project were evaluated via the Berkeley Segmentation Dataset and Benchmark.

Acknowledgment

I would like to extend my sincerest gratitude to my supervisor, Dr Michael Spratling, of the Department of Informatics, King’s College London, for his expert guidance, patience and constant support during the whole period of the project.

Many thanks are also given to my parents and close friends, for their encouragement and support during this intense period of life.

Furthermore, I also would like to thank all teachers and staffs in the Department of Informatics, King’s College London.

Contents:

List of figures and tables:

Figures:

Tables:

1 Chapter 1 Introduction

1.1 Overview

Image segmentation is a process of partitioning an image into multiple parts which are corresponding to different object areas[1]. In recent years, a wide range of applications of image segmentation are arisen, from traffic control (e.g. cameras can catch all scenes of road and analyze the license numbers of cars for traffic violation) to medical diagnosis (e.g. from processing Computed Tomography (CT) results to identify the location of disease[2]).

Image segmentation can be implemented in many approaches - for example, an image could be split into partitions via analyzing the discontinuities in image features (e.g. texture, color and brightness), or employ deep learning architectures to split images based on learning object category knowledge. Although many approaches have been explored which attempt to solve the image segmentation problems, each method has advantages and has some shortcomings as well. Therefore, there is no existing algorithm for simply segmenting any image extremely perfect as human vision. This project will implement some algorithms for image segmentation based on analyzing image feature discontinuities and object category knowledge respectively. Then, combining these two approaches together to achieve better performance than singly using one of these ways. Standard image segmentation benchmarks utilized in the project to evaluate the performance of each algorithm.

1.2 Background

To drew the problems and project objectives, some background information and concepts are necessary to be explicated first. Image segmentation is a high-level image process and has been studied in past two decades[3]. As mentioned above, image segmentation is a process of splitting digital images into salient image regions[4]. These regions are corresponding to individual objects or surfaces[4]. Image segmentation has two main objectives[5]. Locating objects and decomposing images into parts in any complicate image for further analysis is the first objectives[5]. For example, some algorithms have been devised for detecting human faces from color images. The second goal of image segmentation is to change the representation of an image[5]. The images could be performed in more efficient or more meaningful ways to make the analyzation simpler[6]. Moreover, in recent years, image segmentation has many new applications in several fields. From traffic problems like autonomous driving to recognition tasks such as fingerprint detection. It seems that image segmentation technique can enhance the quality of human life and offer tremendous convenience. Thus, this project is focusing on image segmentation.

The process of segmentation can be performed in many ways. Traditionally, features associated with image pixels are always utilized for grouping the objects and determining the locations of boundaries between objects[6]. To be specific, pixels of the same kind of objects should have similar appearance and hence should be grouped together. In addition, image features may be discontinuities at the locations of boundaries between objects. To sum up, image segmentation is a process of analyzing each pixel in an image and marking a label on it, then grouping pixels with the same label into a group[6]. The adjacent areas should have significant differences on some image features - for example, brightness, texture, or colors[1]. The pixels grouped in the same region sharing the similar characteristic[1]. Many methods based on analyzing image feature discontinuities for image segmentation are already exist, such as clustering method, thresholding method and region growing method. Moreover, all these methods are bottom-up image segmentation. The bottom-up approach is an information processing strategy uses image-based criteria, that is, the image properties[7]. The pixels are likely to belong together for they are locally coherent[7].

In addition, another approach to image segmentation, the top-down segmentation, is to detect objects in an image uses object knowledge learned from examples[7]. This approach utilizes prior experience or internal knowledge about an object, like its possible shape, or texture, to direct the segmentation process[7]. Human vision[8][9] suggests that this kind of object recognition method can facilitate the performance of segmentation. Therefore, many researches are focusing on top-down segmentation by employing deep learning architectures based on object knowledge recently.

However, both approaches have some difficulties to achieve excellent segmentation as human vision. For bottom-up approach, the main difficulty is that it is hard to provide accurate delineation of object boundaries[7], that is, an object might be merged with its background or splitter into multi-region. Furthermore, the main difficulty for top-down segmentation comes from the huge variability in the appearances of objects[7]. The merits and demerits of both bottom-up and top-down segmentations are demonstrated in Figure 1[7].

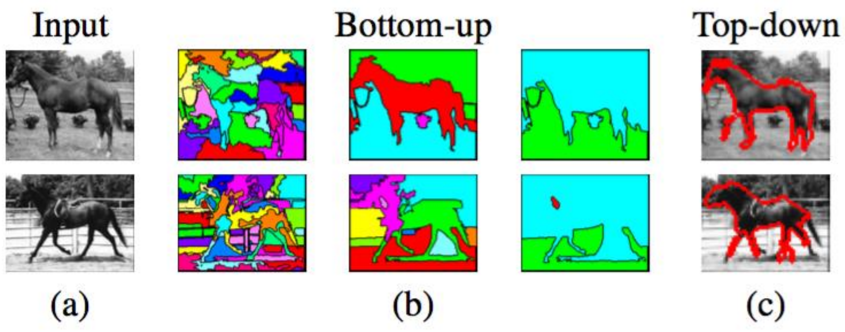


Figure 1[7]: The merits and demerits of bottom-up and top-down segmentations.
(a) Input images. (b) The bottom-up hierarchical segmentations at three different scales[10](the discontinuities of image features can be accurately detected but objects were segmented into several regions and parts of objects were merged within the background). (c) The results of top-down approach[11] (provides an approximation for the object segmentation but it has not find accurately image discontinuities).

This project implements one top-down segmentation approach provided by University of Cambridge: SegNet - a deep convolutional encoder-decoder architecture for image segmentation[12]. Moreover, two bottom-up segmentation approaches are also implemented: Globalized probability of boundary(gPb) and region growing. Then, combining the top-down approach with a bottom-up approach to achieve better performance that cannot be obtained by either the bottom-up or top-down segmentation alone. Thus, the combination offers a final segmentation that merges the merits of both methods. To be specific, the final segmentation results will contain the approximation of object boundaries produced by the top-down approach, and it also keeps consistent with image feature discontinuities found by bottom-up segmentation. Several combining results produced by some different ways will be compared using the standard image segmentation benchmarks.

1.3 Motivation and Significance

The reasons for choosing the project ‘Combining Object Knowledge and Image Feature Discontinuities for Image Segmentation’ are multiple. First, image segmentation is used in a wide range of fields recently. For example, self-driving vehicle is an extremely new artificial intelligent application to release people from driving, which requires the image segmentation technique to understand the street scenes. The algorithm is operated on low-power vehicles in real-time to control them drive automatically. This application based on image segmentation provides much convenience to human. Another field benefit from image segmentation is medical diagnosis. For more precise diagnosis and treatment, medical images processing became essential in recent years[13]. Moreover, the core of the medical images processing is image segmentation. Many medical techniques based on image segmentation were developed, like Computed Tomography(CT) and X-ray[13]. The benefits of these techniques are enormous, doctors can make diagnosis and therapy for patients more precisely due to the high precision of the digital information processing. Moreover, traffic police can benefit from the image segmentation technique as well. For cameras can act as an electronic police to monitor the road scenes and recognized the license numbers of cars which violate the traffic rules, such as furious driving. In conclusion, all these applications will improve the quality of life and release human from the heavy work.

Secondly, the precision of the image segmentation results is extremely important in many applications. For example, it is significant to have precise segmentation results in a self-driving system, because it is closely related to the human safety. Although some algorithms are already devised for this application, the performance of scene-understanding is looked forward to being more accuracy. Moreover, the precision of the image processing for medical diagnosis has high requirement either. Thus, this project will design more accuracy image segmentation by combining top-down and bottom-up approaches.

1.4 Research Aims

The main aim of the project is to find ways to combine the top-down and bottom-up segmentation approaches which can improve the performance of image segmentation for road scene images (because the top-down approach used in this project is focusing on rad scene understaning). To achieve this main aim, several sub-tasks need to be done:

- Implement the top-down segmentation algorithm: SegNet
- Test the performance of top-down segmentation algorithm using the standard image segmentation benchmarks.
- Implement the bottom-up segmentation algorithms
- Test the performance of bottom-up segmentation algorithm using the standard image segmentation benchmarks.
- Combining the top-down and bottom-up segmentation algorithms together to obtain a final segmentation results.
- Evaluate the performance of combined segmentation results using the standard image segmentation benchmarks to see whether it is better than either the bottom-up or top-down segmentation alone.

1.5 Organization

This paper will introduce the project from several aspects and is organized as follows.
Chapter 2 (Background) provides a critical review of the relevant literature about image segmentation, including the existing segmentation approaches for both bottom-up (based on image feature discontinuities) and top-down (based on object category knowledge) segmentation.

Chapter 3 (Design and Implementation) describes the requirement of the implementation of the algorithms and then presents the methodology for implementing the algorithm for bottom-up and top-down approaches respectively. It also explains the ways for combining bottom-up and top-down segmentations together to obtain a final segmentation result.

Chapter 4 (Evaluation) Evaluates the performances of image segmentation produced by either bottom-up or top-down approach alone using the standard image segmentation benchmarks. Using the same benchmark to evaluate the combined segmentation results as well. Then, compares these evaluation results.

Chapter 5 (Conclusion) concludes the achievement of the project. Moreover, what can be done in the further studies are roughly stated.

2 Chapter 2 Background

2.1 Overview

The previous chapter introduced image segmentation, bottom-up approach and top-down approach roughly. This chapter provides more detailed background for several necessary concepts used in this project. Some useful algorithms and existing methods for image segmentation will be stated clearly. Some of these algorithms will later be used in this project. Two bottom-up approaches (gPb and region growing) and one top-down approach(SegNet) which are mainly used in this project will be introduced more detailed. In addition, the pros and cons of these algorithms will be discussed as well.

2.2 Digital image Representation

To perform an image processing process like image segmentation, the representation of an image should be understood clearly. An image can be referred as a two-dimensional function, *f*(x,y), where elements x and y represent spatial coordinates (plane coordinates), and the amplitude of *f* for each pair of (x, y) is the intensity of the image at point (x, y)[14]. Moreover, each point (x, y) is defined as a pixel in an image. This concept is illustrated as Figure 2.

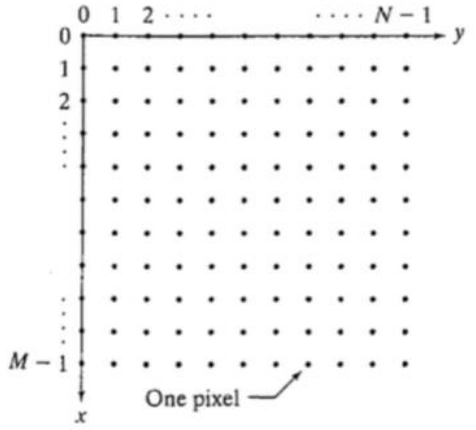


Figure 2[14]: Coordinate conventions used for images

The coordinate system shown in Figure 2 leads a significant representation method for a digital image, which is represents images as matrixes:

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Each element in this array is defined as a pixel. Furthermore, color images can be formed by a combination of several individual two-dimensional images and each individual two-dimensional image can be represented as a matrix. For example, a RGB color image includes three individual component images, which are red, green and blue component images[14].

2.3 Bottom-up segmentation

Bottom-up segmentation is based on analyzing each pixel in an image to find out pixels have similar appearance and features. Then the boundaries between objects can be located guided by the discontinuities in image feature. According to these boundaries, objects in an image should be grouped into several regions. Therefore, detect and localize boundaries precisely is vital in image segmentation process. Moreover, boundary is defined by David(2004) as "a contour in an image plane that represents a change in pixel ownership from one object or surface to another[15]". A great amount of boundary detectors guided by image features have been developed so far. The features for analyzing are multitudinous, such as brightness, color and texture of objects. Therefore, several image features utilized for bottom-up segmentation and some existing approaches use these features will be described in this section.

2.3.1 Basic segmentation methods

A great number of contour detection approaches are devised for image segmentation. For example, the normalized cuts, clustering methods, histogram-based methods, compression-based methods and region-growing methods. This section will introduce thresholding and region growing approaches.

a). Thresholding

Thresholding could be the simplest method for image segmentation. It aims to segment an image into two regions based on discontinuities in thresholds of brightness[16]. The results are in two groups, foreground and background. A suitable threshold value need to be set at first, then the intensity of each pixel is checked against to the threshold value. If the value for a pixel meets the threshold it will be colored black, otherwise colored in white. The pseudo-code for this approach is shown below:

```
Algorithm  Thresholding Algorithm on Greyscale Image of Size (w*h)
1: $threshold \leftarrow C3C3C3$ (colour hex)
2: $i \leftarrow 0$
3: $j \leftarrow 0$
4: for $i$ from 0 to $w-1$ do
5:   for $j$ from 0 to $h-1$ do
6:     if $colour(pixel(i,j)) > threshold$ then
7:       $pixel(i,j) \leftarrow BLACK$
8:     else
9:       $pixel(i,j) \leftarrow WHITE$
10:    end if
11:  end for
12: end for
```

The important problem is that how to choose a proper value for threshold. Three possible methods for setting a threshold are: use average intensity, plot intensity histogram and hysteresis thresholding. For the method: plot intensity histogram, it is to find peaks in the histogram and assume histogram is bimodal, then place threshold between the peaks like the illustration in Figure 3.

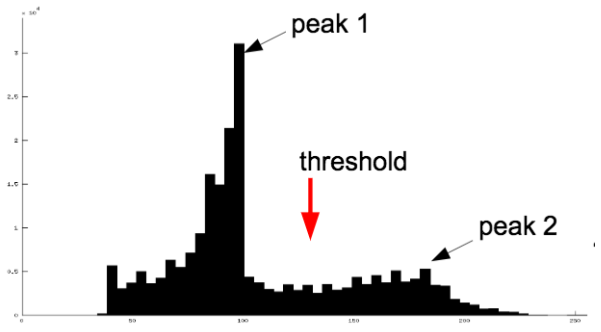


Figure 3[17]: Histogram for an image

This thresholding algorithm runs in O(hw) space, where h and w are the height and width of the image. Moreover, this algorithm can be considered running in linear time with the number of pixels in the image, due to the algorithm is simple and without nested loops. Moreover, there are some more thorough approaches to set threshold, which is using fuzzy sets to analyze threshold levels[18].

