Докторска дисертација

Паралелно дигитално процесирање на ЕКГ сигнали

Ервин Домазет

Универзитет „Св. Кирил и Методиј“
Факултет за информатички науки и компјутерско инженерство
Скопје, Република Северна Македонија

Датум: 17 март 2019 год.
Супервизор: проф. Марјан Гушев

Ервин Домазет

Паралелно дигитално процесирање на ЕКГ сигнали

18 март 2019 год.

Оваа докторска дисертација беше реализирана на Универзитет „Св. Кирил и Методиј“, Факултет за информатички науки и компјутерско инженерство, Скопје, Република Северна Македонија. Мојата мотивација е базирана на продолжувањето на моето истражување во областа: Пресметување со високи перформанси (HPC), како што и направив на моите дипломски и магистерски тези. Мојот фокус, во првиот циклус, беше ставен на искористувањето на графичкото процесирање (GPU) за оптимизирање на алгоритмот Белман-Форд, додека во вториот циклус беше ставен на оптимизирањето на кодот за индустриска печка на повеќе основни средини (CPU).

Предмет на истражувањето на оваа докторска дисертација е целосна оптимизација на алгоритмите за процесирање на ЕКГ-сигнали. Сè започна кога првпат се сретнав со мојот ментор, почитуваниот професор Марјан Гушев, при што се согласивме со визијата да го оптимизираме циклусот на обработка на ЕКГ, со техники на HPC, во замена за зголемен животен квалитет на човештвото. Со мојата полна мотивација и активно учество на мојот ментор во секој чекор, постигнавме оптимизирање на ЕКГ-процесирањето.

Предмет на истражување

Статистичките податоци, доставени од страна на Светската здравствена организација (СЗО/WHO), го откриваат феноменот, според кој, болестите, поврзани со срцевиот удар, се причина за речиси една третина од смртните случаи во светот.

Научно е докажано дека, во одредени случаи, анализата на ЕКГ-сигналите, во реално време, може да спаси еден живот. Сепак, достапните IoT решенија се соочуваат со голем предизвик, т.е. обработката на податоците кои доаѓаат со одредена брзина и во огромни количини.

Достигнувањата на информатичко-комуникациската технологија во решавањето на различните проблеми е фактор за иницирање нови можности. Една ваква иновација е анализата и интерпретацијата на Електрокардиограмот (ЕКГ), во реално време.

Овие системи, генерално, се базирани на најмалку четири актери. Првиот и најважниот е пациентот кој го носи ЕКГ-сензорот, со намера да се собираат ЕКГ-сигнали во реално време. Податоците се скенираат од ЕКГ-сензорот и се пренесуваат на блискиот мобилен уред, преку дефиниран комуникациски медиум. Мобилниот уред, со соодветни апликации, е одговорен за иницијална обработка на ЕКГ-сигналот. Понатаму, апликациите се должни да ги пратат податоците до центарот за преработка на облакот, што го претставува следниот сегмент.

Податоците кои доаѓаат со одредена брзина и во огромни количини се обработуваат во облакот, со цел да се идентификуваат потенцијалните

нерегуларности на срцето. Во случаи на детектирани проблеми, соодветниот сегмент на ЕКГ-сигналот се праќа на медицинскиот кадар, за понатамошна процена и појаснување. Ако тие ја потврдат потенцијалната појава на срцев удар, се повикува Брза помош, на местото каде што се наоѓа пациентот. Со овој процес може да се спречи неочекувана смрт, особено кога навреме е регистриран почетокот на срцевиот удар.

Поради интензивна обработка на податоците, секвенцијалните алгоритми не се доволни да се изврши овој процес во реално време, особено кога серверот во облак процесира илјадници податочни сигнали кои доаѓаат од далечинските носечки сензори за ЕКГ.

Предмет на истражувањето на оваа докторска дисертација ќе биде анализата на алгоритмите за процесирање на ЕКГ-сигналите, дизајнирање методологии за оптимизација и нивна имплементација со користење на начините за паралелизација во областа „Пресметување со високи перформанси (HPC)“.

Општо земено, анализата на Електрокардиограмот се состои од повеќе фази, а најдоминантните и најважните се следниве: елиминирање шум, вејвлет-трансформација и откривање карактеристики.

Нашата цел во ова теза е да презентираме оптимизирани алгоритми за горенаведените фази, со користење на методите за паралелизација.

Образложение на работните хипотези и тези

Главната хипотеза во докторската дисертација гласи на следниов начин: Употребата на различните начини за паралелизација во областа „Пресметување со високи перформанси“ (HPC) може сериозно да ги забрза алгоритмите за дигитално процесирање на ЕКГ-сигналите, споредбено со секвенцијалните.

Дополнително, ги дефинираме и следните хипотези:

- Паралелизацијата на дигиталното процесирање на ЕКГ-сигналите дозволува процесирање во реално време, на облакот на податоци, од илјадници сензори, истовремено;
- Користењето на различните платформи за паралелизација резултира со оптимално решение за облакот;
- Користењето на различните пристапи за оптимизирање резултира во повеќе ефикасни алгоритми;
- Паралелизацијата на дигиталното процесирање на ЕКГ-сигналите резултира во скалабилно решение за облакот, во однос на секвенцијалните решенија.

Цели на истражувањето

Со цел да се постигнат целите, целокупниот екосистем мора да работи во реално време. Паради фактот дека актерите на системот не се цврсто поврзани во комбинација едни со други (loosely coupled to each other), работењето во реално време, не зависи од апликацијата туку од целиот екосистем.

Како резултат на предизвиците, нашата цел е да се оптимизира целокупниот процес на ЕКГ-анализа, со користење начини за паралелизација, во областа „Пресметување со високи перформанси“ (HPC), со цел да се овозможи работење во реално време. Паралелизацијата ќе се спроведе со користење на следните платформи:

1. **Maxeler:** Паралелни пресметки во dataflow;
2. **OpenMP:** Паралелни пресметки во централниот процесор;
3. **CUDA:** Паралелни пресметки во графичкиот уред.

Финалната цел е да се споредат резултатите и да се донесе заклучок, кој оптимизиран алгоритам ќе покаже најдобар резултат. На овој начин ќе се овозможи да се формира оптимално решение за облак, кое ќе може да изработува податоци во реално време за дефинирани пациенти и тоа ќе биде скалабилно.

Методи

Општи методи, кои се користат во оваа теза, се:

- **Анализа** на современите методи за анализа на ЕКГ-сигналите во реално време, кои ќе се презентираат. Ќе се направи анализа на предностите и на недостатоците на алгоритмите.
- **Синтеза** на паралелните алгоритми за обработка на податоците, според современите методологии.
- **Споредба** на оптимизираните алгоритми со достапните секвенцијални алгоритми.
- **Експериментирање** на предложените алгоритми на различни платформи и податоци.
- **Евалуација** на резултатите, со цел да се постигне оптимално решение, во реално време.

Содржина на тезата

Оваа теза се состои од пет дела: 1) Основни концепти, 2) Паралелизација на DSP-филтрирање, 3) QRS-детекција, 4) Подобрена QRS-детекција и класификација и 5) Заклучок.

Првиот дел ги претставува основите на обработката на ЕКГ, предложената архитектура, платформите за паралелизација и најсовремените алгоритми за оптимизација на ЕКГ. Тој се состои од 5 глави. Главата 1 ја елаборира мотивацијата зад оваа теза. Презентирани се статистички податоци и важноста на обработката на ЕКГ-сигналите во реално време. Исто така, дадени се и основните дефиниции за ЕКГ-сигналите. Во Глава 2 се даваат теоретски детали за областа на дигиталното процесирање на сигналите, што е во основата на оваа студија.

Системската архитектура на мобилната апликација, заснована на време, врз основа на ЕКГ-медицинскиот надзор, е дадена во Глава 3. Обезбедена е анализа на барањата, заедно со спецификациите за дизајнот на таквата апликација. Поточно, разработени се сценаријата за работните процеси, деловните барања, функционалниот опис, нефиктивните барања и системските модели. Деталите за платформите за паралелизација, кои се користат во текот на оваа теза, се претставени во Глава 4, имено: OpenMP, CUDA и Maxeler. Конечно, Главата 5 ги прегледува најсовремените приоди за оптимизација, поврзани со процесирањето на ЕКГ-сигналите.

Вториот дел, во 6 поглавја, ги анализира и дава различни видови оптимизации на филтрите што се користат преку процесирање на ЕКГ. Во Глава 6, се користи масивната моќ на графичките процесори (CUDA библиотеката), со цел да се паралелизира операцијата на конверзија на DSP-филтерот. Исто така, дадени се алгоритамските детали и резултати. Понатамошни оптимизации на наивната верзија на CUDA се дадени во Глава 7, со цел да се најде оптимално решение. Глава 8 претставува нов метод со оптимизирање на протокот на податоците со користењето на илјадници проточни јадра (Maxeler-системи).

Филтрирањето на шумот, базиран на дигитална вејлет-трансформација, е претставено во Глава 9. Поточно, секвенцијалната верзија на дигиталната вејлет-трансформација, што се користи за филтрирање и за екстракција на карактеристиките, е паралелизирана. Глава 10, од друга страна, се обидува да го оптимизира достапниот број јадра, во рамките на паралелниот вејлет-алгоритам. Поглавјето 11 дава преглед на добиените резултати, поединечно, за секоја студија.

Дел 3 доаѓа заедно со 4-тата глава. Општо, се фокусира на оптимизирањето на алгоритмите за откривање на QRS. Оптималното филтрирање на DSP за QRS-откривање е дадено во Глава 12. Ефектот на ресемплирање е презентиран во Глава 13.

Влијанието на амплитудното скалирање врз QRS-детекцијата се разгледува во Глава 14. Глава 15 го заклучува делот за детекцијата на QRS,

со презентирање на деталите за добиените резултати, а исто така, поврзано со работата за откривање на QRS.

Четвртиот дел ги презентира подобрените алгоритми за откривање на QRS и нивната класификација. Се состои од три поглавја. Едно од најдобрите достигнувања на оваа теза е дадено во Глава 16. Главата 17 претставува алгоритам за класификација на QRS, базиран на правилата за одлучување, кој бара едноставни операции, кои може да работат на мобилни уреди. Преглед на овој дел е даден во Глава 18, со поврзана работа на оваа област и добиените подобрувања.

Конечно, петтиот дел го дава заклучокот на оваа дисертација. Поточно, Глава 19 ги дава заклучоците и главните резултати на оваа теза. Идната работа е, исто така, презентирана, т.е. што ќе претставува нашиот следен чекор во оваа област.

Главни резултати

Ова докторска дисертација произлегува од повеќе објавени резултати на национални и на меѓународни конференции и во списанија. Фокусот во нив е, првенствено, на перформансите. Сите овие објавени трудови обезбедуваат теоретски како и експериментални резултати. Главните резултати од овие истражувања се дадени подолу.

Dataflow DSP filter for ECG signals

Паралелизиран е ДСП-филтерот кој служи да се издвојат суштинските карактеристики на електрокардиограмските сигнали [41].

Протокот на податоците е оптимизиран со користење на илјадници dataflow-јадра. Со користењето на Maxeler-системите, обележани се значајно високи брзини.

CUDA DSP filter for ECG signals

Со користење на графички уред, ДСП-филтерот е паралелизиран, т.е. со библиотеката CUDA [40]. Значително високите брзини се обележани.

Optimizing high-performance CUDA DSP filter for ECG signals

Претходно паралелизираниот ДСП-филтер[40], е дополнително оптимизиран со повеќе методи[42]. Споредено со претходниот труд, оптимизацијата дава значајни брзини.

Parallelization of digital wavelet transformation of ecg signals

Вејлет-филтерот е детално анализиран и паралелизиран во следниот труд [44]. Резултатите укажуваат дека има забрзување до 20 %, споредбено со секвенцијалната верзија на вејлет-филтерот.

Optimal parallel wavelet ECG signal processing

Претходно паралелизираниот вејлет-филтер е дополнително оптимизиран со повеќе методи[43]. Резултатите укажуваат дека има забрзување.

A cime-critical mobile application based on ECG medical monitoring

Во овој труд [48], нашата цел е да се направи детална анализа за барањата за време-критична мобилна апликација базирана на ЕКГ-следење.

Design specification of an ECG mobile application

Детална спецификација за дизајнот на време-критичната мобилна апликација, базирана на ЕКГ-следење, е направена во овој труд [49].

Optimal DSP bandpass filtering for QRS detection

Во следниот труд [64], нашата цел е да истражуваме како различните вредности на филтрите влијаат врз точноста, чувствителноста и прецизноста на QRS-детекторите. Направената анализа води кон изградба на ефикасен филтер, со мала компјутерска сложеност, наменет да се користи за нослив ЕКГ-сензор, заедно со мобилните уреди.

Optimizing the impact of resampling on QRS detection

Зависноста на перформансот на QRS-детекцијата на фреквенцијата на земањето на примероци и, доколку е можно, да се најде QRS-детектор што ќе биде високоефикасен при различни стапки на земање примероци, е истражувано во [65]. Нашиот пристап резултира со зголемени перформанси за откривање на QRS на оригиналниот алгоритам на Хамилтон.

Amplitude rescaling influence on QRS detection

Влијанието на амплитудното скалирање врз чувствителноста и позитивната прогностичка стапка на Хамилтоновиот алгоритам за откривање QRS истражуван во [46]. Врз основа на оптимизациите, оптимизиран е Хамилтоновиот алгоритам.

Improving the QRS detection for one-channel ECG sensor

Едно од најдобрите достигнувања на оваа теза е објавувано во [47], со главна цел, да се оптимизира Хамилтон алгоритамот за откривање на QRS на првиот канал на рециклирана и репродуцирана МИТ-БИХ ЕКГ база на податоци. Нашето решение работи подобро од алгоритмите кои работат со оригиналните сигнали, семплирани на 360 Hz.

Применливост на резултатите

Трудовите, кои се објавени во рамките на оваа докторска дисертација, добија важни резултати. Алгоритмите, кои се паралелизирани, имаат широк спектар на применливост. Потенцијалните области на нивната применливост се дадени подолу.

Подобрени алгоритми за ДСП-филтрирање

Различните видови на DSP-филтри биле проучени, вклучувајќи нископропусни, високопропусни филтри за пропусници. Дополнително, ги истражувавме алгоритмите за филтрирање, базирани на вејлет-трансформацијата.

Оваа теза обезбеди и експериментално ги потврди низата оптимизирани пристапи за ЕКГ-препроцесирање. Според потребата од литературата, секој од нив може да се искористи ефикасно.

Подобрени алгоритми за откривање на QRS

Ова е еден од главните придонеси на оваа дисертација. Оптимизиравме алгоритам за откривање QRS на Хамилтон, за еден канал, кој работи во 125Hz. Деталите за нашиот метод се веќе објавени во едно списание и може да се применат, особено во средини за кои се потребни ефикасни алгоритми во реално време, како што се преносливите сензори. Трудот е преведен на македонски јазик и тој е поместен во следната глава.

Далечински ЕКГ-мониторинг преку облак

Нашите наоди за време на овие истражувања веќе се интегрирани во порталот за ECG-мониторинг, имено, во порталот ECGAlert, поддржан од Фондот за иновации.

Список на објавените трудови

Во рамките на оваа докторска теза, беа објавени 10 научни трудови и 1 списание со фактор на влијание на меѓународно ниво. Целосната листа на објавени трудови, во рамките на оваа теза, е дадена подолу:

1. „Dataflow DSP filter for ECG signals“ in 13th International Conference on Informatics and Information Technologies, Bitola, Macedonia, 2016.
2. „CUDA DSP filter for ECG signals“ in 6th International Conference on Applied Internet and Information Technologies, Bitola, Macedonia, 2016.
3. „Optimizing high-performance CUDA DSP filter for ECG signals“ in 27th DAAAM International Symposium, Mostar, Bosnia and Herzegovina: DAAAM International Vienna, 2016.
4. „Parallelization of digital wavelet transformation of ecg signals,“ in MIPRO, 2017 Proceedings of the 40th Jubilee International Convention. Opatija, Croatia, IEEE, 2017.
5. „Optimal Parallel Wavelet ECG Signal Processing“ in 14th International Conference on Informatics and Information Technologies, Mavrova, Macedonia, April 2017.
6. „A Time-Critical Mobile Application based on ECG Medical Monitoring“ in 8th Balkan Conference in Informatics, 20-23 September 2017, Skopje.
7. „Design specification of an ECG mobile application“ in Telecommunication Forum (TELFOR), 2017 25th 2017 Nov 21 (pp. 1-4). IEEE.
8. „Optimal DSP bandpass filtering for QRS detection“ in 2018 41st Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE, 2018.
9. „Optimizing the Impact of Resampling on QRS Detection“ in 10th ICT Innovations 2018. Springer, Ohrid, Macedonia.
10. „Amplitude Rescaling Influence on QRS Detection“ in 10th ICT Innovations 2018. Springer, Ohrid, Macedonia.
11. „Improving the QRS Detection for One-channel ECG Sensor“ in journal of Technology and Healthcare 2019, in press, IOS Press.

Подобрување на QRS-детекторот за едноканален ЕКГ-сензор

Вовед

Напредокот на Интернет (IoT) ги охрабри истражувачите да го интензивираат својот фокус на Електрокардиограмската (ЕКГ) обработка, посебно за преносливите уреди. Ова поле се претвори во популарна тема за истражување во биомедицинскиот инженеринг[27]. Нашиот примарен фокус е насочен кон градење квалитетен индустриски QRS-детектор за преносliв ЕКГ-сензор, кој би бил значително подобар од постојните цврсти алгоритми за QRS-детекција, кои не се погодни за мобилни уреди со ограничени ресурси.

