THESIS APPROVAL

GRADUATE SCHOOL, KASETSART UNIVERSITY

Master of Engineering (Information and Communication Technology for Embedded Systems)

DEGREE

Information and Communication Technology for Embedded Systems — **FIELD**

Electrical Engineering — **DEPARTMENT**

TITLE: Design and Development of ECG Paper Conversion Prototype

NAME: Mr. Chaiwat Suwansaroj

THIS THESIS HAS BEEN ACCEPTED BY

..... **THESIS ADVISOR**

(Assistant Professor Dusit Thanapatay, Ph.D.)

..... **THESIS CO-ADVISOR**

(Mr. Chusak Thanawattano, Ph,D.)

..... **THESIS CO-ADVISOR**

(Associate Professor Nobuhiko Sugino, Dr.E.)

..... **DEPARTMENT HEAD**

(Assistant Professor Srijidtra Charoenlarpnopparut, Ph.D.)

APPROVED BY THE GRADUATE SCHOOL ON .....

..... **DEAN**

(Associate Professor Gunjana Theeragool, D.Agr.)

THESIS

DESIGN AND DEVELOPMENT OF ECG PAPER CONVERSION PROTOTYPE

CHAIWAT SUWANSAROJ

A Thesis Submitted in Partial Fulfillment of
the Requirements for the Degree of
Master of Engineering (Information and Communication Technology for Embedded Systems)
Graduate School, Kasetsart University
2010

Chaiwat Suwansaroj 2010: Design and Development of ECG Paper Conversion Prototype. Master of Engineering (Information and Communication Technology for Embedded Systems), Major Field: Information and Communication Technology for Embedded Systems, Department of Electrical Engineering. Thesis Advisor: Assistant Professor Dusit Thanapatay, Ph.D. 72 pages.

The purpose of this research is to develop a method for ECG beat classification from ECG printout. This method composes with image processing and ECG beat classification. Image processing is used for extract time-series ECG signal from ECG printout. It use threshold base and moving average technique for create ECG time-series data. After that the selected beat is classified with SVM classifier. The output of classification is type of ECG signal with confidence measure value. This classifier use MIT-BIH database for training. LIBSVM is used for SVM implementation.

The accuracy of SVM based classifier is 99.682% (correct classification 33326 from 33332) with single lead basis (limb lead II or MLII) on MIT-BIH database. Performance on real ECG printout is good for high quality input. Then an improvement of image processing algorithm is required for real world reliability. This thesis use Python and its library as programming language.

_____ _____ __ / __ / __

Student's signature Thesis Advisor's signature

ACKNOWLEDGEMENTS

I would like to gratefully thank and deeply indebted to Asst.Prof. Dusit Thanapatay my thesis advisor for advice, encouragement and valuable suggestion for completely writing of thesis. I would sincerely like to thank Dr. Chusak Thanawattano my thesis co-advisor from NECTEC, and also Assoc.Prof. Nobuhiko Sugino my thesis co-advisor from Tokyo Institute of Technology for their valuable comments and suggestion.

This research is financially supported by Thailand Advanced Institute of Science and Technology - Tokyo Institute of Technology (TAIST-Tokyo Tech), National Science and Technology Development Agency (NSTDA), Tokyo Institute of Technology (Tokyo Tech) and Kasetsart University (KU).

I am especially appreciated my mother, my family and my friends for their continuing encouragements. Especially, Mr. Napat Parkpien and Mr. Natthapon Kachathorn who also give valuable advice, Dr. Yodyium Tipsuwan for inspiration and support. Finally, I am deeply appreciated to Dr. Chusak Thanawattano who always devotes time and support during my graduate study.

Chaiwat Suwansaroj

June 2010

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF ABBREVIATIONS

SVM	=	Support Vector Machine
PCA	=	Principle Component Analysis
DWT	=	Discrete Wavelets Transform
IDWT	=	Inverse Discrete Wavelets Transform
SVD	=	Singular Value Decomposition
ECG	=	Electrocardiography
HOS	=	Higher Order Statistics
AR	=	Auto Regressive model

DESIGN AND DEVELOPMENT OF ECG PAPER CONVERSION PROTOTYPE

INTRODUCTION

The electrocardiogram (ECG) is a vital sign signal for heart functional investigation. This electric signal is generated from human heart for create the cardiac cycle which generate the blood circulation. It composed with three basic components named P wave, QRS complex and T wave as show in figure 1. P wave is generated when atrium depolarization. After that QRS complex is generated when ventricle depolarization and T wave is generated when ventricle recovery. The cardiovascular disease is one of the leading causes of death around the world. While the lacking of physician who is an expert for analysis on ECG signal is one of serious problem especially on rural area also. Therefore the research on ECG classification method for ECG printout is aimed to relieve this problem. The example of paper based ECG printout is depicted in figure 2.

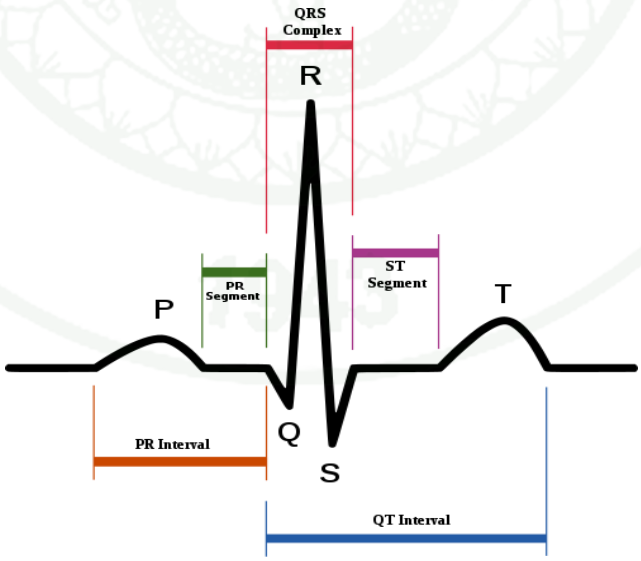


Figure 1 ECG signal and its features.

The ECG classification method for ECG printout composes with three basic components which depict in figure 3. The first component is time series ECG beat detection from ECG printout. It comprise of image processing techniques for extract time-series ECG signal. The second component is data preprocessing such as discrete wavelet transform and PCA. This signal processing techniques are used for enhance ECG signal which aimed to reduce error rate of classification. The last component is ECG beat classifier. This classifier uses SVM technique for classification. MIT-BIH (Mark and Moody, 1988) database is used for training and evaluate this classifier. The MLII or limb lead 2 is used as input for classification. This classifier is trained for distinguish only 4 types of signal as normal beat, left bundle branch block beat, right bundle branch block beat and premature ventricular contraction.

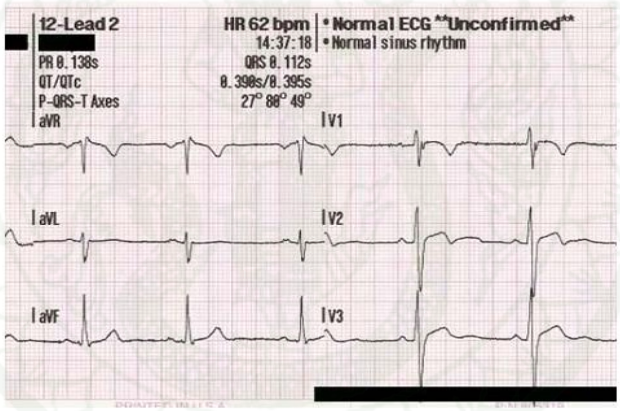


Figure 2 Example of paper based ECG printout.

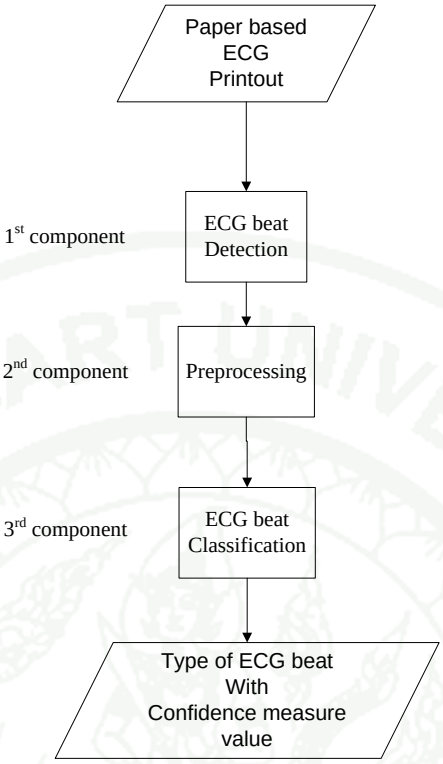


Figure 3 Diagram for ECG beat classification method from ECG printout.

OBJECTIVES

1. To study the method for ECG beat classification.

2. To study the method for ECG beat extraction from paper based ECG printout.

3. To study the method for classify ECG beat with confidence measure value.

4. To propose the method for classify ECG beat from paper based ECG printout with confidence measure value.

LITERATURE REVIEW

This research composes with image processing, signal processing and machine learning for pattern recognition and classification. Machine learning is computer software which classify between normal and abnormal ECG signal. SVM is used for this task. This technique is widely used for machine learning on biomedical signal processing such as ECG detection (Mehta and Lingayat, 2007). Especially to ECG beat recognition, SVM is used with many for reliability and accuracy improvement such as HOS (Osowski et al., 2004), morphological and HOS (Besrour et al., 2008) and CWT with multiple SVM classifiers (Zellmer et al., 2009). The performance and accuracy of classifier highly dependent on its feature vector, then the improvement its feature with signal processing techniques is necessary. For example, DWT (Khadtare and Sahambi, 2004) is applied before categorized with SVM, DWT and AR (Qibin and Li-Qing, 2005) or PCA (Hao and Li-Qing, 2005) is used also. ECG beat extraction from ECG printout is necessary for beat classification with computer. Some research implement with template and rule based for background subtraction (Jian and Mital, 1996) or use suitable threshold value (S. Mitra and M. Mitra, 2003). Most of method which mentioned above use gray scale image. The method for color image is presented with segmentation and mask technique (Kao et al., 2001). A very versatile software which used for digitize electrocardiography is ECGScan (Badilini et al., 2005). After digitize electrocardiography, classification ECG signal with different techniques are used. Such as Fourier transform (Mitra et al., 2004) or shape-based matching (Syeda-Mahmood et al., 2007). LIBSVM (Chang and Lin, 2001) is used for training and evaluation classifier.

MATERIALS AND METHODS

Materials

1. Computer
2. Data from MIT-BIH database
3. Python and its library for programming
   3.1 Numpy for numerical library
   3.2 Scipy for scientific library
   3.3 Matplotlib for plotting library
   3.4 OpenCV for computer vision library
4. Image of paper based ECG printouts

Methods

A method for ECG beat classification with confidence measure value is proposed in this research. This method is composed with image processing, signal processing and machine learning. Image processing is used for extract ECG beat from paper based ECG printout. Signal processing is used for signal enhancement and extract feature which used for beat classification. Machine learning is used for train computer to classify type of ECG beat with confidence measure value. SVM is used for this purpose.

1. Database preparation

This section describes about a method for database construction which required for SVM training. It start from download MIT-BIH database from internet. After that software for extract data from database is downloaded and installed. Finally we extract N (Normal beat), L (Left bundle branch block beat), V (Ventricular premature beat) and R (Right bundle branch block beat) labels from each records (with rdann command) and raw ECG data (with rdsam command) to text file. The

annotation file after extraction is formatted as *.ann while raw ECG data is formatted as *.sam. The procedure which is mentioned above is depicted in figure 4. Shell script for this purpose is shown on appendix A. Output is generated from appendix A is listed below.

1. Directory named raw_data contains every *.sam file.
2. Directory named annotated_file and its subdirectory named N_L_V_R_A which contain annotation file for each case (N, L, V, R). Each annotation has their own directory named N, L, V or R for each record.
3. Directory named 200Hz_segmented_dwt_20D_data which used for store ECG signal after preprocessing.
4. Directory named src, it used for store source code.

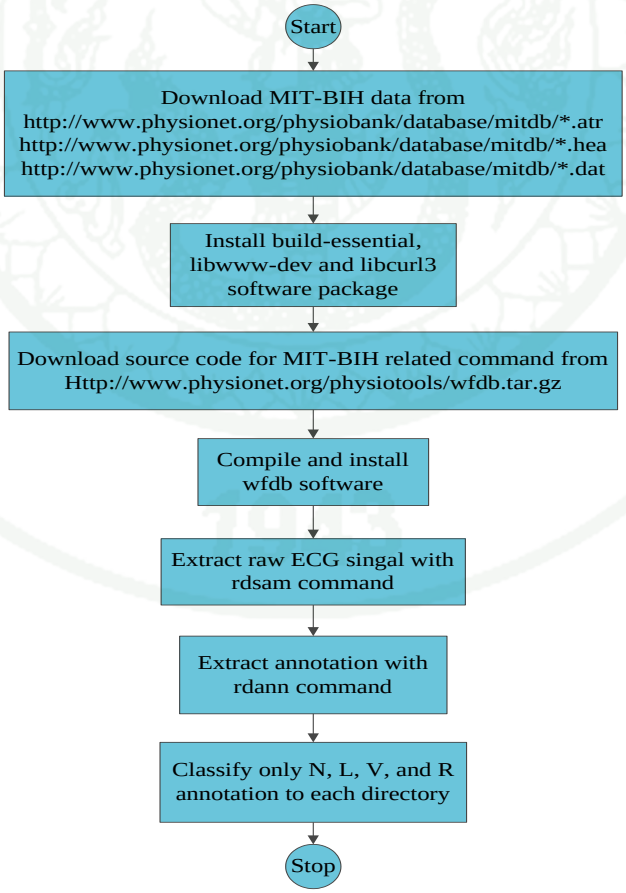


Figure 4 Diagram for database preparation.

2. Data preprocessing for SVM training

After raw ECG signal and its annotation are extracted from MIT-BIH database then we do beat segmentation. Sampling rate of ECG signal is reduced from 360 Hz to 200 Hz. After that we normalize and remove its DC components. Finally DWT is used for construct approximation of ECG signal shape. Figure 5 illustrate method for this purpose. Output of this method is file *.dwt which locate at 200Hz_segmented_dwt_20D_data directory (N, L, V, R subdirectory for each annotation).

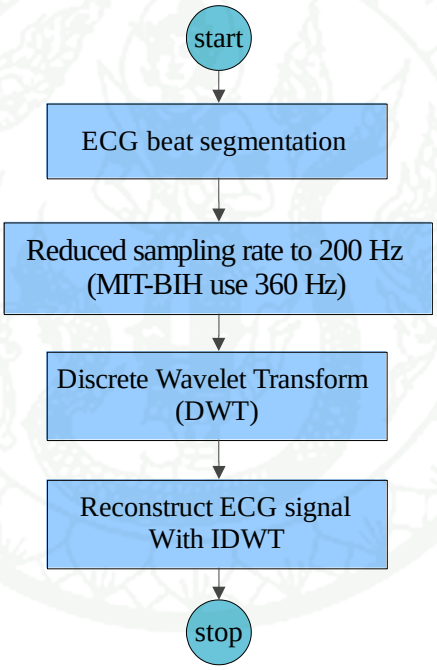


Figure 5 Diagram for data preprocessing.

