行政院國家科學委員會專題研究計畫 成果報告

智慧型心電圖系統之建構及其於臨床環境之應用
研究成果報告(精簡版)

計 畫 類 別 ：個別型
計 畫 編 號 ：NSC 95-2221-E-216-041-
執 行 期 間 ：95 年 08 月 01 日至 96 年 07 月 31 日
執 行 單 位 ：中華大學生物資訊學系

計 畫 主 持 人：曾文慶
共 同 主 持 人：謝瑞建、魏崢、田慶誠
計畫參與人員：共同主持人：謝瑞建、田慶誠、魏崢
計畫參與人員：蔣家正、鄭涵菁

處 理 方 式 ：本計畫可公開查詢

中 華 民 國 96 年 10 月 30 日

行政院國家科學委員會補助專題研究計畫 ▓成果報告 □期中進度報告

智慧型心電圖系統之建構及其於臨床環境之應用

計畫類別：▓ 個別型計畫　　□ 整合型計畫
計畫編號：NSC　95-2221-E-216 -041
執行期間：95　年　8　月　1　日至　96　年　7　月　31　日

計畫主持人：曾文慶
共同主持人：謝瑞建、田慶誠、魏崢
計畫參與人員：蔣家正、鄭涵菁

成果報告類型(依經費核定清單規定繳交)：■精簡報告　□完整報告

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
■出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
□涉及專利或其他智慧財產權，□一年□二年後可公開查詢

執行單位：中華大學生物資訊學系

中　華　民　國　96　年　10　月　30　日

行政院國家科學委員會專題研究計畫成果報告

智慧型心電圖系統之建構及其於臨床環境之應用

The construction of a smart electrocardiogram system and its application in clinical environment

計畫編號：NSC　95-2221-E-216 -041

執行期限：95 年 8 月 1 日至 96 年 7 月 31 日

主持人：　曾文慶 中華大學生物資訊學系

共同主持人：謝瑞建 元智大學資訊管理學系

共同主持人：田慶誠 中華大學通訊工程學系

共同主持人： 魏崢　財團法人振興復健醫學中心心臟醫學中心

計畫參與人員：蔣家正 (交通大學 電機與控制工程學系)、鄭涵菁(交通大學 資訊工程學系)

一、中文摘要

近年來本研究室於 12 導程心電圖資料庫與分析系統之研究，已獲得多項成果。本計劃以這些成果為基礎，應用國人自行研發的輕便型心電儀，與本校通訊工程系、台北振興復健醫學中心及苗栗為恭醫院合作，發展一套自動化之智慧型心電圖系統。系統架構採用模組化設計，以心電圖管理分析單元為核心，藉由網頁式介面之閘道(gateway)整合心電儀、資料庫伺服器及醫療資訊系統(Hospital Information System, HIS)等單元，即可執行 ECG 訊號之擷取、ECG 紀錄之存取、ECG 資料庫管理及 ECG 訊號分析等功能，提供臨床應用、健康照護應用及生醫研究等服務。由於系統以獨立模組之方式提供電子化心電圖服務，既不會干擾醫院現行之資訊系統，也具有易於維護與管理之特性。

考量上述情形，為了能取得供研究用的原始心電圖數據及推動開放式的心電圖檔案格式，本研究設計了一組網頁式心電圖訊號擷取分析系統。本系統主要由兩個部分組成：硬體部分包括個人電腦與數據擷取裝置；軟體部分則包含網頁式心電訊號擷取介面，儲存心電圖資料的關聯式資料庫，管理整個系統的網頁式介面及一組以 PHP 及 Matlab 開發的心電圖格式轉換，數據分析與圖形顯示工具程式。

本研究為利用小波轉換來進行各種疾病的心電圖特徵點自動化萃取，其中心電圖可分類為正常、急性心肌梗塞(Acute Myocardial Infarction)、高血鉀症(Hyperkalemia)與心跳每分鐘大於 150 下。藉由不同尺度的小波係數，套用自行開發的演算法來判別心電圖的 P 波、QRS 波與 T 波。研究結果為:(1) R 波的 Sensitivity 平均為 99%，Specificity 為 99.9%；(2) Q 點與 J 點 Sensitivity 平均為 97%，Specificity 為 99.9%；(3) T 波的 Sensitivity 平均為 95%，Specificity 為 99.9%；(4) P 波的 Sensitivity 平均為 92.5%，Specificity

為 99.9%。此外本研究以 JAVA 語言開發類神經網路(Neural network)，K-means 與隱藏式馬可夫模型(Hidden Markov model)等三組程式，作為未來資料探勘模組之工具程式。

關鍵詞：心電圖、心電圖格式、訊號擷取、小波包分析、離散小波轉換、高血鉀症、心肌梗塞、P 波、QRS 波、T 波、類神經網路、K-means、隱藏式馬可夫模型

Abstract

Based on the portable ECG developed domestically, along with cooperation of Wei Gong Memorial Hospital and Cheng Hsin General Hospital, we developed an ECG system which is built on a module architecture, with the ECG Management and Analysis Unit as its primary core, combined with web-based gateways, portable electrocardiogram, ECG database server and Hospital Information System (HIS), to collect ECG signals, manage the ECG database, and analyze ECG signals and provide web-based interfaces to suit clinical applications, health care applications, and medical research purposes.

To facilitate the acquiring of raw ECG data for research work and to promote the use of open ECG file format, we design a web-based ECG acquisition and analysis system in this study. The system has two major components: one is the hardware part, which includes a personal computer and a data acquisition device; the other is the software part, which includes a web-based data acquisition interface, a relational database for storing ECG's, a web-based management interface and a set of tools developed in PHP and MatLab for format transform, data analysis, and graphic representation of ECG records. The collected data then can be analyzed using the programs in the Matlab-based toolbox to search for parameters which are ECG related characteristics for various heart diseases.

In this research, we used wavelet transform to extract ECG characteristic parameters for various kinds of cardiac diseases, including Acute Myocardial Infarction、Hyperkalemia、Normal and heart rate more than 150 per second. Wavelet coefficients at different scales and a new algorithm were used to find the locations and durations of P wave、QRS complex and T wave. The results are as follows: (1)the average Sensitivity of R wave is 99%, the Specificity is 99.9%; (2) the average Sensitivity of Q and J wave is 97%, Specificity is 99.9%; (3) the average Sensitivity of T wave is 95%, Specificity is 99.9%; (4) the average Sensitivity of P wave is 92.5%, Specificity is 99.9%. In addition, three data mining tools, Neural network, K-means and Hidden Markov model, were developed in JAVA programming language.

Keywords: Electrocardiogram, ECG

format, Signal acquisition, Wavelet packet analysis , Discrete wavelet transform, Hyperkalemia, Acute myocardial infarction, P wave, QRS complex, T wave, Neural network, K-means, Hidden Markov model

二、緣由與目的

近年來由於經濟發展及資訊普及化，人們對於醫療照護及自我健康管理之意識有逐年升高之趨勢。為了能提供品質優良的醫療照護及自我健康管理，生醫訊號量測設備是最前端的必需工具之一。同時生醫訊號之擷取、儲存、管理與分析也是生醫領域病情診斷、病情監控及基礎醫學研究中重要的一環。傳統上許多生醫訊號量測儀器會因本身各裝置間連接線材及原先設計軟硬體功能之限制，造成醫療人員及病人極大的不便，使這些設備不論就使用地點及應用價值而言，都受到很大的限制。另一方面，量測所得之數據往往只能以報表形式輸出，既不利於電子化病歷之建立，生醫研究人員也往往錯失許多寶貴的研究資料。

本有鑒於此，本計劃應用現今資訊與通訊之科技，與振興醫院、為恭醫院之醫師及資訊部門合作，將 ECG 訊號擷取設備予以自動化，並配合資料庫與管理分析軟體，經由網頁式介面之閘道(gateway)與醫療院所內的資訊系統結合後，本智慧型心電圖系統將可達到下述目標：

1. 可直接將 ECG 訊號量測結果整合入電子病歷，利於病患整體病例之管理。
2. 以本研究室先前開發之心電圖資料庫結構為基礎，設計適應醫院內應用環境之 ECG 訊號資料庫。
3. 建立能搭配院內醫療資訊系統之 ECG 資料庫伺服器。此系統獨立於醫療資訊系統之外，在不增加醫療資訊系統負擔及干擾醫療資訊系統運作之原則下，提升醫療資訊系統協助臨床醫師診斷及協助生醫相關研究之功能。
4. 經由資料庫之儲存管理系統，醫師可立即調閱相關之先前量測結果。智慧型心電圖系統配合醫院的醫療資訊系統及醫師之需求，設計整合的介面，醫師使用時，就如同其原有之作業環境，但增加了瀏覽、搜尋、比對及分析這些 ECG 檢測訊號之功能。
5. 運用生物資訊技術，如類神經網路 (Neural network)，K-mean，及小波分析(wavelet analysis)等方法，開發 ECG 訊號分析程式，探討 ECG 訊號特徵與心臟疾病徵狀之關連性。
6. 存於資料庫之 ECG 數據，經判別疾病之軟體分析之後，其報告將有助於醫師對病情之診斷。

三、研究方法

網頁式心電圖訊號擷取分析系統之架構如圖 1 所示。以心電圖資料庫與心電訊號分析系統(ECG Database and Analysis System, ECGDAS)為核心之架構，前端為心電訊號擷取裝置，負責病人心電訊號之量測；資料庫伺