Перформансите на QRS-детекторот се евалвираат со пресметување на тоа, колку вистински QRS-пикови ќе бидат пронајдени (QRS-сензибилитет, Q_{SE}) и колку од тие откриени QRS-пикови се реални отчукувања (QRS позитивна прогнозирачка стапка, Q_{+P}). Стандардната база на податоци за тестирање ја претставуваше Институтот за технологија во Масачусетс - Датабаза за аритмии во болницата во Бет Израел (MITDB)[99], со 48 евидентирани 30-минутни ЕКГ-мерења. Оригиналните сигнали се примероци со конверзија од 360 Hz и 11 бита. Целта на нашето истражување е QRS-детектор за сигнали со користење на фреквентна стапка од 125 Hz и 10-битна AD-конверзија.

Еден од најцитираните трудови за QRS-детекција за мали уреди со ограничени ресурси е алгоритмот на Пан и Томкинс[107]. Неговата цврстина лежи во фактот дека тој е доволно брз, за да биде искористен во реално време и може да се справи со бучни сигнали. Сепак, перформансот на овој алгоритам зависи од битната резолуција во АД-конверзија. Во нашиот случај, кога се користеа едноканален ЕКГ-сензор и помали стапки на фреквенции, перформансот не беше задоволителен, особено за сигнали со помали амплитуди. Поради овие фактори, тоа не беше добро решение за нас.

Друга алтернатива е Хамилтоновиот алгоритам[69]. Споредбено со алгоритмот на Пан и Томпкинс, тој е сосема сличен, но користи поинакви филтри и правила. Ова е стабилно решение, но сепак е недоволно способно за справување со мали амплитуди, или варијации, во консекутивни амплитудни нивоа, особено кога се користи помала битна резолуција во АД-конверзија.

Physionet.org[63] е сеопфатен ресурс, каде што може да се најдат и повеќе алгоритми за детекција на QRS, вклучувајќи *Wavedet*, *gqrs*, *wqrs*, и *sqrs*. Тие претставуваат едноставни и брзи алгоритми што бараат мал број ресурси и поседуваат висока сензибилност и позитивни предиктивни вредности. Сепак, тие ја немаат класификацијата како и добиената сензибилност, и позитивната предитктивна рата, кои се сметаат за пониски

од побарувачката на квалитетен индустриски QRS-детектор за едноканален преносен ЕКГ-сензор.

По овие иницијални напори, вниманието на истражувачите постепено се насочи кон развивање пософистицирани алгоритми за детекција на QRS, вклучувајќи ги машинското учење и другите методи, како што е опишано во Дел 2 (поврзана работа). Иако некои од новите пристапи постигнаа подобри перформанси, тие обично бараат пресметливо интензивни алгоритми, несоодветни за паметните телефони кои собираат континуирани ЕКГ податоци од преносливи сензори.

Во нашето истражување го подобруваме Хамилтоновиот алгоритам [69], за да можеме да го направиме поефикасен за индустриска примена. Подобрувањето беше прилично долг процес поради експоненцијалната природа на напорот за подобрување на алгоритам. Колку сте поблиску до маргината од 100 %, толку повеќе се зголемува напорот за многу мало подобрување на перформансот. Воведовме неколку стотини правила за справување со идентифицираните проблеми во детекцијата на QRS и неколку илјадници тестови, за да ги приспособат параметрите и праговните вредности за идентифицираните решенија. Некои праговни вредности добија добри перформанси на некои тестови, но лоши на други. Кога ги нивелиравме некои од параметрите, се случи некои од правилата да не работат други сетови на податоци за тестирање, и тоа ни стана уште поголем предизвик.

Конечно, резултатите покажаа дека ги реализиравме нашите цели. Ние постигнавме повисоки вредности од сите други објавени резултати, иако ние работевме со речиси трипати помала фреквенција на семплирање и со половина од АД-конверзијата на битната резолуција.

Структурата на овој труд е организирана на следниов начин: во Дел 2 се дискутира за поврзаната работа во областа на QRS-детекциските алгоритми а во Дел 3 се претставува заднинската страна. Анализата на проблеми е презентирана во Дел 4, додека пак, нашиот пристап кон подобрување на алгоритмите, во Делот 5. Резултатите од експериментите се евалвирани во Дел 6 и споредени се со други решенија. Дел 7 е посветен на заклучоците и на идната работа.

Преглед на литературата

Алгоритмите за детекција на QRS, генерално, ја следат истата рутина [106], почнувајќи со филтер за обработка на дигиталните сигнали (ДСП/DSP), којшто ги елиминира шумот и *baseline wander*. Тогаш излезот се совпаѓа со сет на прагови. Алгоритмите, главно, се разликуваат на начинот на кој тие ги калкулираат праговните вредности и ги применуваат правилата, што е само уште еден слој применет по прагот.

Долгите ЕКГ-снимки, генерално, се пренатрупани со шум во форма на суптилна девијација на срцевиот ритам. Покрај тоа, квалитетот на сигналот се менува при алтернативни промени во амплитудата на бранот. Освен ако шумот не се елиминира, детектирањето на таквите QRS-комплекси станува потешко и ја намалува точноста[82]. Важно е да се наведе дека кој било алгоритам, дизајниран за квалитетна индустриска QRS-детекција мора да биде прилагодлив на кој било тип на шум.

Постојат неколку истражувачки трудови[82, 20], кои даваат сеопфатен преглед на популарните методи за детекција на QRS. Во краткиот преглед, објавените алгоритми за откривање QRS, се базираат на следниве техники:

- *Диференцијација (деривација)*, каде што се пресметува разликата помеѓу сегашните и претходните мостри, како начин за идентификување на наклонот, а потоа тоа се споредува со дадената праговна вредност вклучувајќи ги и алгоритамот Пан и Томпкинс[107], Хамилтоновиот алгоритам[69], или други релевантни пристапи [12, 61, 100];
- *Чисти ДСП-алгоритми*, каде што се комбинираат основните ДСП-филтри со различни карактеристики со цел да се произведе појасен филтер и да се елиминира шумот како и да се исфилтрира сигналот, така што прагот ќе ги определи битовите [10, 28, 55, 58, 104];
- *Алгоритми за препознавање на шема*, каде што податоците на сигналот се исти со претходно дефинираните шеми и се открива бранова форма во случај на сличност во дадените ограничувања на амплитудата и наклоните [67, 32, 89, 126, 132];
- *Невро мрежа*, повеќеслојни перцептори (MLP), радијална базична функција (RBF) и Learning Vector Quantization (ЛВО), се користат за адаптивно предвидување на локацијата на следниот пик [138, 23, 39, 74, 89];
- *Дигитална вејлет-трансформаација* ,каде што сигналот е распределен на одредени мерни нивоа, а потоа повторно се рекомпонира, што ефикасно го намалува шумот. Потоа се применува прагот, за избирање соодветни пикови [87], [21], [121], [93], [98];
- *Генетските алгоритми* се користат за оптимизирање на полиномскиот филтер за препроцесирање. ЕКГ-сигналот се споредува со адаптивниот праг и параметрите се оптимизирани со пристап за генетска оптимизација [113];
- *Скриениот марков модел (HMM)*, кој се користи за приспособување на функцијата на веројатност, која варира според скриениот марковски синџир, при што потоа моделот ја предвидува моменталната состојба, која може да биде QRS-комплекс. P и T-брановите, исто така, може да се пресметаат [33, 16, 35];
- *Хилберт трансформација*, каде што Хилберт-трансформацијата на ЕКГ сигналот се пресметува со Fast Fourier-трансформација (ФФТ), а тоа се користи за калкулирање на сигналот [124, 102, 25] и

- *Фазор трансформација*, каде што секој ЕКГ-примерок е претворен во фактор за правилно управување со P и T-брановите, кои, по дефиниција, се карактеризираат со пониски амплитуди од R-пикот со ниска цена на пресметувањеl, а потоа нив ги споредува со прагот [92].

QRS-детекторот со високи перформанси директно влијае на износот и на квалитетот на вредните информации за ЕКГ. QRS-детектирањето е иницијален чекор за понатамошна ЕКГ-анализа.

Основи

Во овој дел , ќе ги објасниме евалвациските метрики и ќе овозможиме преглед на оригиналниот Хамилтонов QRS-алгоритам за детекција.

Мерење на перформансите

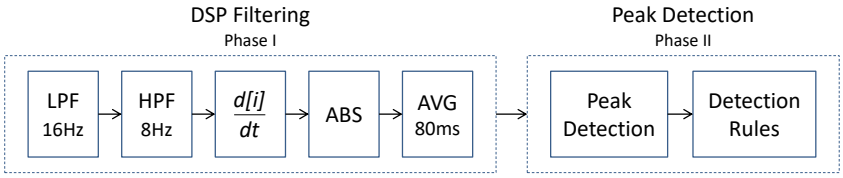
Примерите што се користат во нашата методологија за тестирање се исти за IEC 60601-2-47-стандардот за посебни барања за безбедност, вклучувајќи ги суштинските перформанси на амбулантните електрокардиографски системи и ANSI / AAMI EC57: 2012 за тестирање и известување за резултатите од перформансот на срцевиот ритам и алгоритмите за мерење на CE-сегментот. Овие стандарди ја користат базата на податоци за аритмијата на МИТ-БИХ ЕКГ [99] и Американската асоцијација за срце (AHA)[72].

MITDB содржи получасовни ЕКГ-снимки за 48 анонимни лица, а само 44 записи ги исклучуваат оние кои содржат темпобитови. Овие снимки се јавно достапни на веб-страницата physionet.org [63]. Фреквенцијата на снимањето е 360 примероци во секунда, по канал, со 11-битна резолуција. Иако секоја снимка содржи два канали, во најголемиот дел од записите, го користевме првиот канал, идентификуван како ML II.

Покрај тоа, ние ги следиме барањата според Стандардот IEC 60601-2-47: 2012 за медицинска електрична опрема, особено барањата за суштинските перформанси на амбулантските електрокардиографски системи. Според овие барања, секој пресметан пик се смета за откриен, ако е најмногу 150 ms подалеку од вистинскиот ритам.

Детектираниот QRS се означува како вистински позитивен (TP), ако детекторот QRS најде QRS поблиску од 150 ms од оној што е означен. Лажен негатив (FN) е пропуштен QRS, или ако детекторот QRS пронајде QRS надвор од периметарот од 150 ms, додека лажниот позитив (FP) е погрешно детектиран QRS (дополнително пронајден).

Често користените мерки за изведба се сензибилитетот и позитивната предвидувачка стапка, пресметана со формулата 0.1.



Слика 0.1: Хамилтонов пристап на детекција на QRS.

$$SE = \frac{TP}{TP + FN} \qquad + P = \frac{TP}{TP + FP} \tag{0.1}$$

Покрај тоа, за да се ослободи оптимална врсдност на параметарот, ние овозможивме многу експериментални тестови, и го пресметавме бројот на вкупни грешки, со формула 0.2, каде што збирот на грешките е означен со FP и FN. Кога грешките се помали, се постигнуваат подобри перформанси.

$$Errors = FP + FN \tag{0.2}$$

Анализа на Хамилтоновиот алгоритам

Софтверот со отворен код на EP Limited за детекција на аритмии служи како основа за ова истражување[70, 69]. Тој има целосна имплементација на C-кодот од алгоритмот на Хамилтон, со три различни детектори и со едноставна класификација на отчукувања. Два од детекторите се за општа намена, додека третиот е за средини со мала количина на меморија.

Алгоритмичните детали теоретски се обезбедени во нивната оригинална работа [69]. Слика 0.1 го претставува концептуалното ниво на двете фази и на чекорите, кои се поставени на високо ниво и се спроведени за секоја од тие фази.

Алгоритмот започнува со нископропусен филтер (LPF) со, *cutoff* фреквенција од 16Hz. Потоа, сигналот се пренесува низ високопропусен филтер (HPF) од 8Hz. Двата филтри имаат ефект на канален филтер (BPF). Потоа, наклонот на сигналот се пресметува со метода на диференцијација ($\frac{d[i]}{dt}$), проследено со пресметка на апсолутната вредност (ABS). Последниот чекор (AVG) се состои од пресметување на просечното време од 80 ms.

По елиминирање на шумот во фазата на ДСП-филтрирањето, алгоритамот продолжува со фазата на детекција на пикот. Веќе има два прагови за AVG-сигналот, класифицирани како:

- *Статичен праг* со фиксирана вредност и

- *Динамичен прилагодлив праг* (DAT) кој е под влијание на амплитуди од најновите пикови.

Оригиналниот алгоритам го поставува статичкиот праг ($STHR$) во вредност $MIN\_PEAK\_AMP = 7$. Општо правило е дека за пониската вредност на статичкиот праг, ќе има повеќе пикови, но, исто така, ќе се откријат многу артефакти. Од друга страна, повисоката статичка праговна вредност дава помалку пикови, но и помал број артефакти.

Кога ќе се најде нов локален пик, динамичниот адаптивен праг се пресметува со земање на средните вредности за вистинските пикови, но и шумот влегува во предвид. Средната вредност се пресметува со формулата 0.3.

$$mean = \frac{\sum_{n=1}^{8} X_n}{8} \tag{0.3}$$

Нека се означи средната вредност за реалните отчукувања и пикови настанати од шум, qmean и nmean, соодветно, а исто така, TH нека е постојан мултипликатор (со стандардна вредност од 0.3125), а потоа DAT се пресметува од формулата 0.4.

$$DAT = nmean + (qmean - nmean) * TH \tag{0.4}$$

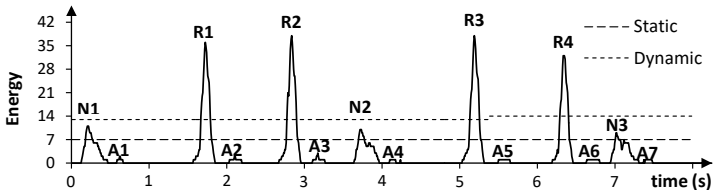
Кога ќе се детектира нов локален максимум, со калкулирање на AVG-вредност, и двата прага се споредуваат со оваа вредност. Ако вредноста е повисока од статичкиот праг, тогаш се смета за потенцијален пик, инаку тоа би било артефакт. Тоа е класифицирано како пик настанат од шум, ако пресметаната вредност е пониска од динамичкиот пик и, како вистинско отчукување, ако е повисок од динамичниот адаптивен праг.

Слика 0.2 ги прикажува и динамичките и статичките прагови. Забележете дека откриените пикови A1, A2, ..., A7 се категоризирани како артефакти (помали од статичкиот праг), а R1, R2, R3 и R4 како реални отчукувања. N1, N2 и N3 се сметаат за пикови настанати од шум, бидејќи локалниот максимум од секоја етикета се помали од динамично пресметаниот праг.

Идентификација на проблемите

Несовпаѓањето во амплитудите на пиковите може да воведе лоша детекција. Идентификувавме два случаи кога тоа се случи:

- низа од ниски амплитудни пикови по изолирана висока амплитуда на пик;
- изолиран низок амплитуден пик по низа од висока амплитудни пикови.



Слика 0.2: Детектирање на артефакти, пикови настанати од шум и вистински пикови, врз основа на вредностите на статичките и динамичките прагови во оригиналниот алгоритам на Хамилтон, презентиран на сигналниот извадок од MITDB-запис 124 (1046 сек.).

Покрај тоа, освен прецизното поставување на прагот и очекувањето на пониските перформанси на исечените сигнали, ги анализиравме сегментите каде што алгоритмот покажа помала сензибилност и спецификација, иако сигналот не бил попречен од шумот. Заклучено е дека пониските перформанси биле добиени за конкретни случаи на сегменти и проблеми, кои може да се класифицираат како:

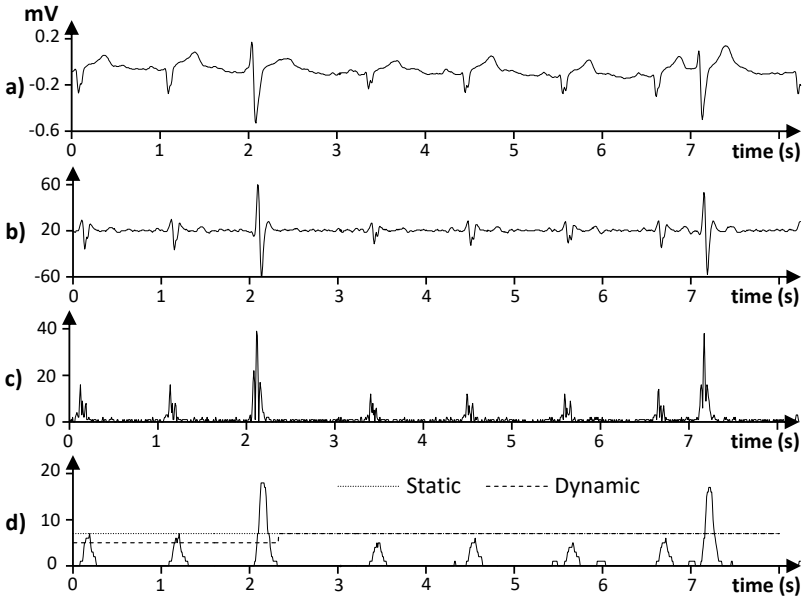
- мешавина од ниски и високи амплитудни пикови;
- елиминација на артефакти и
- погрешна R-пик-локација.

Лоша детекција на пикови со ниски амплитуди

Слика 0.3 го претставува случајот кога претходи изолиран пик со висока амплитуда, а следи низа од ниски амплитудни пикови. Се прикажува сегмент од 8 секунди од MITDB запис 114, вклучувајќи ги оригиналниот сигнал и излезите по извршувањето на секој од чекорите за обработка BPF, ABS и AVG.