Beat segmentation start with read each annotation and use it as R-peak of QRS complex. After that the consequence 231 samples is used as ECG beat which R-peak is located on the middle of this signal. This signal is reduced its sampling rate to 200 Hz and normalize its. The equation for normalization is shown in (1). Diagram for downsampling is shown on figure 5.

$$x[n] = \frac{x[n] - \min(x)}{\max(x) - \min(x)} \qquad (1)$$

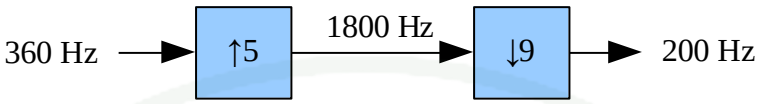


Figure 6 Downsampling diagram for data from MIT-BIH database.

Each of 200 Hz ECG beat is transformed with DWT. Because of shape of ECG signal is a prominent feature for classification. Then level 3 of Daubechies 1 is used for construct an approximation of ECG shape. Diagram for this method is shown in figure 6. Figure 7 is a 200 Hz-normalized ECG signal while output after DWT of it is shown in figure 8. More detail of DWT is described on the Wavelet tutorial (Polikar, 2001).

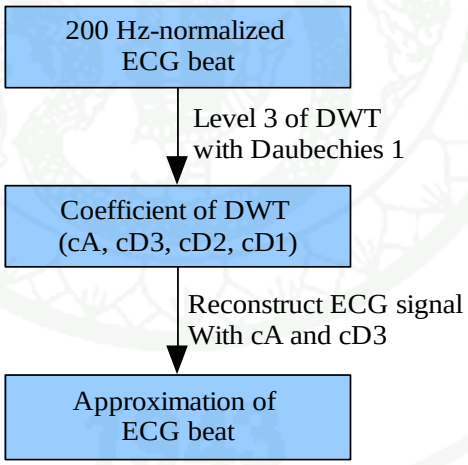
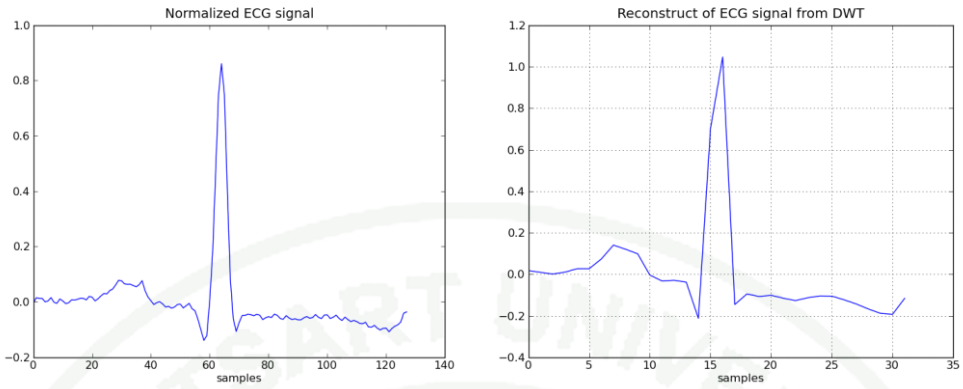


Figure 7 Method for DWT.



(a) (b)

Figure 8 Result of DWT on ECG signal. Figure (b) is an approximation of figure (a).

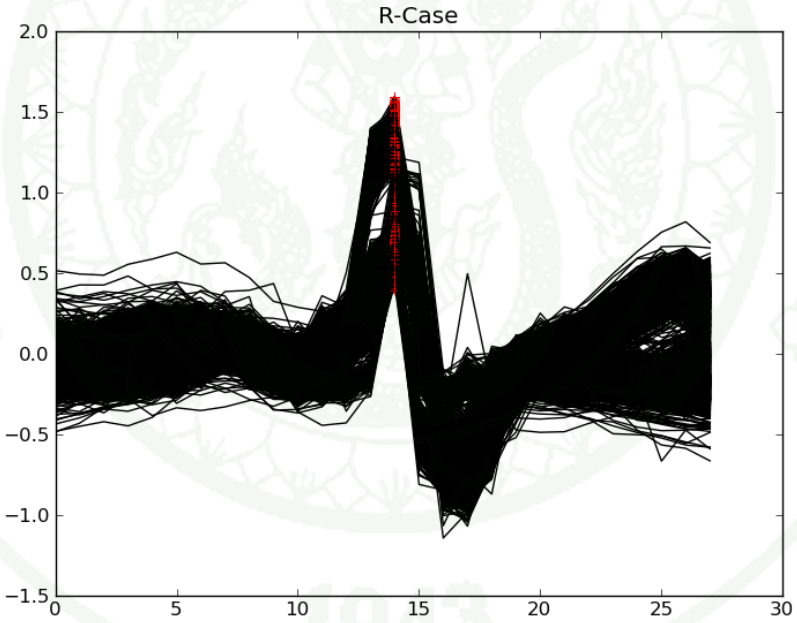


Figure 9 Right bundle branch block ECG signal.

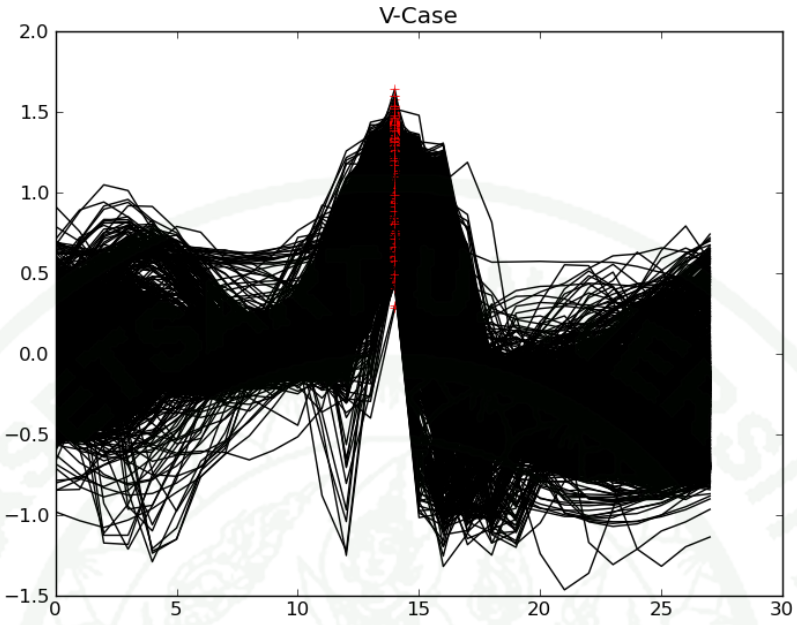


Figure 10 Premature ventricular contraction ECG signal.

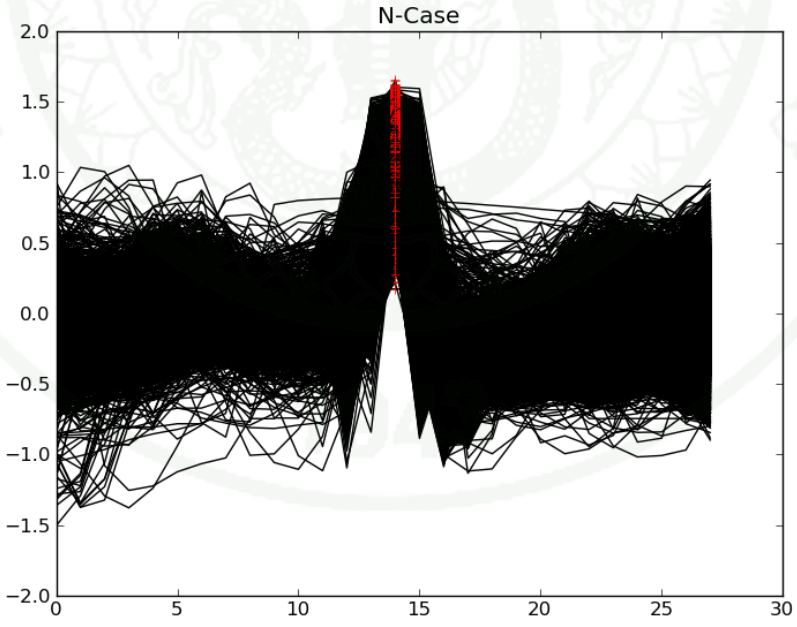


Figure 11 Normal case ECG signal.

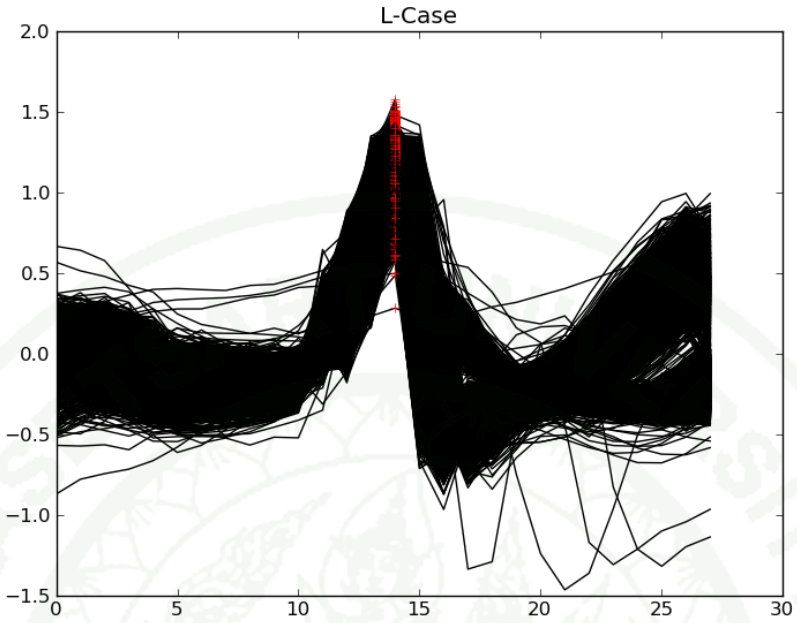


Figure 12 Left bundle branch block ECG signal.

3. Feature extraction with PCA

PCA is a tool for reduce dimension of data. It base on concept of maximum variant. Dimension of data is used for this SVM training is 20. Implementation of PCA based on SVD technique. Source code which implements PCA can be found on appendix C. A tutorial on Principle Component Analysis (Jonathan Shlens, 2005) is a good document for PCA explanation. The result of source code in appendix C is train_ecg_NLVR.svm and test_ecg_NLVR.svm for SVM training and testing respectively.

4. SVM training

Because SVM is a supervise learning, then we need to create new database with class label. An integer value is used for represent each class such as -1 for Normal, -2 for Left bundle branch block beat, 1 for Right bundle branch block beat and 2 for Premature ventricular contraction. Source code for labeling each class is shown on appendix C.

LIBSVM is used as a tool for training and evaluation SVM classifier. Gaussian RBF is used as a kernel function. A 5-fold cross validation is used for parameter estimation for kernel function. After training, we know each parameter for kernel function and SVM model which is 2.0 and 2.0 for C and γ respectively. A course material for 6.867 Machine learning from MIT (Jaakkola, 2006) is a good resource for SVM explanation.

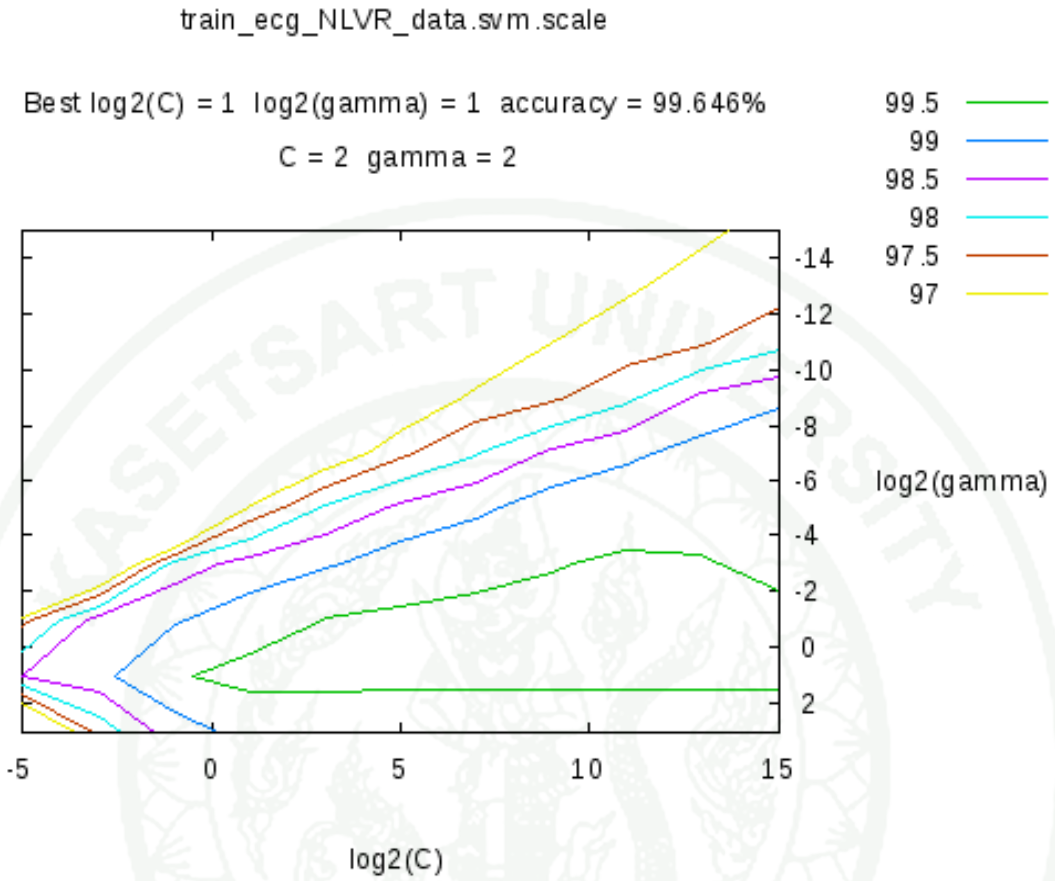


Figure 13 Contour plot of cross validation accuracy.

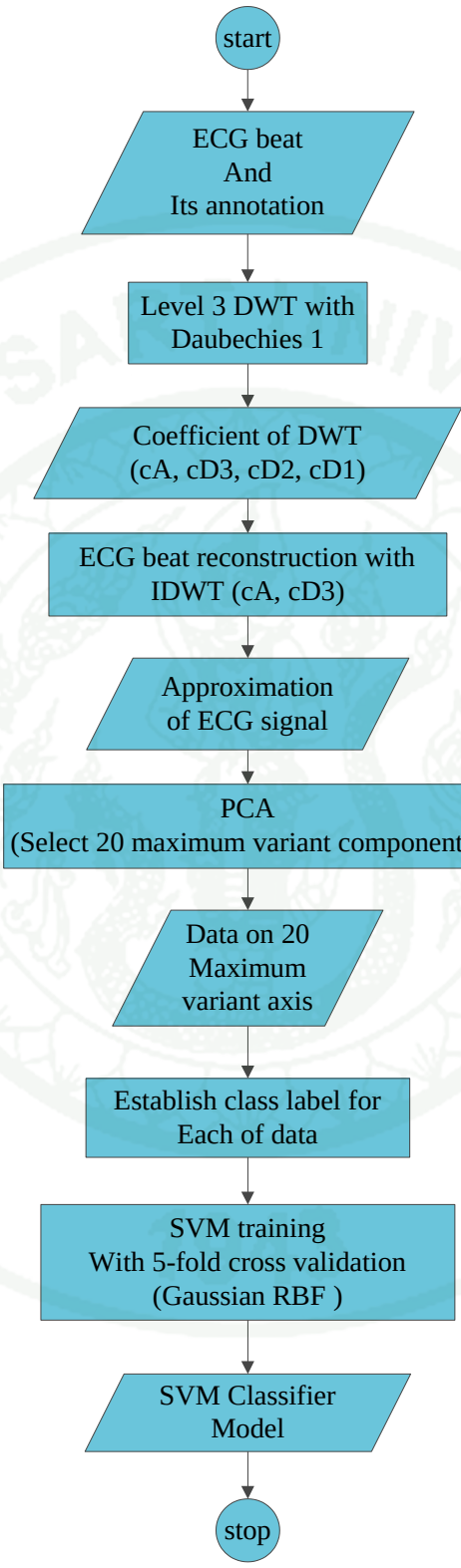


Figure 14 Flowchart for SVM training method with MIT-BIH database.