Although thresholding algorithm often results in missing parts and marking unwanted parts in an image, it can be efficient to identify the regions of interest if using the proper threshold values. Thus, this algorithm can be used as a preprocessor to locate the regions of interest in other image segmentation algorithms.

b). Region growing

A fundamental drawback of thresholding approach is that it does not include the spatial information[17]. Region based segmentation approaches can take into account the location of each pixel in an image as well as its other features like color, intensity, or texture[17]. Region growing is one of the region based segmentation approaches. The first step in region growing is to select seed points. Then, the regions are growing from these seeds to neighboring pixels depending on a featured criterion[19]. The criterion could be many features, for example, intensity, or color[19]. The pseudo-code of it is showed as follows.

Algorithm: Region growing

```
1: Start with one "seed " pixel, chosen arbitrarily
2: Mark this "seed" pixel a label
3: Examine all the unlabelled pixels neighboring labelled pixels
4:   if they are within the similarity threshold,
         give them the same labels as labelled pixels
5: Repeat 4 until region stops growing
6: Choose another "seed" pixel which does not yet belong to any region
7: Start 1 again
8: Repeat the whole process until all pixels have been assigned to a region.
```

Region growing has several advantages, for example:

- It can correctly split the regions that have the same features defined as the grouping criterion[19].
- Region growing can produce good segmentation results for those images have clear edges.
- The concept is simple and easy to be understand.
- Multiple criterions can be selected as the same time.

Although region growing takes into account the location of each pixel and has many advantages, it still has some problems:

- The algorithm is limited in some kinds of images, because the assumption of the algorithm is that regions are approximately constant in image intensity[20].
- The algorithm does not work in non-smoothly varying regions, such as the textured regions[20].
- It is sensitive to noise.
- It is a local approach without global view.
- Computationally expensive.

2.3.2 Advanced segmentation methods

One of the state-of-the-art contour detector combines several features to produce the outputs. For example, combines analyzation for color, local brightness and texture contrasts to obtain one result, its output is called transitional gradient-based approach[15]. This approach was developed as follows. Finding discontinuities in brightness is the most common way for boundary detection[21]. For example, Canny detector is one algorithm guided by the changes of brightness[22]. However, David(2004) indicated that only use brightness to locate boundaries in a natural image is inadequate, because the diversity of texture is ubiquitous[15]. In addition, the textured difference may cause high-contrast edges, whereas no boundary exists[15]. Fowlkes(2004) suggested one possible solution for this problems, which is to exam gradients at multiple orientations around a pixel[15]. However, this solution only can reduce false in a limited class of textures[15]. One feasible method was then developed by Martin(2008) to solve suppressions of edges and corners in textured regions when detecting boundaries. This approach searches local featured discontinuities for each pixel in an image in four feature channels simultaneous and the searching process is over a range of scales and orientations[15]. The four features are two brightness features (oriented energy and brightness gradient), one color feature (color gradient) and one texture feature (texture gradient)[15]. This approach can provide good segmentation results for natural images, because it offers proper treatments for texture, which is one of the most essential elements in a natural image and the algorithm combines several individual features and searches for contour at different orientations and scales.

Moreover, boundaries are multi-scaled in nature images, so processing edges in multi-scales can distinctly improve the precision of segmentation. Gaussian smoothing is used in the early studies on multi-scale edges detection[23]. However, Gaussian-based multi-scale edge detectors are impracticable for natural images[23]. Because general boundary detections are almost reliable at large scales but they always performs a low-levels performance at localization[23]. Globalized probability of boundary(gPb) is a high performance contour detector, which combining both local and global image information. To be detailed, it couples multi-scale local color, texture and brightness cues to a globalization framework via spectral clustering[24]. The local cues are produced by using oriented gradient operators for each pixel in an image[24]. The final gPb algorithm will combine all local gradient cues obtained in three different scales[24]. The precision of gPb algorithm with respect to other single-scale *Pb* is improved over 10%. However, this method also has some disadvantages like it is high cost.

2.4 Top-down segmentation

Most image segmentation approaches are only based on color information of pixels in the past years. Recently, many studies are focusing on using object category knowledge to do image segmentation and some trainable segmentation approaches have been devised to modeling some of these knowledges.
Deep learning architecture is extreme efficient for trainable segmentation so it has been employed for it in many recent researches[12]. In addition, several neural network architectures have been explored, such as Pulse-coupled neural networks (PCNNs), efficient neural network (ENet) and SegNet. This section will introduce SegNet, which is used in this project.

SegNet, an efficient deep encoder-decoder neural network architecture, developed for multi-class pixel-wise semantic segmentation[12]. It was design by Kendall(2016) and Badrinarayana(2016) - members of the computer vision and robotics group at the University of Cambridge, UK[25]. This architecture aims for understanding road scenes. It requires to detect the appearances if buildings, cars, roads, pedestrians and as well as understand the spatial relationships between these different classes[12]. The illustration of the SegNet architecture is shown in Figure 4.

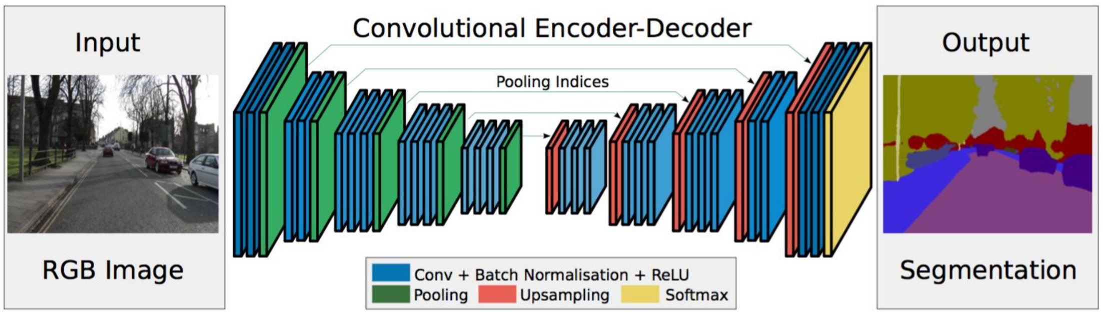


Figure 4[12]: The illustration of SegNet

It does not have any fully connected layers, hence it is only convolutional and it makes SegNet easier to be trained than many other architectures[26][27]. There is a encoder network and a decoder network in the SegNet architecture, followed by a final pixel-wise classification layer[12]. The key learning module of SegNet is the encoder-decoder network[12]. The encoder network is formed by 13 convolutional layers in VGG16 architecture[12], which is a large model for the multi-class classification[28]. Each encoder layer has a corresponding decoder layer, so there are 13 decoder layers as well[12]. Figure 4 shows these decoders are in hierarchy. Therefore, decoders can non-linearly upsamples its inputs guided by the transferred pool indices from its encoder, the output of upsampling process is a sparse feature map[12]. The reason for decoders are important is that the function of them can significantly enhance the delineation of boundaries[12]. SegNet architecture then performs convolution to the feature map with a trainable filter bank to densify it[12]. Finally, these feature maps are fed to a soft-max classifier for the classifications of pixel-wise[12].
SegNet segmentation provides superior permanence on segmenting road scene images. It can reliable labels objects into different classes. Moreover, it is efficient in both computational time and memory.

3 Chapter 3

Design and Implementation

3.1 Overview

The main contribution for this project is implementing some existing algorithms guided by top-down and bottom-up respectively. Then, explore ways to combining these top-down and bottom-up approaches together to obtain a final segmentation results, which can enhance the performance of using either one of these approaches alone. Moreover, an algorithm of bottom-up approach using region growing is designed as well. This chapter is consisted of three sub-sections - top-down approach, bottom-up approach and combination of top-down and bottom-up. In each sub-section, the specification and methodology for implementing the algorithms will be presented. Moreover, the design of combining two approaches will be introduced in the last sub-section.

3.2 Top-down approach (SegNet)

This sub-section describes the fully process of implementing an existing state-of-the-art top-down image segmentation approach - SegNet.

3.2.1 Specification

The specifications for running SegNet are listed as follow:

a. Programming language: Python 2.7
b. Platform: SegNet can be ran on Ubuntu 16.04-12.04 or OS X 10.11-10.8. OS X 10.11 is used in this project.
c. Framework: the implementation of SegNet is built on top of the Caffe, which is a deep learning framework[29]. It is developed by Berkeley Vision and Learning Center (BVLC)[29]. Caffe provides large convenience due to its advantages:
   - Expression: models are defined in plaintext schemas, not in code schemas[29].
   - Speed: the speed for researching is significant for a state-of-the-art models[29]. It has high speed performance.
   - Modularity: it has high flexibility for new tasks and settings[29].
   - Openness: applied progress call for reference models.

3.2.2 Methodology

This sub-section will describe the methodology for implementing the algorithm of SegNet. Many steps will be listed in this part because it is a complicated process. The detailed process is described as follows:

1. Get Caffe installation. This is the most difficult step for running SegNet, many sub-steps need to be carried out for Caffe installation, any step fails will lead the installation failed. After many times trying to install it, Caffe finally can run on the laptop. The sub-steps are introduced below:
   1). Install Homebrew package manager (under the path of /usr/local and ideally start from cleaning /usr/local/ to avoid conflicts.). This package manager will be of enormous use throughout the process of installing Caffe.
   2). Install Anaconda Python 2.7 into the same path as the Homebrew.
   3). Install CUDA 7.0
   4). Install standalone CUDA driver.
   5). Install libstdc++.

In OS X 10.11, clang++ is the default C++ compiler and it use libc++ as the standard library. However, CUDA 7.0 links with libstdc++ currently. Therefore, the Homebrew formulae need to be modified before installing any packages to ensure all packages are linked to libstdc++.

6). Install dependencies via Homebrew, the dependencies are:
   a. edit the installation file for OpenCV.
   b. install snappy, gflags, glog, leveldb, lmdb, szip and opencv.
   c. install protobuf.
   d. install boost libraries for python.

7). Download Caffe
   Create a directory named <caffe> under the path /usr/local/ to install Caffe. Then, from inside the <caffe> directory to run the following commands to download Caffe:

```
$> git clone https://github.com/BVLC/caffe.git
$> cd caffe
$> cp Makefile.config.example Makefile.config
```

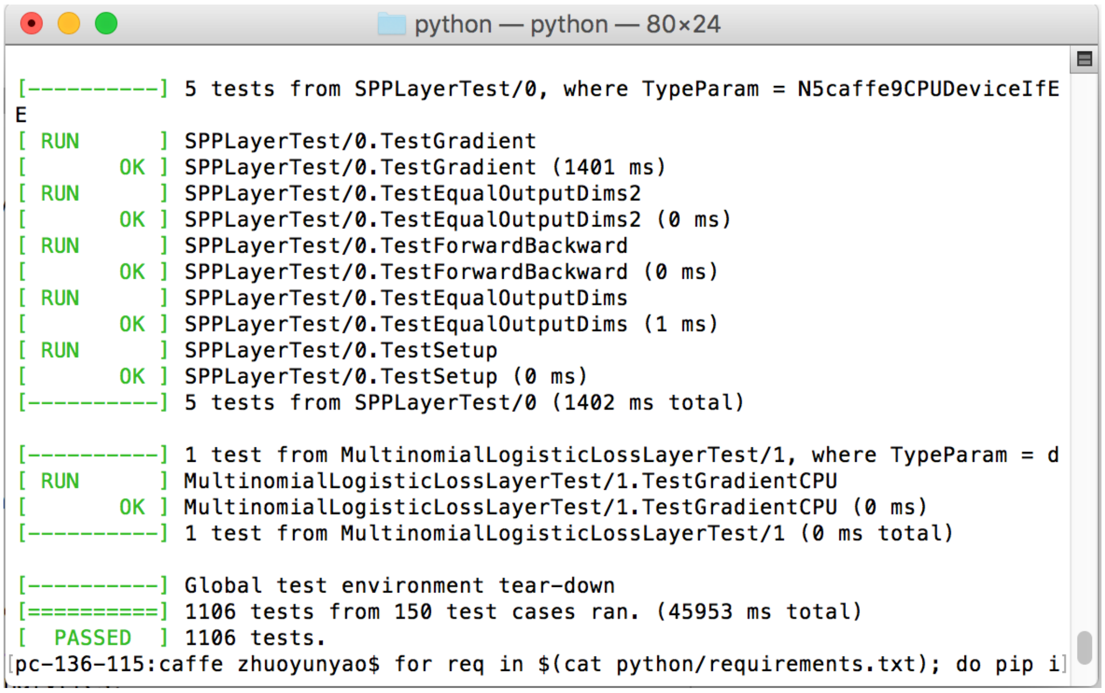
8). Then, it is a very import step, which is to edit the Makefile.config file. Keep track of all the path and variables set in the makefile.config file, even one small mistake will cause problems. The modification of it is intricate and will not be described here step-by-step. However, one special point from general modification is that Caffe will be built without GPU in this project. Thus, one commends need to be added:

```
# CPU-only switch (uncomment to build without GPU support).
CPU_ONLY := 1
```

9). Compile Caffe
   Finally, Caffe can be compiled by executing following commends from the directory <caffe>:

```
$> make clean
$> make all
$> make test
$> make runtest
```