Слика 0.3 г), ги идентификува статичките и динамичките прагови и го покажува случајот каде што отчукувањата меѓу двете високи амплитуди се прикажуваат како артефакти, иако тие треба да бидат вистински QRS-отчукувања.

Причината за лошата детекција на ниските амплитудни пикови по високи амплитудни пикови првенствено се должи на високиот степен на статичкиот праг. Дури и ако некој прави исправка со намалување на вредноста на статичкиот праг, за да ги вклучи овие пикови, сè уште ќе има проблем, без оглед на фактот дека пикот ќе се третира како кандидат и ќе се примени проверка на динамички праг. Тоа е така, затоа што високиот амплитуден пик ќе ја зголеми динамичката вредност на прагот предизвикан од пресметката на средната вредност и така пиковите ќе бидат класифицирани како пикови настанати од шум.

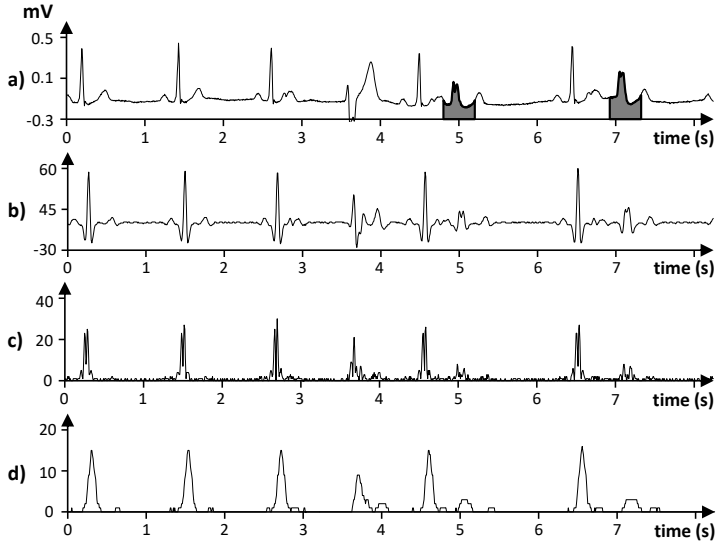


Слика 0.3: Извадоци од сигналот за извршување на Хамилтоновиот алгоритам над MITDB-запис 114 (240 сек) а) Оригинален ЕКГ-сигнал; б) Излез по канален филтер; в) Излез по диференција и пресметка на апсолутни вредности; г) Излез по просекот за интервал од 80 ms.

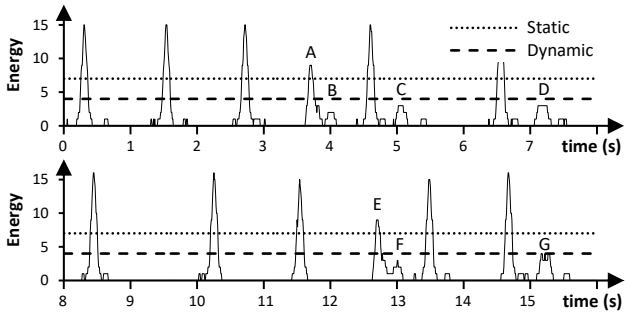
Изолирани пикови по секвенции од пикови со високи амплитуди

Високите амплитудни пикови директно влијаат на пресметката на динамично адаптивниот праг. Се пресметува повторно, секогаш кога има нов локален максимум, со амплитуда повисока од статичкиот праг. Во овој случај, потенцијалниот пик и вредностите пониски од динамичниот праг, се сметаат за пикови настанати од шум, додека другите се сметаат како вистински QRS-пикови. Оригиналниот алгоритам ги складира последните 8 амплитуди на пикот и ја пресметува DAT-вредноста, со формулата 0.4.

Широката анализа на MITDB архива 201 покажува премногу промашувања, особено во случаи на аберирани преткоморни предвремени удари (класифицирани како отчукувања) како што е илустрирано на слика 0.4. Две од отчукувањата, истакнати како стандардни, не може да бидат опфатени, поради динамичниот адаптивен праг и средната пресметка, бидејќи повеќе од најновите отчукувања имаат висока амплитуда. Во овој случај, ниту статичниот ниту динамичниот адаптивен праг нема да работи. Примерот е обележан на сликата 0.5, со пиковите C и



Слика 0.4: Извадоци од сигналот за извршување на Хамилтоновиот алгоритам над MITDB-запис 201 (424 сек.) а) Оригинален ЕКГ-сигнал; б) Излез по канален филтер; в) Излез по диференција и пресметка на апсолутни вредности; г) Излез по просекот за интервал од 80 ms



Слика 0.5: Статички и динамички праговни вредности, добиени на излез, по просекот за интервал од 80 ms на MITDB-запис 201 (424 сек.)

D, кои треба да се класифицираат како QRS-пикови, но тие се откриени како артефакти, бидејќи нивната вредност е помал од статичкиот праг.

Класификација на артефакти

Динамичкиот адаптабилен праг идентификува пикови настанати од шум и реални пикови. Иако, во повеќе случаи, динамичкиот приспособ-

лив праг реагира соодветно, сè уште постојат случаи каде што пикот настанат од шум е погрешно пресметан и прикажан е како вистински. Точна класификација на артефактите е од примарна важност за квалитетен индустриски QRS детектор.

Зголемувањето на статичкиот праг, директно го намалува бројот на артефактите. Сепак, ова драстично ги зголемува пропуштените отчукувања. Ова, исто така, важи и за динамички приспособливот праг. За да се најде оптимумот на статичките и на динамичките пикови, треба да се бара компромис што ќе достигне високи вредности и на сонзибилитот, и позитивна предвидувачка стапка.

Еден пример е илустриран на слика.0.5, каде што пиковите означени како A, C, D, E и G, се ниски амплитудни пикови. Од друга страна, пиковите означени како B, F се артефакти. Со стандардниот праг, пиковите A и E се сметаат за кандидати за QRS, додека останатите се сметаат за артефакти. За овој конкретен случај, со намалување на статичниот праг до 3, ќе се опфатат сите вистински пикови, но артефакниот пик F ќе се смета како пик. Од друга страна, задржувањето на статичниот праг на 4, само ќе го детектира G како пик, а другите пикови повторно ќе останат артефакти.

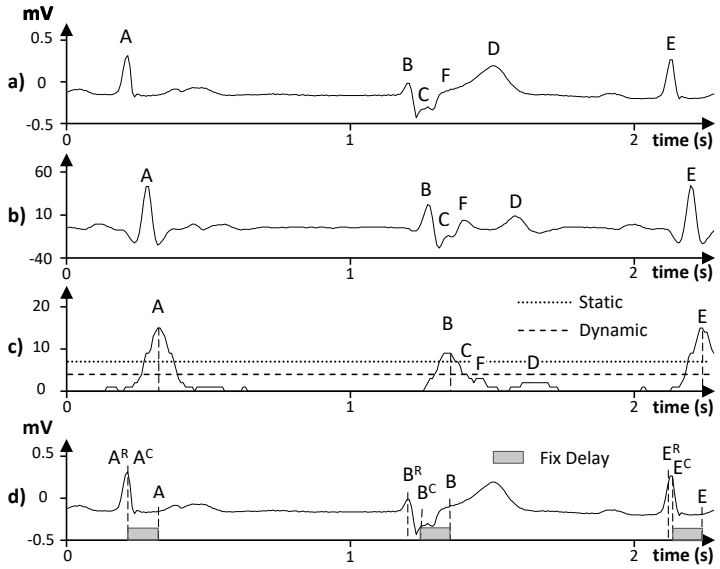
Пресметка на локацијата на R-пикот

Еден од проблемите во извршувањето на оригиналниот алгоритам на Хамилтон е правилното откривање на QRS-пикот. Сликата 0.6 го илустрира таков случај, каде што локалните максимуми се обележани со A, B, C, D и E. Уште еден пик се појавува на излезот од каналниот филтер, со ознака F, како што е прикажано на сликата 0.6 б). Блискоста на означените пикови B, C и F, предизвика два локални пикови, на излез по просекот за интервал од 80 ms, означени како B и C, на сликата 0.6 с). Кога статичкиот праг се применува на просечното време во текот на интервал од 80 ms, пиковите C, F и D се детектираат како артефакти, додека A, B и E, се идентификуваат како потенцијални пикови. Бидејќи динамичкиот праг е понизок од статичниот, овие отчукувања се класифицираат како реални.

Забележете го постојаното доцнење кај филтерот, кое се одзема од локацијата на највисоките вредности (како што е прикажано на просечниот сигнален излез). Иако правилно го одредува претходниот QRS-пик A, и следниот QRS-пик E, сепак прави грешка во одредувањето на пикот B.

Оригиналниот алгоритам на Хамилтон ја детектира локацијата на R-пиковите, тие да бидат во точките A^C, B^C, и E^C, иако имаат различни реални пиковни локации A^R, B^R, и E^R.

Алгоритмот на Хамилтон открива пик базиран на претходно пресметано задоцнување[70]. Иако филтрите создаваат фиксно задоцнување, авторот истакнува дека одложувањето на детекцијата може лесно да ва-



Слика 0.6: Извадоци од сигналот и детекција на R-пик во текот на MITDB-запис 201 (426.4 сек): а) Оригинален ЕКГ-сигнал и локални пикови A, B, C, D и E; б) Излез по канален филтер; в) Излез по диференција и пресметка на апсолутни вредности; г) Излез по просекот за интервал од 80 ms.

рира од 395 ms до 1 сек, во зависност од срцевиот ритам и правилото за детекцијата. Важно е да се напомене дека методот на назадно побарување може да предизвика фиксирано одлагање, ако алгоритмот за назадно пребарување не забележи никакви локални пикови. Оттука, можеме да заклучиме дека оригиналниот алгоритам на Хамилтон ја обезбедува најдобрата можна позиција на R-пикот, иако тоа не значи дека таа е секогаш точна. Овој проблем може особено да влијае на зголемувањето на вкупниот број FP.

Овој конкретен случај е забележан во речиси сите MITDB-записи. Иако разликата не е толку голема, постојат случаи каде што разликата меѓу реалната и откриената локација е поголема.

Подобрување на алгоритмот

Зголемувањето на перформансот на алгоритмот е директно поврзано со поставувањето на праговите и со алгоритамското подобрување.

Подобрување на детекцијата на ниско амплитудните пикови

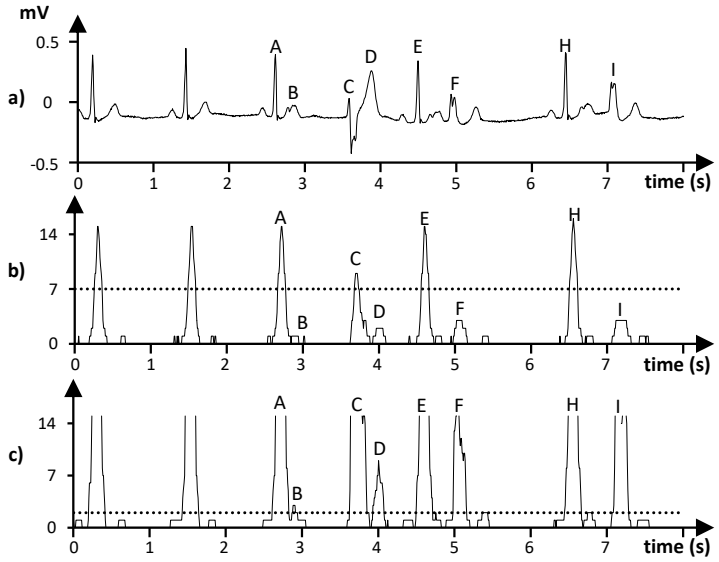
Деталната анализа покажува дека чекорот (ABS), извршен пред просечното време, нема да може да се справи со лошите резултати во откривањето на нискоамплитудните пикови. Ова е особено важно во случаите кога сигналот преставува мешавина од едно високо амплитудно отчукување, а потоа е проследен со неколку отчукувања со ниски амплитуди. Ние ја искористивме идејата претставена во алгоритам на Пан Томпкинс [107] за квадрирање на сигналот, наместо пресметување на апсолутната вредност.

Слика 0.7 претставува случај во кој комбинацијата од квадрираниот и оптимизираниот статички праг ќе ја подобри детекцијата на ниските енергетски пикови. Пиковите означени како A, C, E, F, H и I се вистински R-пикови, додека B и D се артефакти. Оригиналниот алгоритам, кој користи пресметка на апсолутната вредност и просекот заедно со статичкиот праг, не е во состојба да ги класифицира F и I како вистински пикови. Сепак, квадрираниот и просечниот сигнал, во комбинација со новиот (помал) статички праг, може да детектира дека постои доволно енергија за потенцијален пик. Просекот на квадрираниот сигнал, исто така, ги означува B и D како потенцијални пикови. Бројот на таквите пикови може да се намали со динамичкиот праг, или со воведувањето на правилата за детекција на артефакти. Сепак, постојат несакани ефекти, особено во пресметувањето на динамичкиот адаптивен праг.

Овој праг се зголемува со зголемувањето на амплитудата и со тоа се отежнува приспособувањето кон ненадејните промени во амплитудите.

Оваа операција се однесува како важен засилувач, особено ако е тоа проследено со пресметка со просек над интервал од 80 ms. Ова покажува дека овој начин за справување со идентификуваните проблеми е подобар. Сепак, ова не е доволно, бидејќи овој алгоритам не може да се извршува со статички праговни вредности и потребна му е динамичка пресметка со помош на други правила.

Ние спроведовме неколку тестови за експериментирање со праговните вредности STHR од 2 до 50, за да се пронајде оптимална праговна вредност, при што бројот на грешките е минимален. Левиот дел од сликата 12 ја претставува лажната детекција за спроведениот експеримент. Перформансот на алгоритмот постепено се намалува како што прагот се зголемува, најмногу поради високите вредности на FP. Најдобри перформанси се добиваат за $\mathrm{STHR} = 2$. Забележавме намален број на FN, од каде што се доби идејата за тоа, како да се добијат подобри перформанси ако го намалиме бројот на FP со други методи.



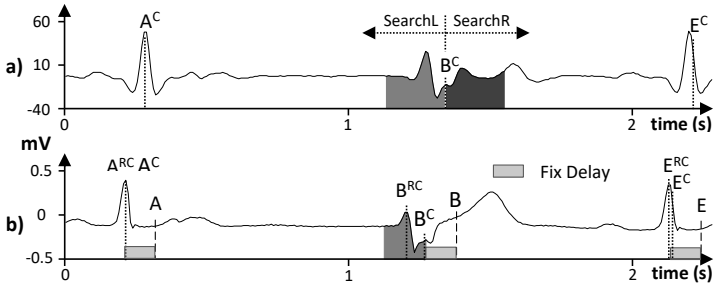
Слика 0.7: Извадоци од сигналот за извршување на алгоритам над MITDB-запис 201 (424 сек.): а) Оригинален ЕКГ-сигнал; б) Излез по просекот за интервал од 80 ms со користење на оригиналниот алгоритам на Хамилтон; в) Излез по квадратен просек за интервал од 80 ms со оптимизиран статички праг.

Подобрување на пресметувањето на локацијата на R-пикот

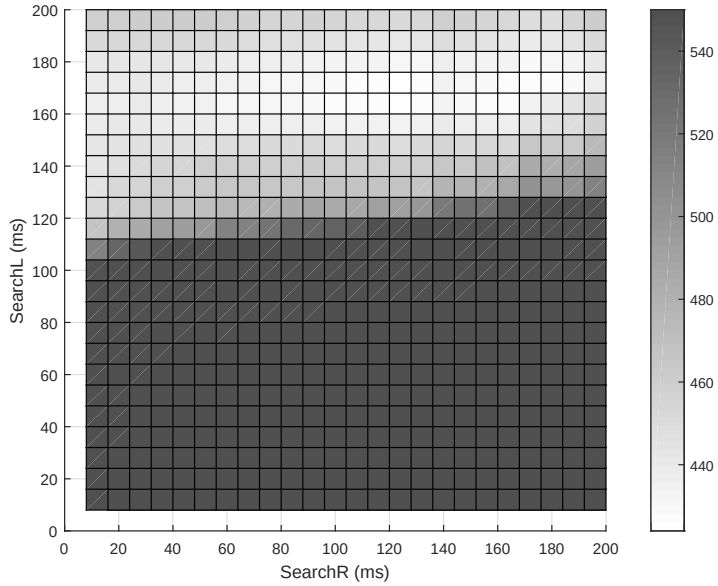
Бидејќи алгоритмот Хамилтон само приближно го лоцира пикот, еден начин да се намали бројот на FP, што се случува поради упорноста за утврдување соодветна локација на R-пик, е да се бара вистинскиот пик во близина. Идејата е да се најде најпогодниот локален максимум, со анализа на излезот што е елиминиран од шумот по филтрирање со канален филтер.

Оригиналниот алгоритам на Хамилтон ќе го одреди најдобрата приближна точка за локацијата на R-пикот. Ова е почетна точка за пребарување на локалниот максимум, во опсег, со најдени SearchL ms лево и SearchR ms десно од приближната R-пик локација. Откако ќе се најде локалниот максимум на излезот на пропусниот опсег, продолжуваме да го бараме локалниот максимум на вистинскиот сигнал, иако опсегот за пребарување ќе биде лимитиран на 48 ms.