5. Image processing

This method is required because most of physician still analyze heart function from ECG printout. Then our method still base on ECG printout. For analyze ECG signal with computer, we need to extract ECG beat from paper. This method will describe in this section.

5.1 ECG image retrieval with scanner

ECG printout is scanned with 300 dpi which equivalent to 295 Hz (3.4 msecs/pixel) sampling rate for data recorded with speed of 25 mm/sec.

5.2 Select ECG beat for classification

An interesting ECG beat is selected from the image for image and signal processing.

5.3 Image binarization

The selected segment of ECG image is loaded as gray scale because the color of ECG signal from the original paper is black and the color of paper grid is red. After that the binary image is created with suitable threshold selection (in this experiment use 130). But noise will appear in sometimes as shown figure 14. Then it needs to eliminate noise after binarizing the image.

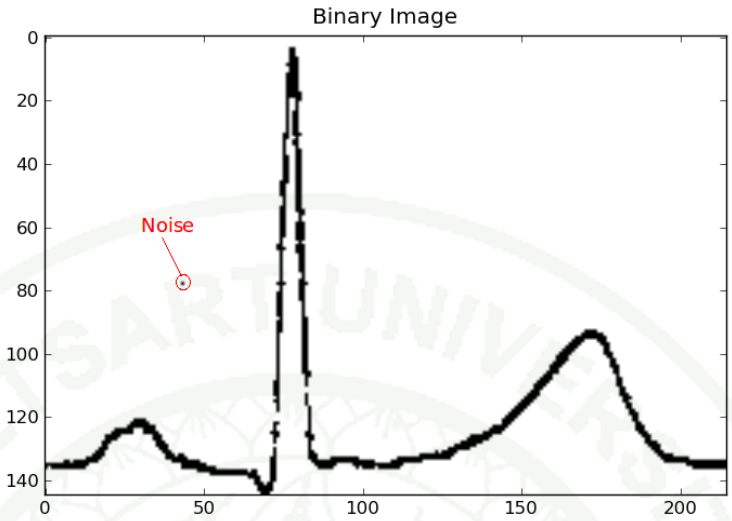


Figure 15 Binary image of ECG and noise.

5.4 Noise rejection

To remove the noise on the image after binarization process, the following process of noise rejection is applied to the binary image. The process starts by scanning vertically to find out the black pixel. If the black pixel is found and all adjacent pixels around it are white background color, then this black pixel will be considered and treated as a noise which will be replaced with white background color.

5.5 Image thinning

Since the line of ECG trace of original scanned image from ECG printout has a thickness which is a redundant of data in time series domain. Then thinning process with moving average algorithm is used to eliminate this redundant of data. Moving average in top-down fashion is used for thinning binary ECG images. When applying 5-point moving average to the binary image after noise rejection, the location of maximum value within 5-point window is used to determine the position of the thinning output pixel. This concept is illustrated in figure 15. The result of thinning process and noise rejection is shown in figure 16.

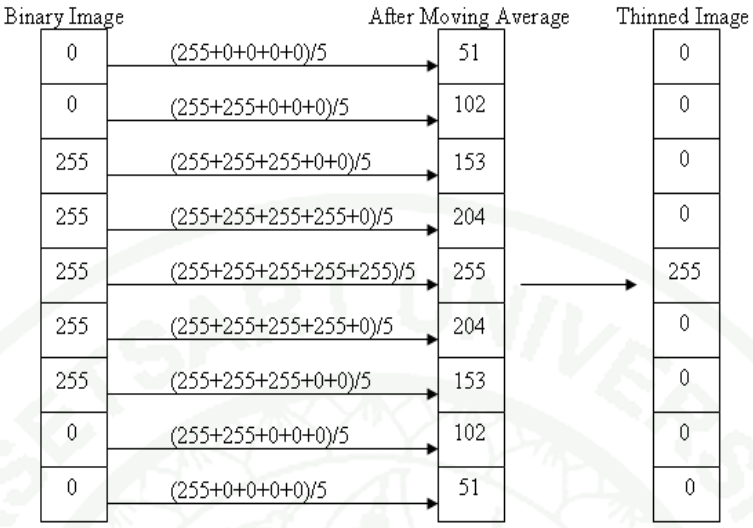


Figure 16 Concept of image thinning with moving average algorithm.

5.6 Time-series data extraction

Time series data extraction is started from black pixel searching on each column with bottom-up style. After black pixel is found, its row index is used as the data value. This process is repeated on every column of image.

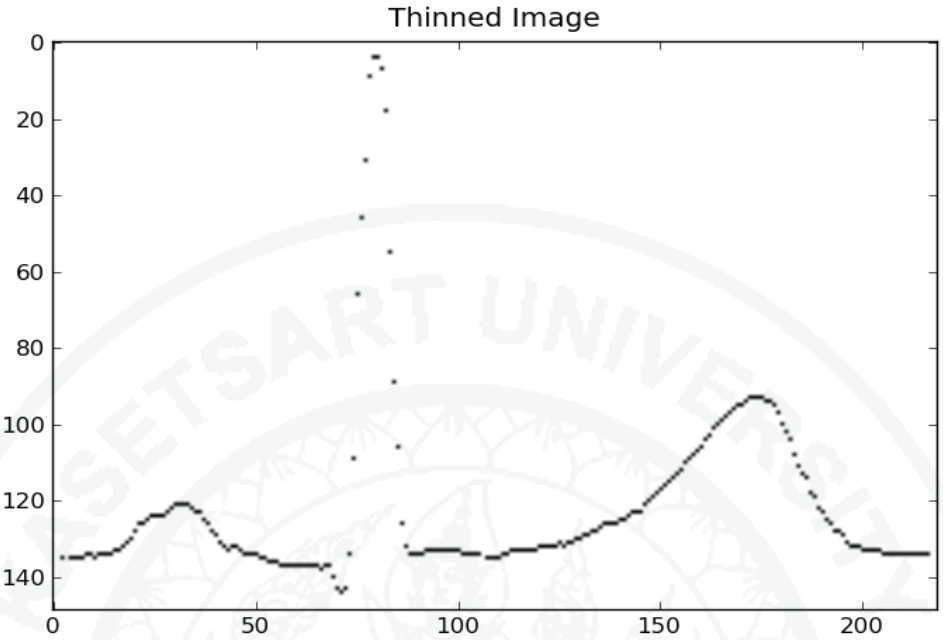


Figure 17 Output of image thinning and noise rejection.

5.7 Time-series data preprocessing

For classification ECG beat, the classifier use its shape for classify. Then data preprocessing is required. This preprocessing comprises of normalization and mean subtraction for generate zero-mean signal. The equation for normalization is shown in (1).

5.8 QRS complex detection

After we create zero-mean normalized ECG signal. The absolute value of zero-mean normalized signal is used to find its envelope with Hilbert transform (green line on figure 17). The equation below is a Hilbert transform of real function $x(t)$ when $x(t)$ is ECG time series data.

$$H[x(t)] = \frac{1}{\pi} \int_{-\infty}^{\infty} x(\tau) \frac{1}{t-\tau} d\tau \qquad (2)$$

Finally the peak of QRS complex is located by looking for the top of its envelope. The result of QRS detection after plotting the peak location of the QRS complex together with zero-mean normalized data and the envelope of its absolute value is shown in figure 17.

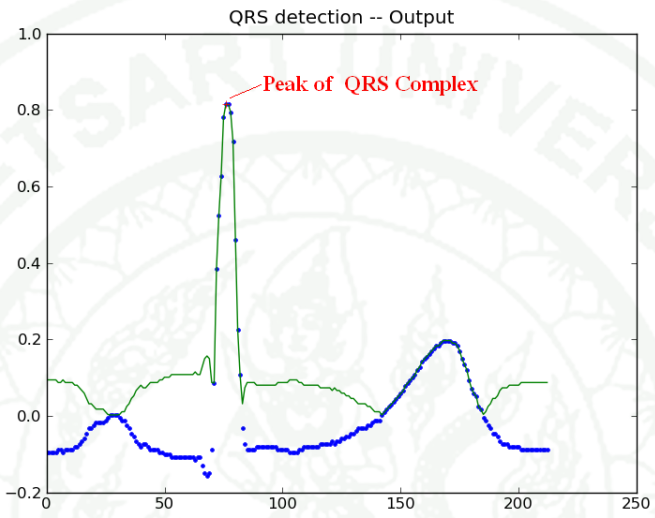


Figure 18 Output of QRS complex detection

5.9 ECG beat extraction

After the position of QRS complex in selected beat is located. ECG beat is extracted with 185 samples where the peak of QRS complex is the 93th sample. After that downsampling method with rational fraction M/L is applied to ECG beat for generate 200 Hz ECG signal. M is a downsample factor and L upsample factor which is 59 and 40 respectively. The linear interpolation is used for upsampling method. Downsampling method is shown below.

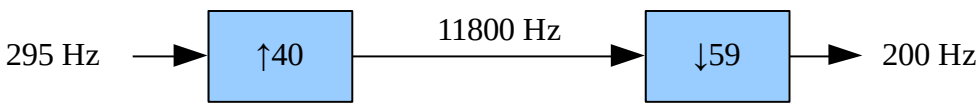


Figure 19 Downsampling diagram for ECG signal from ECG printout**.**

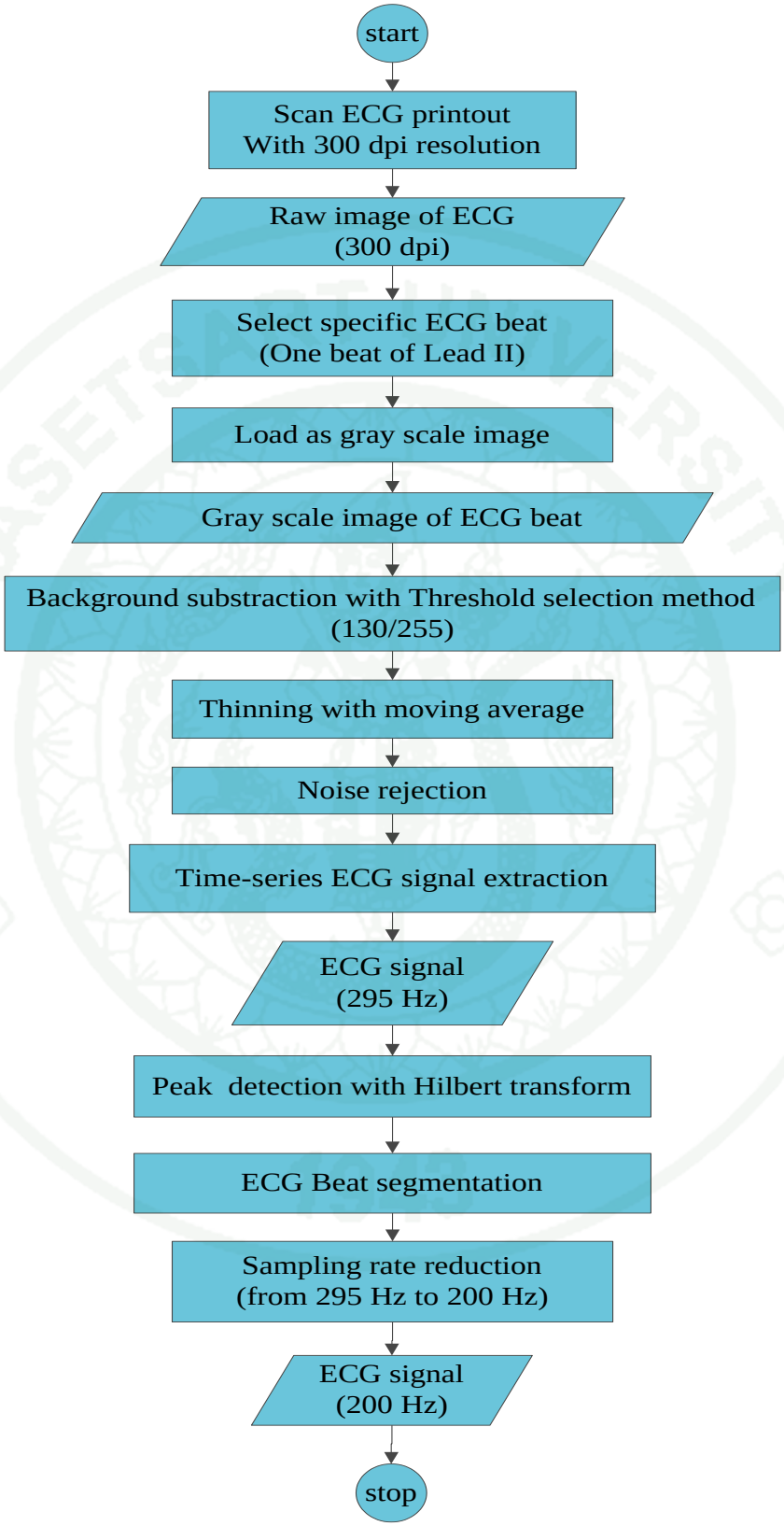


Figure 20 Flowchart for image processing method.

6. Implementation

ECG beat classification from paper based ECG printout composed with model of SVM classifier and image processing method. The model of SVM classifier is an output of SVM training (train_ecg_NLVR_data.svm.model). The flowchart of implementation is shown below. The source code of this implementation is shown on appendix D.