After several minutes, the output is shown like the image below, which means the compilation of Caffe is successful and Caffe is installed to the laptop now.



```
[----------] 5 tests from SPPLayerTest/0, where TypeParam = N5caffe9CPUDeviceIfE
E
[ RUN      ] SPPLayerTest/0.TestGradient
[       OK ] SPPLayerTest/0.TestGradient (1401 ms)
[ RUN      ] SPPLayerTest/0.TestEqualOutputDims2
[       OK ] SPPLayerTest/0.TestEqualOutputDims2 (0 ms)
[ RUN      ] SPPLayerTest/0.TestForwardBackward
[       OK ] SPPLayerTest/0.TestForwardBackward (0 ms)
[ RUN      ] SPPLayerTest/0.TestEqualOutputDims
[       OK ] SPPLayerTest/0.TestEqualOutputDims (1 ms)
[ RUN      ] SPPLayerTest/0.TestSetup
[       OK ] SPPLayerTest/0.TestSetup (0 ms)
[----------] 5 tests from SPPLayerTest/0 (1402 ms total)

[----------] 1 test from MultinomialLogisticLossLayerTest/1, where TypeParam = d
[ RUN      ] MultinomialLogisticLossLayerTest/1.TestGradientCPU
[       OK ] MultinomialLogisticLossLayerTest/1.TestGradientCPU (0 ms)
[----------] 1 test from MultinomialLogisticLossLayerTest/1 (0 ms total)

[----------] Global test environment tear-down
[==========] 1106 tests from 150 test cases ran. (45953 ms total)
[  PASSED  ] 1106 tests.
pc-136-115:caffe zhuoyunyao$ for req in $(cat python/requirements.txt); do pip i
```

Figure 5: Caffe installed successfully

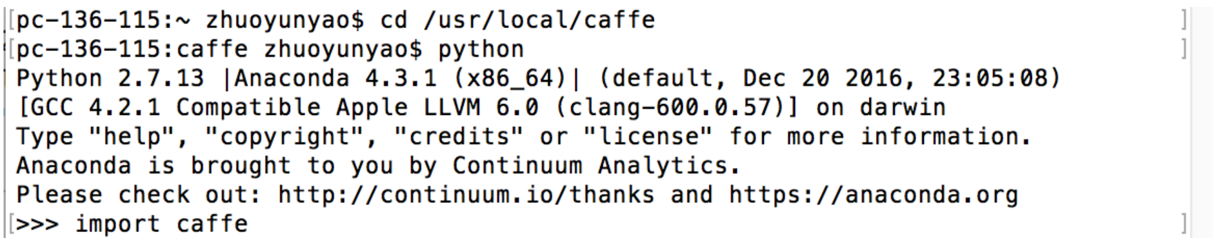
2. Compile Caffe's Python wrapper.
For SegNet is devised via Python, it is necessary to install and configure the Python interface to Caffe. First is to install all the Python dependencies with:

```
for req in $(cat requirements.txt); do pip install $req; done
```

Then, import the Caffe Python module by adding the module to the path of Python. Finally, compile the Caffe Python interface by:

```
make pycaffe
```

To check whether the compilation of Caffy's Python wrapper is successful, try to enter Python from the directory of Caffe. Figure 6 indicates the compilation is succeed.

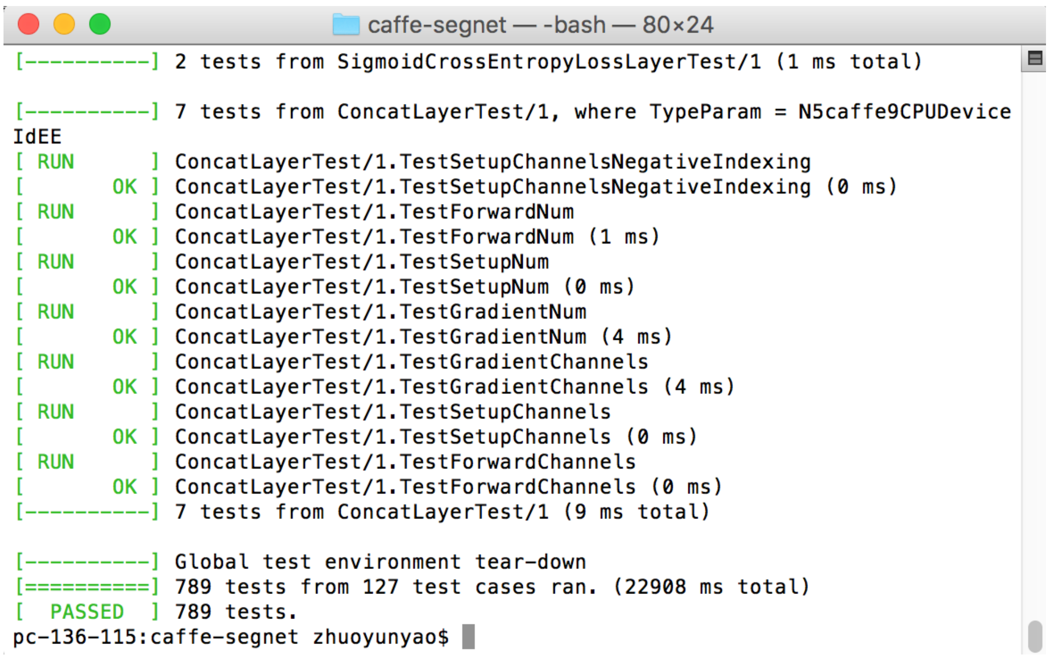


```
pc-136-115:~ zhuoyunyao$ cd /usr/local/caffe
pc-136-115:caffe zhuoyunyao$ python
Python 2.7.13 |Anaconda 4.3.1 (x86_64)| (default, Dec 20 2016, 23:05:08)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
>>> import caffe
```

Figure 6: Caffe's Python wrapper compiled successfully

3. Install SegNet-caffe.

As SegNet is based on the Caffe framework and it makes some modification on it, install the SegNet-caffe as the same procedures as the installation of Caffe from step 7) described in step 2. Moreover, for Caffe is already installed on the laptop, the installation of SegNet-caffe will be slightly easier. Figure 7 shows SegNet-caffe is installed successfully.



```
[----------] 2 tests from SigmoidCrossEntropyLossLayerTest/1 (1 ms total)

[----------] 7 tests from ConcatLayerTest/1, where TypeParam = N5caffe9CPUDevice
IdEE
[ RUN      ] ConcatLayerTest/1.TestSetupChannelsNegativeIndexing
[       OK ] ConcatLayerTest/1.TestSetupChannelsNegativeIndexing (0 ms)
[ RUN      ] ConcatLayerTest/1.TestForwardNum
[       OK ] ConcatLayerTest/1.TestForwardNum (1 ms)
[ RUN      ] ConcatLayerTest/1.TestSetupNum
[       OK ] ConcatLayerTest/1.TestSetupNum (0 ms)
[ RUN      ] ConcatLayerTest/1.TestGradientNum
[       OK ] ConcatLayerTest/1.TestGradientNum (4 ms)
[ RUN      ] ConcatLayerTest/1.TestGradientChannels
[       OK ] ConcatLayerTest/1.TestGradientChannels (4 ms)
[ RUN      ] ConcatLayerTest/1.TestSetupChannels
[       OK ] ConcatLayerTest/1.TestSetupChannels (0 ms)
[ RUN      ] ConcatLayerTest/1.TestForwardChannels
[       OK ] ConcatLayerTest/1.TestForwardChannels (0 ms)
[----------] 7 tests from ConcatLayerTest/1 (9 ms total)

[----------] Global test environment tear-down
[==========] 789 tests from 127 test cases ran. (22908 ms total)
[  PASSED  ] 789 tests.
pc-136-115:caffe-segnet zhuoyunyao$
```

Figure 7: SegNet-Caffe installed successfully

4. Set up the dataset
SegNet learns to speculate pixel-wise class labels from the supervised learning[12]. Thus, a dataset of input images with corresponding ground-truth labels is required. Because SegNet is devised for understanding road scenes, the dataset used for running SegNet is the CamVid dataset[30], it contains 367 training images and 233 testing images for street scenes. Each pixel in an image need to be labelled with its class, among 11 classes in total. Moreover, all images are in the size of 360 by 480.
5. Download SegNet
To run SegNet smoothly, the file structure should be stored in a fixed mode.
6. Training SegNet
The batch size for training the SegNet model is completed depending on the size of GPU. According to the official document, even a batch size of as low as 2 or 3 should still be able to train it well[12]. However, the laptop used in this project does not have a GPU, so only CPU is used for training SegNet. It is the reason for compiling Caffe without GPU in the above steps.
The disadvantage for training with CPU is that the speed is deadly slow. Train the model for 250 times iteration cost 18 hours. After training 250 times iterations, the output is converged and had a high training accuracy, which is greater than 90%.
7. Testing SegNet
After obtaining a great training accuracy, it is time to test SegNet. The batch normalization layers[31] in SegNet can shift the input feature maps guided by their variance and mean statistics for each batch during the process of training[31]. In the test, the statistic for the entire dataset need to be used. Therefore, a training weight file contains test weights will produced in this step. This procedure will cost about 30 minutes.

8. Run the scripts of SegNet and obtained the segmentation results.

3.2.3 Results

Some example results will be listed below. The results for the same images provide by the official document will also be displayed to see the implantation of SegNet is successfully.

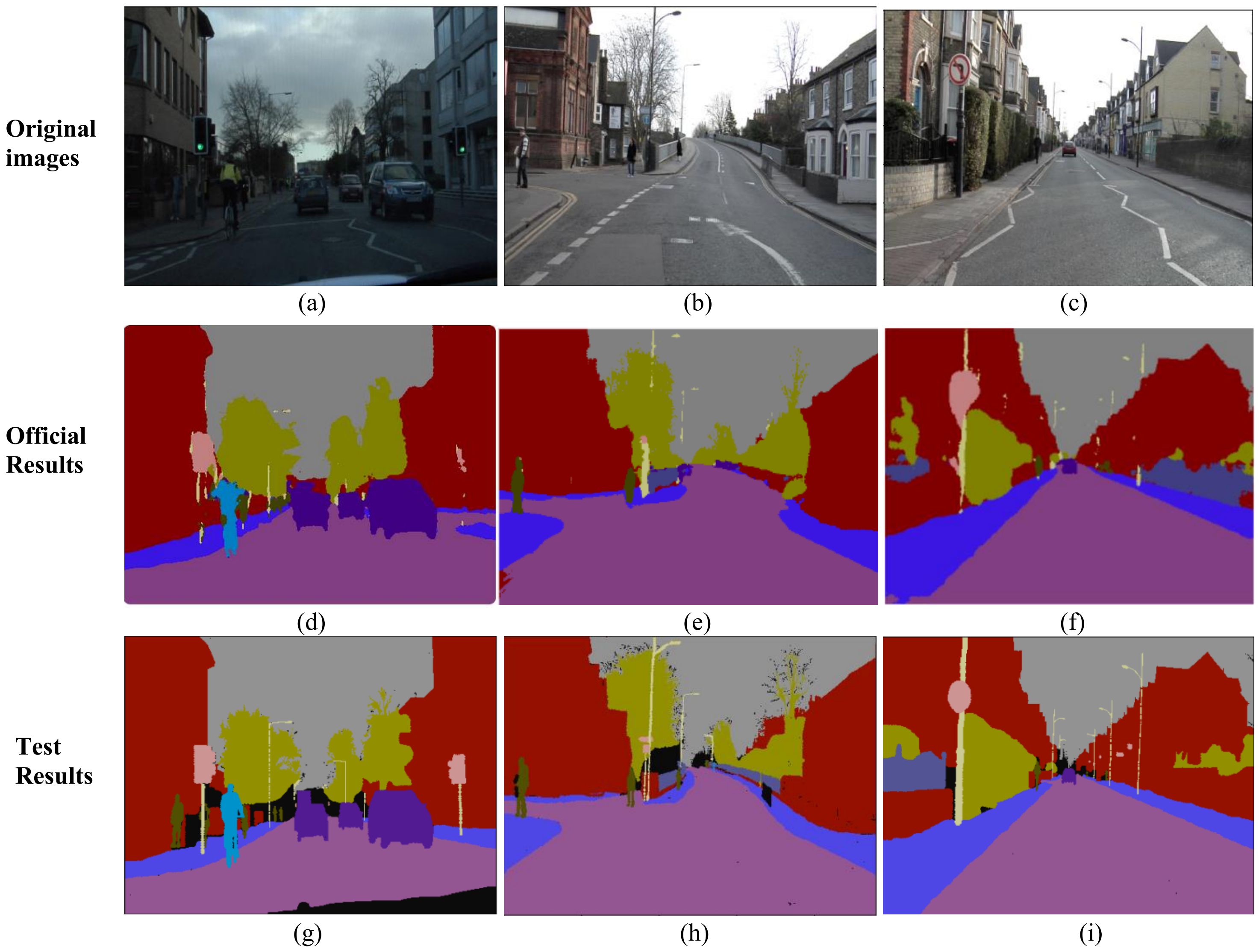


Figure 8: example results of SegNet. (a),(b),(c) are input images. (d),(e),(f) are the segmentation results provided in official document. (g),(h),(i) are results produced by the SegNet implemented in this project

These results produced by SegNet are labelled in 11 different colors for different classes. All labels used in SegNet are:

⬜	🟥	🟨	🟪	🟦	🟩	🟥	🟦	🟪	🟫	🟦
Sky	Building	pole	Road	Pavement	Tree	Sign	Fence	Vehicle	Pedestrian	Bike

Figure 8 indicates the implement of SegNet is successful and reliable. Moreover, the results produced by the SegNet implemented in this project are more precise than official results in some aspects. For example, almost all contours in Figure 8 (g) are more clearly than Figure 8 (d), such as the contours of traffic lights. In additional, pedestrians are detected more accuracy in Figure 8 (g). It is obvious that there are five pedestrians on the left-hand side of the pavement, whereas the contours of pedestrians are fuzzy in Figure 8 (d). For the last pair of comparison, boundaries of lamps in Figure 8 (i) are much more clearly. Additionally, fence on the left-hand side is detected correctly in Figure 8 (i), but deficient in Figure 8 (f). Furthermore, there is a mistake in the official results Figure 8 (f), it shows some fences are on the right-hand side of the road. However, from the original input image Figure 8 (c), it is clear that the right-hand side of the road is wall not fence. Thus, there should be labelled in red and Figure 8 (i) labelled it correct.
However, the results produced by the SegNet implemented in this project also have some shortcomings. There are some areas might cannot be recognized and marked in black in Figure 8(g),(h) and (i).

The evaluation of SegNet algorithm will be discussed in the next chapter via the standard segmentation benchmarks.

3.3 Bottom-up approach (gPb)

This sub-section will describe the process of implementing gPb algorithm, which is guided by bottom-up image features and illustrate some image segmentation results of it.