Слика 0.8 ги илустрира основните чекори на овој алгоритам за подобрување во примерот прикажан на слика 0.6. Соодветните сегменти за пребарување се означени на оригиналниот сигнал. Оригиналниот Хамилтонов алгоритам го открива локалниот пик B^C, и нашиот алгоритам за подобрување открива дека B^{RC} е вистинската локација. Создадовме уш-



Слика 0.8: Правилна пресметка на R-пиковната локација на MITDB-записот 201 (426.4 сек.): а) Пресметана R-пик локација B^C и интервали за пребарување над излезот на каналeн филтер; б) Пресметаната R-пиковна локација B^{RC} на посоченият сигнал.



Слика 0.9: Лажни детекции за подобрување на пресметката на локацијата на R-пикот.

те еден експеримент за лоцирање на оптималните вредности за SearchL и SearchR. Лажните детекции на вредностите на праговите се нацртани на површинскиот графикон, прикажан на сликата 9, за различни вредности за SearchL и SearchR, користејќи ја статичката праговна вредност, STHR = 4. Најдобри резултати се добиваат кога SearchL = 160 ms и SearchR = 120 ms.

Создадовме уште еден експеримент за лоцирање на оптималните вредности за SearchL и SearchR. Лажни детекции за вредностите на праговите се исцртуваат на површинскиот графикон прикажан на сликата 0.9, за различни вредности на SearchL и SearchR, со користење на статичка праговна вредност STHR = 4. Најдобри резултати се добиваат кога SearchL = 160 ms и SearchR = 120 ms.

Подобрување на детекцијата на нискоамплитудните пикови кои следат по секвенции од високо амплитудните пикови

Две функции, означени како *mean*(формула 0.3) и *thresh* (формула. 0.4) играат клучна улога во оригиналниот алгоритам за детекција на QRS.

Ние предложивме промена во *mean* функцијата, со цел да го олесниме ефектот на високо амплитуден комплекс, проследен со значително пониозок комплекс. Наместо пресметување на *mean* вредноста, од последните 8 пикови, нашиот алгоритам смета само на половина од оваа вредност, кога пиковната амплитуда е повисока од динамичкиот праг (DTHR) и од точната вредност во сите други времиња, како што е определено со формулата 0.5. Ова го спречува линеарното зголемување на прагот, особено за сигнали со висока амплитуда, и го решава идентификуваниот проблем.

$$mean = \frac{\sum_{n=1}^{8} \begin{cases} X_n, & \text{if } X_n < DTHR. \\ \dfrac{X_n}{2}, & \text{otherwise.} \end{cases}}{8} \tag{0.5}$$

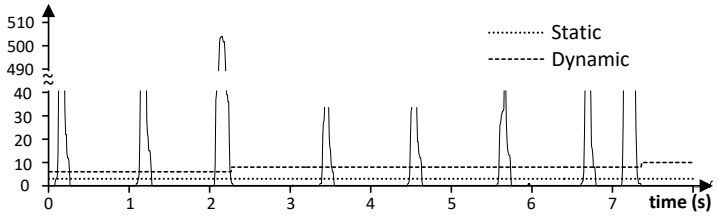
Покрај тоа, го сменивме методот *thresh*. Претходно, пресметаниот праг се множи со константата $TH = 0.3125$. Бидејќи се користи квадратниот режим, ја ажуриравме константата за множење со нејзиниот квадрат, односно $TH * TH$. Така, новата пресметка на DAT е одредена со формулата 0.6. И двете интервенции овозможуваат детекција на таквите отчукувања.

$$DAT = nmean + (qmean - nmean) * TH^2 \tag{0.6}$$

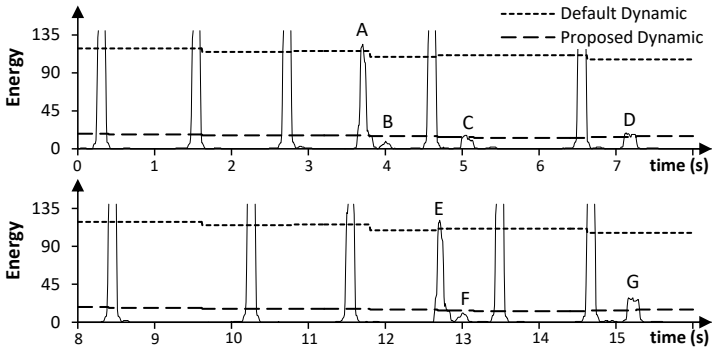
Добар перформанс се постигнува во двата случаи со:

- Низа од ниски амплитудни пикови по изолиран високоамплитуден пик;
- Изолиран ниско амплитуден пик, по низа од високи амплитудни пикови.

Слика 0.10 ја илустрира идејата на подобрување за презентираниот пример на сликата 0.3, каде што низата од ниско амплитудни пикови е последена од низа пикови со висока амплитуда, што ќе го зголеми ди-



Слика 0.10: Ефект на динамички праг за откривање на пикови по просек на квадратни вредности над интервал од 80 ms над MITDB-запис 114 (240 сек).



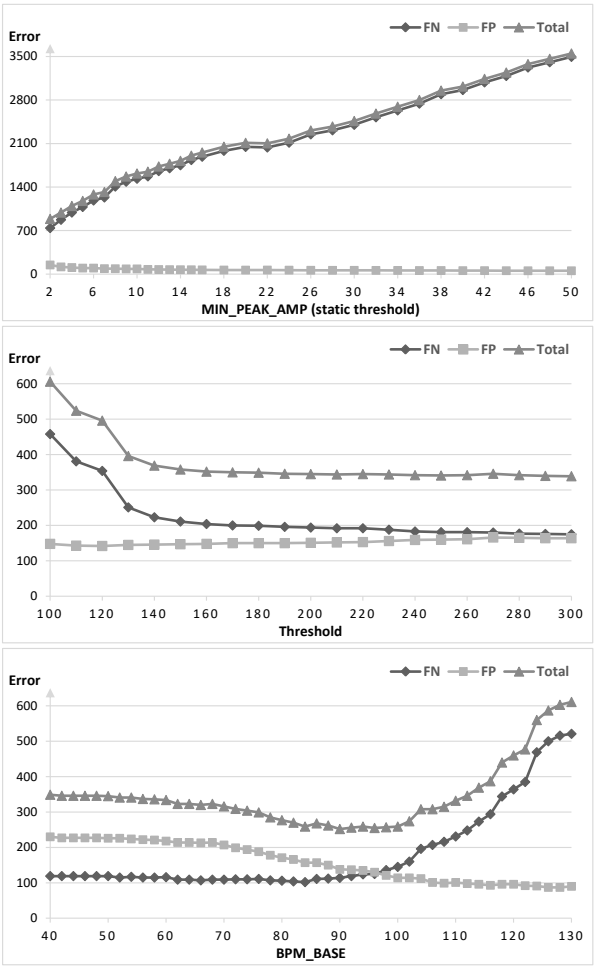
Слика 0.11: Ефект на динамички праг за откривање на пикови по просек на квадратни вредности над интервал од 80 ms над MITDB-запис 201 (424 сек.).

намичниот праг до таа вредност до кајшто сите последователни ниско амплитудни пикови се означени како пикови настанати од шум.

Сликата го покажува излезот по просекот за интервал од 80 ms, реализиран на квадратни (не апсолутни) вредности заедно со статични и динамички прагови. Забележете дека откриените потенцијални пикови и нивните апсолутни вредности, прикажани на сликата 0.3, се помали од оригиналната динамичка праговна вредност.

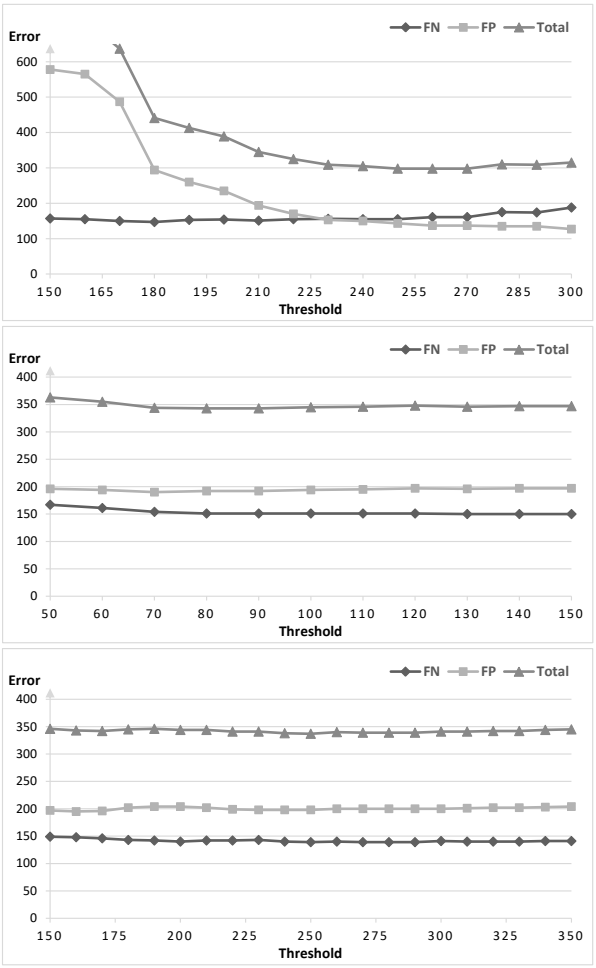
Ефектот на примената на новиот начин за пресметка на динамичниот праг може да се забележи и на слика 0.11, со претставување на изолираните нискоамплитудни пикови, по низа од високо амплитудни пикови. На новиот минимален праг, детектирани се 17 можни пикови, додека пак оригиналниот алгоритам со стандарден динамички праг не е во можност да ги прикаже пиковите означени како C, D и G. Ново оптимизираниот праг е во состојба да ги прикаже точно сите 15 пикови, а исто така да ги класифицира B и F како пикови настанати од шум.

Ние спроведовме многу експерименти, за да ја детерминираме оптимизираната вредност за DTHR праговната вредност. Тестовите вклучу-



Слика 0.12: Лажни детекции на пристапи на оптимизација на пиковите со ниска амплитуда (лево); секвенции на пикови со висока амплитуда (во средината), влијание на рата на срцев ритам (десно).

ваа тестирање на праговните вредности од 100 до 400. Средниот дел од сликата 0.12 претставува лажна детекција како функција од праговните вредности. Праговните вредности околу 200 се најдобрите кандидати за најефикасните перформанси.

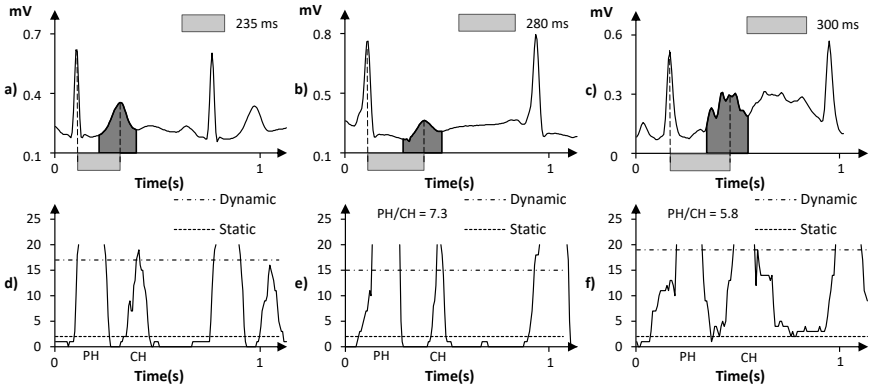


Слика 0.13: Неточни детекции на пристапи за оптимизација A0 (лево), A1 (средината) и A2 (десно).

Подобрување на елиминацијата на артефактите

Во финалниот чекор, воведуваме нова фаза на класификација. Целта на ова воведување е да се одреди дали пресметаното отчукување преставува реален или артефакт. Три важни правила за одлучување укажуваат дали пикот е артефакт. Ако ниедно од овие правила не е задоволено, отчукувањето се смета за вистински пик.

Од прелиминарната анализа, забележуваме дека артефактите генерално следат по вистинските отчукувања и се поставени на далечина помала од 320 ms . Второто важно прашање е дека артефактот очиг-



Слика 0.14: Извадоци на сигналот и излези по квадратен просек за интервал од 80 ms за извршување на нашиот алгоритам над MITDB: а) сигнал и г) излез за артефакт тип А0 во записи 103 (1304,4 сек); б) сигнал и д) излез за артефакт од А1 тип запис 124 (413,3 сек); в) сигнал и ѓ) излез за А2 тип артефакт за запис 101 (132,2 сек).

Табела 0.1: Листа на параметри за правила за детекција на артефакти.

Параметар	**Опис**
C	Current Detected Peak
P	Previous Detected Beat
CH	Current Peak Time Average Height
PH	Previous Beat Time Average Height
RR	Current beat to peak interval in ms
TAx	Time in ms optimizing Ax, $x \in \{0, 1, 2\}$
THRAx	Parameter optimizing Ax, $x \in \{1, 2\}$
MSxxx	xxx ms interval

ледно има помала енергија кога ќе се спореди со претходно откриеното отчукување. Оригиналниот алгоритам на Хамилтон ги елиминира артефактите на далечина помала од 195 ms. Нашите резултати покажуваат дека оваа вредност може исто така, да биде оптимизирана. Во табела 0.1 се опишуваат некои параметри кои се користат во нашите пристапи за оптимизација. Ние ги воведовме следниве оптимизирачки правила за елиминација на артефактите:

C е Артефакт, ако:

A0: $RR \leq TA0 = MS250$
A1: $RR \leq TA1 = MS260$ & $PH/CH > THRA1$
A2: $RR \leq TA2 = MS320$ & $PH/CH >= THRA2$

На сликата 0.14 се презентирани примери за различни типови детектирани артефакти. Идентификуваните сегменти покажуваат дека детектираните пикови се артефакти, бидејќи нивната енергија е повисока од

статичната, и од динамичните прагови, но сепак задоволува едно од правилата A0, A1 и A2. Во спротивно, ако ниедно од овие правила не е задоволено, тогаш откриеното отчукување C не е артефакт.

Резултатите од експериментот со користење на A0 оптимизацискиот пристап, се прикажани во левиот дел од слика 0.13. Праговната вредност од TA0 = MS250 = 250 ms дава ветувачки резултати како резултат на најниското ниво на лажни детекции. Во средишниот дел на слика 0.13 се демонстрира влијанието на праговните вредности врз пристапот A1. X-оската ги означува вредностите кои се множат со 100 додека пак оптимизираната вредност 80, за THRA1, одговара на 80 / 100 = 0.8 .

Десниот дел од сликата 0.13 покажува како праговниот параметар влијае на ефикасноста на методот за оптимизација A2. X-оската означува вредности што се множат со 100 и оптимизираната вредност 250 одговара на 250 / 100 = 2. 5 за THRA2.

Влијание на стапката на отчукувањата врз артефактите

Нашите експерименти покажаа дека стапката на отчукување влијае на интервалите во A0-A2 оптимизациските пристапи. Нека RR_{avg} ни биде просечната вредност на последните интервали од R до R, додека пак, f_s нека биде фреквенцијата на земање примероци. Тогаш , стапката на отчукувања BPM се мери со отчукувања во минута, според формулата 0.7.

$$BPM = \frac{60}{(RR_{avg}/f_s)} = \frac{60f_s}{RR_{avg}} \tag{0.7}$$

За да се направи класифицијата дали пикот е артефакт, поставивме три временски константи $TA0 = 250\ ms$, $TA1 = 260\ ms$ и $TA2 = 320\ ms$. Обично, пиковите на далечина помала од 250 ms се сметаат за пикови (кореспондира со рата на срцев ритам што е повисока од 240 BPM). Вториот и третиот праг, соодветно, проверуваат дали пикот е на далечина помала од 260 ms (одговара на 230 BPM) или од 320 ms (што одговара на 188 BPM).

Сепак, нашата анализа покажа дека предвремените отчукувања може се појават поблиску од овие вредности. Затоа воведовме фактор на скалирање на претходното подобрување и ги искористивме временските прагови, пресметани со формулата 0.8, кои се множат со факторот на скалирање BPM_BASE и BPM на срцевиот такт.

$$RR \leq TAx\frac{BPM\_BASE}{BPM} \quad x \in \{0, 1, 2\} \tag{0.8}$$

Десниот дел од сликата 0.12 покажува како факторот на скалирање BPM_BASE влијае на перформансите. X-оската ги покажува вредностите на скалирачкиот фактор BPM_BASE во опсег од 40 до 130, со

Табела 0.2: Споредба на перформансите на алгоритми со базата на податоци за аритмијата на МИТ-БИХ.

MIT-BIH Arrhythmia database					**all 48 records**			**no paced records (44)**		
Algorithm	f_S (Hz)	TP	FP	FN	**Tot Err**	Q_{SE}	Q_{+P}	**Tot Err**	Q_{SE}	Q_{+P}
Our Work	125	109382	110	112	222	99.90	99.90	194	99.91	99.90
Ghaffari [62]	360	109327	129	101	230	99.91	99.88	N/A	N/A	N/A
Bahouraa[21]	250	109625	133	174	307	99.83	99.88	303	99.82	99.88
Elgendi [53]	360	109775	82	247	329	99.78	99.92	322	99.76	99.92
Martinez [92]	360	109111	35	317	352	99.71	99.97	N/A	N/A	N/A
J.Martinez [93]	360	109208	153	220	373	99.80	99.86	N/A	N/A	N/A
Cvikl [37]	250	109294	200	200	400	99.82	99.82	373	99.81	99.82
Chiarugi [29]	360	109228	210	266	476	99.76	99.81	443	99.75	99.81
J.Lee [85]	N/A	109146	137	335	472	99.69	99.87	459	99.68	99.87
Zidemal [139]	360	109101	193	393	586	99.64	99.82	540	99.64	99.83
Hamilton [69]	360	108927	248	340	588	99.69	99.77	569	99.68	99.76
Choi [30]	360	109118	218	376	594	99.66	99.80	561	99.65	99.79
GQRS [63]	360	109196	302	298	600	99.73	99.72	562	99.72	99.72
Christov [31]	360	109615	386	288	674	99.74	99.65	670	99.72	99.62
Arzeno [20]	360	109099	405	354	759	99.68	99.63	N/A	N/A	N/A
Tompkins[107]	200	109532	507	277	784	99.75	99.54	771	99.73	99.50
Paoletti [109]	360	109430	565	379	944	99.65	99.49	924	99.64	99.45
Poli [113]	120	109522	545	441	986	99.60	99.51	N/A	N/A	N/A
Elgendi [52]	360	109397	97	1715	1812	98.31	99.92	1798	98.33	99.91

инкремент 2, а у-оската е бројот на грешки. Ние забележавме дека вредноста на $BPM\_BASE = 90$ ги минимизира грешките.

