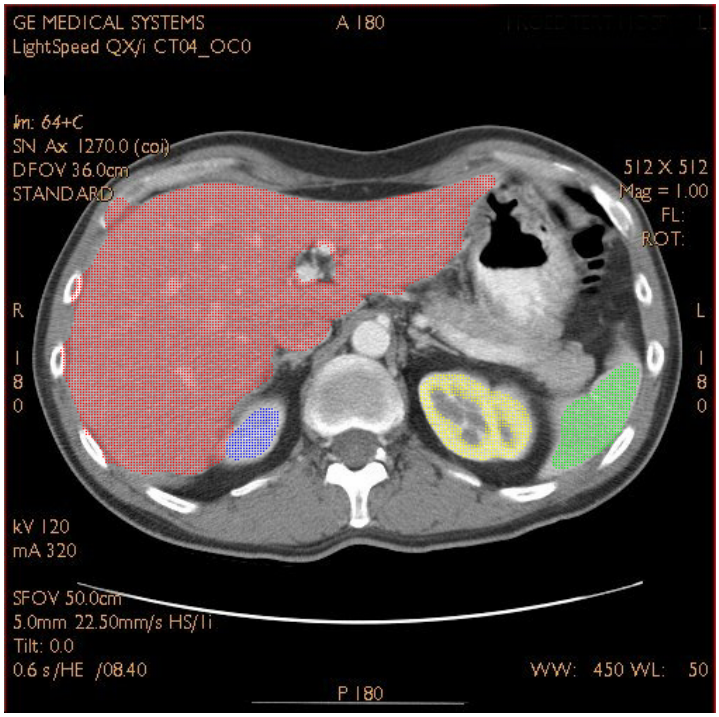
18-551 – Digital Communication & Signals Processing Systems Design

Group 5, Spring 2004



FINAL REPORT

What's in YOUR tummy?
Organ Segmentation and Identification on an Abdominal Computer Tomography Image

Jin Qian (jinq@andrew.cmu.edu)

Faisal Luqman (fbl@andrew.cmu.edu)

Varun Bandi (vbandi@andrew.cmu.edu)

TABLE OF CONTENTS

1. INTRODUCTION

In our project, we designed a system that performs organ segmentation and identification of an abdominal Computer Tomography (CT) imagery.

More specifically, our project is designed to identify 4 major organs in an abdominal CT scan image: liver, left kidney, right kidney and the spleen. We implemented several methods and algorithms in the organ segmentation and identification process, such as median filtering, image segmentation, thresholding and blob labeling. In addition, we also employed three methods in our organ identification process, using the location of the organ, size of organ and center of mass. After the four organs have been identified on the image, the organs are then shaded with a unique color on the original image to make it easier for the user to identify them. Our project is implemented on a Texas Instrument C67 Digital Signal Processing Board, and the procedures and methods we used in our project is described in detail in this report.

1.1 *Problem description*

In the field of radiology, the slowest process in the radiotherapy planning process is organ identification. Currently, the procedure in which organs are identified on a CT image is by manually labeling them one image at a time. A clinician generally performs this process. As such, there is a clear need to be able to provide a quick and accurate identification of various organs in the human abdominal area.

1.2 Solution

We're presenting a solution to this problem by providing a fast an accurate method of identifying the four major organs on the EVM hardware. Our system reads in a CT image of the abdomen, and is able to identify the four major organs based on size, location and center of mass, and colors them with unique colors back on the original image.

This solution uses real time signal processing and accurate organ identification. Our system is able to identify the four major organs after it has been trained using a series of training sets of abdominal CT images.

1.3 Project Goal

The goal of our project is to provide some assistance to radiologist so that organ identification can be done rapidly and accurately. Our hope is that with a system that is able to automatically identify the four major organs in a CT image with a high level of accuracy, we are able to eliminate the manual classification of organs currently performed by clinicians. Hence, by removing this manual classification process, we are able to expedite the overall radiotherapy planning process.

2. PRIOR WORK

2.1 Previous 551 projects

Spring 2003, Group 13 did a similar project which involves classification of an MRI imagery of a brain into 3 different layers.

2.2 Other related work

A lot of research is being done in the medical area on automated organ segmentation. In addition to identifying organs, there is also various related research, such as modeling organs in 3D based on a CT/MRI imagery.