3.3.1 Specification

The specifications for running gPb are listed as follow:

a. Programming language: Matlab 2017a
b. Platform: Ubuntu 16.04

3.3.2 Methodology

This sub-section will describe the methodology for implementing the algorithm of gPb. The process is described as follows:

1. Install Ubuntu 16.04.
2. install Matlab 2017a.
3. configure mex in the Matlab prompt.
4. export the Matlab into the path.
5. download and install some required image libraries.
6. download gPb scripts into a file.
7. edit Rules. make file for the later compilation.
8. compile gPb to build all the packages and copy the resulting files in .mex forms into a fixed directory.
9. run gPb scripts to obtain the segmentation results.

3.3.3 Results

In order to get a better comparison between individual algorithms, the original input images used for testing gPb are also some road scenes images. Some example results will be listed as below:

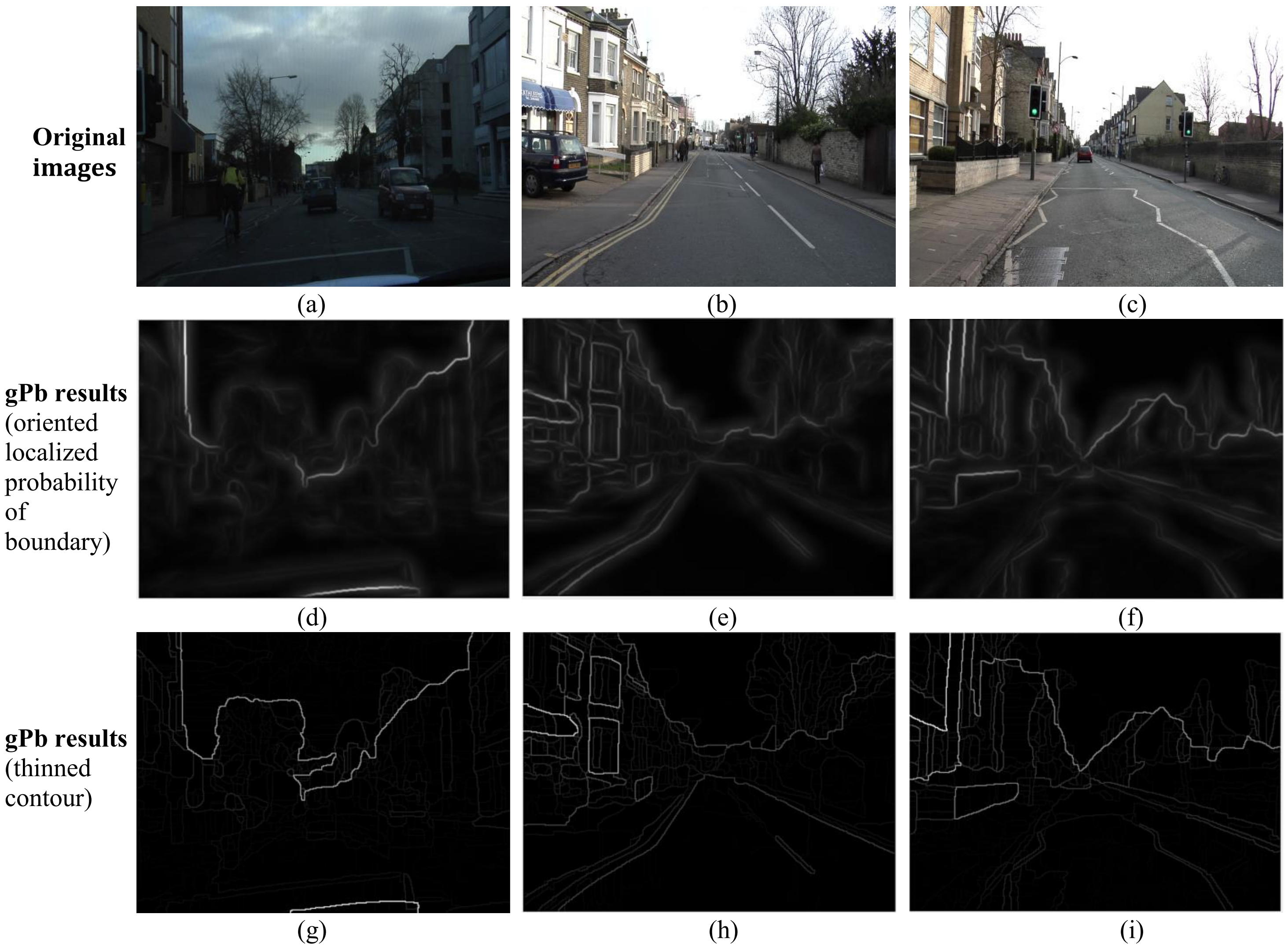

Original images
(a)
(b)
(c)
gPb results (oriented localized probability of boundary)
(d)
(e)
(f)
gPb results (thinned contour)
(g)
(h)
(i)

Figure 9: example results of gPb on road scenes images. (a),(b),(c) are input images. (d),(e),(f) are the oriented localized probability of boundaries produced by gPb. (g),(h),(i) are thinned contours produced by gPb

The algorithm of gPb can produce three results for each input image. Two image outputs - one is the oriented localized probability of boundaries and another is thinned contours for objects in an image. Moreover, an ultrametric contour map (ucm) for the result of thinned contours will generated along with other two outputs. From Figure 9, it is obvious that the integrity of contours for objects in gPb results are much poor than SegNet. However, it can depict many fine edges which might be the inner edges in an object, not only detect the outer boundaries of an object.

Moreover, the evaluation of gPb algorithm will be discussed in the next chapter by using the standard segmentation benchmarks.

3.4 Bottom-up approach (Region growing)

A region growing algorithm is designed in this project. This sub-section will introduce this algorithm and illustrate some image segmentation results of it. The programming language used for designing the region growing algorithm is Matlab. Actually, the common ideal for region growing method is already introduced in Chapter 2. Some points will be selected as the seeds and then the regions are growing from these seeds to neighboring pixels. The detailed process of it will not be introduced again in this part.

After using region growing to segment objects into regions, canny edge detector is used to produce the contour images of the results, which will be utilized in the later combination process. Some example results are listed as below:

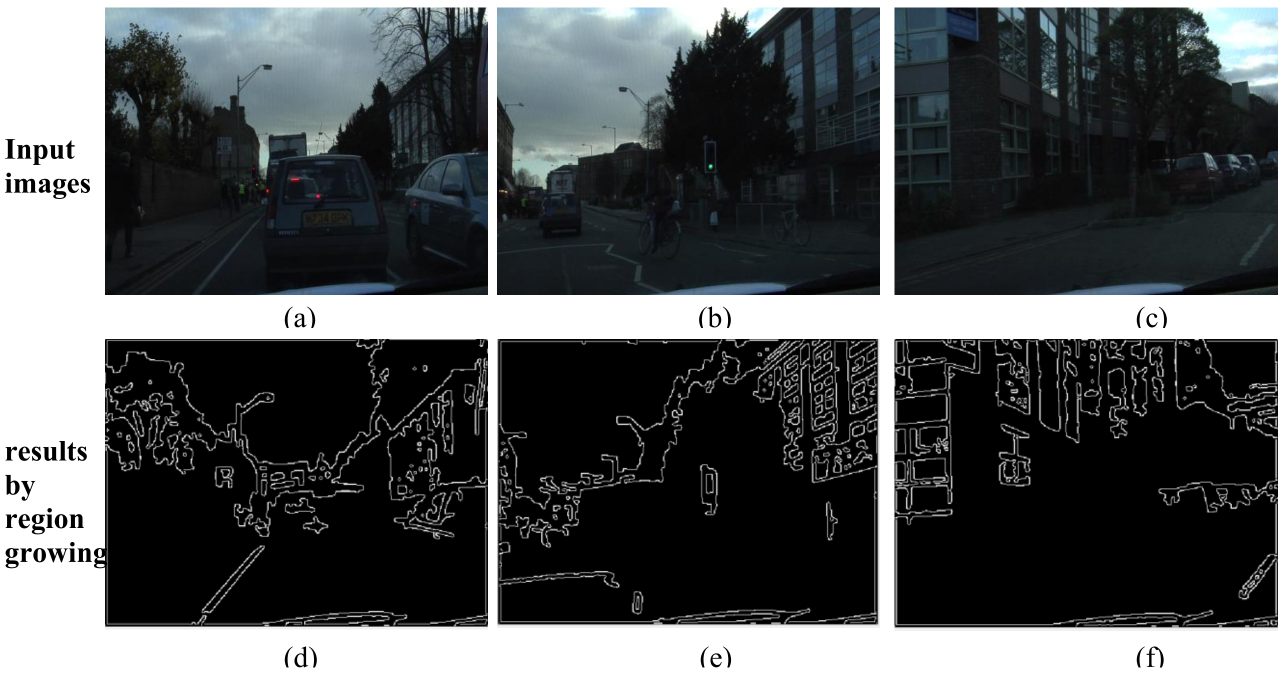


Figure 10: example results of region growing alogrithm on road scenes images. (a),(b),(c) are input images. (d),(e),(f) are the results produced by region growing algorithm.

The results produced by region growing algorithm cannot detect objects precisely. Contours for most objects are incomplete. Moreover, it is sensitive to noise and has bad performance in low-brightness areas. The evaluation of region growing algorithm will be discussed in the next chapter via the standard segmentation benchmarks.

3.5 Combination of Bottom-up and Top-down

This sub-section will introduce the ways explored to combining top-down and bottom-up approaches implemented in the above sub-sections to obtain a final image segmentation results.

3.5.1 Specification

The specifications are listed as follow:

c. Programming language: Matlab 2017a
d. Platform: OS X 10.11

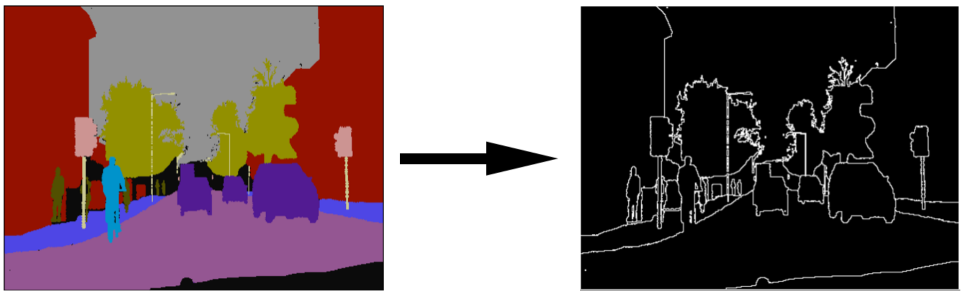
3.5.2 Methodology

The methodology for designing methods to combine two image segmentation approaches will be introduced in this section. The whole design process is consisted of two parts: preparation and combination. Additionally, there are two ways in the combination part.

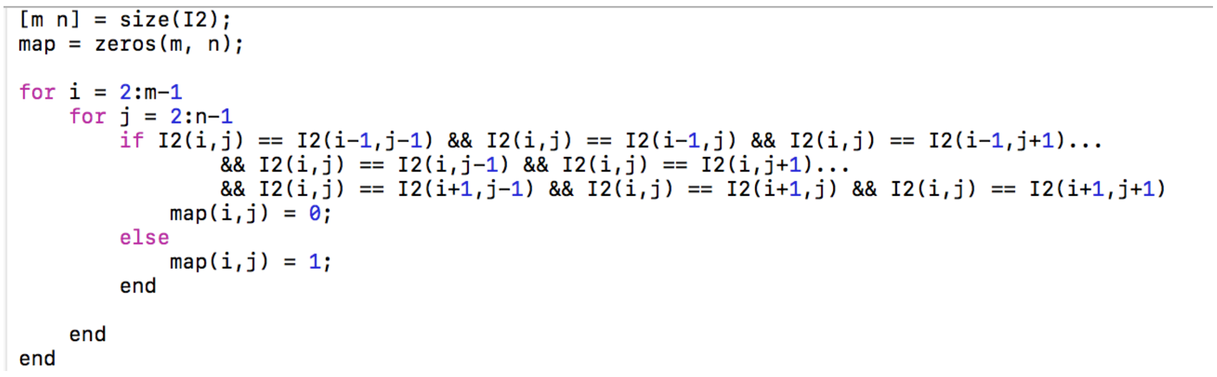
A. Preparation

a. Convert the results of SegNet into contour images

The result of SegNet is a set of segments that cover the whole image collectively and the results of gPb and region growing are a set of contours extracted from the input image. To combine two segmentation results together, they first need to be in the same format. Thus, the first step is to convert the results of SegNet into contour images. To illustrate it more visualized, the demonstration of it is shown below:



The approach is to analyze each pixel in the result images provided by SegNet. For every object in the result of SegNet is labelled in a color, all pixels in one single object should have the same value. Therefore, it will be straightforward to find contours. To be detailed, if the pixel value is changed from one number to another number at one place, it means there is the boundary between two objects. Moreover, a digital image can be represented as a matrix, which have already mentioned in chapter 2. Thus, matrix maps will be created to represent contour images for the results of SegNet. To generate the matrix map, make judgments for each pixel with its neighboring pixels. If the value of this pixel is the same as the neighboring pixels, set the value of this pixel in the new matrix map as 0. Otherwise, set the value as 1 to indicate the edges. Use *if* loop in Matlab to realize it:



```
[m n] = size(I2);
map = zeros(m, n);

for i = 2:m-1
    for j = 2:n-1
        if I2(i,j) == I2(i-1,j-1) && I2(i,j) == I2(i-1,j) && I2(i,j) == I2(i-1,j+1)...
                && I2(i,j) == I2(i,j-1) && I2(i,j) == I2(i,j+1)...
                && I2(i,j) == I2(i+1,j-1) && I2(i,j) == I2(i+1,j) && I2(i,j) == I2(i+1,j+1)
            map(i,j) = 0;
        else
            map(i,j) = 1;
        end

    end
end
```

Figure 11: algorithm for converting the results of SegNet into contour images.