We observe that a value of $BPM\_BASE = 90$ minimizes the errors.

Евалвација и дискусија

Следната комбинација на параметри ги постигнува најдобрите севкупни перформанси.

$STHR = 2$, $SearchL = 152\ ms$, $SearchR = 56\ ms$, $DTHR = 200$, $TA0 = 250\ ms$, $TA1 = 260\ ms$, $TA2 = 320\ ms$, $THRA1 = 0.8$, $THRA2 = 2.5$, и $BPM\_BASE = 90$.

Табела 0.2 дава преглед на добиените резултати и покажува дека нашиот алгоритам постигнал подобро комбинирани вредности на сензибилитет и позитивна предикативна стапка.

Може да се појават неколку проблеми при споредувањето на кој било метод на детекција на QRS со други објавени трудови, како на пример:

- Нема изворен код предвиден за проверка на други пристапи;
- Нема информации за позитивната предвидувачка стапка; или

- Нема информации за бројот на грешки.

При анализирање на перформансите, само мал број трудови даваат информации за постигнатите позитивни предвидувачки стапки и тие обично се стремат да постигнат повисоки вредности на сензибилитет. Сепак, многу е лесно да се постигне повисока вредност на сензибилитет и претставување на повеќето од резултатите што сакате да ги вклучите во вашиот алгоритам, со релаксирање на ограничувањата на параметрите за оптимизација, но, во исто време, ова ќе произведе мноштво дополнително генерирани пикови кои не претставуваат QRS-пик. Затоа е многу важно да ги адресираме и сензибилитетот и позитивното предвидување за евалвација на перформансите.

Ова значи дека не постои метод за директна споредба. За справување со овој проблем, го анализиравме бројот на грешки како мерка на изведба (како што е определено од грешките во формула 0.2). Иако оваа мерка на изведба може да се постигне со пресметка на сумата на FP и FN, исто така може да се пресмета преку хармонична средина (HM) на QRS-сензибилитетот и позитивна предикативна стапка од формула 0.9.

$$Errors = TotalQRS * \left(\frac{1}{Q_{SE}} + \frac{1}{Q_{+P}} - 2 \right) \tag{0.9}$$

Ние ја искористивме формула 0.10, за да ја оцениме оваа релација. Покрај тоа, ние претпоставуваме дека бројот на дополнително детектирани (лажно негативни) пикови е многу помал од бројот на правилно детектирани пикови QRS, односно $TP >> FN$, што доведува до $TotalQRS \approx TP$.

$$\begin{aligned} \frac{1}{Q_{SE}} + \frac{1}{Q_{+P}} &= \frac{TP + FN}{TP} + \frac{TP + FP}{TP} \\ &= 2 + \frac{FP + FN}{TP} \end{aligned} \tag{0.10}$$

Беа пронајдени голем број несогласувања при анализирање на сродните истражувања. Табелата 1 од [107] покажува дека бројот на грешки е 782, иако тој е 784. Пресметката на вкупните отчукувања е најнезабележителна. На пример, TP + FN е поголем од TB во [52]. Ли објавува два труда [84] и [85], обезбедувајќи соодветно 109486 и 109481 вкупно отчукувања. Поранешниот има уште 6 дополнителни отчукувања, кои се од документите 118, 201, 205, 220, 221 и 233, додека вториот има 1 дополнително отчукување на запис 114.

Откривме дека различни автори користеле различен број од детектираните пикови. Тие треба да го користат вкупниот број детектирани отчукувања, бидејќи вкупниот број пикови вклучува исто нетактични отчукувања, на пример, локации каде што постои промена на ритамот. Тоа

е причината зошто ние го користиме вкупниот број забележани отчукувања во МИТ-БИХ Аритмија-базата на податоци 109494.

Табела 0.2, исто така покажува дека истражувачите, кои се фокусираат на алгоритмите за детекција на QRS, најчесто имаат тенденција да ги користат оригиналните фреквенции за земање примероци и резолуцијата на референтна ЕКГ-база на податоци. Вклучувањето на алгоритмот на 125 Hz значи дека речиси 3 пати помалку податоци се храна на QRS-детекторот, така што времињата на извршување се намалуваат соодветно. Исто така, приспособувањата што ги предложивме, ги зголемија перформансите на метриките, што во целост дава подобар QRS-детектор наменет за мал, преносliв, едноканален ЕКГ-сензор.

Заклучок

Воведовме неколку методи за подобрување на QRS-откривањето во алгоритмите, базирани на диференцијација. Иако нашиот пристап е демонстриран врз основа на Алгоритмот за детекција на QRS на Хамилтон, тој, може да се имплементира во други алгоритми. Резултатите покажуваат супериорен перформанс над другите објавени резултати.

Подобрувањата беа ефикасно вградени во индустрискиот QRS-детектор, за преносliв ЕКГ-монитор, каде што фреквенцијата што се зема како примерок изнесуваше 125 Hz, со 10-битна резолуција на AD-конверторот. Поставувањето на вредностите на прагот може да ја зголеми перформансата, но треба да се развијат нови начини, за да се генерираат прагови, како на пример:

- колку претходни удари може да се анализираат, за да се процени средната вредност,
- кои пикови ќе бидат класифицирани како QRS-отчукувања, бидејќи тие се многу слични по формата и
- влијанието на ратата на срцевиот ритам врз класификацијата на артефактите.

За оваа цел воведовме неколку нови правила за 1) пресметка на прагот за подобро класифицирање на пиковите на QRS, 2) елиминација на артефактите слични на QRS и 3) елиминација на артефактите врз основа на ратата на срцевиот ритам.

Поради рескалирање, слабите ЕКГ-контакти и шумот предизвикан од мускулите, 62 отчукувања не може да се детектираат со анализа на првиот ЕКГ-канал, без анализа на вториот канал. Ова дава зголемување на перформансата на QRS-сензибилноста на 99,96%, и нашиот алгоритам достигнува 99,90% QRS-сензибилност, со 99,90% позитивна стапка на предиктивност кога сите записи се анализирани во MITDB и 99,91% QRS-сензибилност за 44 записи без интензивни удари.

Генерално, објавените трудови за алгоритмите за детекција на QRS не ги нудат нивните изворни кодови, а само некои од нив се потврдени со референтни бази на податоци на ЕКГ, како што е МИТ-БИХ Аритмија-базата на податоци. Повеќе од алгоритмите даваат краток опис на прашања без детали за имплементацијата, посочувајќи само на теоретските прашања. Ова е причината зошто не може директно да се споредат резултатите. Различните пристапи, генерално, не постигнуваат резултати како оние, постигнати во реалната имплементација.

Ова истражување може да ги олесни можните алгоритми за детекција на QRS, за реконструкција и ресемплирање на референтни ЕКГ-бази на податоци, во замена за подобри перформанси. Нашите пронаоѓања покажуваат дека приспособената верзија на Хамилтоновиот алгоритам за детекција на QRS дава подобри резултати со 125Hz податоци за речиси трипати пократко време на извршување.

Нашата понатамошна работа ќе биде во насока на класификација на грешките во детекцијата на QRS, сè додека не успееме да ги елиминираме, како и за со цел да се моделира зависноста на фреквенцијата и битната резолуција. Дополнително, покрај нашиот алгоритам за детекција, планираме да создадеме квалитетен индустриски QRS-класификатор, со повисоки перформанси.

Скопје, *Ервин Домазет*
март 2019

Ervin Domazet

Parallel Digital Processing of ECG Signals

PhD Thesis

March 17, 2019

Dedicated to my parents Ergin Domazet and Güner Domazet who are very proud with this thesis.

Preface

The PhD research was realized at the Ss. Cyril and Methodius University, Faculty of Information Sciences and Computer Engineering in Skopje, North Macedonia. My motivation is based on continueing my journey on High Performance Computing area as I did on my B.Sc and M.Sc theses. My focus on the former was utilizing Graphical Processing Unit (GPU) to optimize Bellman–Ford algorithm, whereas on the latter was to optimize an Industrial Furnace Simulation code on multi core environments (CPU).

This PhD research is deeply focussed on the end-to-end optimization of digital processing of ECG signals. It all started when I first met with my supervisor, i.e respected professor Marjan Gusev, where we agreed on the vision to optimize the cycle of ECG processing with HPC techniques in return for an increased life quality of humankind. With my full of motivation and my supervisor's active involvement in every step we achived to optimize ECG processing.

Background

Recent advances in Information and Communication, have stimulated lots of possibilities. One such innovation is, when cloud processing center gathers streams of continuous data from wearable ECG sensors. Electrocardiogram (ECG) is a stream of electric impulses generated by the beating heart muscle. They are detected by electrodes placed on human skin, by measuring the electric potential that reaches the skin surface [111]. ECG stream holds cardiovascular condition of the patient and is represented by P, QRS and T waves.

Interpretation of an ECG stream is essential for a better quality of life. Interpretation along with signal analysis is achieved by Digital Signal Processing (DSP) [17, 123]. Nevertheless, ECG signals are exposed to noise that stem from several sources, varying from environment (electrical switching power, radio waves or other related sources) to the internal noises generated by the human breathing physical movement or similar sources.

Precise interpretation and analysis of the ECG signal can be achieved by eliminating the noise. Essential data preprocessing phase is conducted by the DSP filters. Hereinafter, the main ECG features can be detected and analyzed for further determination of the complex heart condition. Processing of the signal is based on detecting hidden information and the subtle deviation of the heart rhythm to alternating changes of the wave amplitude [8].

Lugovaya [88] had focussed on revealing the efficiency of an ECG signal for identification, when compared to the three efficient biometric methods, i.e identification based on fingerprints, iris or retina, and face. Her experimental results showed that the rate of correct identification was 96%, which gave an insight for considering ECG signal as a new biometric characteristic. What is more important, she was successful in showing the persistence of an individual's ECG characteristics over time (slow and gradual variations on ECG signal). This in turn makes it possible to detect subtle deviations of the heart rhythm and alternating changes of the wave amplitude.

Methodologies for processing and analyzing ECG signal consist of at least three stages: data pre-processing, feature space reduction and feature extraction [88].

ECG signals usually are contaminated by noise. They can stem from several sources, varying from the environment (electrical switching power or other related sources) or the internal noises generated by the human breathing physical movement. The preprocessing phase is responsbible for eliminating this noise, in order to make further analysis possible. There are several operators to eliminate noise. Digital Signal Processing (DSP) filters are essential in eliminating the noise and extracting the essential characteristic signal. In addition to them, the discrete version of Wavelet Transform has also the capability to remove the noise.

In the next stage, the main aim is to cancel the baseline wandering. It is basically a low-frequency component in the ECG system. This stems from offset voltages in the electrodes, respiration, and body movement. There are methods to eliminate this, whereas the mostly used one is the Discrete Wavelet Transform (DWT) algorithm.

In order to initialize the process of analysis and interpretation of the ECG signal, these features should be correctly detected. Several algorithms are capable, such as Nearest Mean Classifier (NMC), Weighted NMC, Linear Discriminant Analysis and others.

In the literature, there are several methods proposed to optimize the above stages of ECG processing. This research aims at presenting and further optimizing the general efforts on the process of ECG signal processing.

Problem Description and Objectives

In some specific cases real time processing of the ECG signal can save lives. However, this phases the big data challenge where data comes with a certain velocity and huge quantities. A server needs to receive these streams from a lot of sensors and needs to star various digital signal processing techniques initiating huge processing

demands. Due to intensive data processing, sequential algorithms are insufficient to run in real-time, especially when a cloud data server processes thousands of data streams coming from remote wearable ECG sensors.

In order real-time analysis to be done processing needs to be fast. Sequential algorithms are insufficient of real time processing of ECG signals. Several solutions have been proposed in order to optimize the processing. The goal of this thesis is to optimize the process of ECG signal processing.

The main research questions of this thesis are:

- Does parallelizing of Digital Processing of ECG signals enable real-time processing especially when data comes with a certain velocity and huge quantities?
- Will using different platforms for parallelization result in optimal cloud based solution?
- Will using different approaches for optimization result in more efficient algorithms?
- Is parallel Digital Processing of ECG signals a scalable solution compared to the sequential solutions?

Scope and Aim

The thesis scope is to develop and optimize algorithms that will detect heart anomalies in real-time, with the aim to increase life quality of patients. Thus the algorithms should have a very high corectness rate while being very efficient.

In this manner, we aim to fully optimize the overal process of ECG analysis, by using High Performance Computing techniques. This thesis utilizes Maxeler, CUDA and OpenMP technologies, for providing an optimum and scalable algorithm that run can on cloud.

Methods

General methods that are used in this thesis can be summarized as:

- **Analysis** of the bottlenecks of algorithms
- **Synthesis** of applicable intelligent solutions to overcome bottlenecks
- **Comparison** of optimized algorithms with available sequential algorithms
- **Experiment** the proposed algorithms on various platforms and dataset
- **Evaluation** of results in order to achieve an optimal solution in real time

The benchmarks used in our testing methodology are the same used in the IEC 60601-2-47 standard for particular requirements for the safety, including an essential performance of ambulatory electrocardiographic systems, and ANSI/AAMI EC57:2012 for Testing and Reporting Performance Results of Cardiac Rhythm and

ST Segment Measurement Algorithms. These standards use the MIT-BIH ECG arrhythmia database [99], and the American Heart Association's (AHA database) [72].

Content of the Thesis

This thesis consists of five parts: 1) Basic Concepts, 2) Parallelization of DSP Filtering, 3) QRS Detection, 4) ECG Feature Extraction, and 5) Concluding Remarks.

Part I presents the basics of ECG processing, proposed architecture, parallelisation platforms and the state-of-the-art ECG optimization algorithms. It consists of 5 chapters. Chapter 1 provides the motivation behind this thesis. It gives recent statistics, and describes the importance of real-time ECG processing. Basic definitions of ECG Signal representation are also provided. Chapter 2 gives theoretical details on the Digital Signal Processing area, which is the basis of this study. Details about Low-Pass, High-Pass and Band-Pass filter are provided. Three important stages of ECG processing is described in detail.

System architecture of a time-critical mobile application based on ECG medical monitoring is provided in Chapter 3. Requirement analysis together with Design specifications of such an application is provided. Specifically, workflow scenarios, business requirements, functinonal description, nonfunctinal requirements and system models are elaborated. Details about the parallelisation platforms used throughout this thesis are presented in Chapter 4, namely the OpenMP, CUDA and Maxeler. Finally, Chapter 5 briefly overviews ECG signal processing and mHealth related state-of-the-art optimization approaches.

The next Part II in 6 chapters analyzes and provides different type of optimizations to the DSP filters used throuhout ECG processing. Chapter 6 utilizes massive power of GPUs by using CUDA library, with the aim to parallelize DSP filter convolution operation. Algorithmical details and results are also provided. Further optimizations to the naive CUDA version is provided in Chapter 7, with a goal to find an optimized solution. Shared and constant memories of GPU are utilized, as well as loop unrolling and precision methods are also investigated. Chapter 8 presents a novel method by using Dataflow engines for parallelizing DSP filters.

DWT based noise filtering is presented in Chapter 9. Specifically, the sequential version of DWT used for filtering and feature extraction is parallelized. Eventhough DWT has high dependency between data, faster codes are reported. Chapter 10 on the other hand tries to optimize the available number of cores that can execute a node, within the parallel DWT algorithm. Finally, Chapter 11 gives an overview of obtained results, separatelly for each study. General related work on DSP filters is also provided.

Part III comes along with total of 4 chapters. General focus is on optimizing QRS detection algorithms. Optimal DSP bandpass filterring for QRS detection is provided in Chapter 12. FIR, IIR and Wavelet based filters are investigated, with the intention to find the optimal values of the central frequency, bandwith and -3 db cutoff frequencies of the filter. Impact of resampling is investigated in Chapter 13.

Experimental research is used to measure the performance of different samping rates and find optimal values.

Ampitude rescaling influence on QRS detection is considered in Chapter 14. Optimized versions of Hamilton's QRS detection algorithm is provided, where lower sample rates and amplitues are used to improve the original algorithm. Chapter 15 concludes the QRS detection part by presenting details about obtained results, and also related work on QRS detection.

Part IV provides improved algorithms for QRS detection and classification phase. It consists of three chapters. One of the best achievements of this thesis is provided in Chapter 16. Details about the improved version of Hamilton's QRS detection algorithm is presented. On the other hand, Chapter 17 presents a pipelined QRS classification algorithm based on decision rules, requiring simple operations, which can run on mobile devices. An overview of this part is given in Chapter 18, with related work on this area and the obtained improvements.

Finally, Part V provides the conclusion of this thesis. Specifically, Chapter 19 provides the conclusions and main results of this thesis. Future work is also provided, which would be our next steps on this area.