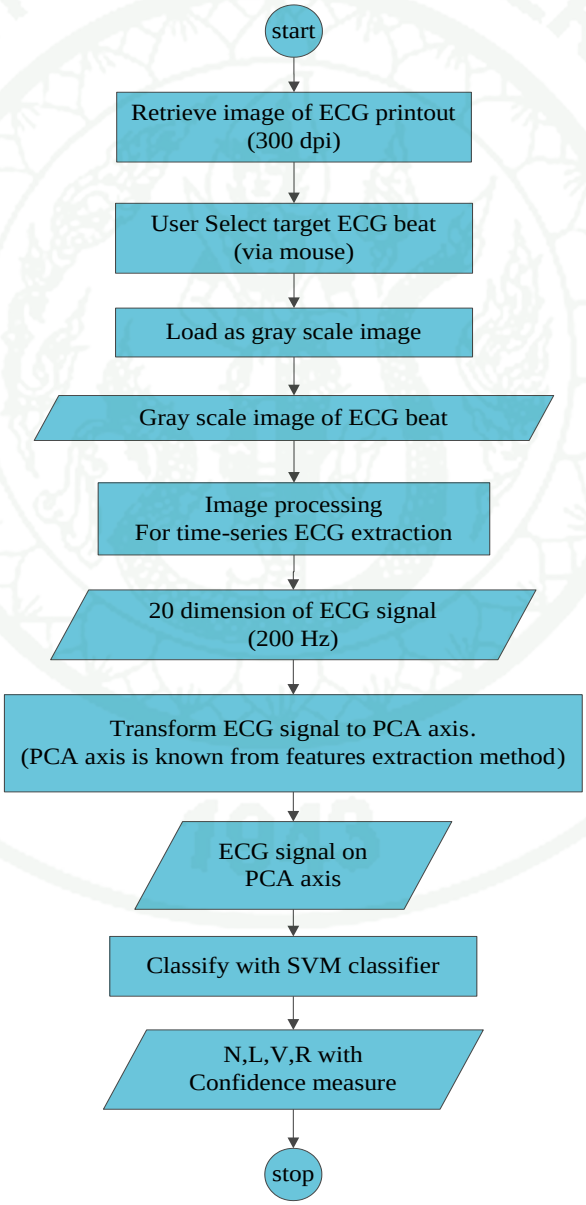


Figure 21 Flowchart for implementation**.**

RESULTS AND DISCUSSION

Results

This SVM classifier has a good performance because its accuracy is 99.682% (correct classification 33326 beats from 33332 beats). Data for MIT-BIH database is used for training and classification. It is used for establish new database which is training and testing database. Annotation label N, L, V and R from MIT-BIH database is used for train and validate classifier.

Discussion

Performance of SVM classifier is excellent. It can use for classify ECG beat in to 4 types, normal beat, left bundle branch block, right bundle branch block and premature ventricular contraction. But training method of SVM is a time-consuming task. It takes about 4 hours on core-i5 PC (with 100% CPU usage). Otherwise this model is train from MIT-BIH database which different from paper based ECG printout. Then performance of this model might be suffered from different input method on real implementation. This problem is relieved with downsampling ECG to 200 Hz. But downsampling input from ECG image depends on image processing method. Then the improvement of image processing techniques for enhance ECG beat extraction is required. SVM training with ECG image is exact solution for eliminate this problem.

The kernel selection is an important method for SVM. The RBF kernel is selected because its performance better than linear kernel and 2-degree polynomial as shown in table 1. The proposed classifier has a competitive with another classifier as shown in table 2.

Image processing is one of most important method. If an algorithm for image processing sensitive to noise and image rotation then the result of classification is not reliable as shown in figure 23-32. Table 3 and 4 show an severe problem because the confidence value of classification increase to normal case when noise is appear or image is rotated. Then an improvement of image processing is required

Table 1 Comparison result between linear, RBF and 2-Polynomial

Data set	Accuracy (%)		
	Linear	**RBF**	**2-Polynomial**
a9a	84.99	85.04	85.06
real sim	97.51	97.9	98.00
ijcnn1	92.21	98.69	97.84
MNIST38	96.82	99.69	99.29
covtype	76.35	96.07	80.09
webspam	93.15	99.20	98.44

Source: Chang et al. (2010)

Table 2 Comparison result of classification accuracy between proposed classifier and another work

Method	Number of ECG types	Accuracy (%)
Proposed	4	99.682
Zellmer et al.	6	99.72
Sung-Nien & Kuan-to	8	98.70
Amir et al.	6	99.49
Zhao et al.	5	97.55
Ubeyli	4	98.61
Prasad and Sahambi	12	96.77

Source: Zellmer et al. (2009)

Table 3 Performance of classifier when noise is appeared on image

% of Noise	Probability estimation for normal case, p(N)	
	Random Pick Noise	Random Slur Noise
0	0.706997	0.706997
10	0.751271	0.726205
25	0.738179	0.700163
50	0.767048	0.730248

Table 4 Performance of classifier when image is rotated

Degree of rotation	Probability estimation for normal case, p(N)
0	0.706997
3	0.868966
6	0.874932
9	0.871669

CONCLUSION AND RECOMMENDATION

Conclusion

In this research, we propose a method for classify ECG beat from paper based ECG printout. It composes with many engineering techniques such as image processing, signal processing and machine learning with SVM. Image processing is used for extract time-series ECG signal from ECG printout. Signal processing techniques is used for signal enhancement. After that we create feature vector with DWT and PCA technique. Finally, we create new database for train and validate SVM classifier. LIBSVM is used for this task. Output of LIBSVM is a classifier model with parameter for kernel function. Gaussian RBF is used as kernel function. Each parameter for kernel function is 2.0 and 2.0 for C and γ respectively. Python and its library is used for implement this research. This method can be used for other kind of input such as ECG signal via skin electrode also.

Recommendation

In this research, we propose a method for classify ECG beat from paper based ECG printout. It composes with many engineering techniques such as image processing, signal processing and data classification with SVM. Although the validation output of purposed SVM is excellent but improvement of image processing method is required for real world implementation. Furthermore this classifier can be extended for normal or abnormal beat classification with database adaptation. For example, create new database with 2 types of ECG beat which is normal and abnormal types. Then train new database with LIBSVM for create new model of SVM classifier. Moreover some network capability will increase value to this research. Because physician cannot be replaced by computer in every case of ECG signal analysis, then network connectivity can be used for remote analysis from an expert. Last but not least, the reliability of classifier can be improved with cooperation with another classifier model such as HMM and voting each result base on its confidence measure value for the final result. Figure 22 depict this concept. Finally if

time-series ECG signal is used as an input of our classifier then the result will be reliable than another kinds of input because the classifier is trained with time-series database.

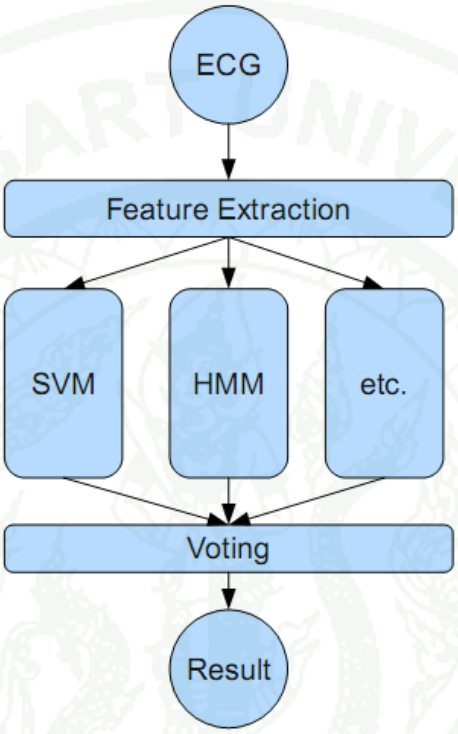


Figure 22 Diagram for classifier improvement.

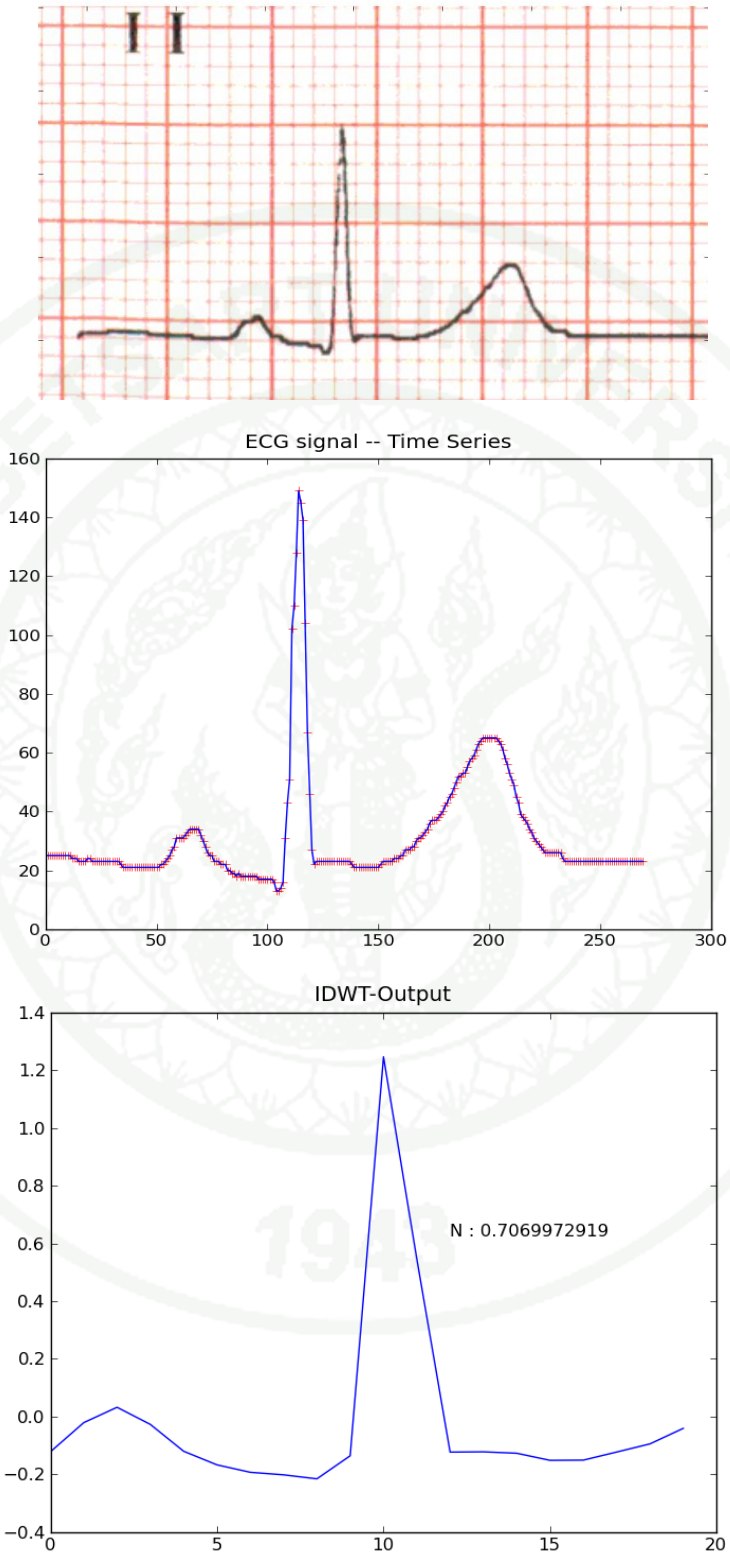


Figure 23 Original image of ECG printout and classification output.

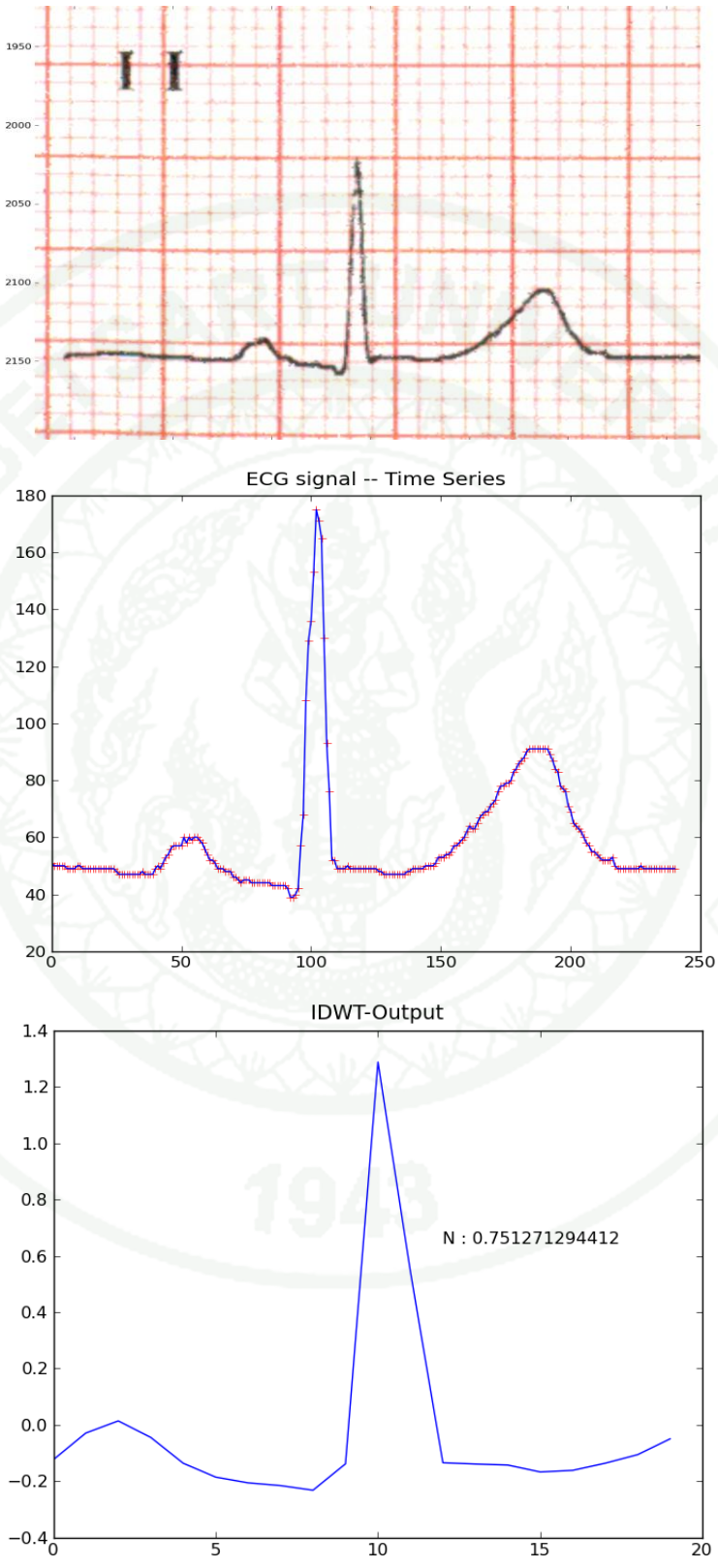


Figure 24 Adding 10% of Random Pick noise to image and classification output.