服器負責心電圖之儲存；使用者端則經由網頁式介面，提供心電圖格式轉換，瀏覽與分析之功能。

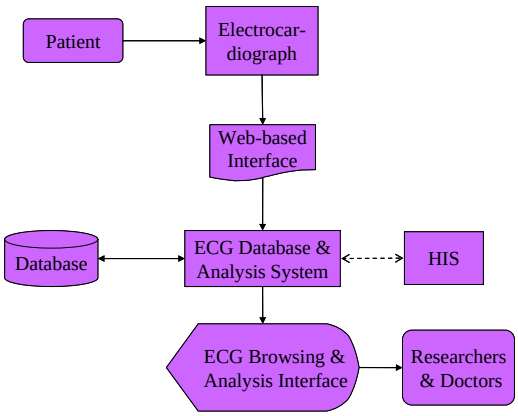

圖 1 網頁式心電圖訊號擷取分析系統架構

(一) 心電訊號擷取

由生訊科技股份有限公司(台北市)開發，內含感測器、濾波器、放大器、安全電路及類比數位轉換器之 12 導程心電訊號擷取裝置(BEST-B728A25A-12C, 圖 2)之組成元件如圖 3 所示。

操作 12 導程心電訊號擷取裝置時，使用者經由本研究開發之網頁式心電訊號擷取介面設定各項參數，經數位化後之心電圖數據經由 RS232 介面傳輸至電腦，進行後續之儲存、管理及分析。

圖 2 12 導程心電訊號擷取裝置

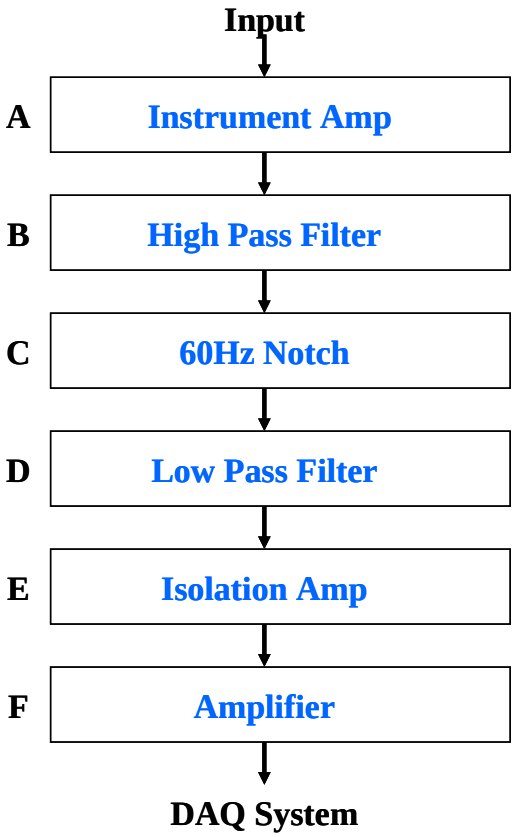

圖 3 心電訊號擷取裝置之組成元件

(二) 心電圖資料庫

資料庫系統以 Apache、PHP、MySQL 開發，採用三層式架構：第一層為 MySQL 資料庫與檔案目錄系統；第二層以 Apache 與 PHP 為系統之核心，負責與其他元件的溝通及邏輯運算，心電圖的分析與格式的轉換也在此進行；第三層為使用者瀏覽界面(圖 4)。

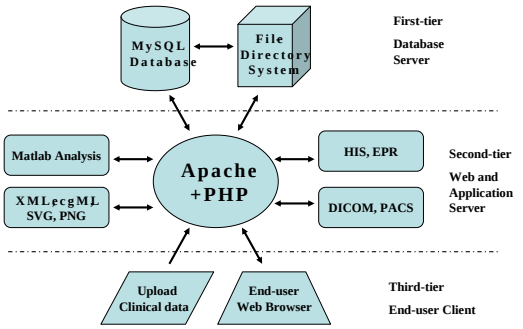

圖 4 三層式架構之心電圖資料庫

(三) 心電圖格式轉換

本研究分析數種開放原始碼之心電圖格式，以 PHP 及 Matlab 開發這些格式之轉換工具程式，心電圖資料可以透過所建立的伺服器轉換成 ASCII、XML、SVG、ecgML、PNG 和 DICOM 等格式。

由心電訊號擷取裝置取得之資訊(包括文字內容及心電訊號數據)先以 ASCII 碼儲存為文字檔，接著以 HL7 公佈之標準為依據，將這些資訊轉為 FDA_XML 及 ecgML 格式。至於擷取裝置所得心電圖資訊之呈現，分別以向量圖形及點陣式圖形兩種方式，產生近似於臨床所用之標準 12 導程心電圖報告。向量圖檔採用以 XML 為基礎之二維圖形描述語言 SVG 撰寫，只要支援 SVG 解讀功能之瀏覽器即可呈現心電圖報告之內容，且可在不失真之情況下，放大局部之內容。點陣式圖檔則採用 PNG 與 DICOM 兩種格式。PNG 檔案以 CSIRO 所開發的 SVG 工具程式，直接將 SVG 檔案轉檔產生。這些 PNG 圖檔加上原存於文字檔之資訊，即可以 MatLab 程式碼再轉為 DICOM 檔案。

(四) 心電訊號分析

典型心電圖波形及特徵參數之定義如圖 5 所示。要分析心電圖特徵，就須將這些特徵參數由心電訊號數據中萃取出來。由於 R 波在心電圖波形中通常最高且明顯，因此心電圖特徵參數之萃取，可由 R 波之定位開始，再由其左右找出其他之特徵參數。

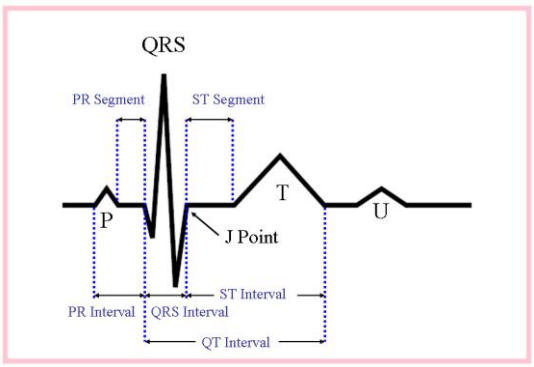

圖 5 心電圖波形及特徵參數示意圖

心電圖特徵參數方面，本研究提出一種新的複合式小波轉換法來定位心電圖的 P 波、QRS 波與 T 波，其方法之敘述與說明如本報告文末之附文所示。至於資料探勘模組，本研究以 JAVA 語言開發類神經網路(Neural network)，K-means 與隱藏式馬可夫模型(Hidden Markov model)等三組程式，作為未來資料探勘之工具程式。

三、結果與討論

(一) 網頁式心電訊號擷取介面

心電儀操作介面採網頁式設計，由使用者觀點出發，畫面內容主要分為兩大部分：一為操作參數輸入區；一為心電訊號顯示區。如圖 6 所示心電儀操作人員可輸入或設定之參數包括病歷號碼(Patient ID)、流水號碼

(Accession number)、訊號擷取長度、取樣頻率(sampling rate)及儲存檔案名稱等。至於所測得心電訊號之顯示則可選擇一次顯示 12 個導程，或以三個導程為一組分開顯示在畫面上。

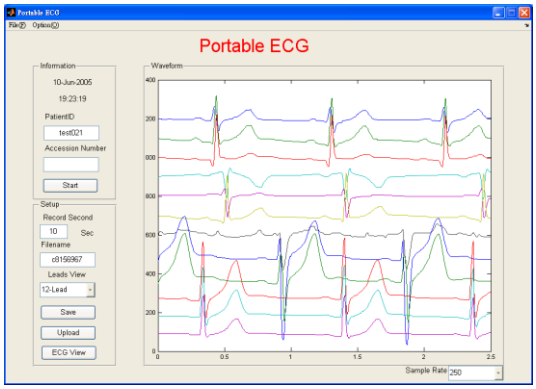

圖 6 網頁式心電訊號擷取介面

(二) 心電圖資料庫

本研究建立之心電圖資料庫為關聯式資料庫管理系統(relational database management system)，其結構(schema)由 7 個表格組成(圖 7)。心電圖資料庫與心電訊號分析之整合系統(ECG Database and Analysis System, ECGDAS)經由網頁式介面管理，其登入後首頁如圖 8 所示。

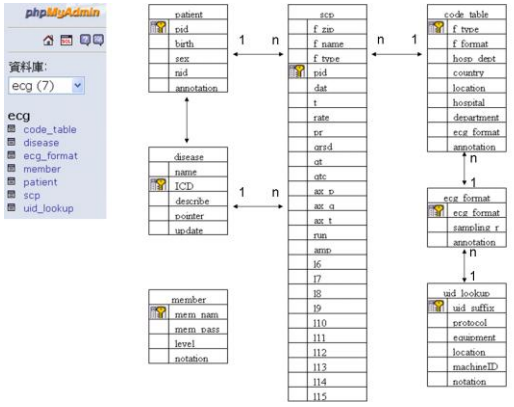

圖 7 心電圖資料庫之結構

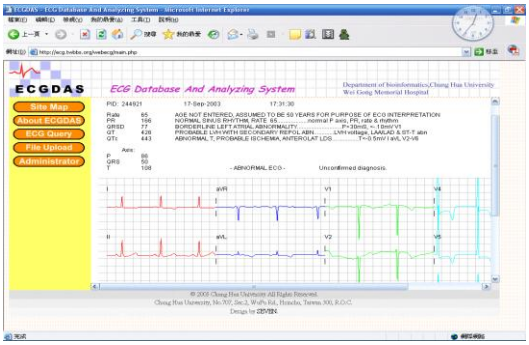

圖 8 ECGDAS 之首頁

本心電圖資料庫除了儲存 12 導程心電儀所量測之資訊外，也可遠端收取來自於苗栗為恭醫院病患之心電圖資料。在過去數年間，本研究的結果已蒐集超過 8 千個以上完整 12 導程心電圖紀錄，這些資料可轉換成 XML、SVG、PNG 和 DICOM 等便於交換之格式，並可進一步做為心電圖特徵參數與心臟疾病相關性分析之用。