2.3 Uniqueness of our project

Our project differs from Spring 2003 Group 13's work on MRI brain segmentation in three different aspects: input imagery, methods and algorithms employed, and overall purpose of the project. First of all, the Group 13 of Spring 2003 used an MRI (Magnetic Resonance Imaging) image of the brain as the input image, while we are using CT (Computer Tomography) imagery in our 551 project. More importantly, the algorithms and methods used in our project differ entirely from the previous years project. While they are using binary mapping and K-means Clustering in their methodology, we are applying a mean-shift segmentation algorithm to segment our abdominal image to several distinct organs.

In addition, the overall purpose of our project is different from theirs. The purpose of their project was to break down an MRI image of a brain (one organ) into several different layers. On the other hand, our project is intended to identify several major organs on a CT scan image.

Previous work	Our project
MRI image of brain	CT scan of abdomen
Binary Mapping and Kmeans Clustering to segment images	Applied a mean-shift algorithm to segment images
Breaking down brain matter into several different layers	Identifying several major organs in the abdomen

Figure 1: Comparison with previous 551 projects

3. CODE

3.1 Available code

We were able to obtain the mean-shift segmentation algorithm from the "Edge Detection and Image Segmentation" (EDISON) system from Rutgers University, New Jersey. Dr. D. Comanicu, who was one of the authors of the algorithm, assisted us in obtaining the code.

The mean shift segmentation code is written in C++. A more detailed explanation of the mean shift algorithm is discussed in section 6.

Apart from the mean shift segmentation algorithm, we reused part of the code provided to us in lab 2 and lab 3 in our final project.

3.2 What code we had to write

Apart from the segmentation algorithm, we had to write all the other algorithms by ourselves, as most of these methods are unique to our project. We wrote code for the median filter, thresholding, blob labeling, organ identification methods, as well as organ coloring (details in section 6).

4. Dataset

4.1 Source of dataset

One of the weak points of our project was in obtaining a comprehensive database of images. The primary reason for this was because of the privacy issues that come with giving away medical images of patients, hence, many of the researchers and institutions from whom we requested for a database of images were unable to provide much help.

The General Electric Healthcare website did however, have a reasonable amount of images, and in the end, we choose to using their abdominal CT scan images for our project.