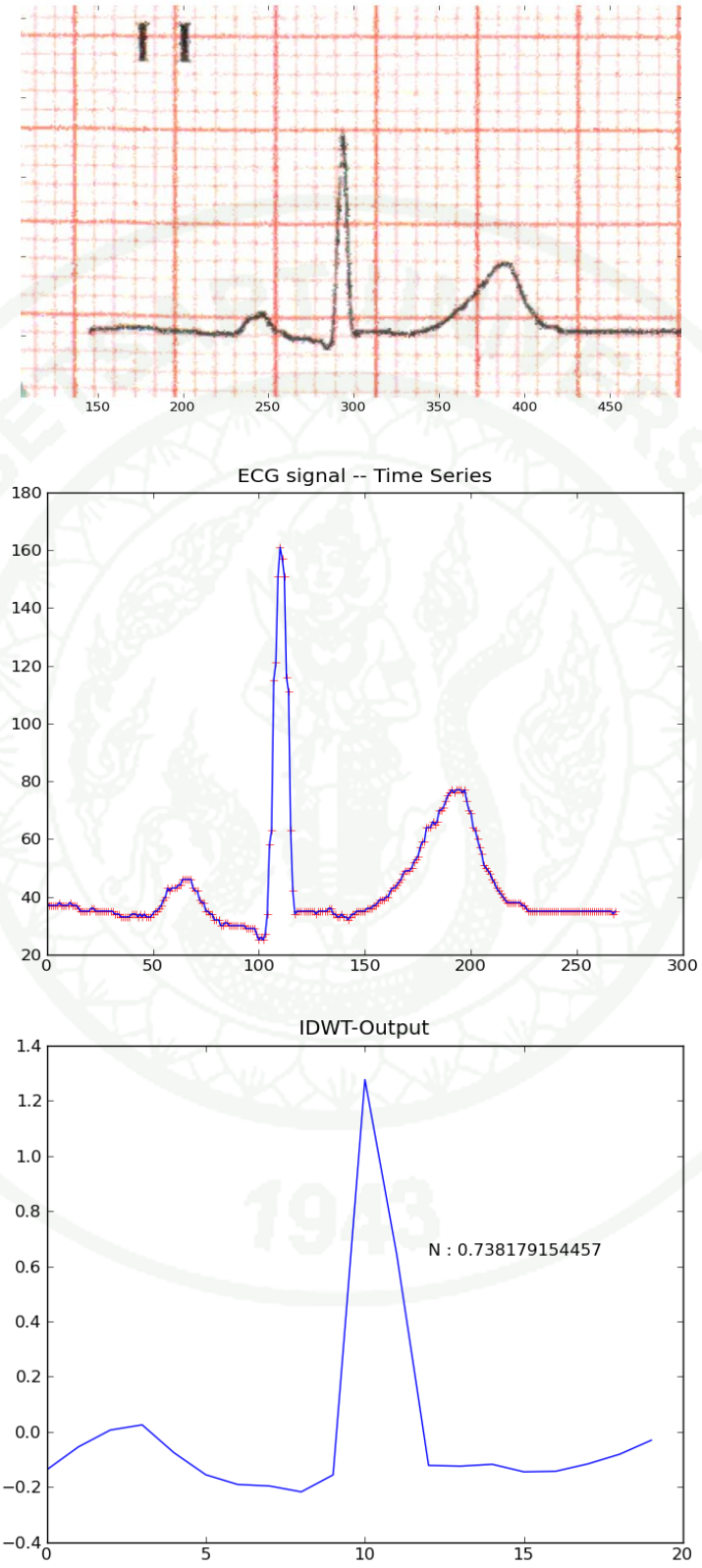


Figure 25 Adding 25% of Random Pick noise to image and classification output.

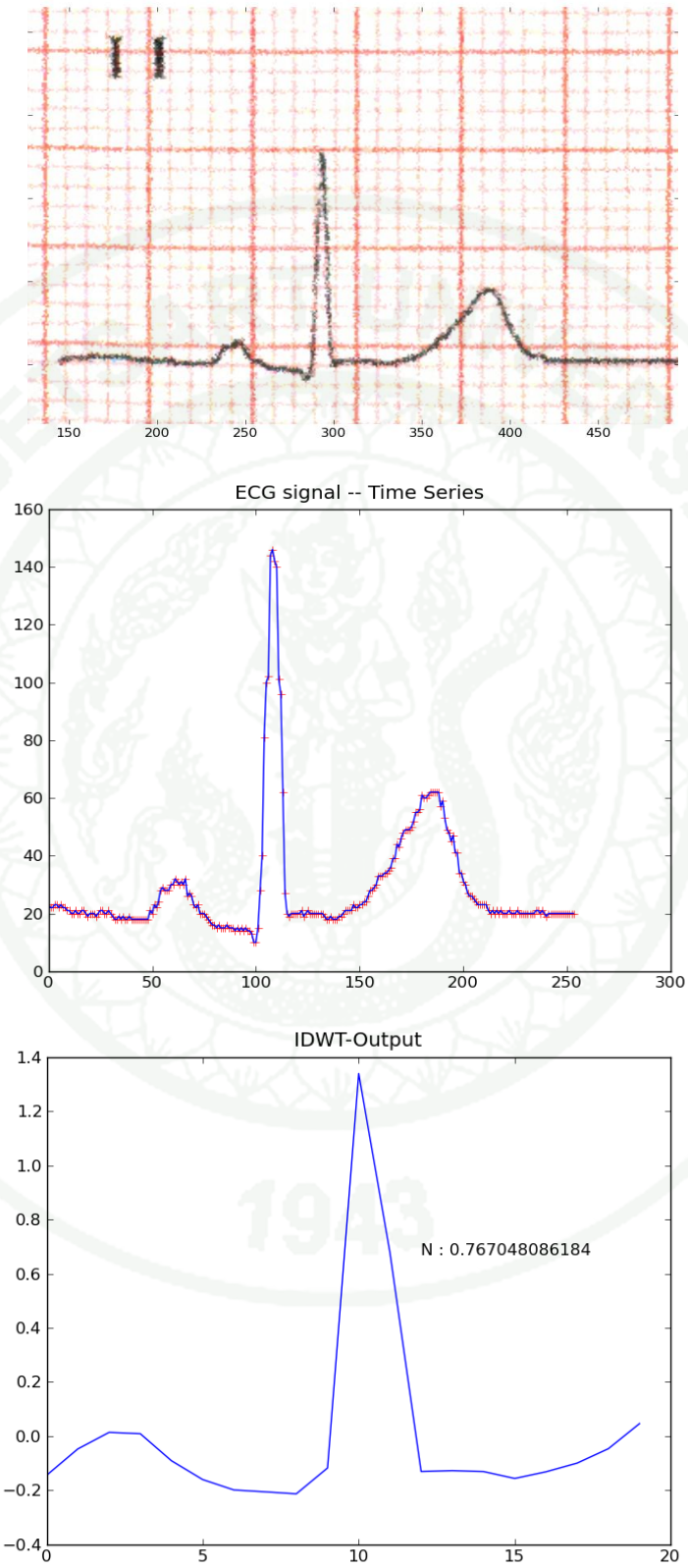


Figure 26 Adding 25% of Random Pick noise to image and classification output.

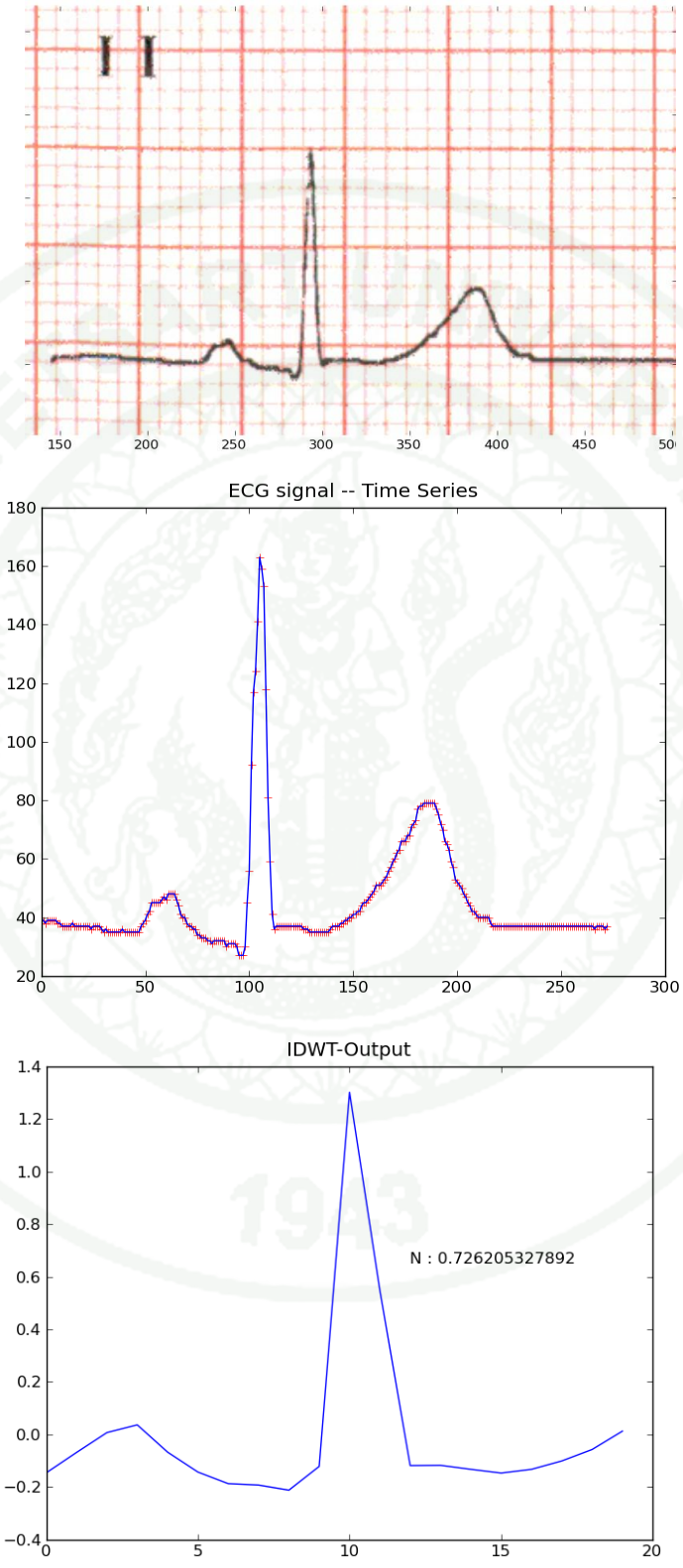


Figure 27 Adding 10% of Random Slur noise to image and classification output.

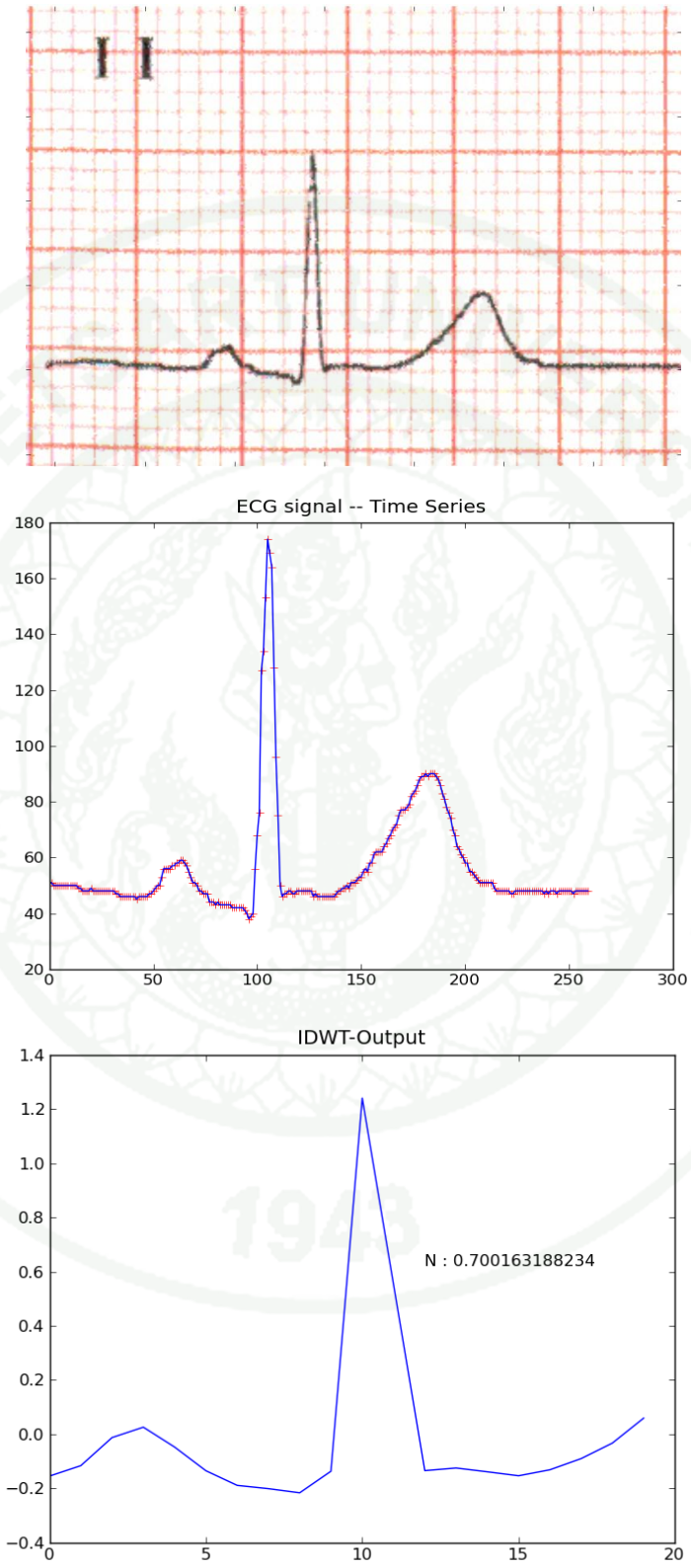


Figure 28 Adding 25% of Random Slur noise to image and classification output

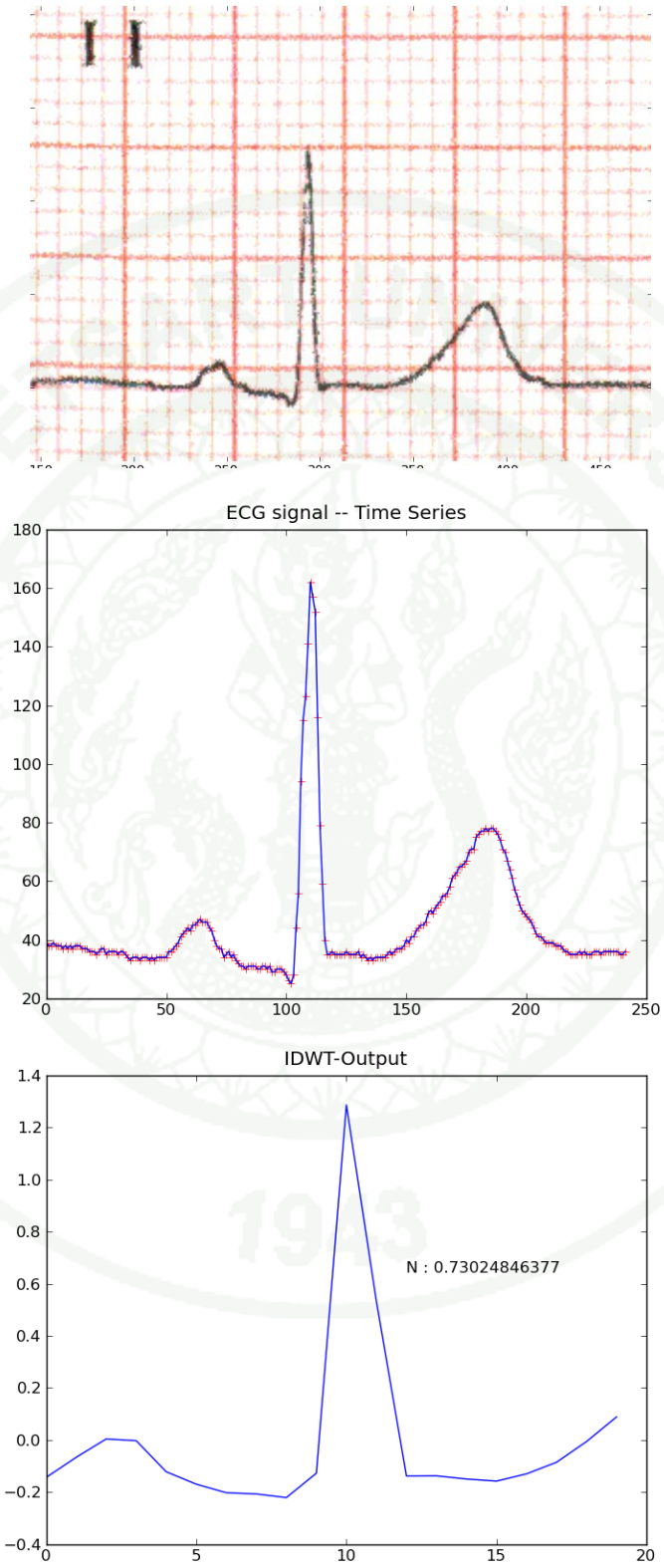


Figure 29 Adding 50% of Random Slur noise to image and classification output.