Some of the transformed contour images are shown as below:

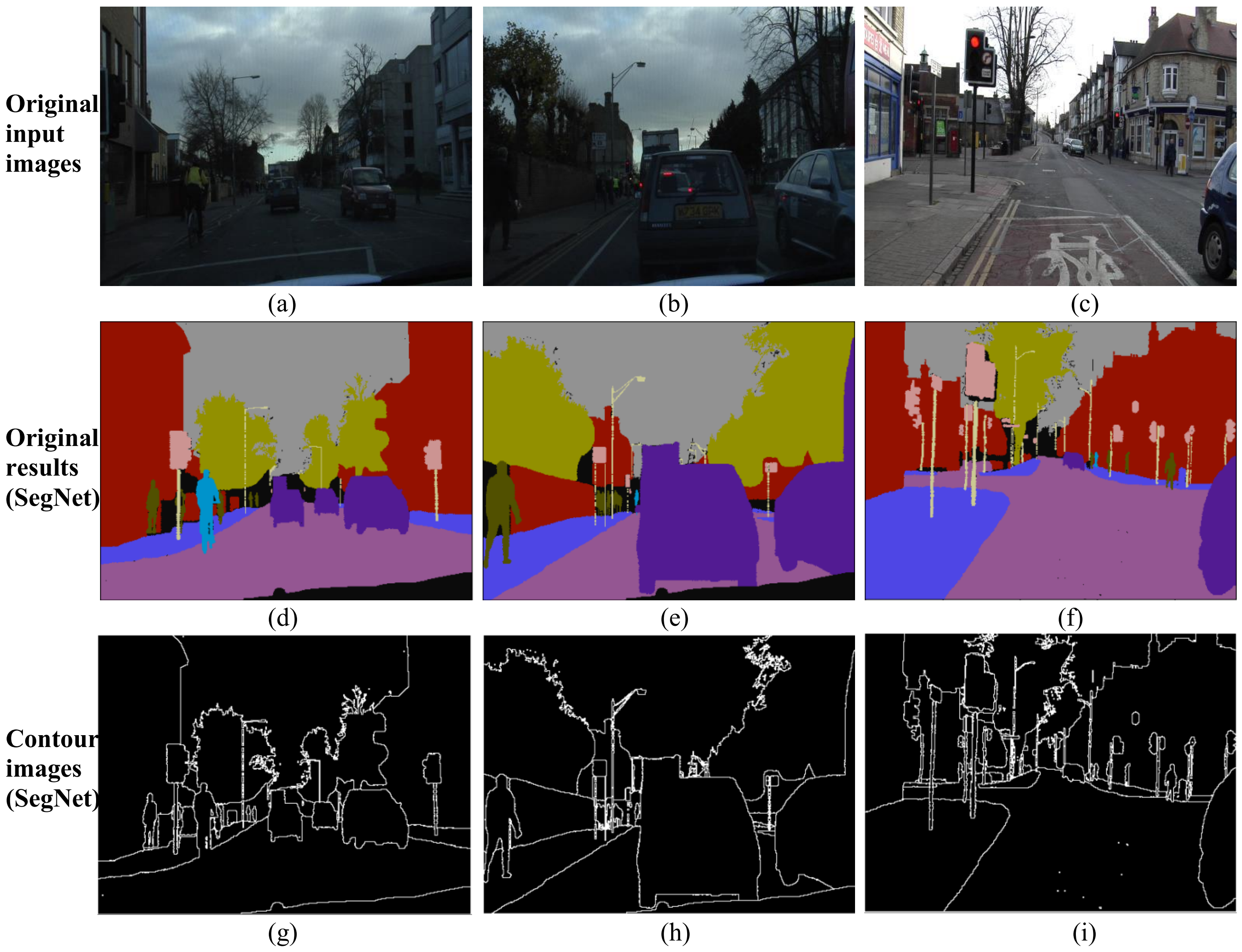


Figure 12: convert the results of SegNet into contour images. (a),(b),(c) are input images. (d),(e),(f) are the original results produced by SegNet. (g),(h),(i) are contour images for (d),(e),(f)

b. Resize images

Make sure the contour images generated in the las step are in the same size as the results of gPb and region-growing. Because to combine two images, they need to be in the same size. If they are not in the same size, resize them by:

```
mask = imresize(mask, [m n], 'bicubic');
```

B. Combination

As a digital image can be represented as a matrix, the process of combining two images can converted into a process of analyzing two matrixes. The gPb algorithm can generate an ultrametric contour map for each segmentation result along with all outputs, which is a matrix with all values in range [0 1]. The region growing algorithm is also designed to output the matrix with the segmentation results. Additionally, the matrixes for the SegNet's results are obtained in the preparation section. Thus, the simplest way to combine two images is to overlay two matrixes. However, the approaches for overlaying two matrixes are various. Two efficient methods will be described in this part.

a. Method No.1

The first method is to overlay two matrixes directly by analyzing the pixel in the first matrix with the corresponding pixel in the second matrix. Knowing the values in the matrix of SegNet's results can only be 0 or 1 and values in the matrix of other algorithms are in the range of [0,1], design the algorithm:

```
Algorithm: Combing method No.1

Set Map_1 = matrix for SegNet results
    Map_2 = matrix for gPb or region-growing results
    Map_3 = matrix for combined results
For each pixel in an image
    if Map_1(i, j) equals 0 and Map_2(i, j) equals 0, Set Map_3(i, j) = 0;
    if Map_1(i, j) equals 1 and Map_2(i, j) equals 1, Set Map_3(i, j) = 1;
    if Map_1(i, j) equals 1 and Map_2(i, j) not equals 0, Set Map_3(i, j) = 1;
    if Map_1(i, j) not equals 0 and Map_2(i, j) not equals 0, Set Map_3(i, j) = Map_2(i, j);
end
```

Map_3 represents the final segmentation results combined both bottom-up and top-down approaches. This algorithm will be evaluated in the next chapter.

b. Method No.2

The generalized conception of the second method is similar with the method No.1. The difference is that a threshold value is setting for the overlay process.

For gPb is a hierarchical segmentation algorithm, many fine edges might be detected as well. All the detected edges are shown in different shade of colors. The edges shown in a light shade means it has low possibility to be a true contour. In fact, some of these edges with extreme low possibilities are not necessary to shown in the segmentation results. Thus, set a threshold value when overlaying. If the value in Map_2 is higher than the threshold value, set the value of this pixel to a big number (e.g. 0.7) to show this contour clearly in the final results as this pixel has high probability to be a meaningful contour. Otherwise, set the value of this pixel to a small number (e.g. 0.1). The algorithm is shown below:

```
Algorithm: Combing method No.2

Set Map_1 = matrix for SegNet results
    Map_2 = matrix for gPb or region-growing results
    Map_3 = matrix for combined results
For each pixel in an image
     if  Map_1(i, j) equals 0 and Map_2(i, j) equals 0, Set Map_3(i, j) = 0;
     if  Map_1(i, j) equals 1 and Map_2(i, j) equals 1, Set Map_3(i, j) = 1;
     if  Map_1(i, j) equals 1 and Map_2(i, j) not equals 0, Set Map_3(i, j) = 1;
     if  Map_1(i, j) not equals 0 and Map_2(i, j) not equals 0
         if  Map_2(i, j) meets threshold value, Set Map_3(i, j) = 0.7;
         elseif Set Map_3(i, j) = 0.1;
         end
end
```

In addition, the method for finding a proper threshold value is to observe the histograms of as many as gPb's results. Here, three histograms will be illustrated to see the common property of gPb's results.

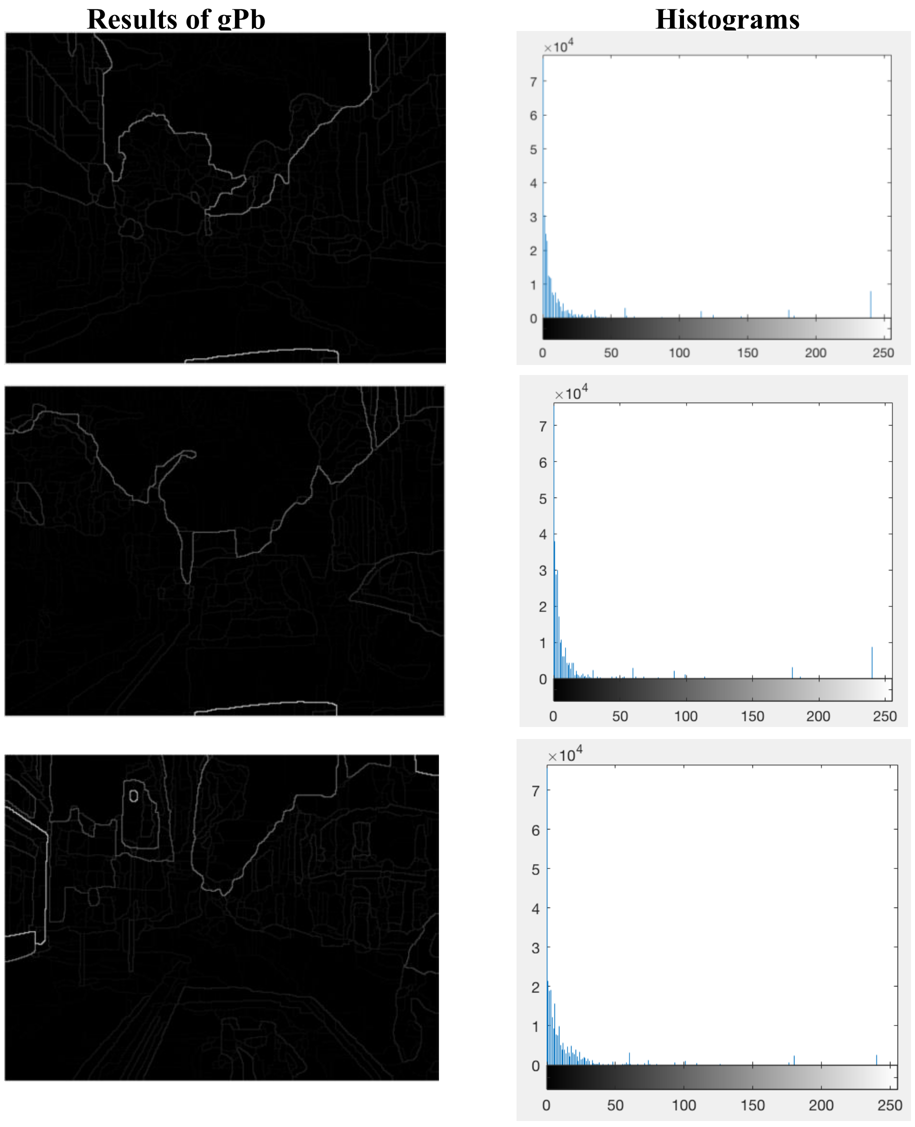


Figure 13: histograms for some of the gPb's Results

From Figure 13, it is clear that there are many inner edges in all of the results produced by gPb and these edges are all concentrate under about 25. They are in a very light shade of color. Thus, set this value as the threshold value for all images. Tests proof that this value is proper for all images in the dataset.

3.5.3 Results

Some example results will be listed as below:

A. SegNet + gPb

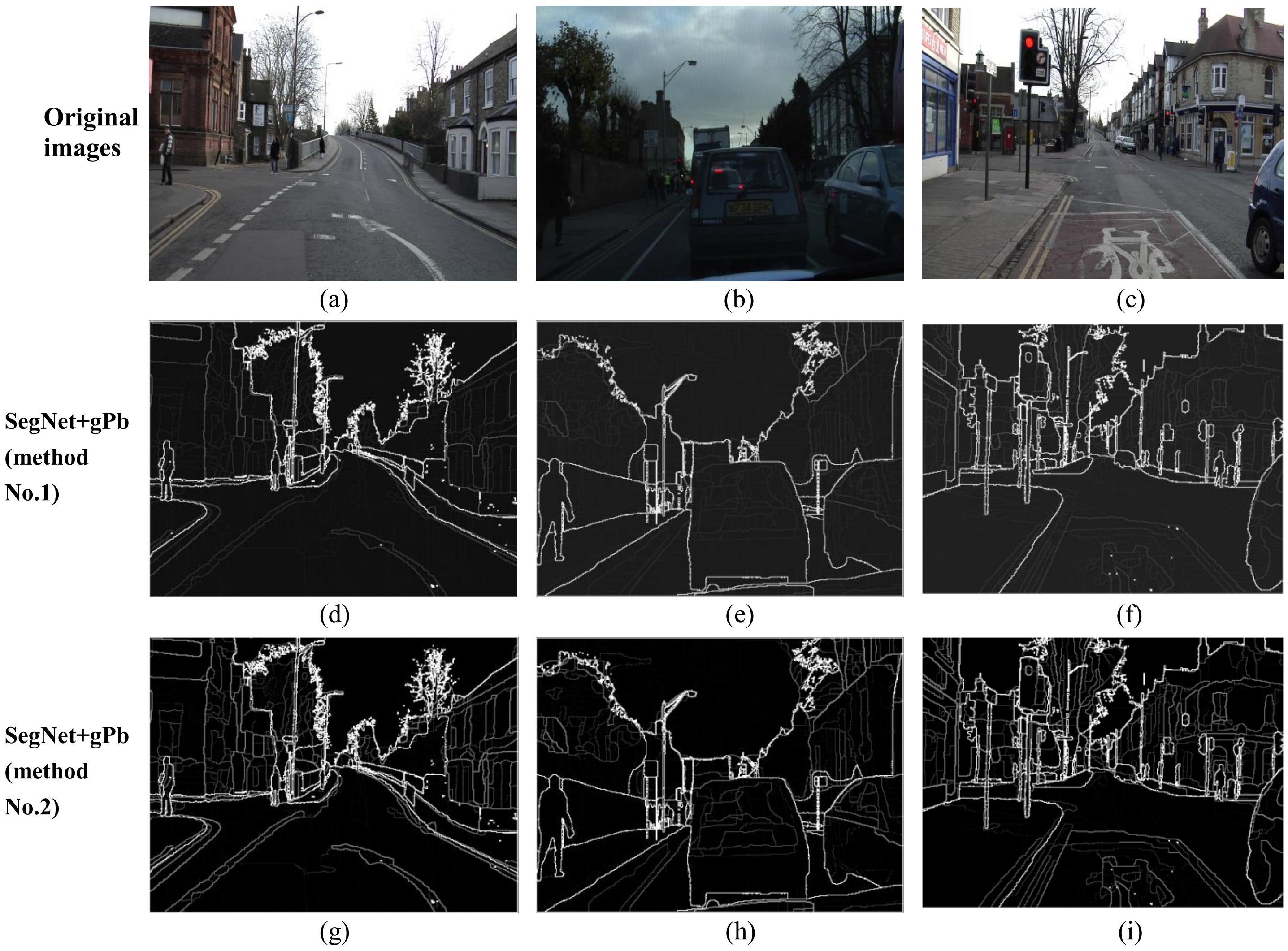


Figure 14: final segmentation results by combining SegNet and gPb. (a),(b),(c) are input images. (d),(e),(f) are final results using method No.1. (g),(h),(i) are final results using method No.2.