Main Results

This PhD thesis research comes with lots of published results on national and international-wide conferences and journals. General focus of them is primarily on the performance. All of these published papers provide theoretical as well as experimental results. Main outcomes of these researchs are provided below.

Dataflow DSP filter for ECG signals

Parallelization of the sequential DSP filter for processing of heart signals on GPU cores is addressed in [41]. Dataflow Computing is a completely different paradigm of computing than conventional CPUs, where instructions are parallelized across the available space, rather than time. It is a revolutionary way for High Performance Computing (HPC) solutions. Data streams are optimized by utilizing thousands of dataflow cores, providing order of magnitude speedups. We consider using Maxeler Systems for dataflow computing. The performance of the parallelized code will be compared to that of the sequential code. Our analysis shows speedups linear to the kernel size of the filter.

CUDA DSP filter for ECG signals

Utilizing massive power of GPU cores to parallelize the sequential DSP filter for processing of heart signals was considered in [40]. We set a hypothesis that a GPU

version is much faster than the CPU version. In this paper we have provided several experiments to test the validity of this hypothesis and to compare the performance of the parallelized GPU code with the sequential code. Assuming that the hypothesis is valid, we would also like to find what is the optimal size of the threads per block to obtain the maximum speedup. Our analysis shows that parallelized GPU code achieves linear speedups and is much more efficient than the classical single processor sequential processing.

Optimizing high-performance CUDA DSP filter for ECG signals

Optimization of our previous work on GPU parallelism[40] of DSP filters was adressed in [42]. The goal is to find an optimized solution that speed ups the parallel CUDA solution. The hypothesis set in this paper is to confirm whether the utilization of shared and constant memories on GPU can yield faster execution times on ECG signal filtering. To test the hypothesis we measured the execution times of naive GPU solution over the optimized solution. We were also interested in determining whether loop unrolling and precision has an effect on the speedup.

Results obtained by executing optimized algorithms show they are identical for each of the filter types. From the obtained results, we can conclude that proper usage of shared and constant memory has a positive impact on the performance. Our analysis showed that their combined effect yield 2.4 times faster executions compared to the previous code. We can, therefore, conclude that the hypothesis is confirmed. Considering loop unrolling speeds up the code by 1-5%. Moreover, we tested the decreased precision effect on the performance and got nearly 1.5 faster code when on 1000 filter length. We observed that the element version is not effective when ported on GPU. It is important to note that each of the proposed optimization techniques adds up to the combined speedup. We observed that the best-combined effect had a speedup of 6.

Parallelization of digital wavelet transformation of ecg signals

The advances in electronics and ICT industry for biomedical use have initiated a lot of new possibilities. However, these IoT solutions face the big data challenge where data comes with a certain velocity and huge quantities. In this paper[44], we analyze a situation where wearable ECG sensors stream continuous data to the servers. A server needs to receive these streams from a lot of sensors and needs to star various digital signal processing techniques initiating huge processing demands. Our focus in this paper is on optimizing the sequential Wavelet Transform filter. Due to the highly dependent structure of the transformation procedure we propose several optimization techniques for efficient parallelization. We set a hypothesis that optimizing the DWT initialization and processing part can yield a faster code. We have provided several experiments to test the validity of this hypothesis by using

OpenMP for parallelization. Our analysis shows that proposed techniques can optimize the sequential version of the code.

Optimal Parallel Wavelet ECG Signal Processing

Real time detection of heart abnormalities can prevent serious health problems. This requires real time processing of ECG data by a corresponding web service. Considering the case of wearable devices to collect ECG data, the signal is actually contaminated by noise. Noise can seriously change the ECG signal and occur in the form of a baseline drift representing various physical movements and breathing. Unless it is removed, correct analysis on ECG data is impossible. Being characterized by very low frequencies, its elimination can not be efficiently realized by simple DSP filters, such as Finite Response Filters (FIR) or Infinite Response Filters (IIR).

Wavelet Transformation is a promising technique to eliminate the noise with very low frequencies, and its digital version (DWT) is capable of efficient removing the ECG baseline drift. In this paper[43], we set a research question to investigate the dependence between the nodes in the DWT implementation (and therefore to their corresponding threads) and the available number of cores that can execute the code. This analysis leads to valuable conclusions that will allow construction of even better optimizations. Results indicate that proper allocation of cores can yield faster code.

A Time-Critical Mobile Application based on ECG Medical Monitoring

Recent statistics indicate that at least one in every three deaths in the world occurs due to a heart attack. Scientific studies show that certain types of such attacks can be detected before their occurrence. This makes the real-time processing of wearable ECG sensors and smartphones an extremely important topic. Delivering healthcare solutions for a mobile platform is an emerging field and a lot of research and development projects have started for this purpose. The concept of mHealth involves the use of provided technologies and the telecommunication infrastructure to deliver healthcare solutions. In the case when a mobile device acts as a data collector and performs the initial processing, possible disorders can be predicted at an earlier stage. This paper [48] contributes to the mobile healthcare solutions area by providing a requirement analysis of possible implementations of time-critical mHealth solutions.

Design specification of an ECG mobile application

Latest developments on IoT, stimulated new innovations. Electronic health solutions are among trending opportunities. Statistics indicate that cardiovascular diseases are the primary cause of deaths globally. Our knowledge of this type of diseases shows

that specific types of heart attacks can be prevented. Real-time acquisition and processing using wearable biosensors enable the detection of cardiac symptoms at an early stage. The concept of mHealth is defined as the combination of current state of the technology and the telecommunications infrastructure, in order to provide healthcare services. This paper[49] contributes to such solutions by provisioning generic design specifications of an ECG mobile application.

Optimal DSP bandpass filtering for QRS detection

An electrocardiogram refers to the process of recording the electrical activity of the heart over a certain time interval. ECG signal holds vital information for the current health condition of the patient. Detection of cardiac disorders is based on detection of sudden deviations from the mean line. Detection of heartbeat functions is based on extracting ECG characteristic features, especially the R-peak. Although in this paper, we address a general approach, we focus on using wearable ECG sensors and developing an efficient QRS detector to determine the heartbeat function. The real problem in detection and ECG signal analysis is processing the noise contaminated ECG signal and the way one can reduce the feature space to extract the relevant features. In this paper [64], we set a research question to investigate how the filter affects the accuracy, sensitivity and precision values on QRS detectors. We report our findings on optimal filter design with a central frequency of 8.33 Hz and -3db cutoff frequencies at 4 Hz and 20 Hz. The analysis is towards the construction of an efficient filter with small computing complexity intended to be used for wearable ECG sensors.

Optimizing the Impact of Resampling on QRS Detection

QRS detection is an essential activity performed on the electrocardiogram signal for finding heartbeat features. Even though there is already a lot of literature on QRS detection, we set a research question to find the dependence of QRS detection performance on the sampling frequency, and, if possible, to find a QRS detector that will be highly efficient at different sampling rates.

Our synthesis technique aims to find the optimal value of the threshold parameters that define if the detected peak is artifact, noise or real QRS peak. In addition, in this paper [65] we conducted experimental research to find the dependence and estimate the optimal threshold values for the best QRS detection performance.

Our approach results with increased QRS detection performance on the original sampling frequency by improving the original Hamilton algorithm.

We tested with the MIT-BIH Arrhythmia database. Lastly, QRS detection sensitivity and positive predictive rate are used to evaluate the performance of the algorithm.

Amplitude Rescaling Influence on QRS Detection

When we record the electrical activity of the heart we generate a signal called an electrocardiogram. Within the electrocardiogram, the information that explains the heart's health is based on the detection of QRS complexes. The focus of this paper[46] is on a wearable ECG sensor that uses a low sampling frequency and bit resolution while it converts the analog signal to digital data. The overall goal is to see if an efficient industrial QRS detector can be developed within these constraints. In particular, we set a research question to investigate how amplitude rescaling affects sensitivity and positive predictive rate of the Hamilton algorithm for QRS detection and improved it by optimizing it based on amplitude ranges.

We used the MIT-BIH Arrhythmia ECG database to evaluate performance. The original recordings are sampled with a sampling frequency of 360 Hz with a 11-bit resolution over a 10 mV range. Our experiments include testing rescaled signals on a sampling frequency of 360 Hz using different maximum amplitudes. We found that rescaling impacts performance and that the optimization parameters need to tuned to obtain the expected performance. However, the performance decreases when the maximum amplitude is lower than 9 bits.

Improving the QRS Detection for One-channel ECG Sensor

We analyzed several QRS detection algorithms in order to build a quality industrial beat detector, intended for a small, wearable, one channel electrocardiogram sensor with a sampling rate of 125 Hz, and analog-to-digital conversion of 10 bits. The research[47] was a lengthy process that included building several hundred rules to cope with the QRS detection problems and finding an optimal threshold value for several parameters. We obtained 99.90% QRS sensitivity and 99.90% QRS positive predictive rate measured on the first channel of rescaled and resampled MIT-BIH Arrhythmia ECG database. Even more so, our solution works better than the algorithms for the original signals with a sampling rate of 360 Hz and analog-to-digital conversion of 11 bits.

Applicability of the Results

Papers that are published within the research for this PhD thesis have obtained important results. Below we provide potential areas of their applicability.

Algorithms that are parallelized have broad range of applicability. An enormous research and analysis has been done in order to specify algoithmic parts that can be parallelized, and to show the effects of given calculations. OpenMP, CUDA and Maxeler based approaches were reviewed.

Improved ECG Data Pre-Processing

Different types of DSP filters are studied, including Low-pass, High-pass, Band-pass filters. Additionally, we also explored wavelet based filtering algorithms, namely the Digital Wavelet Transformation algorithms.

This thesis provided and experimentally verified a range of optimized approaches for ECG data pre-prepocessing. According to the need of literature, all of them can be utilized efficiently.

Improved QRS Detection

This is one of the main contribution of this thesis. We optimized Hamilton's QRS detection algorithm for one channel wearable Sensor, operating in 125Hz. Our proposed method produces even better results on rescaled and resampled data, when compared to the original daa of MIT-BIH Arrhythmia ECG database. The details of our method is already published in a journal, and can be applicable especially in environments requiring real-time efficient algorithms such as wearable sensors.

Cloud Based Remote ECG Monitoring

Our findings during this PhD research are already integrated to the Cloud Based Remote ECG Monitoring portal ECGAlert, supported by the Macedonian Fund for Innovations.

Review of the Published Work within this PhD Thesis

Within this PhD thesis, 10 scientific papers and 1 journal with impact factor have been published on international scope.

Optimization of DSP filters on OpenMP, CUDA and Maxeler platforms are adressed in [41, 40, 42]. Results indicate that superliniar speedups are possible especially when combining OpenMP and CUDA related approaches.

Parallelisation of Digital Wavelet Transformaton of ECG signals is considered in [44], where proposed techniques optimize the sequential code. On the other hand, an optimal Parallel wavelet ECG signal processing is provided in [43]. Our findings indicate that proper allocation of cores can yield faster executions.

Requirement analysis of a possible time-critical mobile healthcare solution is elaborated in [48], whereas generic design specifications of an ECG Mobile application are provided in [49]

We have conducted several studies to optimize QRS detection algorithms. We started with investigating the filter effect on accuracy, sensitivity and precision

values[64], where we report optimal filter design with a central frequency of 8.33 Hz and -3db cutoff frequencies at 4 Hz and 20 Hz.

Next approach was to optimize the thresholding parameters of the original Hamilton algorithm for detecting a real QRS[65]. Obtained results showed that we achieved increased QRS detection compared to the original algorithm. We were also interested on the effect of amplitude rescaling on QRS detection [46], with an observation that rescaling affecting the performance.

Our best achievement was to Improve the QRS detection for one-channel ECG sensor [47]. We obtained 99.90% QRS sensitivity and 99.90% QRS positive predictive rate measured on the first channel of rescaled and resampled MIT-BIH Arrhythmia ECG database. Our solution runs with a sampling rate of 125Hz, and produces better results compared to the original signals with a sampling rate of 360Hz.

Complete list of published papers within this thesis is given below:

1. “Dataflow DSP filter for ECG signals” in 13th International Conference on Informatics and Information Technologies, Bitola, Macedonia, 2016.
2. “CUDA DSP filter for ECG signals,” in 6th International Conference on Applied Internet and Information Technologies, Bitola, Macedonia, 2016.
3. “Optimizing high-performance CUDA DSP filter for ECG signals,” in 27th DAAAM International Symposium, Mostar, Bosnia and Herzegovina: DAAAM International Vienna, 2016.
4. “Parallelization of digital wavelet transformation of ecg signals,” in MIPRO, 2017 Proceedings of the 40th Jubilee International Convention. Opatija, Croatia, IEEE, 2017.
5. “Optimal Parallel Wavelet ECG Signal Processing” in 14th International Conference on Informatics and Information Technologies, Mavrova, Macedonia, April 2017.
6. ”A Time-Critical Mobile Application based on ECG Medical Monitoring” in 8th Balkan Conference in Informatics, 20-23 September 2017, Skopje.
7. ”Design specification of an ECG mobile application” in Telecommunication Forum (TELFOR), 2017 25th 2017 Nov 21 (pp. 1-4). IEEE.
8. ”Optimal DSP bandpass filtering for QRS detection.” in 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE, 2018.
9. ”Optimizing the Impact of Resampling on QRS Detection.” in 10th ICT Innovations 2018. Springer, Ohrid, Macedonia.
10. ”Amplitude Rescaling Influence on QRS Detection” in 10th ICT Innovations 2018. Springer, Ohrid, Macedonia.
11. ”Improving the QRS Detection for One-channel ECG Sensor” in journal of Technology and Healthcare 2019, in press, IOS Press.

Skopje, *Ervin Domazet*
March 2019

Acknowledgements

First and foremost I offer my sincerest gratitude to my supervisor, Prof. Marjan Gushev, who has supported me throughout my PhD thesis with his patience and broad knowledge while allowing me to work in my own way. I attribute the level of my Doctoral degree to his encouragement and effort and without him this thesis would not have been completed or written. One simply could not wish for a better or friendlier supervisor.

In addition, I express my kindest thanks to each member of ECGAlert team for their support and guidance. Special thanks to all of my colleagues from Technoperia, especially to my business partner Affan Hasan, for their great support.

Besides my advisor and supporters from ECGAlert and Technoperia, I would also like to thank to the thesis committee for serving with my thesis.

I would like to thank my family; my grandfather Abdurrahman Emuş, my parents Ergin Domazet and Güner Domazet, my brothers Furkan Domazet, Harun Domazet the ones that are in my heart for supporting me spiritually throughout my life.

Last but not the least, I would like to thank my wife, Nazife Yakupova Domazet and my lovely daughter Neslihan Domazet. They are always there cheering me up and stood by me through the good times and bad.

Contents

List of Figures

List of Tables

Acronyms

AAMI	Association for the Advancement of Medical Instrumentation
ABS	Absolute Value
AD	Analog-to-Digital
AHA	American Heart Association
ANN	Artificial Neural Network
ANSI	American National Standards Institute
AV	Atrioventricular Junction
AVG	Average Value
BLAS	Basic Linear Algebra Subroutines
BPM	Beats Per Minute
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
EC2	Amazon Elastic Compute Cloud
ECG	Electrocardiogram
EUR-ST	European ST-T database
DFE	Dataflow Engine
DSP	Digital Signal Processing
DWT	Discrete WAvelet Transform
FBGA	Field Programmable Gate Array
FFT	Fast Fourier Transform
FIR	Finite Response Filters
FN	False Negative
FP	False Positive
GB	Gigabyte
GFLOP	Giga-Floating Point Operations
GHz	Gigahertz
GPU	Graphics Processing Unit
HMM	Hidden Markov Model
HPC	High Performance Computing
HPF	High-Pass Filter
ICT	Information and Communication Technology

IEC International Electrotechnical Commission
IIR Infinite Response Filters
IoT Internet of Things
IOS iPhone Operating System
IT Information Technology
IWCU Inverse Wavelet Compute and Update
LPF Low-Pass Filter
MHz Megahertz
MIT-BIH Massachusetts Institute of Technology - Beth Israel Hospital
MLP Multilayer Perceptron
MOE Mixture of Experts
NMC Nearest Mean Classifier
OpenMP Open Multi-Processing
OS Operating System
PAC Premature Atrial Contraction
PHR Personal Health Records
PVC Premature Ventricular Contraction
QRS Combination of Q, R and S characteristic features
RAM Random-access Memory
RBF Radial Basis Function
SA Sinoatrial Node
TCA Total Classification Accuracy
TN True Negative
TP True Positive
TPB Threads Per Block
WCU Wavelet Compute and Update
WHO World Health Organization
WIFI Wireless Networking

Part I
Basic Concepts

Chapter 1
Introduction

The content of this Chapter was published at the 13th International Conference on Informatics and Information Technologies [41], 2016, 8th Balkan Conference in Informatics [48] and 25th Telecommunication Forum (TELFOR) [49], 2017, and the Journal of Technology and Healthcare [47], 2019.

Recently, statistics provided by the World Health Organization(WHO) on heart diseases [60], had revealed the phenomenon that different types of heart attack is the cause of nearly one third of deaths in the World.

From the other side, researchers have proven that detection of heart disorders can occur before their existence [83]. Motivated by this type of statistics and the research results, a lot of projects have started to develop real time processing of ECG data and increase the level of medical care and life expectancy.

The Internet of Things (IoT) represents a trending concept of connecting *things* rather than *computers* [95]. Mobile Health (mHealth) is an emerging technology where mobile devices are used for medical monitoring. Many innovative solutions are competing on the market for achieving effective ways of mobile medical monitoring.