將這個資料庫與醫院之醫療資訊系統(Hospital information system, HIS)結合，經由醫師確認後，本研究建立了正常人(Normal)及四種心臟疾病患者之 ECG 資料庫：急性心肌梗塞(Acute myocardial infarction, AMI)、高血鉀症(Hyperkalemia) 、低血鉀症(Hypokalemia)及心房顫動(Atrial Fibrillation, Af)。到目前止，各種病患之心電圖筆數如表 1 所示。

表 1 各種心臟疾病患者於 ECG 資料庫中之筆數

疾病	ECG 筆數
Normal	12
AMI	44
Hyperkalemia	40
Hypokalemia	50
Af	19

本心電圖資料庫伺服器能提供有效的心電圖訊息服務，來幫助臨床醫生的診斷以及協助研究人員的心電圖訊號處理。將來，我們的工作可以包括其他醫學訊號，比如 24 小時心電圖(Holter ECG)、運動心電圖，病患監視器和心音圖等。

(三) 心電圖格式轉換

圖 9 所示為經過轉換程式處理後所得之 FDA_XML 文件，其原始碼可用一般的網頁瀏覽器顯示。圖 10 所示則是利用 AMPS LLC website 所取得之 FDA_XML viewer 所得到之結果，這與臨床上醫師所習慣的 12 導程心電圖有很大的不同。

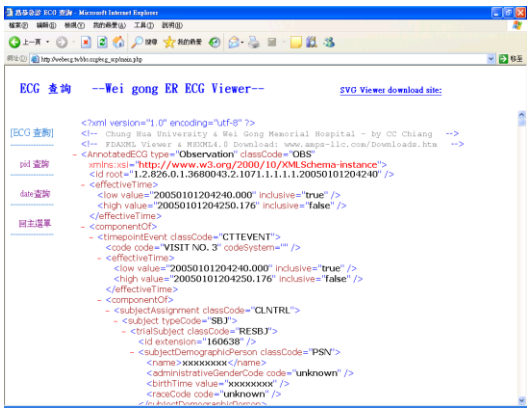

圖 9 網頁瀏覽器所呈現之 FDA_XML 文件原始碼

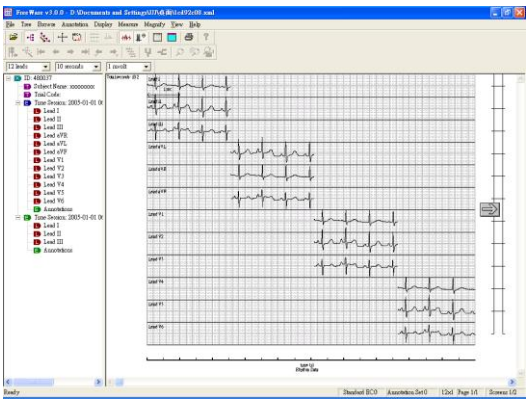

圖 10 FDA_XML viewer 所得之 12 導程心電圖

為了能提供臨床醫師所熟悉的典型 12 導程心電圖及未來能與醫院資訊系統(Hospital information system, HIS)相容，本研究發展工具程式將 FDA_XML 內容以 SVG 語言描述，其結果如圖 11 所示。為了進一步提高系統於網路上之相容性及應用性，由 SVG 描繪之心電圖，可再轉為圖像相同之 PNG 及 DICOM 格式。

圖 11 SVG 文件在瀏覽器中以 Adobe SVG viewer 3.0 plug-in 所得之 12 導程心電圖

(四) 心電訊號分析

心電圖特徵參數之萃取，本研究

提出一種新的複合式小波轉換法來定位心電圖的 P 波、QRS 波與 T 波。結果發現，採用本研究開發之方法，對於各種疾病之心電圖，均能得到很高之靈敏度 (sensitivity) 與專一性 (specificity)，其詳細之結果與討論，請參閱本報告文末之附文。至於資料探勘模組，本研究以 JAVA 語言開發類神經網路(Neural network)，K-means 與隱藏式馬可夫模型 (Hidden Markov model)等三組程式，作為未來資料探勘之工具程式。此三個程式之主程式內容如本報告文末之附文所示。

四、計畫成果自評

在本研究中，我們完成了：

1. 建立一個由心電訊號擷取裝置與個人電腦為硬體組成之網頁式心電訊號擷取分析系統。搭配以 PHP 及 MatLab 開發之心電圖資料庫與心電訊號分析系統 (ECGDAS)，可做為臨床醫師診斷之輔助工具，也可協助研究人員進行心電圖資料之蒐集與分析。
2. 經由與為恭醫院之合作，在過去二年間，本研究蒐集超過 8 千個以上完整的 12 導程心電圖紀錄。將資料庫與醫院之醫療資訊系統結合，再經由醫師確認後，本研究建立了正常人及四種心臟疾病患者之 ECG 資料庫，其中包括急性心肌梗塞、高血鉀症、低血鉀症及心房顫動等。
3. 藉由以 PHP 或 Matlab 開發之心電圖格式轉換工具程式，心電圖資料可以透過所建立的伺服器轉換成 ASCII、XML、SVG、ecgML、PNG 和 DICOM 等格式。一方面有利於心電圖資料跨平台之交換，另一方面可將心電訊號擷取所取得之數據以典型 12 導程心電圖呈現並且使用網頁瀏覽器即可進行檢閱。
4. 藉由不同尺度的小波係數，套用自行開發的演算法來判別心電圖的 P 波、QRS 波與 T 波。本研究之參數萃取法能適用於多種心臟疾病，且都能得到很搞的靈敏豆與專一性，其結果為:(1) R 波的 Sensitivity 平均為 99%，Specificity 為 99.9%；(2) Q 點與 J 點 Sensitivity 平均為 97%，Specificity 為 99.9%；(3) T 波的 Sensitivity 平均為 95%，Specificity 為 99.9%；(4) P 波的 Sensitivity 平均為 92.5%，Specificity 為 99.9%。
5. 以往的研究所用的資料來源，大都是取自 MIT/BIH 的資料庫，而本研究的資料來源為苗栗為恭醫院急診室的臨床資料，這顯示本研究發展之 ECG 特徵參數偵測程式能適合臨床醫療環境所使用。
6. 本研究以 JAVA 語言開發類神經網路(Neural network)，K-means 與隱藏式馬可夫模型 (Hidden Markov model)等三組程式，將作為未來資料探勘模組之工具程式。

五、參考文獻

Akay M. Wavelets in biomedical engineering. Ann Biomed Eng. 1995 Sep-Oct; 23(5):531-542.

Analyzing Medical Parameters for Solutions, AMPS LLC., 2005. website: http://www.amps-llc.com/

Baxt WG, Shofer FS, Sites FD, and Hollander JE. A neural network aid for the early diagnosis of cardiac ischemia in patients presenting to the emergency department with chest pain. Annals of Emergency Medicine. 2002; 40(6):575-582.

Chiang CC, Yang YC, Tzeng WC, Tseng WD, Hsieh JC, "An SCP Compatible 12-Lead Electrocardiogram Database for Signal Transmission, Storage and Analysis" Computers in Cardiology 2004;31:621-624.

EUROPEAN COMMITTEE FOR STANDARDIZATION. Standard Communications Protocol for Computer-Assisted Electrocardiography. prENV 1064:2000 2000.

Everitt BS. Cluster Analysis. 4th ed. Oxford University Press. 2001.

Fischer R, and Zywietz C, HowToImplement SCP-ECG_Part I. http://www.openecg.net/SCP_howto/How_To_Implement_SCP.pdf 2003.

Fischer R, and Zywietz C. HowToImplement SCP-ECG_Part II. http://www.openecg.net/SCP_howto_II/HowToImplementSCP_PartII.pdf 2003.

Li C et al., Detection of ECG characteristic points using wavelet transforms. IEEE Trans. Biomed. Engng. 1995;42:21-28.

Li T., Li Q., Zhu S., and Ogihara M. A survey on wavelet applications in data mining. SIGKDD Explorations 4(2):49-68, 2003.

JC Hsieh, **WC Tzeng**, YC Yang, SM Shieh, **"**Detecting ECG Characteristic Points by Novel Hybrid Wavelet Transforms: An Evaluation of a Clinical SCP-ECG Database," Computers in Cardiology, 2005:32:751-754

Olmez T and Dokur Z. Classification of heart sounds using an artificial neural network. Pattern Recognition Letters. 2003; 24: 617-629.

Penny W and Frost D. Neural networks in clinical medicine. Med Decis Making. 1996; 16(4):386-398.

Porter RS, Kaplan J, Zhao N, Garavilla LDE, Eynon CA, Wenger FG, and Dalsey WC. Prediction of hyperkalemia in dogs from electrocardiographic parameters using an artifificial neural network. Academic Emergency Medicine. 2001; 8(6):599-603.

Sahambi JS, Tandon SN, and Bhatt RKP, Using Wavelet Transforms for ECG Characterization (An On-Line

Digital Signal Processing System). IEEE Engineering in Meducune and Biology. 1997；：77 – 83

Turkoglu I, Arslan A, and Ilkay E. An intelligent system for diagnosis of the heart valve diseases with wavelet packet neural networks. Comput Biol Med. 2003 Jul;33(4):319-331.