According to Figure 14, the final segmentation results of combining SegNet and gPb in both methods are acceptable. The evaluation and comparison of them will be mentioned in next chapter.

B. SegNet + region growing

As the region growing algorithm designed in this project is not hierarchical segmentation, only the first combining method is practicable to combine SegNet and region growing. Some example results will be listed as below. The corresponding SegNet's results are also demonstrated.

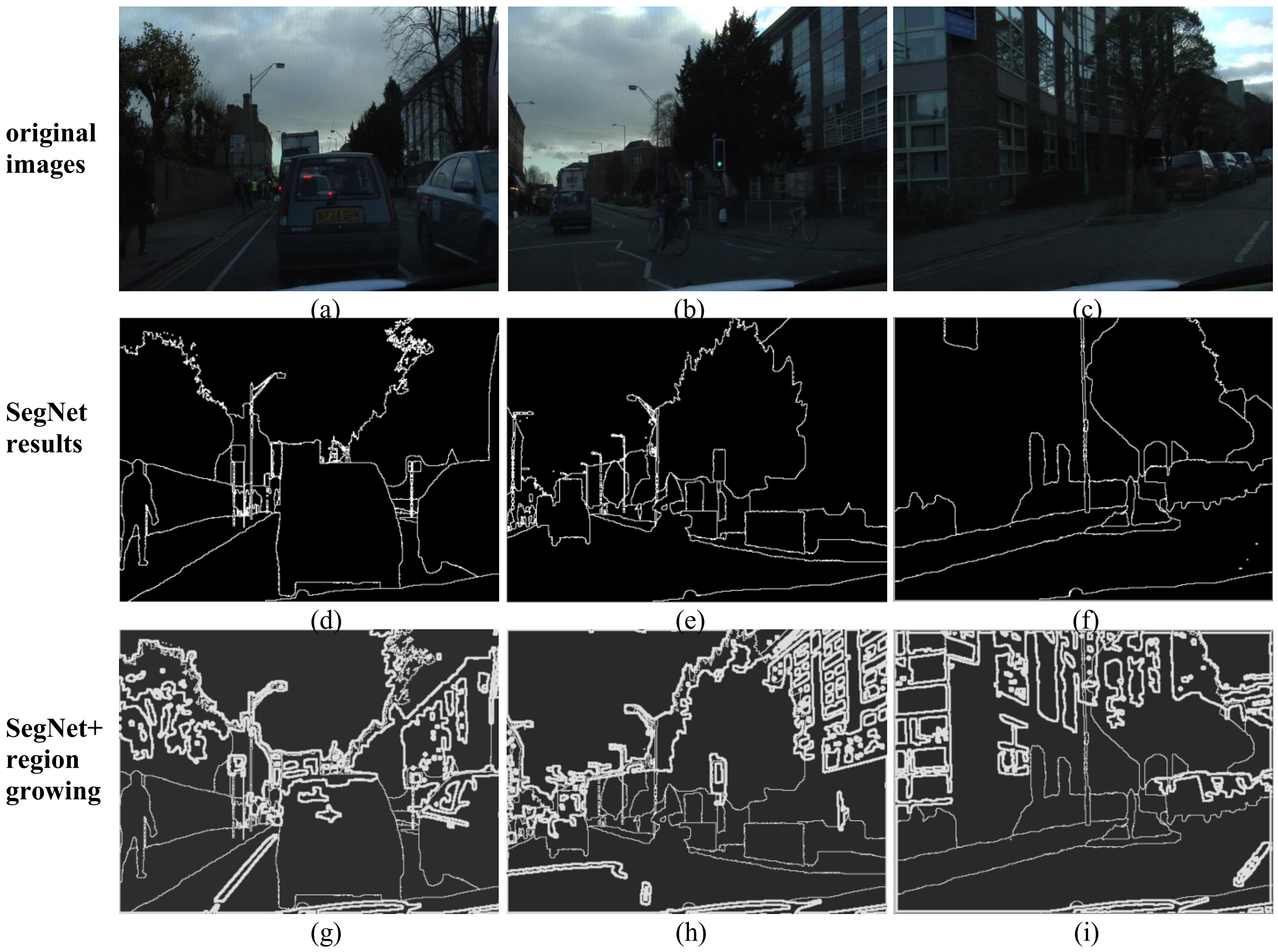


Figure 15: final segmentation results by combining SegNet and region growing. (a),(b),(c) are input images. (d),(e),(f) are segmentation results produced by SegNet. (g),(h),(i) are final results.

From Figure 15, it is clear that the results of combining SegNet and region growing can add many inner edges to the results produced by running SegNet alone. For example, in Figure 15(h),(i), the windows of buildings are depicted, whereas Figure 15 (c),(f) just detect the outer contours of the buildings. However, region growing algorithm adds many inaccurate edges as well, which makes the final segmentation result looks a bit messy. The overall performance of SegNet + region growing is worse than the combination of SegNet and gPb, the evaluation of it will be mentioned in the next chapter.

4 Chapter 4

Evaluation

4.1 Overview

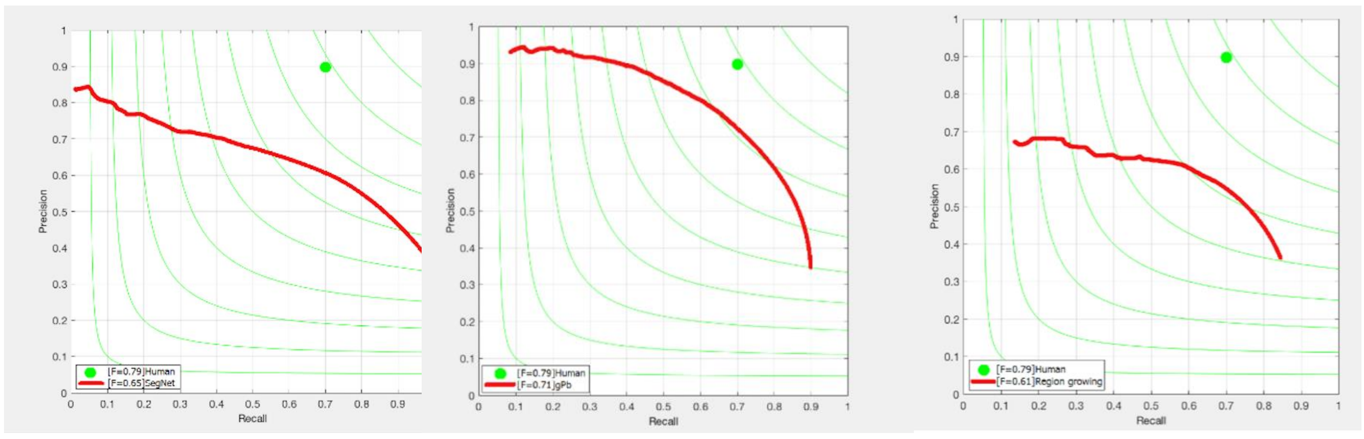
In the last chapter, results for three individual segmentation algorithms are obtained. Moreover, three kinds of final segmentation results for combining two individual algorithms are also generated. The evaluation and comparison of their performances will be discussed in this chapter.

4.2 Evaluation results & analyze

The Berkeley Segmentation Dataset and Benchmark(BSDS) is a standard protocol which allows the comparison of many different boundary detection or segmentation algorithms by quantifying their performance with respect to human vision[12]. Basically, the comparison of machine-detected contours with human-marked contours is based on the Precision-Recall framework, which is a standard information retrieval technique[12]. It means two quality measures will be considered in the evaluation. One is Precision(P), given by the fraction of detections that are real contours in the images[12]. Another is Recall(R), defined as the fraction of real contours have been detected[12]. The output of BSDS for each algorithm is a Precision-Recall curve after quantifying a set of images. Two quality measures will be combined in a single quality measure: F-measure. It is defined as:

$$F(P, R) = \frac{2PR}{P+R}.$$

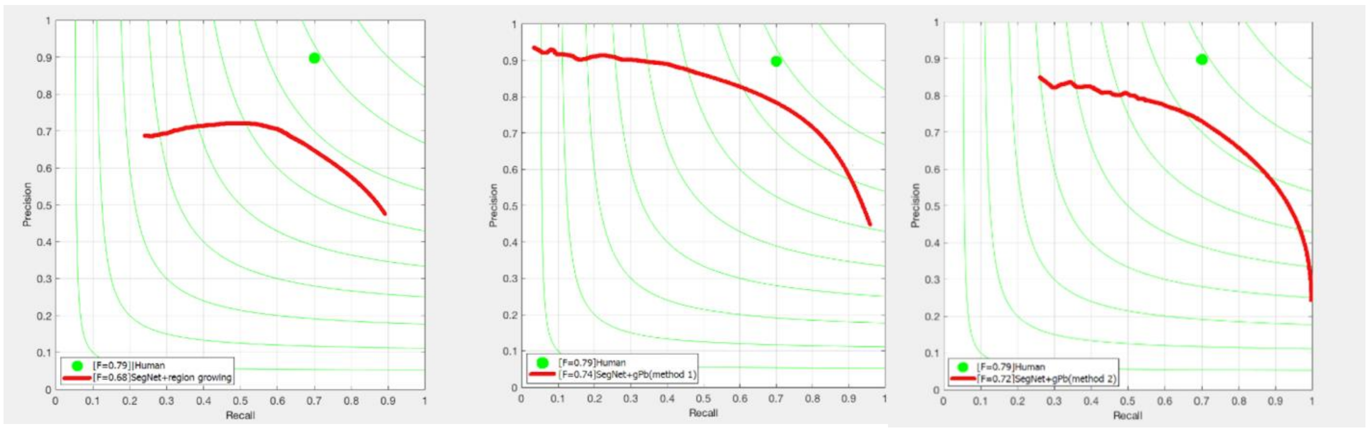
Benchmark BSDS500 is used in this project for evaluation, which contains 300 training images and 200 test images. First, test the performance of three individual algorithms on the benchmark. The outputs are:



(a) (b) (c)

Figure 16: Evaluation three individual algorithms on the BSDS500. (a) SegNet (F=0.65) (b) gPb (F=0.71) (c) region growing (F=0.61)

Then, use benchmark BSDS500 to evaluate three kinds of combination results, which are:
(a) Combination of SegNet and region growing
(b) Combination of SegNet and gPb using method No.1
(a) Combination of SegNet and gPb using method No.2
The evaluation results are illustrated as:



(a) (b) (c)

Figure 17: Evaluation combined results on the BSDS500. (a) SegNet+region growing (F=0.68) (b) SegNet+gPb by method No.1 (F=0.74) (c) SegNet+gPb by method No.2 (F=0.72)

To compare these 6 evaluation results more directly, a Table will be created：

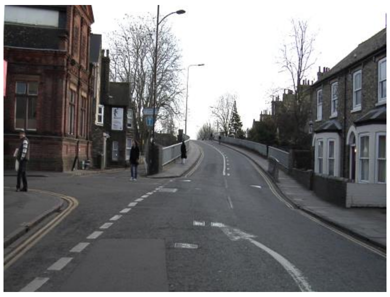
		individual algorithm			combined algorithm		
algorithm	Human vision	SegNet	gPb	region growing	SegNet + region growing	SegNet+gPb (method No.1)	SegNet+gPb (method No.2)
F-measure	0.79	0.65	0.71	0.61	0.68	0.74	0.72

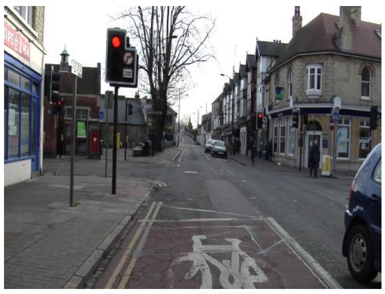
Table 1: Evaluation results for all algorithms on the BSDS500

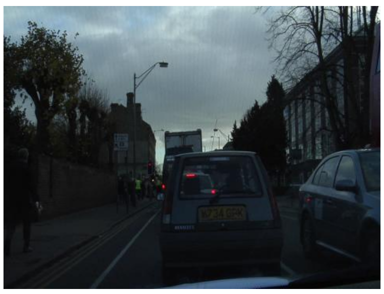
According to Table 1, all the final segmentation results which combined two individual algorithms (one top-down approach and one bottom-up approach) can improve the performance of using either one approach alone. For example, the combination of SegNet and gPb using method No.1 has the F-measure of 0.74, but if using SegNet or gPb alone will get 0.65 or 0.71 F-measures. For the combination of SegNet and region growing, the F-measure is 0.68. Although this value is lower than singly using gPb, it is higher than using SegNet (F=0.65) or region growing (F=0.61) alone.

The approach with the best performance should be SegNet + gPb using method 1, which is overlapping the results of SegNet and gPb directly. Using a threshold value to overlap them also performs well. The F-measures for these two approaches are close (one is 0.74 and another is 0.72), so some results of them will be listed together for comparison.