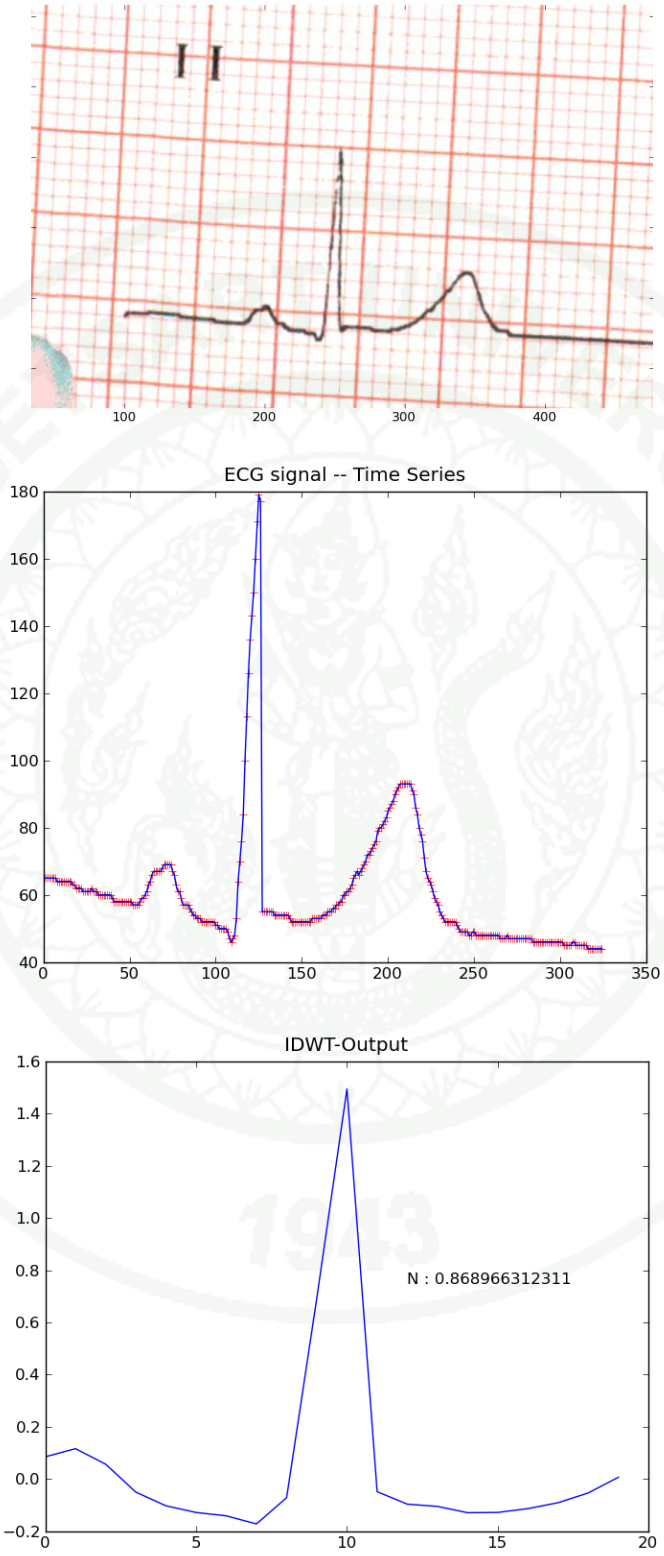


Figure 30 Image is rotated 3 degree and classification output.

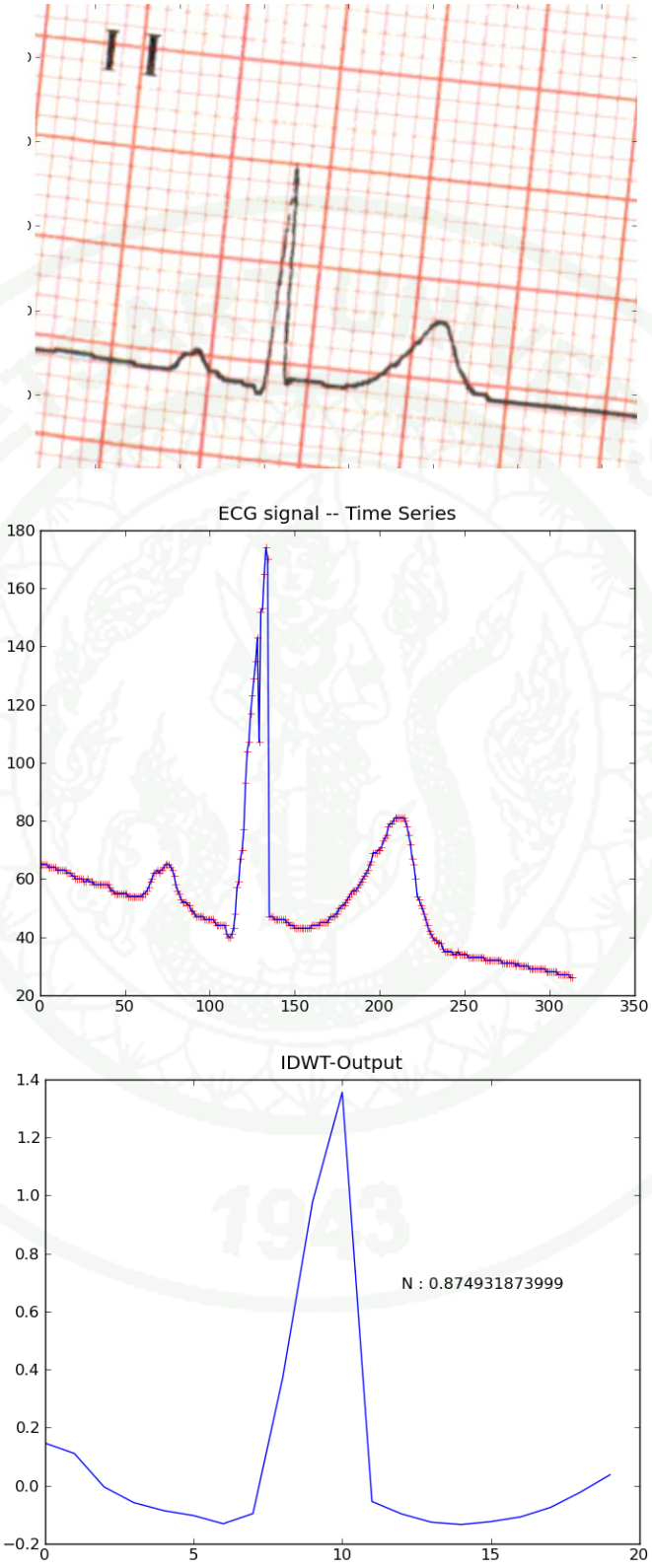


Figure 31 Image is rotated 6 degree and classification output.

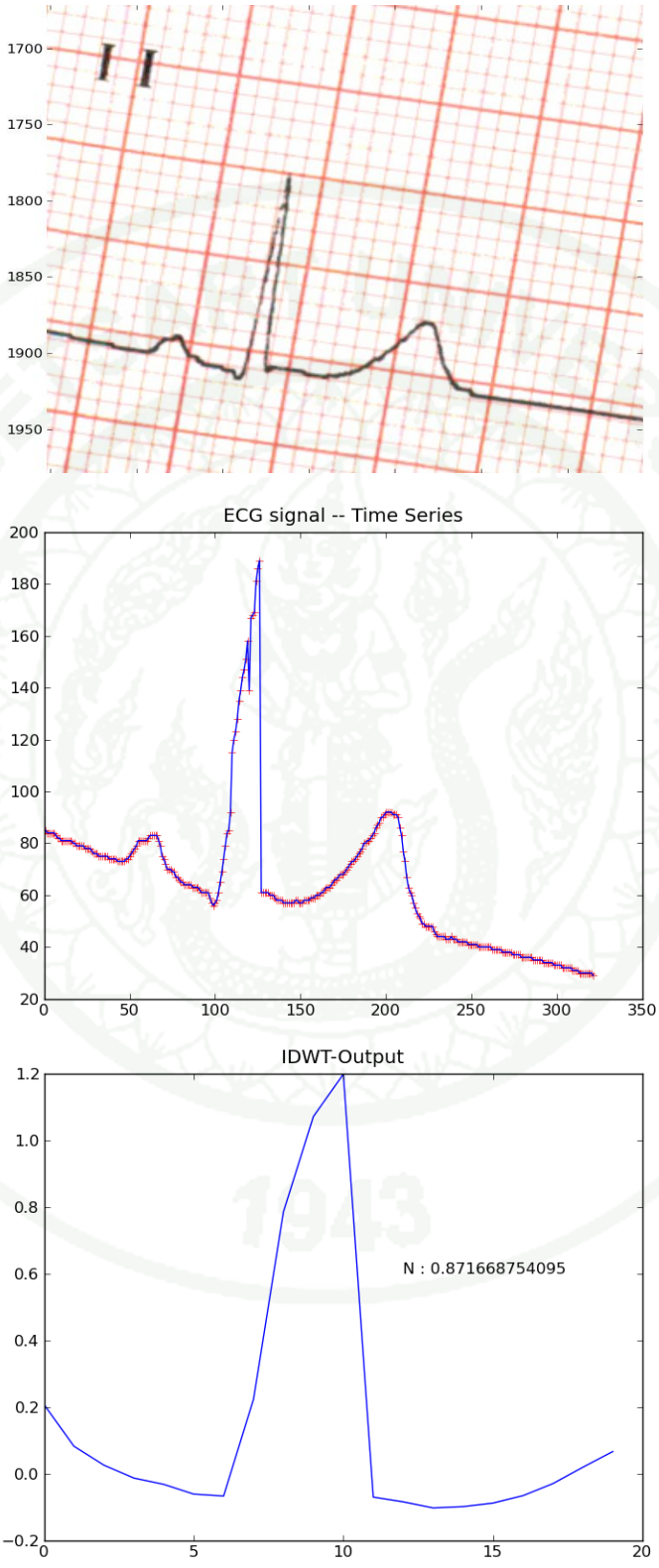


Figure 32 Image is rotated 9 degree and classification output.

LITERATURE CITED

Badilini, F., Tanju Erden, Wojciech Zareba, and Arthur J. Moss. October 2005. ECGscan: a method for conversion of paper electrocardiographic printouts to digital electrocardiographic files, **Journal of Electrocardiology**, Vol. 38, issue 4, 310-318.

Baeza, R., October 2005. Electrocardiogram digitization: a practical perspective on the usefulness of a new tool to convert paper electrocardiograms into digital waveform, **Journal of Electrocardiology**, Vol. 38, issue 4, 321-323.

Besrour, R., Z. Lachiri and N. Ellouze. 2008. ECG Beat Classifier Using Support Vector Machine, **Information and Communication Technologies: From Theory to Applications** (ICTTA 2008), 1-5.

Chang, C-C. and Chih-Jen Lin. 2001. **LIBSVM : a library for support vector machines**, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

Chang, Y. W., Cho-Jui Hsieh, Kai-Wei Chang, Micheal Ringgard and Chih-Jen Lin. 2010. Training and Testing Low-degree Polynomial Data Mappings via Linear SVM, **Journal of Machine Learning Research** 11, 1425-1444.

Hamilton, P. S. and Willis J. Tompkins. 1986. Quantitative Investigation of QRS Detection Rules Using the MIT/BIH Arrhythmia Database, **Biomedical Engineering, IEEE Transaction**, Vol. BME-33 Issue: 12, 1157-1165.

Hao, Z. and Li-Qing Zhang. 2005. ECG analysis based on PCA and Support Vector Machines, **Neural Networks and Brain** (ICNN&B '05), 743–747.

Jaakkola, T., Fall 2006. Course materials for 6.867 Machine Learning, MIT OpenCourseWare. **Massachusetts Institute of Technology**. Available Source: http://ocw.mit.edu/

Jian, T. and Dinesh P. Mital. July 1996. A microcomputer-based prototype for ECG paper record conversion, **Journal of Network and Computer Applications**, Vol. 19, issue 3, 295-307.

Kao, T., Len-Jon Hwang, Yui-Han Lin, Tzong-Huei Lin and Chia-Hung Hsiao. Octuber 2001. Computer analysis of the electrocardiograms from ECG paper recordings, **Engineering in Medicine and Biology Society**. Vol. 4, 3232-3234

Khadtare, M. S. and J.S. Sahambi. 2004. ECG Arrhythmia Analysis by Multicategory Support Vector Machine, **Springer Berlin / Heidelberg**, vol. 3285/2004, 100-107.

Mark, R. and G. Moody. 1988. **MIT-BIH arrhythmia database directory**, Massachusetts Inst. Technol. (MIT).

Mehta, S. S. and N. S. Lingayat . 2007. Biomedical signal processing using SVM, **Information and Communication Technology in Electrical Sciences** (ICTES 2007), 527-532.

Mitra, S. and M. Mitra. May 2003. An autotmated data extraction system from 12 lead ECG images, **Computer Methods and Programs in Biomedicine**, vol. 71, issue 1, 33-38.

Mitra, S., M. Mitra and B.B. Chaudhuri. October 2004. Generation of digital time database from paper ECG records and Fourier transform-based analysis for disease identification, **Computer in Biology and Medicine 34**, vol. 4, issue 2, 551- 560.

Osowski, S., L.T. Hoai and T. Markiewicz. 2004. Support vector machine-based expert system for reliable heartbeat recognition, **Biomedical Engineering, IEEE Transactions**, Vol. 51, issue 4, 582-589.

Polikar, R., January 12, 2001. **The Wavelet Tutorial**. Available Source: http://users.rowan.edu/~polikar/WAVELETS/WTtutorial.html

Qibin, Z. and Liqing Zhang. 2005. ECG Feature Extraction and Classification Using Wavelet Transform and Support Vector Machines, **Neural Networks and Brain**, 2005(ICNN&B '05), 1089-1892.

Shlens , J., **A Tutorial on Principal Component Analysis**. Available Source: http://www.cs.cmu.edu/~elaw/papers/pca.pdf

Syeda-Mahmood, T., D. Beymer and Fei Wang. 2007. Shape-based Matching of ECG Recordings, **Engineering in Medicine and Biology Society** (EMBS 2007), 2012–2018.

Zellmer, E., Fei Shang and Hao Zhang. 2009. Highly Accurate ECG Beat Classification Based on Continuous Wavelet Transformation and Multiple Support Vector Machine Classifiers, **Biomedical Engineering and Informatics** (BMEI '09), 1-5.