WC Tzeng, YZ Chan, JC Hsieh, "Predicting Hyperkalemia by the Use of a 12-Lead Temporal-Spatial Electrocardiograph: Clinical Evaluations and Model Simulations," Computers in Cardiology, 2005:32:215-218

Wu MF, Chiang JJ, Yang YC, Chao IH, Shieh SM, **Tzeng WC**, and Hsieh JC. Predicting Hyperkalemia by a Two-Staged Artificial Neural Network. Computers in Cardiology. 2003；30：433 – 435

行政院衛生署 93 年死因摘要，2005: http://www.doh.gov.tw/statistic/data/ 死因摘要/93 年/93.htm

吳旻芳。以 12 導程心電圖辨識高血鉀症。中華大學資訊工程系碩士論文，2004。

曾文慶、謝瑞建。高低血鉀症心電圖資料庫，心電圖資料特徵分析及其數學模式之建構與應用。行政院國家科學委員會補助專題研究計畫期末報告，2004 (NSC 92-2213-E-216-012)。

曾文慶、謝瑞建。網頁式心電圖訊號擷取分析系統之建構與應用。行政院國家科學委員會補助專題研究計畫期末報告，2005 (NSC 93-2213-E-216-011)。

楊雅筑。以複合式小波轉換方法偵測 R 波之研究。中華大學資訊工程系碩士論文，2005。

蔣家正。網頁心電圖資料庫處理系統的設計及其在臨床環境的。中華大學資訊工程系碩士論文，2005。

蔣家正、**曾文慶**、謝瑞建、楊雅筑、張靜芳、陳朝慶、林順木，"網頁心電圖資料庫的設計及其在臨床環境的運用," 94 年度急診醫學會學術討論會，2005。

以小波轉換為基礎之 12 導程心電圖特徵點萃取
Wavelet-Based 12 Lead ECG Characteristic Waves Extraction

吳思謙[1]、曾文慶[2]、謝瑞建[3]
[1]中華大學資訊工程所
[2]中華大學生物資訊學系
[3]元智大學資訊管理學系

摘要

為了發展心臟疾病的電腦自動化分析，12 導程心電圖特徵點萃取是最關鍵的一部份。而一般除了 HP、PHILIPS 等國外廠商的 12 導程心電圖儀外，其他市場上的心電圖儀皆不具有自動診斷分析的功能。有鑒於此，希望藉由心電圖自動化萃取特徵點的開發，能為國人發展一套具有自動化病徵分析的心電圖儀。

由於臨床的心電圖資料，圖形的變異度非常的大，因此特徵點的判別變的非常具有挑戰性。本研究為利用小波轉換來進行各種疾病的心電圖特徵點自動化萃取，其中心電圖可分類為正常、急性心肌梗塞(Acute Myocardial Infarction)、高血鉀症(Hyperkalemia)與心跳每分鐘大於 150 下。藉由不同尺度的小波係數，套用自行開發的演算法來判別心電圖的 P 波、QRS 波與 T 波。研究結果為:(1) R 波的 Sensitivity 平均為 99%，Specificity 為 99.9%；(2) Q 點與 J 點 Sensitivity 平均為 97%，Specificity 為 99.9%；(3) T 波的 Sensitivity 平均為 95%，Specificity 為 99.9%；(4) P 波的 Sensitivity 平均為 92.5%，Specificity 為 99.9%。

關鍵字：心電圖、心電圖特徵點、12 導程心電圖。

Abstract

In order to develop cardiac disease diagnosis by auto-computer analysis, the key part is 12-lead ECG characteristic points extraction. But 12-lead ECG instrument is not having diagnosis and analysis by auto-computer besides HP、PHILIPS and so on. Therefore, rely on the development of 12-lead ECG characteristic point extraction, establish a instrument which has cardiac disease diagnosis by auto-computer analysis for Taiwan.

Because ECG clinical data is much variability, the characteristic points extraction is very challenge. Our research is ECG characteristic points extraction based wavelet transform for all kinds of cardiac diseases. The cardiac diseases in our research include Acute Myocardial Infarction、Hyperkalemia、Normal and heart rate more than 150. Depend on different scale wavelet coefficients, and implement the new algorithm to find the P wave、QRS complete and T wave. The result is follow:(1)the average Sensitivity of R wave is 99%, the Specificity is 99.9%; (2) the average Sensitivity of Q and J wave is 97%, Specificity is 99.9%; (3) the average Sensitivity of T wave is 95%, Specificity is 99.9%; (4) the average Sensitivity of P wave is 92.5%, Specificity is 99.9%.

Keywords: ECG, Characteristic ,12-lead ECG
Abbreviation: ECG.

1. 前言

心電圖是用來診斷心臟疾病主要的工具之一。常見的心臟疾病如急性心肌梗塞(Acute Myocardial Infarction)、高血鉀症(Hyperkalemia)與心律不整等。心電圖可以將這些疾病一一的表現出來，其中特徵點的位置更是格外重要。心電圖特徵點主要由 P 波、QRS 複合波、T

波構成，每個波形分別代表心臟各部位的電生理意義。而藉由特徵點的參數，可以達成心臟病徵的分析，判別出各種的心臟疾病。而由於病人所測量的心電圖資料數量龐大，醫生用肉眼判斷時往往耗費許多時間，所以希望能藉由儀器設備自動化判斷出特徵點來爭取寶貴的時間。但是，心電圖特徵點參數的萃取程式卻很少由國人自製開發，大部分都依賴國外的儀器設備。所以特徵點萃取的程式開發是一塊非常重要且必需達成的領域，在電腦自動判別病徵中，心電圖特徵點自動化萃取扮演了極度重要的腳色。

在近幾年中的心電圖波形偵測，以小波轉換[1-8]的技術最為熱門。1995 年，由 Cuiwei Li 等人提出利用不同尺度的小波轉換進行心電圖特徵點的偵測[2]，可以在雜訊與基線漂移的訊號中將 QRS 複合波與 P 波、T 波區分開。而[1,6]也利用相似的方法進行 QRS 波的偵測。雖然皆運用了小波轉換的多尺度觀察進行 QRS 波的偵測，但是在偵測過程中耗費了很多時間在篩選不同尺度上所需要的係數及其位置。1998 年，由 Yanli Zheng 等人提出由偶對稱小波與奇對稱小波分別搭配三階與四階係數來搜尋 R 點[3]。利用了奇對稱小波的極值對中過零點位置，相對偶對稱小波位置附近的波峰點來判斷是否為 R 波。如此也花費了相當二次的時間去找候選 R 點。

2005 年，由 Hsu 等人提出的特徵點提取方法[4]，則是利用了小波轉換的三階與四階係數相乘，並且讓相乘過後的係數取絕對值，如此一來更明顯的指出 R 點的位置。

本研究使用了不同尺度的小波係數之搭配來進行心電圖特徵點的偵測，而在原 ECG 訊號上則不作其他額外的加工處理，以不破壞最原始的訊號來做特徵點位置的判定。在利用小波係數的搭配找出各特徵波形附近的位置之後，再用簡單的演算法來搜尋正確的特徵點位置。

為了能加速醫生診斷的效率，爭取寶貴的診斷時間，心電圖電腦自動化特徵點擷取的程式開發即是此研究的目的。藉由此程式的開發，能夠提供醫療人員足夠的時間診斷所有的病人，並且能對未來的自動化病徵分析能有所貢獻。

2. 研究方法與步驟

2.1 程式開發環境與資料取得

ECG 特徵點擷取之程式開發環境為 MATLAB7.0；而 ECG 的資料是由新竹為恭醫院的院內醫療資訊系統(Hospital Information System, HIS)，取出各種類的臨床病歷資料。每筆資料有 12 導程的心電圖，每個導程的取樣頻率為 250 點/每秒，針對每筆資料的每個導程都會以所開發的程式來萃取其特徵點。

2.2 小波轉換(Wavelet Transform)

小波轉換如方程式(1)所示。

$$C(2^j,b)=\frac{1}{\sqrt{2^j}}\int_{-\infty}^{\infty} f(t)\psi(\frac{t-b}{2^j})dt \qquad (1)$$

其中 C 為小波轉換後的係數；j 為小波轉換的漲縮係數，依代入不同的值之代表不同的尺度；$f(t)$ 為原訊號，也就是 ECG 訊號；b 為小波轉換的平移係數。$\frac{1}{\sqrt{2^j}}\psi(\frac{t-b}{2^j})$ 則表示原小波函數在不同尺度上的位移。藉由小波的漲縮係數，我們可以觀察原訊號在什麼尺度下的小波其對應的係數是什麼。而漲縮係數愈小，原訊號中高頻成分所對應的小波係數能量愈大；漲縮係數愈大，其原訊號中低頻成分的小波係數能量愈大。圖 2-1 列出 ECG 訊號相對不同尺度的小波係數。

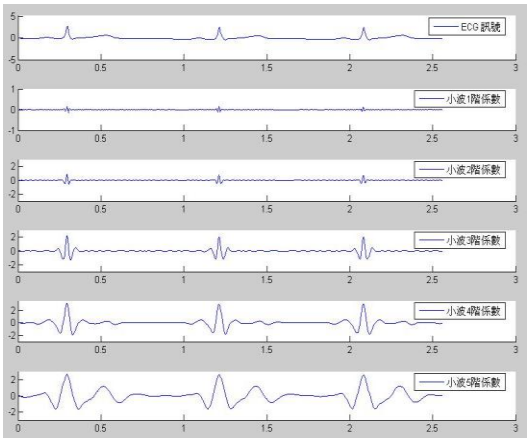

圖 2-1 ECG 訊號與不同尺度小波係數

小波分析的優點在於能夠於時域上指出何時圖形中斷不連續，圖形走向，與對應出相

似小波圖形的位置。利用小波轉換的優點，能夠應用在 ECG 訊號上，找出波形的走向與轉折點，藉此已達成特徵點萃取。本研究中所用的小波母函數為 bior5.5[4]，如圖 2-2 所示。