1.1 Motivation

Advances in the Internet of Things (IoT) field have encouraged researchers to intensify their focus on Electrocardiogram (ECG) processing, especially for wearable devices. This field has become a popular research topic in biomedical engineering [27]. Electrocardiogram (ECG) is a stream of electric impulses generated by the beating heart muscle. They are detected by electrodes placed on human skin, by measuring the electric potential that reaches the skin surface [111].

Interpretation of an ECG stream is essential for a better quality of life. Interpretation along with signal analysis is achieved by Digital Signal Processing (DSP) [17, 123]. Nevertheless, ECG signals are exposed to noise that stem from several sources, varying from environment (electrical switching power, radio waves or other

related sources) to the internal noises generated by the human breathing physical movement or similar sources.

Precise interpretation and analysis of the ECG signal can be achieved by eliminating the noise. Essential data preprocessing phase is conducted by the DSP filters. Hereinafter, the main ECG features can be detected and analyzed for further determination of the complex heart condition. Processing of the signal is based on detecting hidden information and the subtle deviation of the heart rhythm to alternating changes of the wave amplitude [8].

Lugovaya [88] had focussed on revealing the efficiency of an ECG signal for identification, when compared to the three efficient biometric methods, i.e identification based on fingerprints, iris or retina, and face. Her experimental results showed that the rate of correct identification was 96%, which gave an insight for considering ECG signal as a new biometric characteristic. What is more important, she was successful in showing the persistence of an individual's ECG characteristics over time (slow and gradual variations on ECG signal). This in turn makes it possible to detect subtle deviations of the heart rhythm and alternating changes of the wave amplitude.

Potential heart risks can be detected before some time making the real-time processing a critical task. This, in turn, can save lives [88]. Sequential algorithms are insufficient of processing ECG signals real time, especially in the case of a data center intended for real-time processing and analyzing thousands of connected wearable ECG sensors. Our motivation is, therefore, to optimize the high-performance solution for signal processing.

In the case when wearable ECG biosensor collects and transmits data to the mobile device, ECG data can initially be processed on the mobile devices. In this manner certain types of heart-related disorders can be detected in an early stage. This can save lives and reduce the overall mortality rate [88].

1.2 ECG Signal Representation

ECG stream holds cardiovascular condition of the patient. Figure 1.1 reveals general representation ECG signal with its representative P, QRS and T waves. Algorithms for detecting ECG disorders are based on correctly detecting these features. In each of these algorithms, the fundamental step is to detect the R peak as the initial step of the QRS detection.

Fig. 1.1: General Representation of ECG Signal.

Chapter 2
Ecg Signal Processing

The content of this Chapter was published at the 13th and 14th International Conference on Informatics and Information Technologies [41, 43], 2016-2017.

Digital Signal Processing (DSP) area has introduced a powerful set of tools to deal with digital signals, and some of them can be successfully applied to the analysis of ECG signals. Being one of the most important tools of the DSP area, filtering can be used for noise elimination and further on for extraction of signal features. Thus, main features of ECG can be detected and extracted from the hidden information, such as alternating changes of the wave amplitude and subtle deviation of the heart rhythm.

2.1 Digital Signal Processing

DSP is the act of manipulating signals with intention varying from filtering, measuring to producing or compressing analog signals. As the power of computers radically increased during the last decades, so does the power of the DSP [123]. DSP had made a tremendous impact on science and engineering, by providing methodologies to deal with the most powerful technologies.

DSP had revolutionized many fields in science and engineering. There are many industrial sectors benefiting from the advancements on the DSP field such as Medical, Military, Space and Telephone. Electrocardiogram analysis, diagnostic imaging, voice and data compression, radars, secure communication, telephone signal filtering are among the range of revolutionized fields.

ECG signal filtering is applied with the intention to remove the noise that stem from several sources. The commonly used method in DSP filtering is the *convolution*, as one of the most important techniques of signal processing. It is defined as a mathematical operation that combines the *input stream* and the *impulse response* in order to generate a new *output stream*. In case of a filter, the impulse response is known as a *filter kernel*.

Each value of the output stream in digital signal convolution is represented as the sum of input stream multiplied by set of weight coefficients, which define the *impulse response*. The impulse response is the signal that results when a delta function (unit impulse) is the input in the DSP filter.

Denote by $f(i)$ the weight (filter kernel) coefficients in the range $i \in \{-\infty, +\infty\}$ if it is an infinite response filter. We will use finite response filters and the weight coefficients $h(i)$ in the range $i \in \{0, \ldots, M-1\}$, where M is the filter length. Let the input stream consists of elements $x(i)$ and the output stream of elements $y(i)$, for $i = 0, \ldots$. The convolution, as a mathematical operation can be expressed by (2.1).

$$y(i) = \sum_{j=0}^{M-1} h(j)x(i-j) \tag{2.1}$$

During this research, we have used the three classic filters to eliminate the noise: Low-Pass, High-Pass and Band-Pass filters.

2.1.1 Low-Pass Filter

Low-pass filters are designed to thoroughly weaken all the frequencies above the cutoff frequency, known as a *stopband*, while passing all frequencies below the *passband* [123]. These filters are composed of stream of data items. All samples of the output stream are in fact a weighted average of the input with the adjacent points of low pass filter. A simple low-pass filter is presented in Figure 2.1.

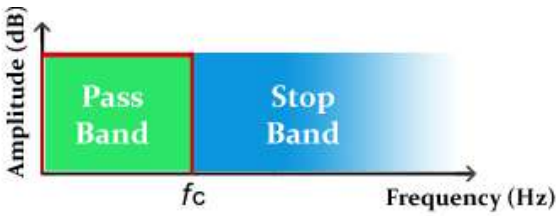


Fig. 2.1: Frequency response of a simple Low-pass filter.

2.1.2 High-Pass Filter

A high-pass filter has opposite characteristics of the low-pass filter. The effect of the filter is to weaken the frequencies below the *cutoff frequency* whereas passing all frequencies above the cutoff frequency.

As in the case of Lowpass filter, the output is generated with a weighted average of the adjacent input stream. The response characteristics of a simple high-pass filter is presented in Figure 2.2.

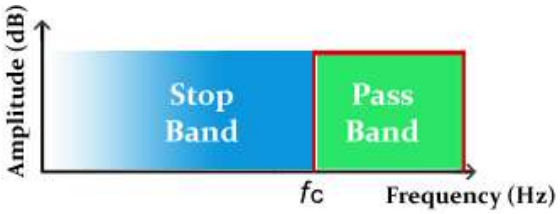


Fig. 2.2: Frequency response of a simple High-pass filter.

2.1.3 Band-Pass Filter

A band-pass filter is a composition of the high-pass and low-pass filters. This type of filter passes certain ranges of frequencies and rejects the frequencies of the remaining region. The frequency response of a simple band-pass filter is shown in Figure 2.3.

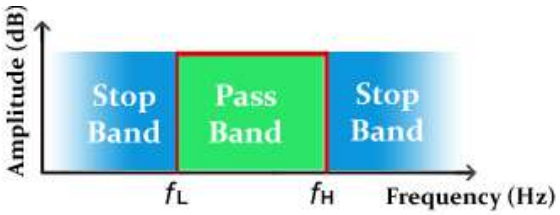


Fig. 2.3: Frequency response of a simple Band-pass filter.

2.1.4 Visual Interpretation of Filter outputs

Figure 2.4 represents a segment of an ECG signal with several QRS complexes, filtered with a low pass filter of 30Hz, high pass filter of 0.5Hz and a band pass filter between 0.5Hz and 30Hz.

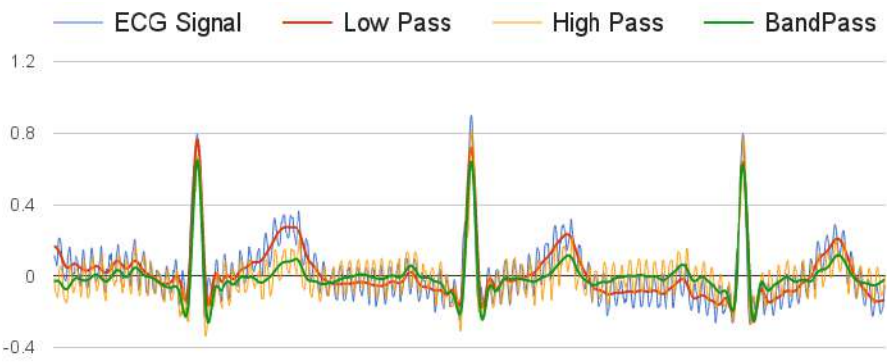


Fig. 2.4: A segment of an ECG signal with several QRS complexes, filtered with a low, a high and band pass filter.

2.2 ECG signal Processing

Methodologies for processing and analyzing ECG signal consist of three stages: data pre-processing, feature space reduction and feature extraction [88]. Figure 2.5 shows the general method for processing and analyzing ECG signals [88].

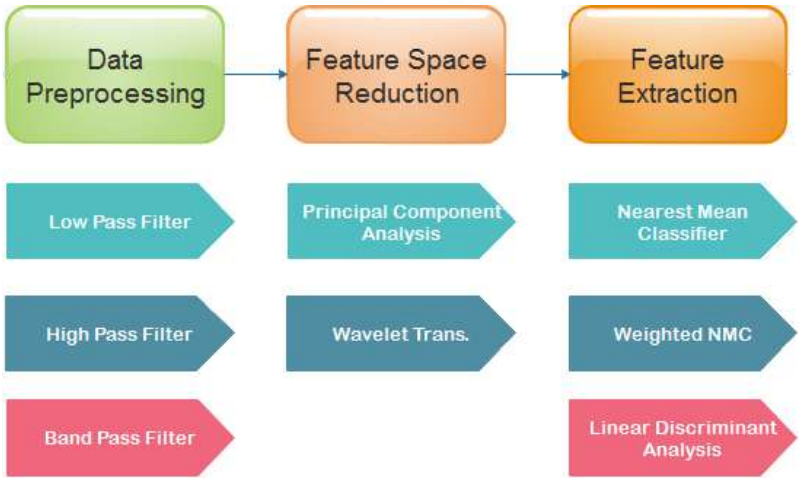


Fig. 2.5: Methodology for ECG signal processing.

DSP filters are generally used in the data pre-processing phase. Low pass filters are usually used to eliminate the noise with high frequencies, such as the electrical switching and radio waves. High pass filters eliminate the noise initiated by physical

movement and breathing, mainly interpreted as baseline drift elimination. Bandpass filters, as a combination of high pass and low pass filters are considered as effective DSP tools for noise elimination. Although, DSP filters eliminate the noise to a certain extent, they provide a relatively clear signal, which can be further processed for feature extraction.

In the feature space reduction phase, the signal is analyzed by detecting the peaks of QRS complexes and locating the peaks of individual P and T waves. A QRS complex is used as the starting point for further analysis, and, therefore, it's exact detection is of a high importance [97]. For example, a Wavelet transformation can be used for baseline drift elimination in this stage. In the final phase, QRS features are extracted, and the ECG signal precisely characterized.

The quality of extracted features, is directly dependent on the correct rate of eliminated baseline drift. Thus, focusing on this step is vital.

Digital filtering is essential for both the first two steps of the ECG signal processing. Wavelet Transformation is an efficient method used in both the elimination of baseline drift and QRS complex extraction.

Chapter 3
System Architecture

The content of this Chapter was published at the 8th Balkan Conference in Informatics [48] and 25th Telecommunication Forum (TELFOR) [49], 2017.

The general architecture of an IoT solution for a time-critical monitoring on mobile devices is based on the following four actors and segments.The first actor is the patient who wears an ECG sensor responsible for collection of real time ECG data. The patient's mobile device is another segment of the same actor. ECG data scanned by the ECG sensor is transmitted to the close by mobile device via personal area network communication. The mobile device is responsible for receiving and applying the initial processing on the signal. This device is used to upload the received ECG signal to the ECG Cloud Processing Centre, as another segment.

The ECG signal is further processed in the cloud, and any potential heart conditions are identified. If heart-related problems are detected then risk alerts are sent to the medical staff for further assessment and clarification. If they confirm a potential occurrence of a heart attack, then an emergency ambulance is being called with the coordinates of the patient. This process can prevent sudden deaths, especially when the onset of a heart attack is registered on time.

The life cycle of this high level architecture of the time-critical mobile application is illustrated in Fig. 3.1.

In this paper, we focus on the requirement analysis and design specification of the mobile application. Next sections provide details about the workflow diagrams and the business requirements.

3.1 Workflow Scenarios

The mHealth solution can be used in three different solutions. These can be listed as:

- *Real time visualization and monitoring*: The main aim would be to receive the ECG signal and to visualize it. This is to be used for healthy patients. Unless the

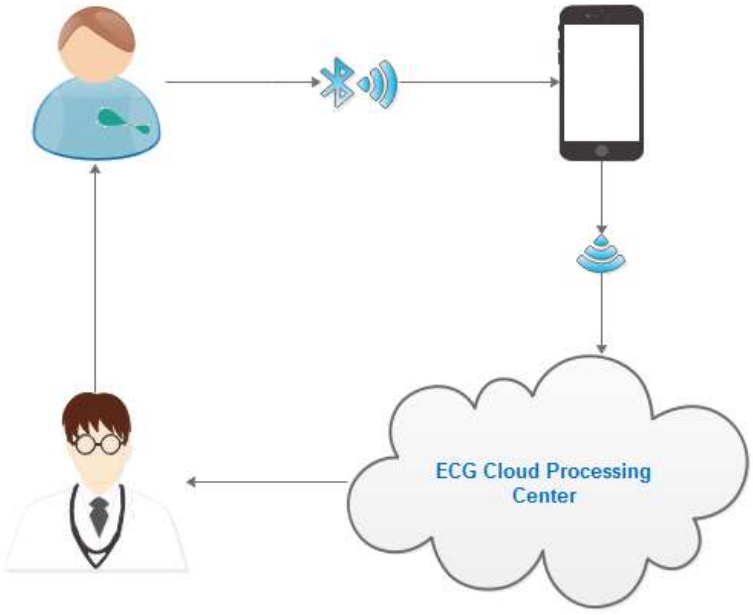


Fig. 3.1: High level architecture of the solution.

Event Alert Mechanism is used, the cloud will not be informed. Once the cloud is informed, an analysis will be performed on the region of interest. results will be analyzed by the medical staff and the cardiac patient informed.

- *Collect data for a certain period*: This scenario has the capability of the preceding one with additional functionalities. This is to be used for cardiac patients who have occasional complaints. It has the ability to collect data for a certain period, such as one week or so. The solution can be treated as an ECG Holter system. When the period completes, the mobile application must transmit the scanned data to the cloud efficiently. This will start the post processing and enable a platform to detect and anlyze abnormalities by the medical staff.
- *Continuous ECG Monitoring*: This has the capability of the preceding ones with all of the functional requirements. This should be used on patients with a certain level of heart disease. The measured ECG signal will be periodically transmitted to the cloud. The cloud server processes the signals and alerts the medical staff.

3.2 Business Requirements

Business requirements generally provide a better understanding of the business objective of the product. In the case of a mobile healthcare solution, the key actors are

cardiac patient, *medical staff*, *health service providers* and the *the solution provider*. Each actor has different business requirements, defined as follows.

3.2.1 Cardiac Patient

The beneficiaries of the clardiac patient is the solution having medical advices from an end-to-end functioning solution that will keep them in a good health status. The value of this is priceless, as it has the capability of detecting possible heart problems in early stages.

3.2.2 Medical Staff

The medical staff must have the opportunity to contribute to the solution by provision of medical devices. A web application needs to be hosted on the cloud to review ECG reports, and overview the alerts to make final decisions. The timing of the medical staff to deliver response is valuable, where they can enroll to the project part-time. Thus, the solution must be efficient in each aspect. There are two aspects on which the medical staff can benefit. The primary one is it increases the reputy of them. By making analysis of the ECG scans on the alerts, they can also make additional income.

3.2.3 Health Service Provider

When the review analysis of the medical staff addresses a serious health problem, the cardiac patient must be responsible by the health service providers. The requirements of the system is to inform them and take care of their health. Medical staff also belongs to the health service provider and their response is essential. The readiness of the health providers increase, while they focus more on the health of the patients.

3.2.4 The Solution Provider

The solution provider assures value added services for means of interoperability of actors. They must continuously maintain each step, with the aim to enable a secure platform to host the overall solution. They will increase the awareness of people and decrease the health costs of the Government. For doing so they will have financial

benefits. This will certainly help them to cover up their operational expenses and continue their research.

3.3 Functional Description

Requirements are considered as operational constraints any product must satisfy. It is a general term that can range from a high-level abstraction to detailed functional specifications [115, 94]. In this section, the focus is on the requirement elicitation of the mobile application part.

Mobile application intended to operate as m-Health solution must satisfy the following functional requirements [110, 68]:

3.3.1 Acquisition of ECG signal

ECG holds significant information regarding cardiovascular condition. When a cardiac patient wears the ECG biosensor, it is activated and starts to scan the heart signals. The scanned ECG signals are then transmitted to the mobile device.