APPENDICES

Appendix A

A Shell script for database and environment preparation

```
#! /bin/bash
A=`seq 100 109`
B=`seq 111 119`
C=`seq 121 124`
D="200 201 203 205 207 208 209 \
  210 212 213 214 215 217 219 \
  "
RECORD="$A $B $C $D"
for i in ${RECORD} ; do
        echo "Download record...${i}"
        wget http://www.physionet.org/physiobank/database/mitdb/${i}.atr
        wget http://www.physionet.org/physiobank/database/mitdb/${i}.hea
        wget http://www.physionet.org/physiobank/database/mitdb/${i}.dat
done

echo "install build-essential"
sudo aptitude install build-essential -y

echo "install libwww"
sudo aptitude install libwww-dev -y

echo "install libcurl"
sudo aptitude install libcurl3 -y

echo "install libsvm"
sudo aptitude install libsvm2 libsvm-dev libsvm-tools python-libsvm -y

echo "Download wfdb-software package"
wget http://physionet.org/physiotools/wfdb.tar.gz
tar xzf wfdb.tar.gz

echo "Intall wfdb-software package"
cd wfdb-10.5.1
./configure --with-libcurl && make && sudo make install

cd ..

echo "Create N directory"
mkdir N
echo "Create L directory"
mkdir L
echo "Create V directory"
mkdir V
echo "Create R directory"
mkdir R

echo "Extract data from each record"
```

```
for i in ${RECORD} ; do
        echo "Extract record#${i} output is ${i}.sam"
        rdsamp -r ${i} > ${i}.sam
        rdann -a atr -r ${i} > ${i}.ann
        cat ${i}.ann | grep N > N/${i}.ann
        cat ${i}.ann | grep L > L/${i}.ann
        cat ${i}.ann | grep V > V/${i}.ann
        cat ${i}.ann | grep R > R/${i}.ann
done

mkdir ./raw_data && mv *sam ./raw_data
mkdir ./annotated_file/ && mv *.ann ./annotated_file/
mkdir ./annotated_file/N_L_R_V_A/
mv N L V R ./annotated_file/N_L_R_V_A/
mkdir ./200Hz_segmented_dwt_20D_data/
mkdir ./200Hz_segmented_dwt_20D_data/N
mkdir ./200Hz_segmented_dwt_20D_data/L
mkdir ./200Hz_segmented_dwt_20D_data/V
mkdir ./200Hz_segmented_dwt_20D_data/R
mkdir ./src
```

Appendix B

Source code for *.dwt file generation

```
# -*- coding: utf-8 -*-
"""
Created on Tue Feb 23 10:30:37 2010

@author: iml
"""

import numpy as np
import pylab as pl
import os
import psyco
import pywt as wt
from scipy import signal


def normalization(x):
    min_signal = min(x);
    return (x - min_signal)/(max(x) - min_signal);


def bp_filter(x):
    """
    This banpass filter use equation from
    Quantitative  Investigation of the QRS Detection Rules
    Using the MIT/BIH Arrhythmia Database
    Patrick s. Hamilton and Willis j. Tompkins
    """

    output = np.zeros_like(x)
    output = np.concatenate((np.zeros(2),output)) # zero padding

    # Low Pass filter
    # y[n] = 2*y[n-1]-y[n-2]+x[n]-2*x[n-6]+x[n-12]
    x = np.concatenate((np.zeros(12),x)) # zero padding

    j = 0
    for i in xrange(2, len(output),1):
        j = i + 10
        output[i] = 2*output[i-1]-output[i-2]+ \
                x[j]-2*x[j-6]+x[j-12]

    x = output[2:] # ignore padded output
    output = np.zeros_like(x)
    x = np.concatenate((np.zeros(4), x))

    # High Pass filter
    # y[n] = (2*x[n]+x[n-1]-x[n-3]-x[n-4])/8
```

```
    for i in xrange(4, len(x), 1):
        output[i-4] = (2*x[i]+x[i-1]-x[i-3]-x[i-4])/8

    return output + 6 # delay 6 samples (From paper)

def resampling(up,down, x):
    """
    resamples the sequence in vector x ,
    using a numpy.interp and scipy.signal.resample.
    """

#    if type(up) != 'int' and type(down) != 'int' :
#        raise TypeError('up and down parameter must be integer')

    t = np.linspace(0, 1, x.size)
    tvals = np.linspace(0, 1, t.size*up)
    xinterp = np.interp(tvals, t, x)    # up sample use linear-interpolation
    return signal.resample(xinterp, len(xinterp)/down) # decimation


if __name__ == '__main__' :
    psyco.full()
    N = 0
    data_dwt = np.zeros(0)
    for rec in ['N', 'L', 'V', 'R'] :
        data_table = {}
        pl.figure()
        pl.title(rec+'-Case')

        for f in os.listdir("../raw_data/") :
            data_table[f.split('.')[0]] = np.loadtxt('../raw_data/'+f, usecols=[1])

        anno_table = {}
        anno_list = os.listdir("../annotated_file/N_L_R_V_A/"+str(rec))
        for f in anno_list :
            try :
                anno_table[f.split('.')[0]] =
np.loadtxt("../annotated_file/N_L_R_V_A/"+str(rec)+"/"+f, usecols=[1],
dtype='int')[1:-1]
            except :
                print '%s-Case annotation : %s Fail!!!'%(rec,f)
                continue

            print '%s-Case annotation : %s'%(rec,f)

        for i in anno_table.keys() :
            mean_ECG = np.asarray(data_table.get(i)) # read signal from record i
```

```
        print 'Process Record : %s'%i
        #pl.figure()
        #pl.title(str(i)+' : V-Case;')

        for peak_index in anno_table.get(i) :
           if peak_index < 200  : continue

           data = mean_ECG[peak_index-20:peak_index+21]
           peak_loc = data.argmax()
           if peak_loc > 20 :
              peak_index = peak_index + ( peak_loc - 20 )
           else :
              peak_index = peak_index - ( 20 - peak_loc )

           data = mean_ECG[ peak_index - 115 : peak_index + 116 ]

           data = resampling( 5 , 9 , data )
           data = normalization( data )
           data -= data.mean()

           cA, cD3, cD2, cD1 = wt.wavedec(data, 'db1', level=3)
           coeff = [cA, cD3]
           recons_ecg = wt.waverec( coeff, 'db1')

           peak_loc = recons_ecg.argmax()

           if peak_loc < 14 or peak_loc > 16 :
                                        continue
           else :
                                        recons_ecg = recons_ecg[peak_loc-
14:peak_loc+14]


           pl.plot(recons_ecg, 'k')
           pl.plot(recons_ecg.argmax(), recons_ecg.max(), '+r')

np.savetxt('../200Hz_segmented_dwt_20D_data/'+rec+'/'+rec+str(i)+'_'+str(peak_inde
x)+'.dwt', recons_ecg)

    pl.show()
```

Appendix C

Source code for *.pca file generation

```
# -*- coding: utf-8 -*-
"""
Created on Wed Feb 24 00:55:41 2010

@author: iml
"""

import numpy as np
import glob

train_nlvr = open('./train_ecg_NLVR_data.svm', 'w')
test_nlvr  = open('./test_ecg_NLVR_data.svm', 'w')

data_dwt = np.zeros(0)
N = 0
dim = 20

count = 0
Ncase = np.zeros(0)
Nlist = glob.glob('../200Hz_segmented_dwt_20D_data/N/*.dwt')
for f in Nlist :
  Ncase = np.concatenate( ( Ncase, np.loadtxt( f ) ) )
  count += 1

dimension = Ncase.size/count
N += count
data_dwt = np.concatenate( ( data_dwt , Ncase ) )
Ncase = Ncase.reshape(count,dimension)

count = 0
Lcase = np.zeros(0)
Llist = glob.glob('../200Hz_segmented_dwt_20D_data/L/*.dwt')
for f in Llist :
  Lcase = np.concatenate( (Lcase, np.loadtxt( f ) ) )
  count += 1

N += count
data_dwt = np.concatenate( ( data_dwt , Lcase ) )
Lcase = Lcase.reshape(count,dimension)

count = 0
Vcase = np.zeros(0)
Vlist = glob.glob('../200Hz_segmented_dwt_20D_data/V/*.dwt')
for f in Vlist :
  Vcase = np.concatenate( (Vcase, np.loadtxt( f ) ) )
  count += 1
```

```
N += count
data_dwt = np.concatenate( ( data_dwt , Vcase ) )
Vcase = Vcase.reshape(count,dimension)

count = 0
Rcase = np.zeros(0)
Rlist = glob.glob('../200Hz_segmented_dwt_20D_data/R/*.dwt')
for f in Rlist :
  Rcase = np.concatenate( (Rcase, np.loadtxt( f ) ) )
  count += 1

N += count
data_dwt = np.concatenate( ( data_dwt , Rcase ) )
Rcase = Rcase.reshape(count,dimension)

##########################################################################
###########
print '+'*40
print 'Do PCA'

# Number of trails equal N
# Number of dimensions equal min_length
data_dwt = data_dwt.reshape( ( N, dimension ) )

# Each rows is dimension of data
# Each columns is number of trail
data_dwt_trans = data_dwt.transpose()

# Create covariance matrix
# Skip mean subtraction step because data is zero mean
cov_mat = 1./(N-1)*np.dot(data_dwt_trans, data_dwt)

U,S,Vt = np.linalg.svd(cov_mat)

PC = Vt.transpose()[:,:20] # Select 20 of most significant componant

for i in xrange( 0, dim, 1):
   np.savetxt('./NLVRpc'+str(i)+'.pca', PC[i])

Npca = np.dot( Ncase, PC )
Lpca = np.dot( Lcase, PC )
Vpca = np.dot( Vcase, PC )
Rpca = np.dot( Rcase, PC )

data = Npca
for i in xrange( 0, data.shape[0], 2 ) :
```

```
   train_nlvr.write('0 1:' +str(data[i,0])+' 2:'+str(data[i,1])+' 3:'+str(data[i,2])+'
4:'+str(data[i,3])+ ' 5:' +str(data[i,4])+\
                ' 6:' +str(data[i,5])+' 7:'+str(data[i,6])+' 8:'+str(data[i,7])+'
9:'+str(data[i,8])+' 10:' +str(data[i,9])+\
                ' 11:'+str(data[i,10])+' 12:'+str(data[i,11])+' 13:'+str(data[i,12])+'
14:'+str(data[i,13])+' 15:' +str(data[i,14])+\
                ' 16:'+str(data[i,15])+' 17:'+str(data[i,16])+' 18:'+str(data[i,17])+'
19:'+str(data[i,18])+' 20:' +str(data[i,19])+'\n')

for i in xrange( 1, data.shape[0] ,2 ) :
   test_nlvr.write('0 1:' +str(data[i,0])+' 2:'+str(data[i,1])+' 3:'+str(data[i,2])+'
4:'+str(data[i,3])+ ' 5:' +str(data[i,4])+\
                ' 6:' +str(data[i,5])+' 7:'+str(data[i,6])+' 8:'+str(data[i,7])+'
9:'+str(data[i,8])+' 10:' +str(data[i,9])+\
                ' 11:'+str(data[i,10])+' 12:'+str(data[i,11])+' 13:'+str(data[i,12])+'
14:'+str(data[i,13])+' 15:' +str(data[i,14])+\
                ' 16:'+str(data[i,15])+' 17:'+str(data[i,16])+' 18:'+str(data[i,17])+'
19:'+str(data[i,18])+' 20:' +str(data[i,19])+'\n')

data = Lpca
for i in xrange( 0, data.shape[0], 2 ) :
   train_nlvr.write('1 1:' +str(data[i,0])+' 2:'+str(data[i,1])+' 3:'+str(data[i,2])+'
4:'+str(data[i,3])+ ' 5:' +str(data[i,4])+\
                ' 6:' +str(data[i,5])+' 7:'+str(data[i,6])+' 8:'+str(data[i,7])+'
9:'+str(data[i,8])+' 10:' +str(data[i,9])+\
                ' 11:'+str(data[i,10])+' 12:'+str(data[i,11])+' 13:'+str(data[i,12])+'
14:'+str(data[i,13])+' 15:' +str(data[i,14])+\
                ' 16:'+str(data[i,15])+' 17:'+str(data[i,16])+' 18:'+str(data[i,17])+'
19:'+str(data[i,18])+' 20:' +str(data[i,19])+'\n')

for i in xrange( 1, data.shape[0] ,2 ) :
   test_nlvr.write('1 1:' +str(data[i,0])+' 2:'+str(data[i,1])+' 3:'+str(data[i,2])+'
4:'+str(data[i,3])+ ' 5:' +str(data[i,4])+\
                ' 6:' +str(data[i,5])+' 7:'+str(data[i,6])+' 8:'+str(data[i,7])+'
9:'+str(data[i,8])+' 10:' +str(data[i,9])+\
                ' 11:'+str(data[i,10])+' 12:'+str(data[i,11])+' 13:'+str(data[i,12])+'
14:'+str(data[i,13])+' 15:' +str(data[i,14])+\
                ' 16:'+str(data[i,15])+' 17:'+str(data[i,16])+' 18:'+str(data[i,17])+'
19:'+str(data[i,18])+' 20:' +str(data[i,19])+'\n')

data = Vpca
for i in xrange( 0, data.shape[0], 2 ) :
   train_nlvr.write('2 1:' +str(data[i,0])+' 2:'+str(data[i,1])+' 3:'+str(data[i,2])+'
4:'+str(data[i,3])+ ' 5:' +str(data[i,4])+\
                ' 6:' +str(data[i,5])+' 7:'+str(data[i,6])+' 8:'+str(data[i,7])+'
9:'+str(data[i,8])+' 10:' +str(data[i,9])+\
```

```
                ' 11:'+str(data[i,10])+' 12:'+str(data[i,11])+' 13:'+str(data[i,12])+'
14:'+str(data[i,13])+' 15:' +str(data[i,14])+\
                ' 16:'+str(data[i,15])+' 17:'+str(data[i,16])+' 18:'+str(data[i,17])+'
19:'+str(data[i,18])+' 20:' +str(data[i,19])+'\n')

for i in xrange( 1, data.shape[0] ,2 ) :
   test_nlvr.write('2 1:' +str(data[i,0])+' 2:'+str(data[i,1])+' 3:'+str(data[i,2])+'
4:'+str(data[i,3])+ ' 5:' +str(data[i,4])+\
               ' 6:' +str(data[i,5])+' 7:'+str(data[i,6])+' 8:'+str(data[i,7])+'
9:'+str(data[i,8])+' 10:' +str(data[i,9])+\
               ' 11:'+str(data[i,10])+' 12:'+str(data[i,11])+' 13:'+str(data[i,12])+'
14:'+str(data[i,13])+' 15:' +str(data[i,14])+\
               ' 16:'+str(data[i,15])+' 17:'+str(data[i,16])+' 18:'+str(data[i,17])+'
19:'+str(data[i,18])+' 20:' +str(data[i,19])+'\n')

data = Rpca
for i in xrange( 0, data.shape[0], 2 ) :
   train_nlvr.write('3 1:' +str(data[i,0])+' 2:'+str(data[i,1])+' 3:'+str(data[i,2])+'
4:'+str(data[i,3])+ ' 5:' +str(data[i,4])+\
                ' 6:' +str(data[i,5])+' 7:'+str(data[i,6])+' 8:'+str(data[i,7])+'
9:'+str(data[i,8])+' 10:' +str(data[i,9])+\
                ' 11:'+str(data[i,10])+' 12:'+str(data[i,11])+' 13:'+str(data[i,12])+'
14:'+str(data[i,13])+' 15:' +str(data[i,14])+\
                ' 16:'+str(data[i,15])+' 17:'+str(data[i,16])+' 18:'+str(data[i,17])+'
19:'+str(data[i,18])+' 20:' +str(data[i,19])+'\n')

for i in xrange( 1, data.shape[0] ,2 ) :
   test_nlvr.write('3 1:' +str(data[i,0])+' 2:'+str(data[i,1])+' 3:'+str(data[i,2])+'
4:'+str(data[i,3])+ ' 5:' +str(data[i,4])+\
               ' 6:' +str(data[i,5])+' 7:'+str(data[i,6])+' 8:'+str(data[i,7])+'
9:'+str(data[i,8])+' 10:' +str(data[i,9])+\
               ' 11:'+str(data[i,10])+' 12:'+str(data[i,11])+' 13:'+str(data[i,12])+'
14:'+str(data[i,13])+' 15:' +str(data[i,14])+\
               ' 16:'+str(data[i,15])+' 17:'+str(data[i,16])+' 18:'+str(data[i,17])+'
19:'+str(data[i,18])+' 20:' +str(data[i,19])+'\n')

train_nlvr.close()
test_nlvr.close()
```