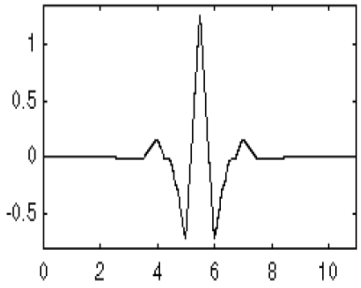

圖 2-2 小波母函數 bior5.5

2.3 開發波形及影像瀏覽器

2.3.1 R 波偵測

在搜尋此 R 波前，先將 0.2 秒內的小波係數(波峰點)收集起來，形成一個待偵測的集合。而又因為 R 波在第三階與第四階小波係數通常能量特別大，尤其是在第四階的小波係數;而第三階與第四階小波係數所相乘出來的值又更加突顯 R 波的位置,所以要找出此集合最大能量係數，也就是找出集合內小波係數最大值。在找到最大能量的係數後,其最大係數相對於原訊號之位置,並在原 ECG 訊號的此位置向左 0.04 秒與向右 0.08 秒區間內搜尋最大正向 peak 與最大負向 peak 的點,其正向、負向波峰做一標誌(布林值),。假設找到最大正向波峰與最大負向波峰，且其最大正向波峰在最大負向波峰的左邊，再加上最大正向波峰的高度與最大負向波峰的高度比例小於 0.035，則判定最大負向波峰為 R 波；假設找到最大正向波峰與最大負向波峰，且其最大正向波峰在最大負向波峰的右邊，再加上最大正向波峰的高度與最大負向波峰的高度比例小於 0.5，則判定最大負向波峰為 R 波；假設只找到最大負向波峰，則判定此波峰為 R 波；假設經過上述三個判斷都沒找到 R 波，但是有找到最大正向波峰，則此波峰為 R 波。若最大正向波峰與最大負向波峰都沒找到，則進入下一組找 R 波。

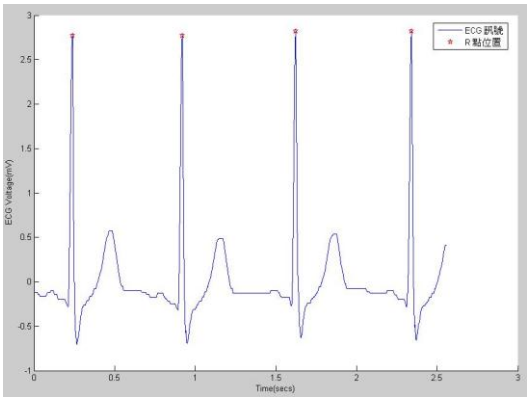

圖 2-3 R 波偵測

2.3.2 Q 波、J 波偵測

在心電圖 Q 與 J 特徵點搜尋的部分，採用小波第三階係數來做偵測的依據。首先由 R 特徵點向右 16ms 開始，找出第三個小波三階轉折位置。由 R 點向右 16ms 開始的原因，是為了避掉由 R 波產生的波峰。接著找出第三個小波三階係數的波峰位置，並對此位置做判斷，如果此位置距離 R 特徵點在 112ms 內，則此位置即為 J 特徵點;若此位置距離 R 特徵點在 112ms 之外，則由 R 特徵點向右 16ms 從新開始找出第一個過零點位置，此即為 J 特徵點。在 Q 特徵點的偵測，是由 R 特徵點向左 16ms 開始，搜尋第一個小波三階係數正向波峰的位置，此即為 Q 特徵點；若沒找到正向波峰位置，則由 R 特徵點向左 16ms 開始，搜尋第一個過零點的位置，此即為 Q 點。

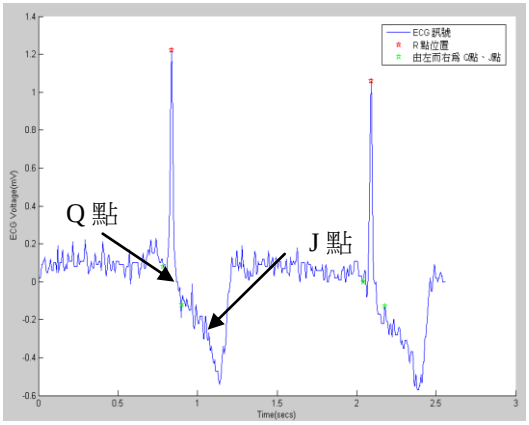

圖 2-4 Q 點與 J 點偵測

2.3.3 T 波偵測

首先找出 R-S 區間所產生的小波四階係數的轉折點，以 R 之後的第一、第二個小波四階係數之轉折點為準。因為 S 波的轉折在某些 ECG 的資料中會特別的大，因此小波四階係數的能量在 S 波的附近也會明顯產生，對 T 波位置的辨認會照成錯誤，所以首先要找出 R 波之後第一、第二個小波四階係數之波峰的

位置。

由所找出的 R 波向右第二個小波四階係數轉折點位置開始，到此心跳的 R 點與下一個心跳的 R 點的中點且向右 40ms 為止，收集小波四、五階相乘係數的轉折點成為一集合。T 波的波形是屬於原訊號中比較低頻的成分，而小波在愈高階的係數對低頻成分的波形靈敏度又愈大，因此以高階的小波係數來偵測 T 波會最為適當，也就是小波第四、第五階係數；而小波第四、第五階係數相乘的結果又更加強化 T 波附近所產生的係數，即 T 波附近的小波能量會更加明顯。然後將集合的係數數值排序(大到小），並且以此最大的係數的位置找出最近的小波四階係數的轉折點，並以此轉折點開始向左找出第一個小波四階係數轉折點；如果沒找到左邊的第一個小波四階係數的轉折點，則回去找出已排序集合的次大係數的位置，然後找出此位置最近的小波四階係數的轉折點，再找出向左邊的第一個小波四階係數的轉折點；重複上述這些動作直到找到所對應的最近的小波四階係數轉折點向左邊的第一個小波四階係數為止。一旦找出所要的四階係數轉折點位置，從此位置開始的四階係數向左直到過第一個零點，此過零點的位置即為 T on 點。而在剛才四階與五階相乘係數所找到最近的小波四階係數轉折點，離此轉折點最近的小波五階係數之轉折點位置即為 T 波峰位置。將 T 波峰 位置向右找出第一個小波五階係數的轉折點位置，此位置即為 T end。

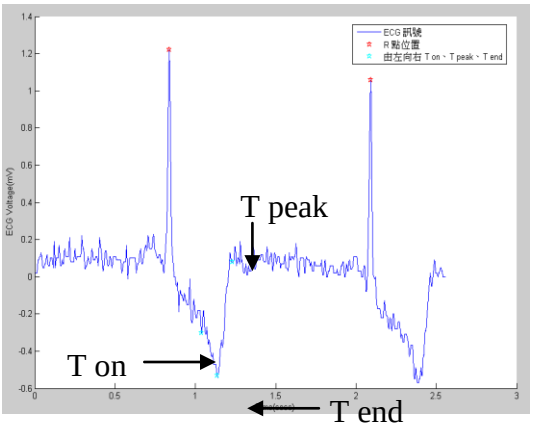

圖 2-5 T 波偵測

2.3.4 P 波偵測

對於 P 波的偵測，採用的係數為小波第四階係數。首先 R 波往左 65ms 開始偵測，收集小波第四階係數的所有轉折點位置形成一個集合。接下來由此集合排序，由最大能量的轉折點位置開始找。此轉折點為 P peak。接著由此轉折點往右，到 Q 點間找第一個轉折點位置，如果沒有找到轉折點，則找第一個過零點位置，到此即找到 P end。如果都沒找到 P end，則回去集合內找次大的位置繼續找 P end。由 P peak 向左找，第一個轉折點即為 P on。

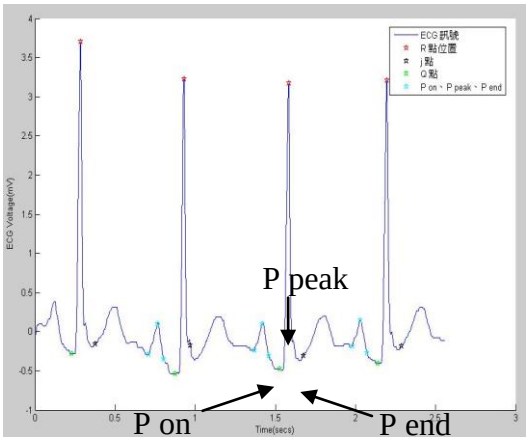

圖 2-6 P 波偵測

3. 結果

3.1 R 波實驗結果

實驗中以四種不同類型的心臟臨床病歷資料來做測試。此四種類型包括正常人(Normal)、急性心肌梗塞(Acute Myocardial Infarction，簡稱 AMI)、高血鉀症(Hyperkalemia)與心跳每分鐘大於 150 下的心電圖。而實驗主要討論 Sensitivity 和 Specificity。Sensitivity 為所有 R 波中且被正確偵測出來的比例；Specificity 為所有不是 R 波的波形中且沒有被誤認為 R 波的比例。如表 3-1 所示。

ECG Classification	Sensitivity	Specificity
AMI	98%	99%
Hyperkalemia	98%	100%
Normal	100%	100%
Heart rate more then 150	98%	99%