Figure 2: Abdominal CT scans from the General Electrics website

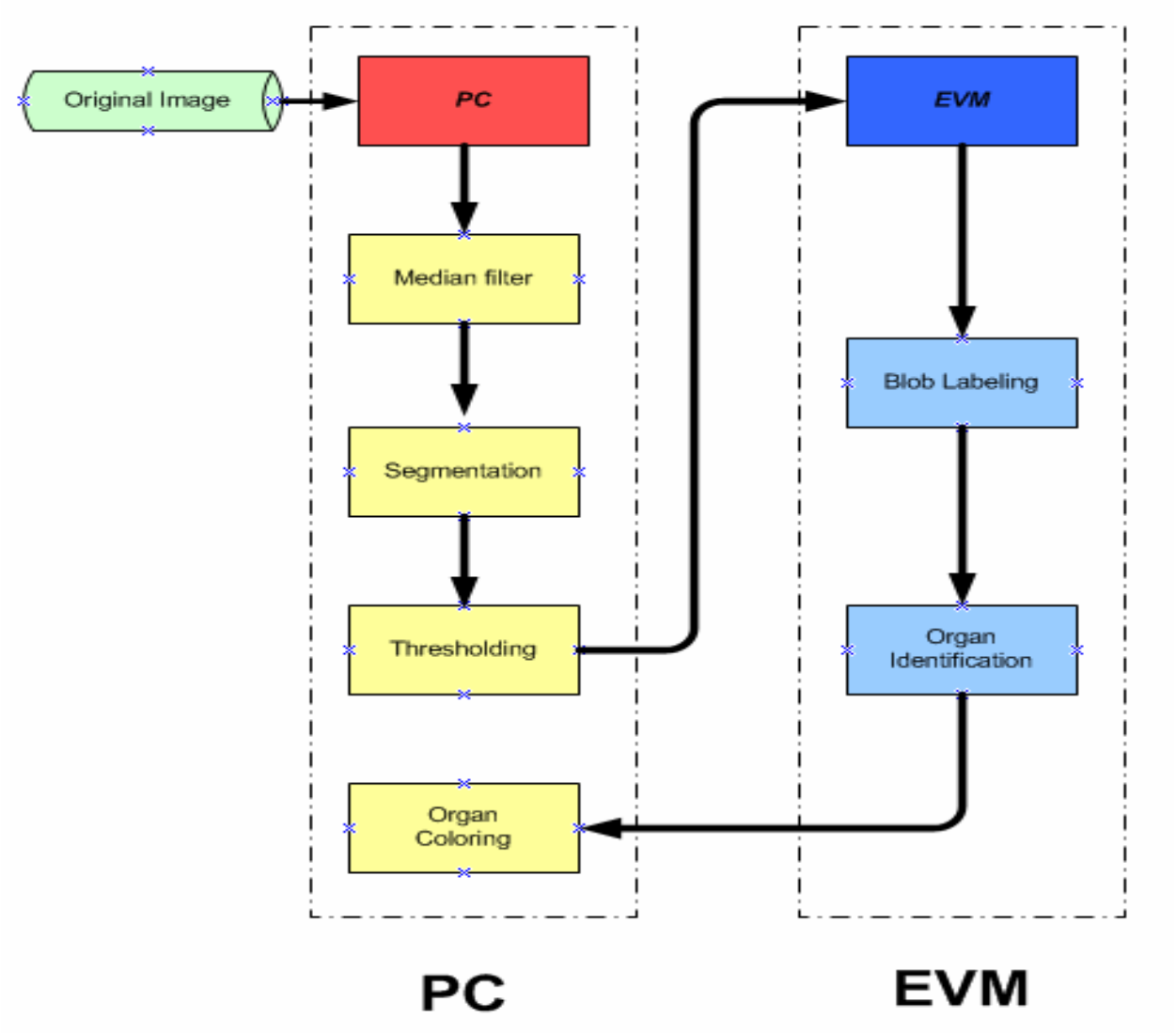
4.2 Training set/Test set

The entire database consisted of the scans of 6 different people. We then divided up the pictures randomly to assign from each person, some to the training set, and some to the test set. The major problem that we ended up with at the end was that we did not properly document which images went where. To the best of our knowledge, we picked the first half of the set of images of every person and assigned it to the training set, while leaving the other half for the test set.

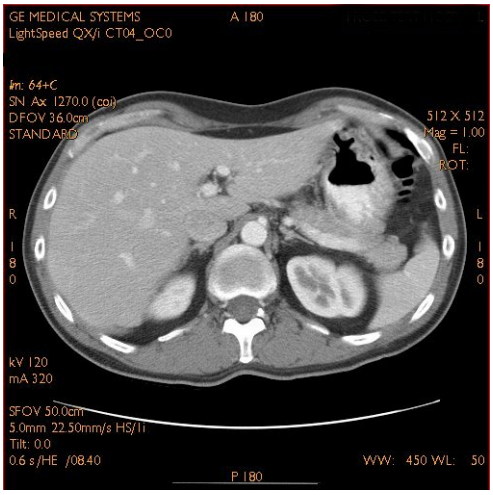
5. Overview of System

The overview of our system is illustrated in the Signal Flow Diagram below:

5.1 Signal Flow Diagram



6 ALGORITHM

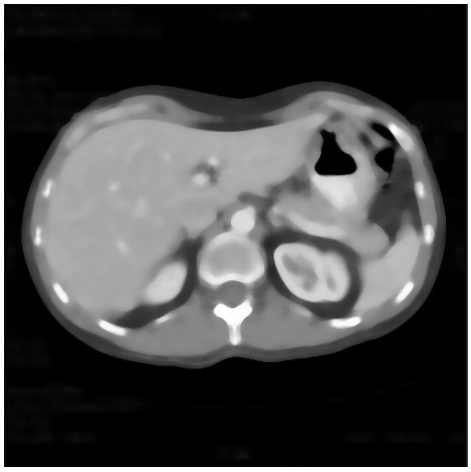


Original Image

6.1 Median Filter

The median filter was applied to our images to remove all extraneous text around the abdomen part of the image, as well as some noise in the abdomen part. The process is to go through the entire image pixel by pixel, and then look at the surrounding 11 by 11-square set of pixels. Of these pixels, we arrange their values in ascending order, and pick the median value and assign it to the same pixel in a new copy of the image. Repeat this process for every pixel in the original image, always using surrounding values from the original image. Note that we cannot traverse the pixels close to the border, as we do not have enough surrounding values. We simply set those pixels to black in the resulting image. However, one of the effects of using an 11 by 11-median filter is that the picture will lose some detail, and