Appendix D

Source code for Application

```
"""
Do a mouseclick somewhere, move the mouse to some destination, release
the button.  This class gives click- and release-events and also draws
a line or a box from the click-point to the actual mouseposition
(within the same axes) until the button is released.  Within the
method 'self.ignore()' it is checked wether the button from eventpress
and eventrelease are the same.

"""

import sys, time

try :
   import numpy as np
except :
   raise SystemExit("""
            python-numpy must be install before use this software.
            """)

try :
   import pywt as wt
except :
   raise SystemExit("""
                pywt must be install before use this software.
                """)

try :
   from svm import *
except :
   raise SystemExit("""
```

```
                libsvm-dev libsvm-tools python-libsvm libsvm2
                must be install before use this software.
                """)

try :
  # for use hilbert transform, resample
  from scipy import signal as dsp
except :
  raise SystemExit("""
                scipy library must be installed to run this software
                """)

try :
  from matplotlib.widgets import RectangleSelector, Button
except ImportError, exc :
  raise SystemExit("""
                matplotlib must be installed to run this software
                """)

try :
  #from pylab import subplot, arange, plot, sin, cos, pi, show
  from pylab import *
except ImportError, exc :
  raise SystemExit("""
                pylab must be installed to run this software
                """)

try:
    import Image, ImageFilter
except ImportError, exc :
```

```
    raise SystemExit("""
                PIL must be installed to run this software
                """)

try :
  from opencv.cv import *
  from opencv.highgui import *
except :
  raise SystemExit("""
                python-opencv, libcv1, libhighgui1 must be installed to run this software
                """)

try :
  import psyco
  psyco.full()
except :
  print 'Run without psyco'


#img_file = '../png/ECG004_300.png'
img_file = './ECG001_300.png'

try :
  img_orig = Image.open(img_file).transpose(Image.FLIP_TOP_BOTTOM)
except :
  raise SystemExit('Cannot open image file')


figure()
ax = axes([0,0,1,1], frameon=False)
```

```
im = imshow(img_orig)


X,Y = img_orig.size


bin_img = None
tmp_img = []
img     = None



BLACK_COLOR       = 0
WHITE_COLOR       = 255
MARKED_FOR_DELETE = 123
BACKGROUND_COLOR  = WHITE_COLOR


def resampling(up,down, x):
    """
    resamples the sequence in vector x ,
    using a numpy.interp and scipy.signal.resample.
    """


#    if type(up) != 'int' and type(down) != 'int' :
#        raise TypeError('up and down parameter must be integer')


    print 'Resampling'
    t = np.linspace(0, 1, x.size)
    tvals = np.linspace(0, 1, t.size*up)
    xinterp = np.interp(tvals, t, x)    # up sample use linear-interpolation
    return dsp.resample(xinterp, len(xinterp)/down) # decimation


def InitTmpImg2Zero():
```

```
  global tmp_img

  for r in range(0,tmp_img.rows,1):
    for c in range(0,tmp_img.cols,1):
      tmp_img[r][c] = 123

def ThinImg():
  global img
  global tmp_img


  for c in range(2, img.cols-2,1):
    max_data  = {"row":0, "col":0, "dat":tmp_img[2][c]}

    for r in range(2, img.rows-2,1):
      if tmp_img[r][c] < max_data["dat"] :
        max_data["row"] = r

    for r in range(2, img.rows-2,1):
      if r != max_data["row"] :
        tmp_img[r][c] = BACKGROUND_COLOR
      else :
        tmp_img[r][c] = BLACK_COLOR

ecg_signal = []
def ExtractSignal():
  global img
  global ecg_signal

  ecg_signal = []
```

```
 for c in range(2,img.cols-2,1):
  for r in range(2,img.rows-2,1):
   if img[r][c] == BLACK_COLOR :
    if True :
     ecg_signal.append(r)

    # if it is not an adjacent pixel
    elif len(ecg_signal) > 1 and abs(ecg_signal[-1] - r) > 10 :
     if ecg_signal[-1] - ecg_signal[-2] > 0 : # rising edge
      ecg_signal.append(ecg_signal[-1]+1)
     else :
      ecg_signal.append(ecg_signal[-1]-1)
    else :
     ecg_signal.append(r)

def UpdateImg():
 global tmp_img
 global img

 bc = 0
 wc = 0
 for r in range(2,img.rows-2,1):
  for c in range(2,img.cols-2,1):
   if tmp_img[r][c] == BLACK_COLOR :
    bc += 1
    img[r][c] = BLACK_COLOR

   elif tmp_img[r][c] == WHITE_COLOR :
    wc += 1
    img[r][c] = WHITE_COLOR
```

```
  print "Fill Black Pixel", bc
  print "Remove Black Pixel", wc
  InitTmpImg2Zero()


def moving_average():
  global img
  global tmp_img

  tmp = 0
  for c in range(2, img.cols-2, 1):
    for r in range(2, img.rows-2, 1):
      tmp += img[r-2][c]
      tmp += img[r-1][c]
      tmp += img[r][c]
      tmp += img[r+1][c]
      tmp += img[r+2][c]
      tmp /= 5
      tmp_img[r][c] = tmp

  for c in range(2, img.cols-2, 1):
    for r in range(2, img.rows-2, 1):
      if tmp_img[r][c]  != BACKGROUND_COLOR and \
        tmp_img[r-1][c] == BACKGROUND_COLOR and \
        tmp_img[r+1][c] == BACKGROUND_COLOR :

        print 'Remove noise @ (%d,%d)'%(r,c)
        tmp_img[r][c] = BACKGROUND_COLOR
```

```
  cvSaveImage("/tmp/moving_average_img.jpg", img)

#  time.sleep(2)
  figure()
  title('Moving average')
  imshow(Image.open('/tmp/moving_average_img.jpg').convert('1'))

def thin_image(file_name):
  """
  This function use openCV.
  """
  global bin_img
  global tmp_img
  global img

  # Load image as gray scale mode
  try :
   image = cvLoadImage( file_name, 0 )
  except :
   print "Can't open image '%s'" % file_name
   sys.exit(1)

  bin_img  = cvCloneImage(image)

  cvThreshold(image, bin_img,
            130, 255,
            CV_THRESH_BINARY)

  cvSaveImage("/tmp/bin_img.jpg", bin_img)
```

```
# for 5 pixel moving average
img = cvCreateImage( cvSize(image.cols+4, image.rows+4),8,1)

print "Init img object"
# Init two upper most rows
for r in range(0,2,1):
  for c in range(0, img.cols, 1):
    img[r][c] = BACKGROUND_COLOR

# Init two lower most rows
for r in range(img.rows - 2, img.rows, 1):
  for c in range(0, img.cols, 1):
    img[r][c] = BACKGROUND_COLOR

# Init two left most rows
for r in range(0,img.rows, 1):
  for c in range(0, 2, 1):
    img[r][c] = BACKGROUND_COLOR

# Init two right most rows
for r in range(0, img.rows, 1):
  for c in range(img.cols-2,img.cols,1):
    img[r][c] = BACKGROUND_COLOR

# Init data
for r in range(0,bin_img.rows,1):
  for c in range(0,bin_img.cols,1):
    img[r+2][c+2] = bin_img[r][c]

tmp_img  = cvCloneImage(img)
```

```
  InitTmpImg2Zero()

  print "Moving Average"
  moving_average()

  print "Thinning Image"
  ThinImg()
  UpdateImg()
  ExtractSignal()

  cvSaveImage("/tmp/thinned_img.jpg", img)

  print "Thin Image Complete"

def normalization(signal):
  min_signal = min(signal)
  return (np.asarray(ecg_signal, dtype='float') - min_signal)/(max(ecg_signal) - min_signal)


def ECG_process(event):
  global ecg_signal

  print 'Process ECG'
  print 'Time series signal extraction'

  figure()
  title('ECG signal -- Time Series')
  #plot(bp_filter(normalization(asarray(ecg_signal))))
  #plot(normalization(asarray(ecg_signal)))
  plot(ecg_signal, '+r')
```

```
plot(ecg_signal)

#normalized_ecg = asarray(ecg_signal, dtype=float)/asarray(ecg_signal).max()
#normalized_ecg = normalization( asarray(ecg_signal, dtype=float) )
normalized_ecg = normalization( ecg_signal )
mean_ecg = normalized_ecg - normalized_ecg.mean()

abs_mean_ecg     = abs(mean_ecg)
hil_abs_mean_ecg = dsp.hilbert( abs_mean_ecg )

max_loc = hil_abs_mean_ecg.argmax()
max_val = hil_abs_mean_ecg.max()

if max_val == hil_abs_mean_ecg[max_loc - 1] :
  max_loc -= 1

#plot(ecg_signal)
figure()
title('QRS detection -- Output')
plot( mean_ecg )
plot(hil_abs_mean_ecg)
plot(max_loc, max_val, '+r')

try :
  data     = mean_ecg[ max_loc-92 : max_loc+93 ].copy()
except :
  do_classify = False
  print 'Cannnot classify because lenght of data < 185'

if data.size == 185 :
```

```
  do_classify = True
  figure()
  title('data')
  plot(data)
else :
  do_classify = False
  print 'data.size != 185'

print 'data', data

if do_classify :
  print 'ECG beat classification'
  # 185 samples --> 125 samples and ignore last sample
  data = resampling(40, 59, data)[:-1]
  print 'Do DWT'
  cA, cD3, cD2, cD1 = wt.wavedec(data, 'db1', level=3)
  coeff = [ cA, cD3 ]
  #recons_ecg = wt.waverec( coeff, 'db1')[6:26] # 20 samples of signal
  recons_ecg = wt.waverec( coeff, 'db1')

  peak_loc = recons_ecg.argmax()
  if peak_loc < 14 or peak_loc > 16 :
     return
  else :
     recons_ecg = recons_ecg[peak_loc-10:peak_loc+10]

  print 'sizeof(recons_ecg) : ', len(recons_ecg)
  figure()
  title('IDWT-Output')
  plot(recons_ecg)
```

```
#if recons_ecg.argmax() != 9 :
#    return
#else :
if True :
    # Load model for SVM evaluation
    print 'Load Model for SVM-Classification'
    try :
        m = svm_model('./train_ecg_NLVR_data.svm.model')
    except :
        raise SystemExit("""
                    Cannot load ./train_ecg_NLVR_data.svm.model
                    for svm model.
                    """)

    # load 20D-PCA for map data
    PC = np.zeros(0)
    N = 0
    for pc in range(20) :
        try :
            pc = np.loadtxt("./NLVRpc"+str(pc)+".pca")
        except :
            raise SystemExit("Cannot load ./NLVRpc"+str(pc)+".pca")


        try :
            PC = np.concatenate( (PC, pc) )
            N += 1
        except :
            raise SystemExit("Cannot concatenate data")
```

```
        PC = PC.reshape( ( N, 20 ) )
        print 'shape(PC) : ', shape(PC)
        #print 'PC :',PC

        print 'Do PCA-20D'
        eval_data = np.dot( recons_ecg , PC )[:20]

        print 'Compute confidence measure'
        pred, prob = m.predict_probability(eval_data)
        if int(pred) == 0 :     T = 'N'
        elif  int(pred) == 1 :  T = 'L'
        elif  int(pred) == 2 :  T = 'V'
        elif  int(pred) == 3 :  T = 'R'

        print ('CM-Output -> %s(%f)')%( T,  prob[ int(pred) ])
        text( recons_ecg.argmax() + 2, recons_ecg.max()-recons_ecg.max()/2, T + ' : ' + str( prob[
int(pred) ]) )


def line_select_callback(event1, event2):

  'event1 and event2 are the press and release events'
  X1, Y1 = int(event1.xdata), int(event1.ydata)
  X2, Y2 = int(event2.xdata), int(event2.ydata)

  print
  print '0,0 --> %d,%d '%(X,Y)
  print "(%d, %d) --> (%d, %d)"%(X1,Y1,X2,Y2)
```

```
if X1 > X2 :
  Right = X1 ; Left  = X2

elif X2 > X1  :
  Right = X2 ; Left  = X1

if Y1 > Y2 :
  Upper = Y1 ; Lower = Y2

elif Y2 > Y1 :
  Upper = Y2 ; Lower = Y1

print "{%d, %d} --> {%d, %d}"%( Left, Lower, Right, Upper)

Upper = Y - Upper
Lower = Y - Lower

#print "[%d, %d] --> [%d, %d]"%( Left, Lower, Right, Upper)
#result = img.crop((Left,Lower,Right,Upper))
print "[%d, %d] --> [%d, %d]"%( Left, Upper, Right, Lower )
im = img_orig.crop(( Left, Upper, Right, Lower ))
im.save('/tmp/crop_img.jpg')
time.sleep(2)
thin_image('/tmp/crop_img.jpg')
time.sleep(2)

figure()
title('Binary Image')
imshow(Image.open('/tmp/bin_img.jpg').convert('1'))
```

```
figure()
subplots_adjust(bottom=0.2)
title('Thinned Image')
imshow(Image.open('/tmp/thinned_img.jpg').convert('1'))

axprev = axes([0.7,  0.05, 0.1, 0.075])
bprev = Button(axprev, 'Process')
bprev.on_clicked(ECG_process)

# drawtype is 'box' or 'line' or 'none'
LS = RectangleSelector(ax, line_select_callback,
                       drawtype='box',useblit=True,
                       minspanx=5,minspany=5,spancoords='pixels')

print "\n      click  -->  release"

show()
```

CIRRICULUM VITAE

NAME : Mr. Chaiwat Suwansaroj

BIRTH DATE : July 19, 1983

BIRTH PLACE : Phang-nga, Thailand

EDUCATION :

YEAR	**INSTITUTE**	**DEGREE/DIPLOMA**
2007	Kasetsart Univ.	B.Eng. (Electrical Engineering)
2010	Kasetsart Univ.	M.Eng. (Information and Communication Technology for Embedded Systems)

POSITION/TITLE : -

WORK PLACE : -

SCHOLARSHIP/AWARDS : TAIST ICTES Master Degree Scholarship