表 3-1 R 波偵測實驗結果

3.2 Q 波、J 波實驗結果

實驗中偵測 Q 波與 J 點的誤差範圍在 -0.032ms ～ +0.032ms 之間。主要探討 Sensitivity 和 Specificity。Sensitivity 為所有 Q 波(J 點)中被正確偵測出來的比例。Specificity 為所有不是 Q 波(J 點)波形中沒被誤判為 Q 波(J 點)的比例。實驗結果 Q 波與 J 點如表 3-2 及表 3-3 所示。

ECG Classification	Sensitivity	Specificity
AMI	96%	100%
Hyperkalemia	96%	99%
Normal	98.9%	100%
Heart rate more then 150	96.5%	99%

表 3-2 Q 波實驗偵測結果

ECG Classification	Sensitivity	Specificity
AMI	96.7%	100%
Hyperkalemia	98%	100%
Normal	96%	100%
Heart rate more then 150	98%	100%

表 3-3 J 點實驗偵測結果

3.3 T 波實驗結果

T 波偵測有三個部份，分別為 T on、T peak、T end。其誤差範圍在 -0.024ms～+0.024ms 之間。而 Sensitivity 為所有 T on(T peak、T end)中有被正確偵測出的比例。Specificity 為所有非 T on(T peak、T end)中沒有被誤認為 T on(T peak、T end)的比例。其偵測實驗結果如表 3-4 所示。

ECG Classification		Sensitivity	Specificity
AMI	T on	92.6%	99%
	T peak	91%	99%
	T end	91.8	99%
Hyperkalemia	T on	96%	99%
	T peak	94%	99%
	T end	95%	99%
Normal	T on	98%	99%
	T peak	97.5%	99%
	T end	96.7%	99%
Heart rate more than 150	T on	97%	99%
	T peak	94%	99%
	T end	93%	99%

表 3-4 T 波實驗偵測結果

3.4 P 波實驗結果

P 波偵測與 T 波偵測相同，分為三個部份，分別為 P on、P peak、P end。其誤差範圍在-0.020ms～+0.020ms 之間。而 Sensitivity 為所有 P on(P peak、P end)中有被正確偵測出的比例。Specificity 為所有非 P on(P peak、P end)中沒有被誤認為 P on(P peak、P end)的比例。其實偵測驗結果如表 3-5 所示。

ECG Classification		Sensitivity	Specificity
AMI	P on	90.5%	99%
	P peak	90.8%	99%
	P end	89%	99%
Hyperkalemia	P on	89.7%	99%
	P peak	91.6%	99%
	P end	90%	99%
Normal	P on	93.3%	100%
	P peak	94%	100%
	P end	93%	99%
Heart rate more than 150	P on	96%	99%
	P peak	96%	99%
	P end	96%	99%

表 3-5 P 波實驗偵測結果

3.5 研究結論

本研究所使用的資料是以臨床的病例，所以心

電圖波形的變異度非常的大;在這些變異度大的資料中,利用一套方法規則來找出特徵點,此方法即是利用不同階層小波轉換係數的搭配,進而逼近各個特徵點,是開發心電圖特徵點萃取的最大貢獻。R 波偵測以第三、第四階小波轉換係數搭配;Q 波與 J 點偵測利用第三階小波轉換係數;T 波偵測則利用了第四、第五階小波係數的搭配;P 波偵測則利用了第四階小波係數。研究中利用小波轉換係數來萃取心電圖的特徵點之優點在於小波對轉折點的高敏感性,但是所對應的位置卻不是百分之百的特徵點之轉折點位置,而是在實際正確特徵點位置的附近。所以除了 R 波的實驗外,其他特徵點如 Q、J、T、P 等皆有一個誤差範圍,讓特徵點萃取之實驗結果更具有彈性。

4. 討論

由於使用臨床的病歷,其心電圖波型除了變異度非常大外,各種雜訊也是常常發生。這些突然出現的雜訊、小波峰、轉折點,可能是因為病人緊張震動,或因為呼吸而產生非常嚴重的基線漂移,皆有可能導致偵測錯誤失敗的主因。

本研究中雖然利用小波對轉折點的高敏感性之特性來萃取特徵點,但是如果面對的資料是變異度大的臨床病歷心電圖,則有可能會因為測量心電圖時外在因素導致多了幾處不該有的轉則點。所以改進的方向有二類:(1)讓原訊號平滑化,使去除不必要的轉折點且予更改前之訊號相似度高;(2)其他偵測技術能夠無視多餘轉折點的影響。

一但心電圖特徵點萃取開發完成,進而朝自動化病徵分析邁進。

參考文獻

1. J.S. Sahambi , S.N. Tandonz , R.K.P. Bhatt : Using **Wavelet** Transforms for ECG Characterization. IEEE ENGINEERING IN MEDICINE AND BIOLOGY. February 1997 . Page(s): 77-83

2. Cuiwei Li, Chongxun Zheng, Changfeng Tai : Detection of ECG characteristic points using wavelet transforms. Biomedical Engineering, IEEE Transactions on Volume 42, Issue 1, Jan. 1995 Page(s):21 - 28

3. Yanli Zheng , Guangshu Hu : QRS complex detection by the combination of maxima and zero-crossing points of wavelet transform. Engineering in Medicine and Biology Society, 1998. Proceedings of the 20th Annual International Conference of the IEEE, 29 Oct.-1 Nov. 1998 Page(s):1 5 6 - 158 vol.1

4. 許建強,袁震東,李高平,林靖宇: 基於小波變換的12-導聯心電圖特徵提取方法. Hua-Tung Normal University Journal. 2005(May), 27-32.

5. Tikkanen P.E., Palmgren J.-E., Sellin L.C. : Wavelet analysis of ECG and MAP signals. Engineering in Medicine and Biology society, 1997. Proceedings of the 19th Annual International Conference of the IEEE , Volume 1, 30 Oct.-2 Nov. 1997 Page(s):317 - 320

6. Martinez J.P. , Almeida R. , Olmos S. , Rocha A.P. , Laguna P. : A wavelet-based ECG delineator: evaluation on standard databases. Biomedical Engineering, IEEE Transactions on Volume 51, Issue 4, April 2004 Page(s):570 – 581

7. Gutierrez A. , Hernandez P. , Lara M. , Perez S. : A QRS detection algorithm based on Haar wavelet Computers in Cardiology 1998 , 13-16 Sept. 1998 Page(s):353 - 356

8. Ya-Chu Yang : Study of R Wave Detection Using Hybrid Wavelet Transform. Chung Hua University, Computer Science and Information Engineering , 2004 , 1-40.

9. Kohler B.-U.; Hennig C.; Orglmeister R. : The principles of software QRS detection. Engineering in Medicine and Biology Magazine, IEEE Volume 21, Issue 1, Jan.-Feb. 2002 Page(s):42 – 57

10. The Mathworks, MATLAB wavelet toolbox user's guide. Natick, MA: The Mathworks Inc., 2005.p.1-33

11. Coast D.A. , Stern R.M.; Cano G.G.; Briller S.A. : An approach to cardiac arrhythmia analysis using hidden Markov models. Biomedical Engineering, IEEE Transactions on Volume 37, Issue 9, Sept. 1990 Page(s):826 - 836