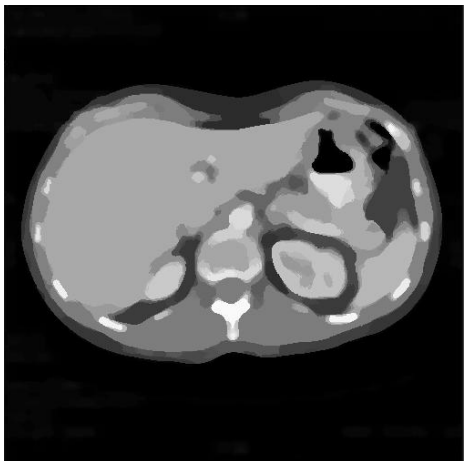
get blurry. However, this is not a problem since we the end result is to apply coloring to a copy of the original image, which will retain all its detail.



Mean Filtered Image

6.2 Segmentation

The next step was to apply the Mean Shift Algorithm to our image. This segments the image into distinct regions by setting a region that has similar color and intensity levels to the same level through out. The Mean Shift algorithm was one of the best algorithms we found online. We attempted to port the code over to our implementation, but it was too large to do so. We then decided to consider the segmentation step as pre-processing. We used the binary executable provided to us by the researchers at Rutgers to apply the segmentation before we ran the image through our code. The algorithm is fairly long and quite complex, so the explanation of the algorithm is to be looked up in the paper listed as a reference.

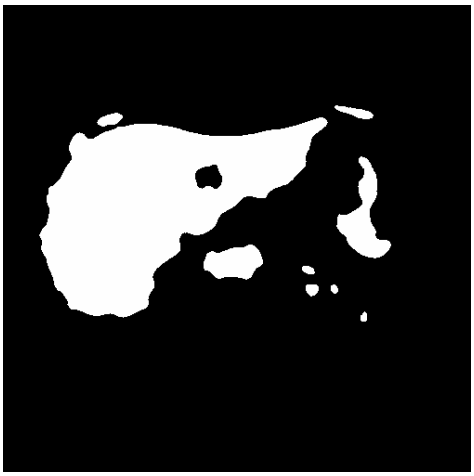


Segmented Image

6.3 Thresholding

To separate the major organs from the rest of the CT image, we used our training set to manually calculate threshold levels for each of the organs. Then, we used them to come up with three separate binary images that would contain blobs representing the organs. Later, we were told that this step was rather unnecessary, because of the segmentation step. Once we had segmented the image, each major region was colored the same. Therefore, we needed to separate the entire image into separate images based on a certain color and intensity. Then, we could simply pick the blob based on factors such as size and location, which we actually ended up using later in our process. In the actual code, we were using an image library provided to us from one of the 700 level courses in image processing. The explicit threshold values were calculated to fit along with the values that the library would set to different colors.

Thresholded Image 1



Thresholded Image 2

6.4 Blob Labeling

After we came up with three separate binary images, we needed to find the blobs in the picture. This is due to the fact that our thresholding levels were far from ideal, and sometimes we ended up with two organs in one image, and none in another. To do this, we used a recursive blob-labeling algorithm. The algorithm would look through the picture until it found an on pixel, and

after it did, it would then look for on pixels in the surrounding pixels, and so on and so forth. We were told that this was a simple algorithm to implement, but ridiculously inefficient. To counter the efficiency problems, we down-sampled the images by a factor of four, so we would not have to look through too many pixels.

6.5 Organ Identification

After labeling the blobs, we had to identify which blobs corresponded to the organs. We did this by considering the two major factors that are consistent in all the CT images in the training set: location and size. For example, to locate the liver, we looked for the biggest blob that was in the left half of the CT image. The EVM would then return a single pixel that was part of the liver.

Here are some of the explicit data values used in our code to aid in identification.