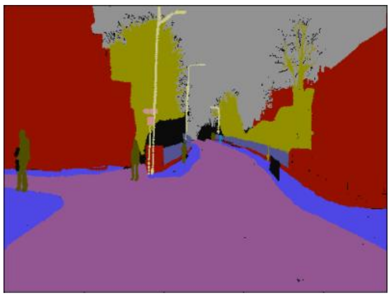
Original images

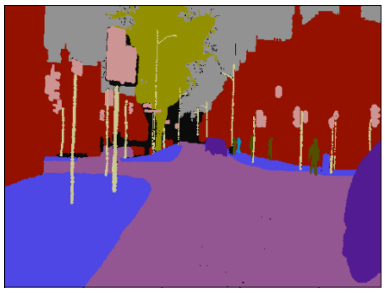

(a)

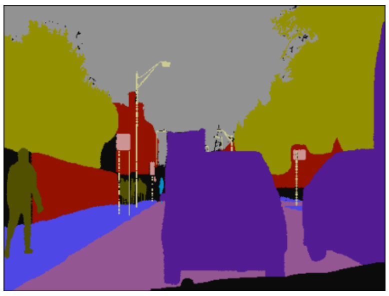

(b)


(c)

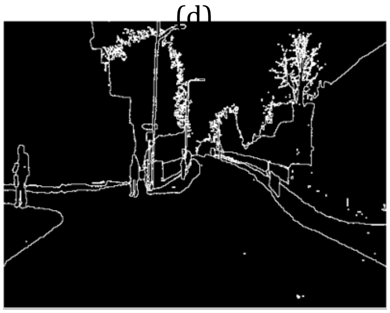
Original SegNet results

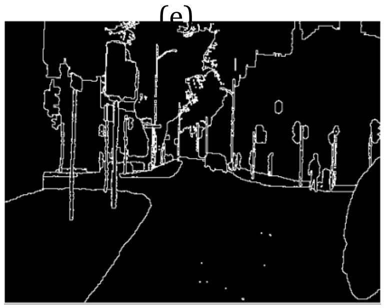

(d)

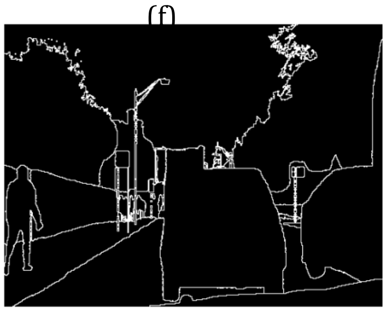

(e)


(f)

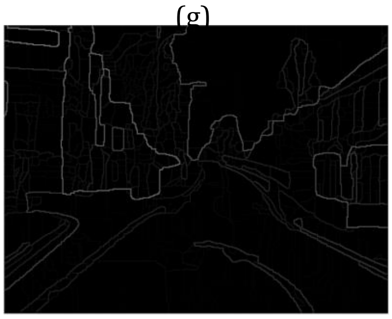
Contour images for SegNet

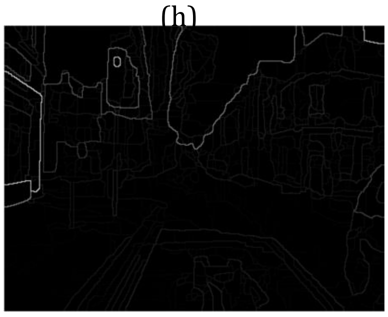

(g)

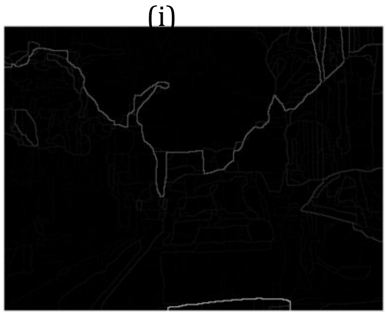

(h)


(i)

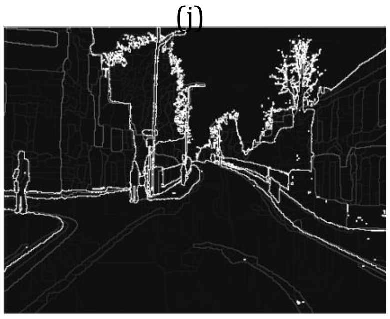
gPb results

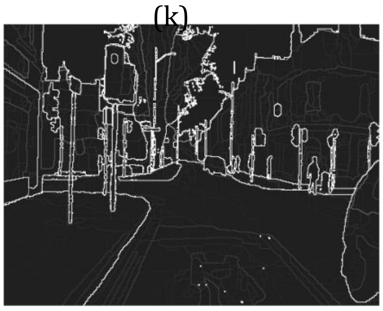

(j)

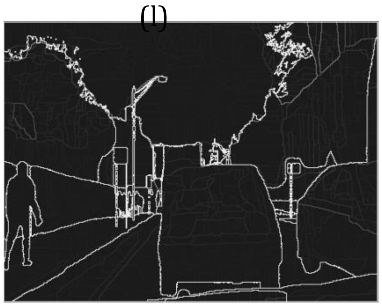

(k)


(l)

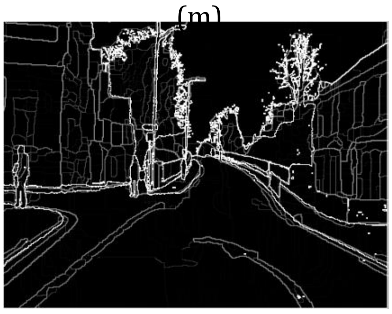
SegNet+ gPb (method No.1)

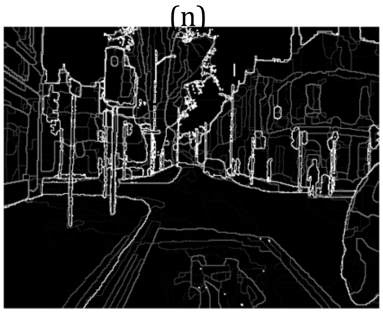

(m)


(n)


(o)

SegNet+ gPb (method No.2)


(p)


(q)

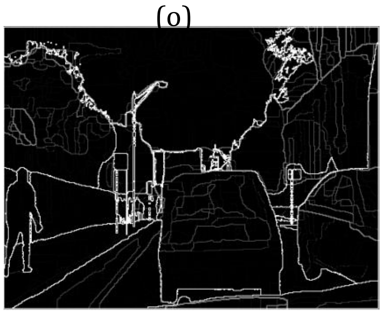

(r)

Figure 18: Comparison of some approaches. (a),(b),(c) are original input images. (d),(e),(f) are the original segmentation results of SegNet. (g),(h),(i) are contour images of SegNet's results. (i),(k),(l) are segmentation results of gPb. (m),(n),(o) are combined results of SegNet + gPb use method 1. (p), (q), (r) are combined results of SegNet + gPb use method 2.

Some advantages and disadvantages of SegNet algorithm can be found from Figure 18. First, the boundaries between objects detected in SegNet are clearly, completed and accurate. It can detect all objects in an image. Moreover, SegNet can get realizable imagination results in even an extremely low-brightness image, such as Figure 18 (c). For the reason that it can map low-resolution features to a higher resolution for classification. However, SegNet also has some drawbacks. The original segmentation results of SegNet are labelled in 11 different colors for different classes. Therefore, only the outer contours of objects will be depicted after transforming them into the contour images as shown in Figure 18 (g),(h),(i). For example, it can detect the location of the buildings but it cannot locate the windows.

Compared to SegNet's results, gPb shows worse performance in a low-brightness image and some of the contours in its results are incomplete. However, the overall performance of gPb is better than SegNet. The gPb is a hierarchical segmentation algorithm, it can show edges in different shade of colors. It means if an edge has high possibility to be a true boundary, it will show in the result clearly. Oppositely, an edge will show in a light shade in the result if its possibility is low. In addition, gPb can not only detect the outer contours of an object or the boundaries between two objects, it also can find the inner edges of an object. For example, in Figure 18 (k), the windows of buildings are depicted. In Figure 18 (j), (k) can detect the traffic instructions on the roadway.

The final combination results can add the advantages of two individual approaches together. The final results contain both the inner edges of objects and boundaries between objects. For this project is aiming for segmenting images for road scenes. The final results indeed enhance the performance of understanding road scenes and improve the precision of segmentation. For example, the traffic instructions on the roadway are detected in the combined results, which will be useful if the algorithm is applied for an automatic driving system. The difference between these two kinds of combined results is that Figure 18 (m),(n),(o) shows all the inner edges in the same shade of color, whereas Figure 18 (p), (q), (r) shows inner edges in different shade of colors. The advantage for using different shade of colors is that many probable boundaries will be added clearly. However, some false boundaries will be shown in the final results clearly as well.

5 Chapter 5
Conclusion

To sum up, one existing top-down segmentation approach called SegNet and one bottom-up segmentation approach named gPb were implemented through this project. One bottom-up approach using region-growing and canny were designed, although it does not have high performance. Moreover, some feasible methods have been explored for combining the top-down and bottom-up segmentation approaches. The benchmark evaluation results show that the final segmentation results produced by these combination methods improve the performance of the individual algorithm.

This report introduced the background and the rationale of this project at the beginning and then provided a literature review about some image segmentation approaches. The methodologies for implementing individual approaches and the combination process of two algorithms were detailed described followed with their segmentation results. All the segmentation results obtained in this project were evaluated and discussed in Chapter 4.

The final segmentation results produced by this project achieved high levels of performance. It can be applied for automatic driving systems. Some further work can be carried out. For example, improve the algorithm to do image segmentation in real-time. Also, extend the algorithm to be efficient for other kinds of images, not only the road scenes images.

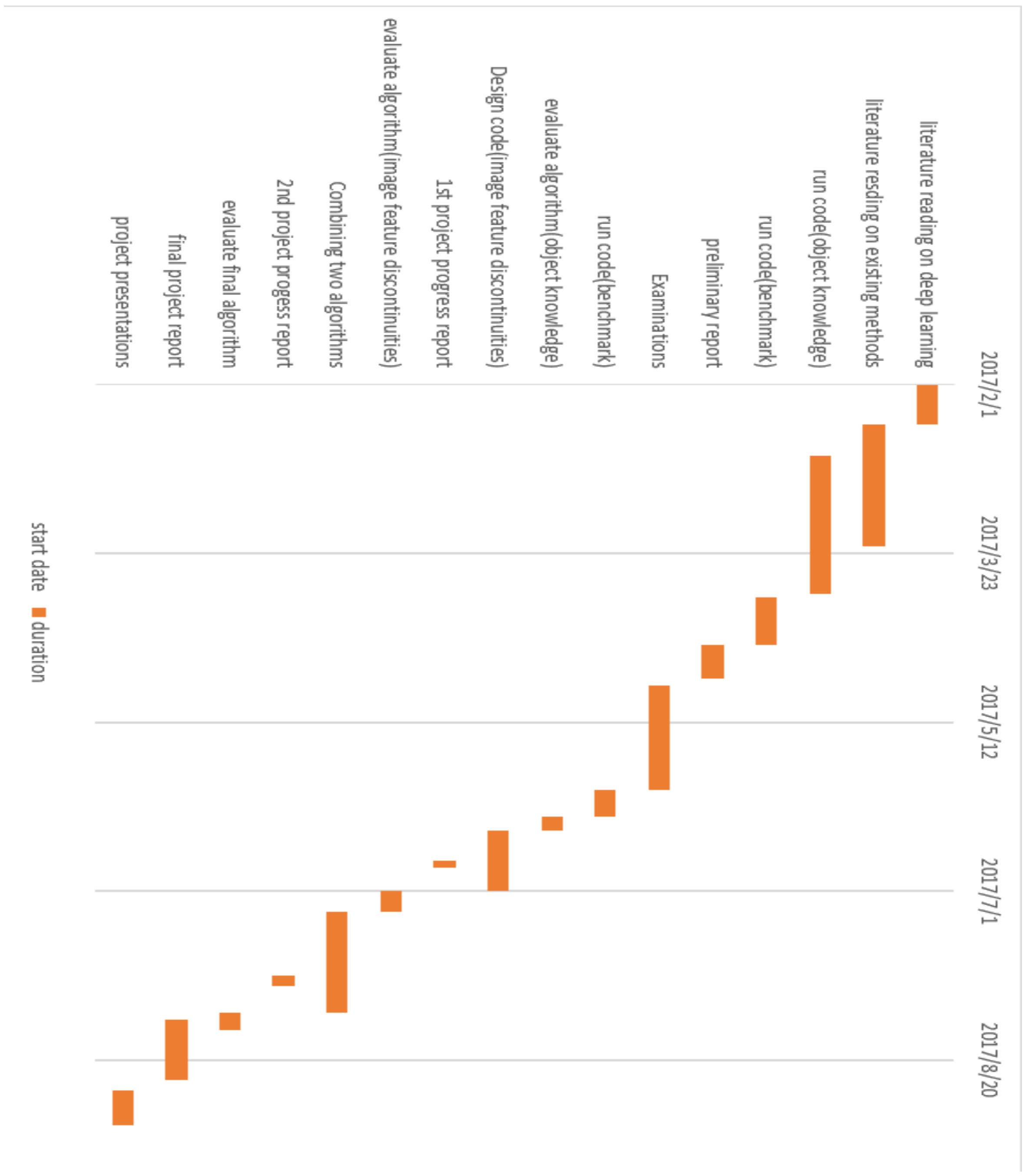
References:

[1] Linda G. Shapiro and George C. Stockman(2001): “Computer Vision”, pp 279-325, New Jersey, Prentice-Hall, ISBN 0-13-03076-3
[2] Soler, L., Devinette, H., Malandrin, G., et. al(2001):“Computer Aided Surgery”, 6(3), pp. 131-142,
Fully automatic anatomical, pathological, and functional segmentation from CT scans for hepatic surgery.
[3] Grigorescu, C., Petkov, N., and Westenberg, M. A. (2004). Contour and boundary detection improved by surround suppression of texture edges. Image Vis. Comput., 22(8):609–22.
[4] (2008): “Image segmentation”
[onine]http://www.cs.toronto.edu/~jepson/csc2503/segmentation.pdf
[5] (2004): “Chapter 10 Image segmentation”
[onine]https://courses.cs.washington.edu/courses/cse576/book/ch10.pdf
[6] Barghout, Lauren&Lawrence W.Lee.(2003): “Perceptual information processing system.” Parade Inc. U.S. Patent Application 10/618,543
[7] Borenstein, Sharon & Ullman (2003): “Combining Top-down and Bottom-up Segmentation” [online]http://www.dam.brown.edu/people/eranb/Combining_td_and_bu.pdf
[8] A. Needham and R. Baillargeon(1998): “Effects of prior experience in 4.5-month-old infants object segmentation”, vol.21, pp. 1-24, Infant Behavior and Development
[9] M. Peterson and B. Gibson(1993): “Shape recognition contributions to figure-ground organization in three-dimensional displays”,vol.25, pp. 383-429, Cognitive Psychology.
[10] E. Sharon, A. Brandt, and R. Basri(2001):“Segmentation and boundary detection using musicale intensity measurements”, CVPR(1), pp. 469-476
[11] E. Borenstein and S. Ullman(2002):“Class-specific, top-down segmentation ”, ECCV(2), pp. 109-124
[12] Badrinarayanan V., and Kendall A.: ”Segnet: A Deep Convolutional Encoder-Decoder Architecture for image segmentation” IEEE trans. 1511.00561v3. 2016
[13] Norouzi A., Uddin M, “Medical Image segmentation Methods, Algorithms, and applications”, IETE Technical Review 31(3): 199-213,(2014), [Online] https://www.researchgate.net/publication/263608069_Medical_Image_Segmentation_Methods_Algorithms_and_Applications
[14] Gonzalez: “Digital image processing”, pp12-14, 194-199
[15] Martin, D. R., Fowlkes, C. C., and Malik, J. (2004). Learning to detect natural image boundaries using local brightness, color and texture cues. IEEE Trans. Pattern Anal. Mach. Intell., 26(5):530–49.
[16] Sezgin (2007): “Survey over image thresholding techniques and quantitative performance evaluation” [Online]http://pequan.lip6.fr/~bereziat/pima/2012/seuillage/sezgin04.pdf
[17] (2017): “slides3.2 segmentation artificial”, King’s College London, [Online]https://keats.kcl.ac.uk/pluginfile.php/1875516/mod_resource/content/2/slides3.2_segmentation_artificial.pdf
[18] Tizhoosh, H. R., 2005. Image thresholding using type II fuzzy sets.
Pattern Recognition, 38(12), pp. 2363-2372.
[19] Jian-Jiun Ding, The class of "Advanced Digital Signal Processing", the Department of Electrical Engineering, National Taiwan University (NTU), Taipei, Taiwan, 2008.
[20] Jain(2002): “Region growing”,
[Online]https://www.cse.unr.edu/~bebis/CS791E/Notes/RegionGrowing.pdf

[21] Ren, X. (2008). Multi-scale improves boundary detection in natural images. In Proc. European Conf. Comput. Vision
[22] Canny,J.: "A computational approach to edge detection". IEEE trans. Pattern Analysis and Machine Intelligence, Vol. 8, 679-698,1986
[23] Ren, X. (2008). Multi-scale improves boundary detection in natural images. In Proc. European Conf. Comput. Vision
[24] Arbelaez, Maire (2013): "Contour Detection and Hierarchical Image Segmentation" [Online]https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/papers/amfm_pami2010.pdf
[25] SegNet official website [Online]http://mi.eng.cam.ac.uk/projects/segnet/#publication
[26] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, pp. 3431–3440, 2015.
[27] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *ICCV*, pp. 1520–1528, 2015.
[28] Chaurasia A., Kim S.:"ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation" , Purdue University, 1606.02147v1, 2016
[29] Caffe official website, [Online] http://caffe.berkeleyvision.org/tutorial/
[30] Brostow, Gabriel J., Julien Fauqueur, and Roberto Cipolla. "Semantic object classes in video: A high-definition ground truth database." Pattern Recognition Letters 30.2 (2009): 88-97.
[31] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).

Appendices

1. Gantt chart



2. Source code

1). Convert results of SegNet into contour images

```
clear all;
close all;

[filename pathname] = uigetfile('*.png');
filename = strcat(pathname, filename);


I = imread(filename);
figure;
imshow(I);
title('input');


I2 = rgb2gray(I);
figure;
imshow(I2);
title('gray');

[m n] = size(I2);
map = zeros(m, n);

for i = 2:m-1
   for j = 2:n-1
      if I2(i,j) == I2(i-1,j-1) && I2(i,j) == I2(i-1,j) && I2(i,j) == I2(i-1,j+1)...
            && I2(i,j) == I2(i,j-1) && I2(i,j) == I2(i,j+1)...
            && I2(i,j) == I2(i+1,j-1) && I2(i,j) == I2(i+1,j) && I2(i,j) == I2(i+1,j+1)
         map(i,j) = 0;
      else
         map(i,j) = 1;
      end

   end
end
figure;
imshow(map);
title('contour');


save map map;
```

2). region growing

```
close all;
clear all;
```

```
A0=imread('2.png');
seed=[100,220];
thresh=15;
A=rgb2gray(A0);

A=imadjust(A,[min(min(double(A)))/255,max(max(double(A)))/255],[]);
A=double(A);
B=A;
[r,c]=size(B);
n=r*c;
pixel_seed=A(seed(1),seed(2));
q=[seed(1) seed(2)];
top=1;
M=zeros(r,c);
M(seed(1),seed(2))=1;
count=1;

while top~=0
r1=q(1,1);
c1=q(1,2);
p=A(r1,c1);
dge=0;
for i=-1:1
for j=-1:1
if r1+i<=r & r1+i>0 & c1+j<=c & c1+j>0

if abs(A(r1+i,c1+j)-p)<=thresh & M(r1+i,c1+j)~=1
top=top+1;
q(top,:)=[r1+i c1+j];
M(r1+i,c1+j)=1;
count=count+1;
B(r1+i,c1+j)=1;
end
if M(r1+i,c1+j)==0;
dge=1;
end
else
dge=1;
end
end
end
if dge~=1
B(r1,c1)=A(seed(1),seed(2));
end
if count>=n
top=1;
end
q=q(2:top,:);
```

```
top=top-1;
end
subplot(2,2,1),imshow(A,[]);
subplot(2,2,2),imshow(B,[]);

E = getE(B);
se = strel('disk', 1);
E = imclose(E, se);
% E = imfill(E, 'holes');
E = bwareaopen(E, 200); %30
subplot(2,2,4), imshow(E, []);

E2 = edge(E, 'canny');
figure, imshow(E2);
E3 = getE(E);
figure, imshow(E3);


map2 = getMap(E2);
save map2 map2;

map3 = getMap(E3);
save map3 map3;

function map = getMap(I2)
[m n] = size(I2);
map = zeros(m, n);

for i = 2:m-1
   for j = 2:n-1
      if I2(i,j) == I2(i-1,j-1) && I2(i,j) == I2(i-1,j) && I2(i,j) == I2(i-1,j+1)...
            && I2(i,j) == I2(i,j-1) && I2(i,j) == I2(i,j+1)...
            && I2(i,j) == I2(i+1,j-1) && I2(i,j) == I2(i+1,j) && I2(i,j) == I2(i+1,j+1)
         map(i,j) = 0;
      else
         map(i,j) = 1;
      end

   end
end
% figure;
% imshow(map);

function E = getE(I2)

   [m n] = size(I2);
   E = zeros(m, n)
```

```
    for i = 2:m-1
        for j = 2:n-1
            if I2(i,j) == I2(i-1,j-1) && I2(i,j) == I2(i-1,j) && I2(i,j) == I2(i-1,j+1)...
                    && I2(i,j) == I2(i,j-1) && I2(i,j) == I2(i,j+1)...
                    && I2(i,j) == I2(i+1,j-1) && I2(i,j) == I2(i+1,j) && I2(i,j) == I2(i+1,j+1)
                E(i,j) = 0;
            else
                E(i,j) = 1;
            end

        end
    end



end
```

3). SegNet+gPb (method No.1)

```
clear all;
close all;


[filename pathname] = uigetfile('*.png');
filename2 = filename;

filename = strcat(pathname, filename);


I = imread(filename);
figure;
imshow(I);




I2 = rgb2gray(I);
figure;
imshow(I2);


[m n] = size(I2);
map = zeros(m, n);

for i = 2:m-1
    for j = 2:n-1
        if I2(i,j) == I2(i-1,j-1) && I2(i,j) == I2(i-1,j) && I2(i,j) == I2(i-1,j+1)...
                && I2(i,j) == I2(i,j-1) && I2(i,j) == I2(i,j+1)...
```

```
            && I2(i,j) == I2(i+1,j-1) && I2(i,j) == I2(i+1,j) && I2(i,j) == I2(i+1,j+1)
            map(i,j) = 0;
        else
            map(i,j) = 1;
        end

    end
end
figure;
imshow(map);


% save map map;
map0 = map;


[filename3 pathname3] = uigetfile('*.mat');
load(filename3);
mask = ucm;
mask = imresize(mask, [m n], 'bicubic');
% mask = im2bw(mask, graythresh(mask));
figure;
imshow(mask);

for i=1:m
    for j=1:n
        if mask(i,j)<0
            mask(i,j)=0;
        end
    end
end

rt = zeros(m, n);
for i = 1:m
    for j = 1:n
        if map0(i,j) == 0 && mask(i,j) == 0
            rt(i,j) = 0;
        elseif map0(i,j) == 1 && mask(i,j) == 1
            rt(i,j) = 1;
        elseif map0(i,j) == 1 && mask(i,j) ~= 0
            rt(i,j) = 1;
        elseif mask(i,j) == 1 && map0(i,j) ~= 0
            rt(i,j) = 1;
        elseif mask(i,j) == 0 && map0(i,j) ~= 0
            rt(i,j) = map0(i,j);
        elseif map0(i,j) == 0 && mask(i,j) ~= 0
            rt(i,j) = mask(i,j);
        end
    end
end

figure;
imshow(rt, []);
title('result');

save rt rt;
```

4). SegNet+gPb (Method No.2)

```
clear all;
close all;


[filename pathname] = uigetfile('*.png');
filename2 = filename;

filename = strcat(pathname, filename);


I = imread(filename);
figure;
imshow(I);



I2 = rgb2gray(I);
figure;
imshow(I2);


[m n] = size(I2);
map = zeros(m,n);
for i = 2:m-1
    for j = 2:n-1
        if I2(i,j) == I2(i-1,j-1) && I2(i,j) == I2(i-1,j) && I2(i,j) == I2(i-1,j+1)...
                && I2(i,j) == I2(i,j-1) && I2(i,j) == I2(i,j+1)...
                && I2(i,j) == I2(i+1,j-1) && I2(i,j) == I2(i+1,j) && I2(i,j) == I2(i+1,j+1)
            map(i,j) = 0;
        else
            map(i,j) = 1;
        end

    end
end
figure;
imshow(map);
```

```
% save map map;
map0 = map;


[filename3 pathname3] = uigetfile('*.mat');
load(filename3);
mask = ucm;
mask = imresize(mask, [m n], 'bicubic');
% mask = im2bw(mask, graythresh(mask));
figure;
imshow(mask);


for i=1:m
   for j=1:n
      if mask(i,j)<0
         mask(i,j)=0;
      end
   end
end


rt = zeros(m, n);
for i = 1:m
   for j = 1:n
      if map0(i,j) == 0 && mask(i,j) == 0
         rt(i,j) = 0;
      elseif map0(i,j) == 1 && mask(i,j) == 1
         rt(i,j) = 1;
      elseif map0(i,j) == 1 && mask(i,j) ~= 0
         rt(i,j) = 1;
      elseif mask(i,j) == 1 && map0(i,j) ~= 0
         rt(i,j) = 1;
      elseif mask(i,j) == 0 && map0(i,j) ~= 0
         rt(i,j)= map0(i,j);
      elseif map0(i,j) == 0 && mask(i,j) ~= 0
         if mask(i,j)<=0.05
            rt(i,j)=mask(i,j);
         end
         if mask(i,j)>0.05 && mask(i,j)<=0.1
            rt(i,j)=0.2;
         end
         if mask(i,j)>0.1 && mask(i,j)<=1
            rt(i,j)=0.7;
         end
```

```
        end
    end
end

figure;
imshow(rt, []);


save rt rt;
```

5). SegNet+region growing

```
clear all;
close all;

[filename pathname] = uigetfile('*.png');
filename2 = filename;

filename = strcat(pathname, filename);


I = imread(filename);
figure;
imshow(I);

I2 = rgb2gray(I);
figure;
imshow(I2);

[m n] = size(I2);
map = zeros(m, n);

for i = 2:m-1
    for j = 2:n-1
        if I2(i,j) == I2(i-1,j-1) && I2(i,j) == I2(i-1,j) && I2(i,j) == I2(i-1,j+1)...
                && I2(i,j) == I2(i,j-1) && I2(i,j) == I2(i,j+1)...
                && I2(i,j) == I2(i+1,j-1) && I2(i,j) == I2(i+1,j) && I2(i,j) == I2(i+1,j+1)
            map(i,j) = 0;
        else
            map(i,j) = 1;
        end

    end
end
figure;
imshow(map);
```

```
% save map map;
map0 = map;


[filename3 pathname3] = uigetfile('*.mat');
load(filename3);
mask = map2;
mask = imresize(mask, [m n], 'bicubic');
% mask = im2bw(mask, graythresh(mask));
figure;
imshow(mask);

for i=1:m
    for j=1:n
        if mask(i,j)<0
            mask(i,j)=0;
        end
    end
end

rt = zeros(m, n);
for i = 1:m
    for j = 1:n
        if map0(i,j) == 0 && mask(i,j) == 0
            rt(i,j) = 0;
        elseif map0(i,j) == 1 && mask(i,j) == 1
            rt(i,j) = 1;
        elseif map0(i,j) == 1 && mask(i,j) ~= 0
            rt(i,j) = 1;
        elseif mask(i,j) == 1 && map0(i,j) ~= 0
            rt(i,j) = 1;
        elseif mask(i,j) == 0 && map0(i,j) ~= 0
            rt(i,j) = map0(i,j);
        elseif map0(i,j) == 0 && mask(i,j) ~= 0
            rt(i,j) = mask(i,j);
        end
    end
end

figure;

imshow(rt, []);

save rt rt;
```