Neural network:

```
import java.util.Random;


/**
 * This class stimulates a simple 3 layer neuron network. User is able
 * to specify the number of neurons at each layer and the maximum iteration
 * that the neuron network is able to run.
 *
 * Reference: "Back-Propagation Neural Network Tutorial", Daniel Franklin (2003).
 *
http://ieee.uow.edu.au/~daniel/software/libneural/BPN_tutorial/BPN_English/BPN_English/BPN_English.html
 *
 * @author Robert Tseng
 */
public class NN
{
	//weight for edges from input to hidden layer
	private double[][] wHidden;

	//weight for edges from hidden layer to output
	private double[][] wOutput;

	//current value of the neurons
	private double[] input;
	private double[] hidden;
	private double[] output;

	//maximum iteration per run
	private int maxIter;


	/**
	 * Constructs a new neuron network with random weights
	 *
	 * @param in number of neurons in input layer
	 * @param hid number of neurons in hidden layer
	 * @param out number of neurons in output layer
	 */
	public NN(int in, int hid, int out, int maxIter)
	{
		input = new double[in];
		hidden = new double[hid];
		output = new double[out];
		wHidden = new double[in][hid];
		wOutput = new double[hid][out];
		this.maxIter = maxIter;

		// initialize the weights to some random number
		initialize();
	}

	/**
	 * This is the sigmoid function
	 *
	 * @param in the input value
	 * @return the value which the sigmoid function returns
```

```
     */
    private double sigmoid(double in)
    {
        double ret =
1.0/(1.0+Math.exp(-in));
        return ret;
    }

    /**
     * Trains the neuron network with
an example
     *
     * @param in the input of the
example
     * @param desired the desired
output of the example
     * @param maxError the
maximum square mean error allowed
     * @param learningRate the
learning rate of the neuron network
     * @return the error obtained after
training the network. -1 if there was no
     *    convergence.
     */

    public double train(double[] in,
double[] desired, double maxError,
double learningRate)
    {
        while(true)
        {
            // runs the neuron
network with the input
            int iteration = 0;
            execute(in);
            iteration++;
            double error =
getError(desired,output);
            // check if the error is
with allowable range
            if(error<maxError)
                return
(maxError-error);

            // check if the NN has
been run too many times
            else if(iteration>maxIter)
            {

    System.err.println("WON'T
CONVERGE!!");
                return -1;
            }

            /* stores the values such
that the partial derivative of error with
respect to the
             * weight of edge from
hidden neuron 'i' to output neuron 'j' can
be found
             * by adjOut[j]*hidden[i]
             */
            double adjOut[] = new
double[output.length];


            /* stores the values such
that the partial derivative of error with
respect to the
             * weight of edge from
input neuron 'i' to hidden neuron 'j' can
be found
             * by adjHid[j]*input[i]
             */
            double adjHid[] = new
```

```
double [hidden.length];

                // calculates the adjOut
array
                for(int i =
0;i<output.length;i++)
                {
                    double delta =
-2.0*(desired[i]-output[i]);
                    double sum = 0.0;
                    for(int j =
0;j<hidden.length;j++)
                    {

    sum+=(hidden[j]*wOutput[j][i]);
                    }

    delta*=((1-sigmoid(sum))*sigmoid(
sum));
                    adjOut[i] = delta;
                }

                // calculates the adjHid
array
                for(int i =
0;i<hidden.length;i++)
                {
                    double sum = 0.0;
                    for(int j =
0;j<input.length;j++)
                    {

    sum+=(input[j]*wHidden[j][i]);
                    }
                    double outputSum =
0.0;
                    for(int j =
0;j<output.length;j++)
                    {

    outputSum+=(adjOut[j]*wOutput[i
][j]);
                    }
                    adjHid[i] =
(1-sigmoid(sum))*sigmoid(sum)*output
Sum;
                }

                // adjusts the weights of
edge from the hidden layer to output
layer
                for(int i =
0;i<wOutput.length;i++)
                {
                    for(int j =
0;j<wOutput[i].length;j++)
                    {
                        double adj =
adjOut[j]*hidden[i];

    wOutput[i][j]-=(adj*learningRate);
                    }
                }

                // adjusts the weights of
edge from the input layer to hidden layer
                for(int i =
0;i<wHidden.length;i++)
                {
                    for(int j =
0;j<wHidden[i].length;j++)
                    {
                        double adj =
adjHid[j]*input[i];

    wHidden[i][j]-=(adj*learningRate);
```

```
                    }
                }

            }
        }


    /**
     * Calculate the square mean error of a set of datas
     *
     * @param desired the desired result
     * @param obtained the obtained result
     * @return the square mean error obtained
     */
    private double getError(double[] desired, double[] obtained)
    {
        double ret = 0.0;
        for(int i = 0;i<desired.length;i++)
        {
            double diff = desired[i]-obtained[i];
            ret+=(diff*diff);
        }
        return ret;
    }

    /**
     * Initialize the weights of the neuron network to some random numbers
     */
    private void initialize()
    {
        Random rand = new Random();
        for(int i = 0;i<wHidden.length;i++)
        {
            for(int j = 0;j<wHidden[i].length;j++)
            {
                wHidden[i][j] = rand.nextDouble();
            }
        }

        for(int i = 0;i<wOutput.length;i++)
        {
            for(int j = 0;j<wOutput[i].length;j++)
            {
                wOutput[i][j] = rand.nextDouble();
            }
        }
    }

    /**
     * runs the neuron network with the input datas
     *
     * @param in the input data
     * @return the output obtained
     */
    public double[] execute(double[] in)
    {
        for(int i = 0;i<in.length;i++)
        {
```

```
                input[i] = in[i];
            }

            for(int i =
0;i<hidden.length;i++)
            {
                double sum = 0.0;
                for(int j =
0;j<input.length;j++)
                {

      sum+=(input[j]*wHidden[j][i]);
                }
                hidden[i] =
sigmoid(sum);
            }

            for(int i =
0;i<output.length;i++)
            {
                double sum = 0.0;
                for(int j =
0;j<hidden.length;j++)
                {

      sum+=(hidden[j]*wOutput[j][i]);
                }
                output[i] = sigmoid(sum);
            }

            return output;
      }

      /**
       * Retrieve the number of neurons
in the input layer.
       *
       * @return the number of neurons
in the input layer
       */
      public int getNumInput()
      {
            return input.length;
      }

      /**
       * Retrieve the number of neurons
in the output layer.
       *
       * @return the number of neurons
in the output layer
       */
      public int getNumOutput()
      {
            return output.length;
      }

      /**
       * A method which sets the weight
of the entire neuron network.
       * Should be used for testing
purpose only.
       *
       * @param newHidden the weights
from input layer to hidden layer
       * @param newOutput the weights
from hidden layer to output layer
       */
      protected void setEdge(double[][]
newHidden, double[][] newOutput)
      {
            wHidden = newHidden;
            wOutput = newOutput;
      }
```

```
}
```

K-means:

```
import java.util.*;

/**
 * Implements a simple K-mean algorithm. It stores both the datas
 * and the result found in the most recent analysis. After a data
 * point is added, it should no longer be manipulated.
 *
 * To use this class, create another class to subclass this one and
 * implement the optimal() method.
 *
 * @author Robert Tseng
 */
public abstract class Kmean
{
	// stores the data points
	private ArrayList<Point> datas;


	// stores the result of the last analysis
	private Centroid[] centers;

	/**
	 * Constructor of the class. Initializes the fields.
	 */
	public Kmean()
	{
		datas = new ArrayList<Point>();
		centers = new Centroid[1];
	}

	/**
	 * Check whether the result found in the most recent anaylsis
	 * is the desired optimal solution.
	 *
	 * @return whether the result in the last anaylsis is optimal
	 */
	public abstract boolean optimal();


	/**
	 * add the data point described by the Point data
	 *
	 * @param data the data point to be added
	 */
	public void addData(Point data)
	{
		datas.add(data);
	}

	/**
	 * Retrieve the result of the most recent analysis
	 *
	 * @return an array of Center indicating the cluster centroids
	 * of the last analysis.
	 */
	protected Centroid[] getLastResult()
	{
		return centers;
	}
```

```
    /**
     * Performs the K-mean analysis on the current set of data points.
     * Will randomly pick the starting position for the centroids.
     *
     * Uses the cluster(int k, int[] start) method.
     *
     * @param k number of clusters desired
     * @return An array of String specifying the coordinates of the
     * centroids of each cluster. The order of the array is random
     */
    public String[] cluster(int k)
    {
        int[] index = new int[k];
        Random rand = new Random();
        int i = 0;

        // randomly generate k unique indices such that the data point associated
        // with each index is unique.
        while(i<k)
        {
            int temp = rand.nextInt(datas.size());

            // checking whether datas.get(temp) is unique
            boolean canAdd = true;
            Point next = datas.get(temp);
            for(int j = 0;j<i;j++)
            {
                Point toComp = datas.get(index[j]);

    if(next.equals(toComp))
                {
                    canAdd = false;
                    break;
                }
            }

            // if datas.get(temp) is unique, update information
            if(canAdd)
            {
                index[i] = temp;
                i++;
            }
        }


        return cluster(k,index);
    }


    /**
     * Performs the K-mean analysis on the current set of data points.
     * The starting position of the centroids will be the data point whose
     * index is specified in the start array. It should not be chosen such that
     * the starting locations of two centroids are equal.
     *
```

```
        * @param k the number of clusters desired
        * @param start an array specifying the indices of the data points that
        * will be picked as the starting position of the centroids.
        * @return n array of String specifying the coordinates of the centroids
        * of each cluster. The order of the array is random
        */
      public String[] cluster(int k, int[] start)
      {
            // initializes the centers array to the specified starting positions
            centers = new Centroid[k];
            for(int i = 0;i<start.length;i++)
            {
                  centers[i] = new Centroid(datas.get(start[i]));
            }

            boolean shouldContinue = true;

            do{
                  // check for each point the centroid that it is closest to
                  for(Point p:datas)
                  {
                        // the index of the centroid that the point is currently closest to
                        int ownBy = 0;
                        // the distance to the centroid that the point is currently closest to
                        double dist = centers[0].distance(p);

                        // loop through all centroid to calculate distance
                        for(int i = 1;i<centers.length;i++)
                        {
                              double temp = centers[i].distance(p);

                              // if point is closer to the current centroid, update information
                              if(temp<dist)
                              {
                                    ownBy = i;
                                    dist = temp;
                              }
                        }

                        // add the current point to the center that it is closest to

                  centers[ownBy].addPoint(p);
                  }

                  // check if another iteration of calculating centroid is required
                  shouldContinue = false;
                  for(int i = 0;i<centers.length;i++)
```

```
                {
                    // generate the next
position for this centroid
                    Centroid next =
centers[i].getNext();

                    // if centroid moves
a lot, then another iteration is required

      if(next.distance(centers[i])>0.0000
001)
                        shouldContinue
= true;

                    // updates the centers
array
                    centers[i] = next;
                }
            }
            while(shouldContinue);

            if(!optimal())
                return cluster(k);

            // parsing of the output
            String[] ret = new
String[centers.length];
            for(int i =
0;i<centers.length;i++)
            {
                ret[i] =
centers[i].toString();
            }
            Arrays.sort(ret);

            return ret;
      }
```

```
    /**
     * Remove all data points stored.
     */
    public void clearData()
    {
        datas.clear();
    }
}
```