Liver:

size > 1000

location: Xmin<70, Ymin<120

Spleen:

size > 500

location: Xmin>140, Ymin>70

Left Kidney:

size>150

location: Xmin>48, Ymin>110

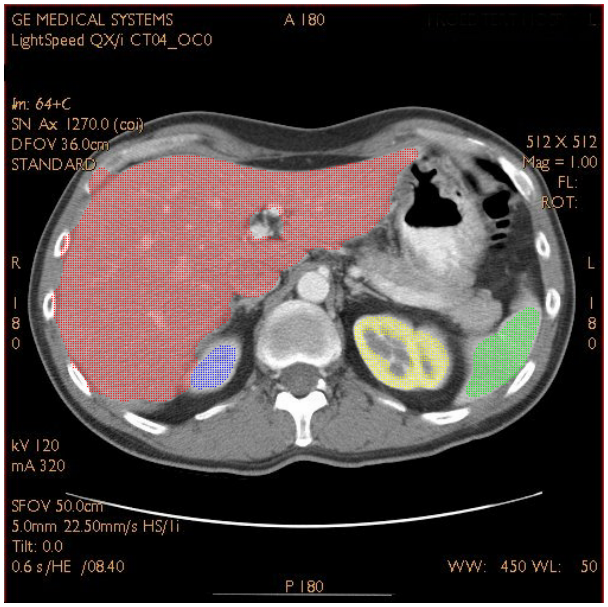
Right Kidney:

size>150

location: Xmin>141, Ymin>114

6.6 Organ Coloring

Using the one on pixel that is part of the organ, we color that pixel in the original picture with the particular color chosen for that organ. Then we recursively check the binary image to see if the surrounding pixels are on. If they are, we continue the coloring process until we hit off pixels. This process is repeated for all the major organs.



Final Result

7 DEMO

7.1 Results of final demo

The final demo consisted of picking 6 images from the test set and coloring all the organs present in each of the images. We received very favorable results, where 5 of the images were all completely correctly colored, and the other image had 2 out of the four organs correctly colored.

7.2 Errors that occurred and why

The reason for the error in the one image used during the demo was that the spleen and right kidney were too close together, and the blob-labeling step in our algorithm clumped the two organs together as one big blob. This could easily have been avoided if we had modified our algorithm to separate the image based on very specific threshold values instead of a range that was too broad in some cases.

7.3 Feedback

Overall feedback for our project was a bit on the critical side. Our algorithm could definitely have undergone some big improvements, which would have allowed us to speed up the entire process, port more code to the EVM as well as improve the accuracy of our algorithm. However, the results were still encouraging, and with improvements, the project could be very close to perfect.

8 Analysis of EVM Performance

8.1 Input/Output Data Rate

We used HPI transfers to send the input to the EVM as well as retrieve the output. At first, our images were too big to fit on the EVM, and we could not send the image by HPI. However, we did not want to transfer one row at a time, since we would need the entire image for our algorithm. So, we down-sampled our image so that it would be a lot smaller, and were able to use the more reliable HPI transfer method. The transfer rate was about 6 Mbits/sec.

8.2 Profiling: Speed and Memory Needs

The major part of our EVM code consisted of the blob detection and labeling. The results of the profiling are given below.

Blob detection loop:

One Image: 11,526,524 cycles (average)
Three Images: 33,579,574 cycles

Entire program:

One Image: 18,060,405 cycles (average)

The images were originally 500 by 500 pixels. Each pixel took one byte of information. This is far too big to store in the EVM. So, we down sampled our image by a factor of 16.

One image = (500x500)/16 = 15625 bytes (approximate)

Transfer time: (15265 bytes/image) / (6Mbits/sec) = 0.02 sec/image

8.3 Improvements Done

We wrote our code to maximize the use of the parallel capabilities of the EVM. We unrolled some loops and rewrote parts of our code to test for improvements. However, since we did not port a lot of code to the EVM, we weren't able to make a lot of improvements.

8.4 Future Improvements

Rather than attempt to use the EVM better, future improvements should be directed toward improving our algorithm. Changing some of the recursive parts to linear time running time would provide incredible improvements in speed. The memory area of our algorithm does not really need any improvements.

9 References

- Mean Shift Segmentation Paper
    - "Mean shift: A robust approach toward feature space analysis.", D. Comanicu, P. Meer, 2002
    - Appears in "IEEE Transactions on pattern analysis and machine intelligence", Vol. 24, No. 5, May 2002

- Mean Shift Segmentation Code
    - http://www.caip.rutgers.edu/riul/research/code/EDISON/

- Image Database
    - General Electrics Healthcare website
    - http://www.gehealthcare.com/rad/ct/products/hi_cte/images_abd.html

10 Code

- In the attached files:
  - main.cpp ----- first median filter
  - main1.cpp ----- everything after the segmentation on PC side
  - image.h ----- given class
  - EVM.c ----- EVM code