Hidden Markov model:

```
import java.util.Arrays;
import java.util.Random;
import java.util.TreeMap;

/**
 * A very simple HMM model where the user is able to specify the number of states,
 * the number of observation symbols, and the symbol set. The observation symbols
 * must be discrete. This class uses the Viterbi Algorithm to find the optimal
 * state sequence and the algorithm proposed by Baum and others to optimize the
 * HMM model. No scaling is done in any of the methods.
 *
 * A user is able to run, train, and/or obtain the maximal state sequence of the
 * HMM model for a specific set of input.
 *
 * reference: Lawrence R. Rabiner, "A Tutorial on Hidden Markov Models and
 * Selected Applications in Speech Recognition" Proceedings of the IEEE Vol. 77,
 * pp 257-286, 1989.
 *
 * @author Robert Tseng
 */
public class HMM
{
    // the state transition matrix
    private double[][] stateT;

    // the observation symbol probability distribution
    private double[][] observationT;

    // the initial state distribution
    private double[] initial;

    // keeps track of the symbols used
    private TreeMap<Character, Integer> symbols;

    /**
     * Constructs a new HMM model with the specified number of states and
     * observation symbols. The state transition matrix, observation symbol
     * probability distribution, and the initial state distribution will be
     * randomly generated.
     *
     * @param states the number of states in the model
     * @param observations the number of discrete observation symbols
     * @param symbolSet the set of symbols that will be used for observations.
     * Each symbol must be a character.
     */
    public HMM(int states, int observations, String symbolSet)
    {
        // initializes the matrices
        stateT = new double[states][states];
```

```
        observationT = new
double[states][observations];
        initial = new double[states];
        symbols = new
TreeMap<Character,Integer>();

        // add the symbol set to the
map
        for(int i =
0;i<symbolSet.length();i++)
        {

    symbols.put(symbolSet.charAt(i),i);
        }



        Random rand = new
Random();

        // assigns random values to the
state transition matrix
        for(int i =
0;i<stateT.length;i++)
        {
            double sum = 0.0;
            for(int j =
0;j<stateT[i].length;j++)
            {
                double temp =
Math.min(0.999,rand.nextDouble()+0.00
1);
                stateT[i][j] = temp;
                sum+=temp;
            }
            // normalize the random
values
            for(int j =
0;j<stateT[i].length;j++)
            {
                stateT[i][j]/=sum;
            }
        }

        // assigns random values to the
observation symbols probability
        // distribution matrix
        for(int i =
0;i<observationT.length;i++)
        {
            double sum = 0.0;
            for(int j =
0;j<observationT[i].length;j++)
            {
                double temp =
Math.min(0.999,
rand.nextDouble()+0.001);
                observationT[i][j] =
temp;
                sum+=temp;
            }
            // normalize the random
values
            for(int j =
0;j<observationT[i].length;j++)
            {
                stateT[i][j]/=sum;
            }
        }

        // assigns random values to the
initial state distribution array
        double sum = 0.0;
        for(int i =
0;i<initial.length;i++)
        {
            double temp =
```

```
Math.min(0.999,
rand.nextDouble()+0.001);
                initial[i] = temp;
                sum+=temp;
            }
            // normalize the random values
            for(int i =
0;i<initial.length;i++)
            {
                initial[i]/=sum;
            }


    }

    /**
     * A helper method which parse
the a string of observation to an
     * integer array.
     *
     * @param obs the observation
sequence observed
     * @return the integer array which
represent the observation sequence
     */
    protected int[] parse(String obs)
    {
        int[] temp = new
int[obs.length()];
        for(int i =
0;i<obs.length();i++)
        {
            temp[i] =
symbols.get(obs.charAt(i));
        }
        return temp;
    }

    /**
     * Run the current HMM model
with the observation string passed in
     *
     * @param observation the
observation symbol sequence observed
     * @return the result of evaluating
the HMM model
     */
    public double run(String
observation)
    {
        int[] obs = parse(observation);
        double ret = 0.0;
        int T = obs.length;
        int states = stateT.length;
        double[][] alpha = new
double[T][states];

        // initialization
        for(int i =
0;i<alpha[0].length;i++)
            alpha[0][i] =
initial[i]*observationT[i][obs[0]];

        // dp to get the alpha array
        for(int i = 1;i<T;i++)
        {
            for(int j = 0;j<states;j++)
            {
                alpha[i][j] =
observationT[j][obs[i]];
                double sum = 0.0;
                for(int k =
0;k<states;k++)
                {

    sum+=(alpha[i-1][k]*stateT[k][j]);
```

```
                    }
                    alpha[i][j]*=sum;
                }
            }

            // getting final result
            for(int i =
0;i<stateT.length;i++)
            {
                ret+=alpha[T-1][i];
            }



            return ret;
        }



        /**
         * Finds the hidden state sequence
of the HMM model for
         * the observation string passed in.
         *
         * @param observation the
observation symbol sequence observed
         * @return the optimal state
sequence with maximum likelihood
         */
        public int[]
findHiddenSequence(String observation)
        {
            int[] obs = parse(observation);
            int[] ret = new int[obs.length];
            int T    = obs.length;
            int states = stateT.length;

            double[][] delta = new
double[T][states];
            int[][] psi   = new
int[T][states];

            // initialization
            for(int i = 0;i<states;i++)
            {
                delta[0][i] =
initial[i]*observationT[i][obs[0]];
                psi[0][i] = 0;
            }

            //dp to get the delta and psi
array
            for(int i = 1;i<T;i++)
            {
                for(int j = 0;j<states;j++)
                {
                    double max =
delta[i-1][0]*stateT[0][j];
                    int maxIndex = 0;
                    for(int k =
1;k<states;k++)
                    {
                        double temp =
delta[i-1][k]*stateT[k][j];
                        if(temp>max)
                        {
                            max =
temp;
                            maxIndex
= k;
                        }
                    }
                    delta[i][j] =
max*observationT[j][obs[i]];
                    psi[i][j] = maxIndex;
                }
            }
```

```
        // getting the final result
        double max = delta[T-1][0];
        int maxIndex = 0;

        for(int i = 1;i<states;i++)
        {
            if(delta[T-1][i]>max)
            {
                max = delta[T-1][i];
                maxIndex = i;
            }
        }
        ret[T-1] = maxIndex;

        for(int i=T-2;i>=0;i--)
        {
            ret[i] = psi[i+1][ret[i+1]];
        }

        return ret;
    }


    /**
     * Set the observation symbol probability distribution equal to the
     * parameter. The method makes a deep copy of the passed in matrix,
     * so the parameter can be modified safely afterward.
     *
     * @param newTrans the matrix indicating the observation symbol
     * probability distribution.
     */
    public void setObservationTrans(double[][] newTrans)
    {
        for(int i = 0;i<observationT.length;i++)
        {
            for(int j = 0;j<observationT[i].length;j++)
            {
                observationT[i][j] = newTrans[i][j];
            }
        }
    }

    /**
     * Trains the HMM model with the given observation symbol sequence
     * such that the output is no more than maxError away from desiredOut.
     *
     * @param observation the observation symbol sequence observed
     * @param desiredOut the desired outcome for the observation sequence
     * @param maxError the max error allowed
     * @return the error obtained after training the HMM model
     */
    public double train(String observation, double desiredOut, double maxError)
    {
        int[] obs = parse(observation);
        int T = obs.length;
        int states = stateT.length;

        double[][][] eta = new double[T][states][states];
```

```
        double[][] alpha = new
double[T][states];
        double[][] beta = new
double[T][states];
        double[][] gamma = new
double[T][states];

        while(true)
        {
            // initialization
            for(int i = 0;i<states;i++)
            {
                alpha[0][i] =
initial[i]*observationT[i][obs[0]];
                beta[T-1][i] = 1;
            }

            // dp to get the alpha and
beta array
            for(int i = 1;i<T;i++)
            {
                for(int j =
0;j<states;j++)
                {
                    alpha[i][j] =
observationT[j][obs[i+1]];
                    double sum =
0.0;
                    for(int k =
0;k<states;k++)
                    {

    beta[T-i-1][j]
+=(stateT[j][k]*observationT[k][obs[T-i
]]*beta[T-i][k]);

    sum+=(alpha[i-1][k]*stateT[k][j]);
                    }
    alpha[i][j]*=sum;
                }
            }

            // the probabilities
obtained in the current HMM
            double curr = 0.0;
            for(int i = 0;i<states;i++)
            {
                curr+=alpha[T-1][i];
            }

            // if error small enough,
the break out of loop and return the error
            double temp =
Math.abs(curr-desiredOut);
            if(temp<maxError)
                return temp;

            // Reestimate the HMM

            // generating the eta and
gamma matrices
            for(int i = 0;i<T;i++)
            {
                for(int j =
0;j<states;j++)
                {
                    gamma[i][j] =
alpha[i][j]*beta[i][j]/curr;
                    for(int k =
0;k<states;k++)
                    {
                        eta[i][j][k]
=
alpha[i][j]*stateT[j][k]*observationT[k]
[obs[i+1]]*beta[i+1][k]/curr;
```

```
                    }
                }
            }

            // iAndObsExpect[j] =
expected number of times in state i and
observation j
            double[] iAndObsExpect
= new double[observationT[0].length];

            // updating the initial,
stateT, and observationT matrices
            for(int i = 0;i<states;i++)
            {
                // update the initial
array
                initial[i] =
gamma[0][i];

                // expected number
of transition from state i
                double iToExpect =
0.0;

                // expected number
of times state i is visited
                double iExpect =
gamma[T-1][i];

                // reset and
initializae the iAndObsExpect array

    Arrays.fill(iAndObsExpect,0.0);

    iAndObsExpect[obs[T-1]]+=
gamma[T-1][i];

                for(int j =
0;j<T-1;j++)
                {

    iToExpect+=gamma[j][i];

    iExpect+=gamma[j][i];

    iAndObsExpect[obs[j]]+=gamma[j
][i];
                }

                // update the stateT
matrix
                for(int j =
0;j<states;j++)
                {
                    // expected
number of transitions from state i to
state j
                    double
iTojExpect = 0.0;
                    for(int k =
0;k<T-1;k++)
                    {

    iTojExpect+=eta[k][i][j];
                    }

                    stateT[i][j] =
iTojExpect/iToExpect;
                }

                // update the
observationT array
                for(int j =
0;j<observationT[0].length;j++)
                {
```

```
        observationT[i][j] =
iAndObsExpect[j]/iExpect;
                        }

                    }
                }
            }
}
```