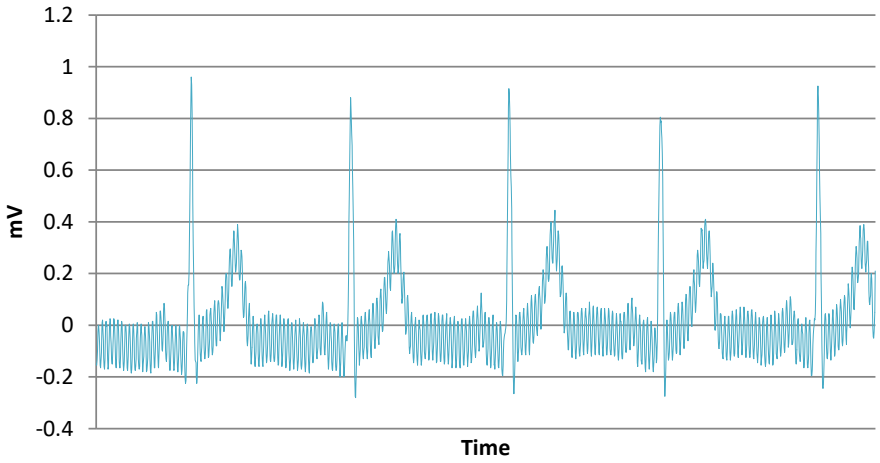
The mobile application within the m-Health solution must receive the signal provided by the ECG biosensor. In order to accomplish that, there should be a functionality to connect (likewise disconnect) to the sensor. When the connection is established, the mobile application should receive the encoded ECG signal network data packets. These packets have a header and data payload. The mobile application should be able to decode the packets, and put them to the processing queue.

The mobile application should also have a procedure for periodically checking the connection to the sensor. This is a common issue where the connection between devices can be lost due to several problems, such as the insufficient battery, the long distance and Bluetooth signal interference.

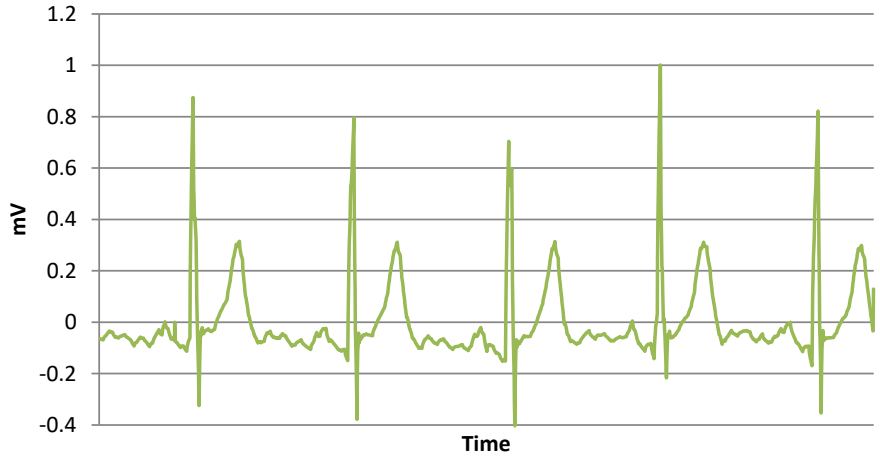
3.3.2 Data preprocessing

Raw ECG signal received from the biosensor is occupied with different types of noise. Radio waves, electrical switching power, internal noise generated by human breathing are physical movement are some of the noise sources.

Digital Signal Processing is based on filters in order to cope with the enviromental noise, where low and high pass filters can effectively be used. Fig. 3.2a shows a segment of ECG signal occupied with noise, whereas the Fig. 3.2b shows the signal with the eliminated noise.



(a) ECG signal contaminated with noise



(b) Filtered ECG signal

Fig. 3.2: Segment of an ECG signal and its representation with removed baseline drift and high frequency noise

Any further determination of the complex heart condition must be possible after the baseline drift and the high frequency noise is eliminated. The mobile application must implement an efficient filter to save battery resources.

3.3.3 Detecting of heart-related abnormalities

The algorithms to process and make decisions on the heart condition based on the ECG signal require a high processing power. The processing power of mobile devices is insufficient of real-time ECG processing, thus a more comprehensive analysis should run on the cloud ECG processing centre.

However, the mobile application must extract basic features from the ECG by algorithms that do not require a high processing power, especially those for detection of obvious heart rate abnormalities. In these cases, the mobile application must mark the region of interest and send an alert to inform the cloud centre immediately.

3.3.4 Visualization and monitoring

The mobile application must visualize the current ECG signal and enable monitoring for the patient. The requirement is to show only the filtered signal. A history of abnormal activities, and a certain depth of last ECG activities should be saved locally. The visualizer module should also be able to reveal them.

3.3.5 Event tracking

As mentioned previously, the mobile device is not capable of running complex algorithms for making decision on the heart condition based on the ECG signal. However, sometimes the cardiac patient may not feel well. The mobile application must provide means of manual marking of an event and act as a kind of *panic button*.

The process must alert the cloud solution and send an ECG segment of the request moment. These marking points will be important for post processing algorithm. In order to prevent any misuse, there must be a limit on the usage of this functionality.

3.3.6 Local storage of ECG data

A functionality for local storage is compulsory. Manual alerts, abnormalities and the preprocessed data must be saved locally. The preprocessed data should be appended to a single file unless the connection is lost.

This module must ensure that there is enough space in the mobile device. In case a threshold is exceeded, old files without detected abnormalities should be deleted.

3.3.7 Transmission to the Cloud server

This is the final and most important functionality of the mHealth application. In order to communicate to the main application, a separate daemon process must be implemented in the background and deal with all transmission issues.

It should have several functionalities. One of them is to connect to cloud server API for transmission of files. User should also be able to specify the synchronization period.

Occasionally, files for transmission can be large due to continuous monitoring. The daemon must be able to split ECG files into smaller segments. The limit should also be customizable. Files should be transmitted on available WIFI connection. There should be an option for transferring on available mobile connection.

When the files are uploaded, they can be deleted as a default option. User should be able to modify this. In any case, when an alarming threshold is reached, all of the transferred files must be deleted. A list of all transmitted and queued files should also be shown.

This study has defined the minimum viable functionalities of any mHealth solution. However, other modules can also be added.

3.4 Nonfunctional requirements

Nonfunctional requirements are not directly related to functionality. They in fact, describe user-level requirements. There is a broad range of non functional requirements. The mHealth solution must consider the following:

3.4.1 Usability

It must be kept in mind that elderly people will dominate the usage of the application. Thus, the interface must be easy to learn and remember. This would decrease the rate of a need for consulting the manual.

3.4.2 User-friendliness

The application must be user-friendly. It should give the feeling of its purpose. Its design should be responsive, in order to run properly in each type of resolution.

3.4.3 User Experience

The mHealth solution must provide a high level of user experience. The solution to this is by enabling users to achieve their objectives while using the product. It is highly likely that elderly patients will be willing to use the product, thus a starting point is to collect more details about their expectations.

3.4.4 Interoperability

In the context of IoT, interoperability is the ability to exchange information between so called *things*. The mHealth solution roughly exchanges data between the ECG biosensor and the ECG cloud processing centre. Thus the interfaces, communication syntax, the information flow and the security protocols must be clearly defined. An important point is that the solution must be loosely coupled to other segments.

Interoperability is the key feature of IoT concept, enabling different devices to connect easily. This is the key factor that have stimulated the advent of cloud-based mHealth solutions.

The interoperability can be addressed on several layers and provide independence of sensors, mobile device platform or cloud. For example, the mobile application must be platform independent and should run at least on Android OS, IOS and Windows Phone. In each of the platforms the design must be equivalent (or compatible as much as possible) and same functionalities should be provided.

3.4.5 Reliability

Reliability is sometimes referred as availability, which is a measure of the percentage of time the application works correctly. In the mHealth application, this can be measured by the time the percentage of time the functional requirements run correctly.

The state of art approach is to have a reliability of 99.99 percent, which means given a 365-day year, in total it can fail to properly function in 52.56 minutes in one year.

3.4.6 Performance

The performance criteria for the application is to perform the above-mentioned functional requirements in real-time. In case the device is not capable to perform the operations, the application must automatically be able to alert.

When focussed on time-critical domain, the requirements must be exactly defined in order to detect, alert and react on a given heart condition.

Performance of this solution must be one of the priorities. Sometimes milliseconds can prevent further damage of the health of the patient. However, it is important to know that performance in IoT solutions does not solely depend on one node. This leads to the fact that each of the components on the health ecosystem must be optimized.

3.4.7 Platform Compatibility

The application must be available for the market dominant mobile operating systems, i.e Android OS, IOS and Windows Phone.

3.4.8 Energy Efficiency

In order to process the functional requirements, application consumes device's energy. Algorithms and procedures must be optimized. Especially the Bluetooth connection listener, and the daemon process must run efficiently. In this manner its overhead to the battery decreases and ensures long battery life.

3.4.9 Data Protection and Security

This is one of the most important nonfunctional requirements. ECG data is sensitive and must be kept private. The mobile device must ensure that any third party applications does not have access to the files. Additionally, the daemon process must encrypt files before transmitting the ECG files.

Health data is sensitive, and can easily be misused, especially following the health-related data protection laws and legislation. Securing must start from the transmission of ECG signals by the ECG biosensor, and continue up to the delivery of application data to the medical staff. One should keep in mind that each part of the cloud solution should be kept on a high security level. In this way patients trustiness level can be increased.

3.4.10 Fault Tolerance

Systems are prone to errors, where each of them has a certain level of ability to continue operations on failures. This is defined as Fault tolerance. In modern systems,

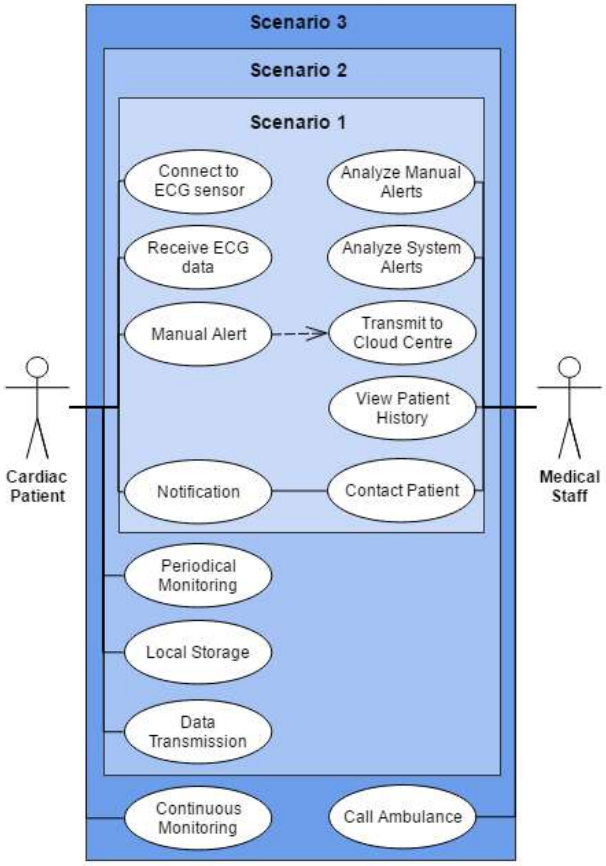


Fig. 3.3: The use case diagram of the mHealth solution.

the applications have procedures to automatically overcome errors. Possible cases of failures of the mHealth solutions must be well-defined, and the system must be able to catch them. The application will maintain a specified level of performance even if faults happen.

3.4.11 Disaster Recovery

The capability of any system to restore to the previous well-known point in cases of incidences or disasters, is defined as the disaster recovery. All of the potential disaster cases must be examined. Prior to this, a disaster recovery procedure document must be prepared, and a team for immediate intervention must be kept on

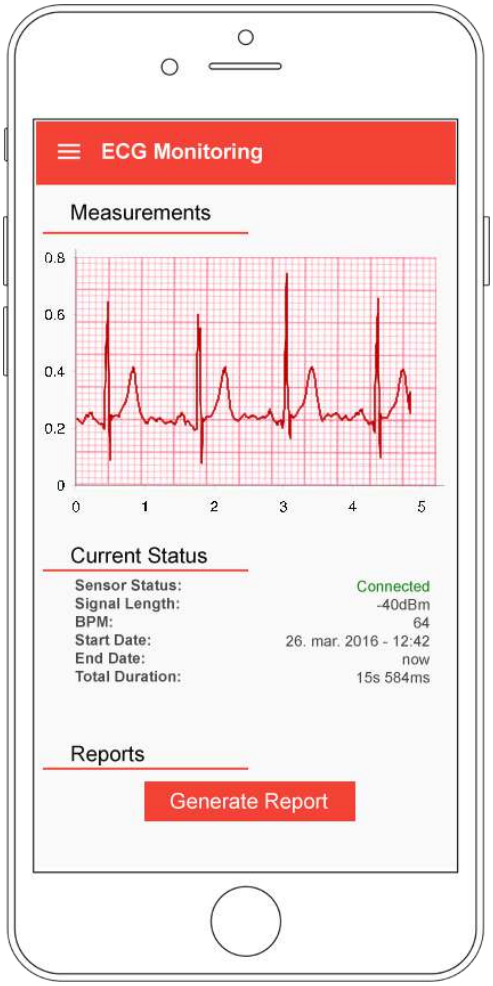


Fig. 3.4: Convenient landing page mock-up design for mHealth application.

hold. As new features are integrated, it is essential that the recovery procedure be kept up-to-date.

3.5 System models

The System Modeling concept is used for a better understanding of the functionality of the system with visual models. These are essential in terms of better communication, visualization and verification.

3.5.1 Use case model

Figure 3.3 presents three different use cases which are based on the provided workflow scenarios. There are two actors, the *Cardiac Patient* and the *Medical Staff*. The observation is that subsequent scenarios include the use cases of the preceding one.

3.5.2 User Interface

Red oriented colors should be selected when designing mock-ups for the mobile application. This would give the feeling of a heart to the user. Simplistic designs should be preferred, with the aim of public usage. Although being a design decision and not binding, Figure 3.4 presents a convenient landing page mock-up design. Current ECG is visualized, and apparent parameters are summarized.

3.5.3 Verification and Validation

Verification and validation are required for checking whether a product meets the defined requirements and the specifications. Verification is the terminology to confirm that the software conforms to its specification, whereas validation focusses on whether the software does the requirements that are previously defined.

In their work Speidel and Sridharan [125] have inferred that conventional verification and validation methods are not sufficient to overcome the challenges exposed by todays mobile devices. They also mention that mHealth concept is in a premature phase, and has difficulties in finding the right method for software verification and validation. Their provided solution is based on the crowd-testing methodology, which they believe can effectively be used in the area of medical monitoring.

This has in fact stimulated researchers to focus on it. Particularly in our case, a separate study must be made to compare the conventional and state-of-the art methodologies to find the most suitable verification and validation technology, and select the best strategy.

Chapter 4
Parallelisation Platforms

This chapter will provide detailed information about the paralellisation and optimization platforms used throughout the thesis.

4.1 Shared Memory Multiprocessor (OpenMP)

In shared memory architectures(Figure 4.1), shared memory locations can be accessible by all the processors. OpenMP is a shared memory programming library, developed and defined by a group of major computer hardware and software vendors[105]. OpenMP provides a portable, scalable model for shared memory parallel applications. OpenMP API provides set of compiler directives to support shared memory parallelism, by supporting C/C++ and Fortran on a wide variety of architectures[50].

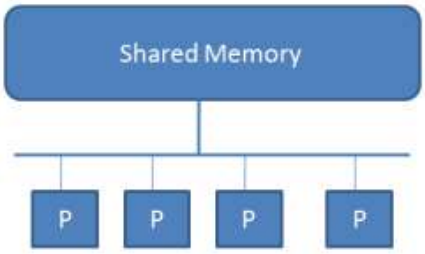


Fig. 4.1: Shared memory architecture [50].

The main aim behind this choice is primarily to minimize the complexity by adding parallel structures. Additionally, it supports incremental parallelism with the ability to parallelize bottlenecks of the application part by part, without changing the

data structures, incrementally locating the loops that have long running times and then parallelizing them [105]. Moreover, since all threads share a common address space, we expect a decrease in the overhead required by the introduced parallelism.

4.2 Graphical Processing Unit (GPU-CUDA)

CUDA, or Compute Unified Device Architecture, is NVIDIA's technology which ensures a significant increase in software performance by using the GPU power [51, 112]. The difference between a CPU and GPU is to compare how they process tasks.

CPU consists of a few cores optimized for sequential serial processing, while the GPU has a massively parallel architecture consisting of thousands of smaller, more efficient cores designed for handling multiple tasks simultaneously [3], presented in Figure 4.2.

Overhead of using GPU instead of CPU, is that the orchestration should be done by CPU. This in turn is an overhead, which is generaly alleviated in programs requiring huge computations which do not have data dependencies.

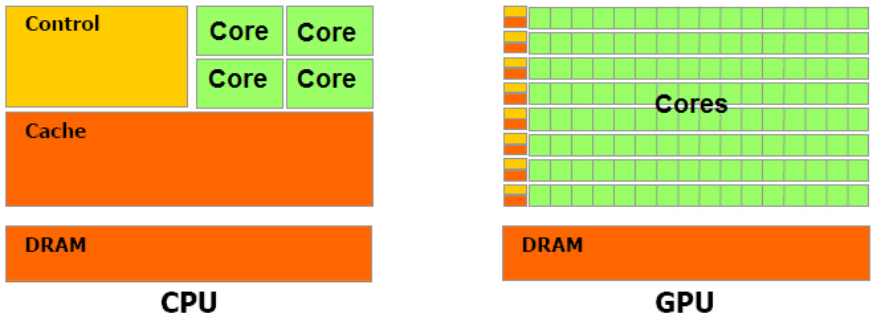


Fig. 4.2: CPU versus GPU-Accelerated processing architecture [2].

4.3 Manycore architectures (Maxeler dataflow)

Manycore architectures are specialized multi-core processors. They can contain thousands of simple and independent cores. These type of architectures can result in higher degree of parallelism.

Maxeler Dataflow [4] provides a very powerful device for this type of computation. It is a completely different computing paradigm compared to the traditional

CPUs. Here, the instructions are parallelized across the available space, rather than time. This is illustrated in Figure 4.3. In each tick, next state of data is passed to dataflow engines. This way, superlinear speedups are possible.

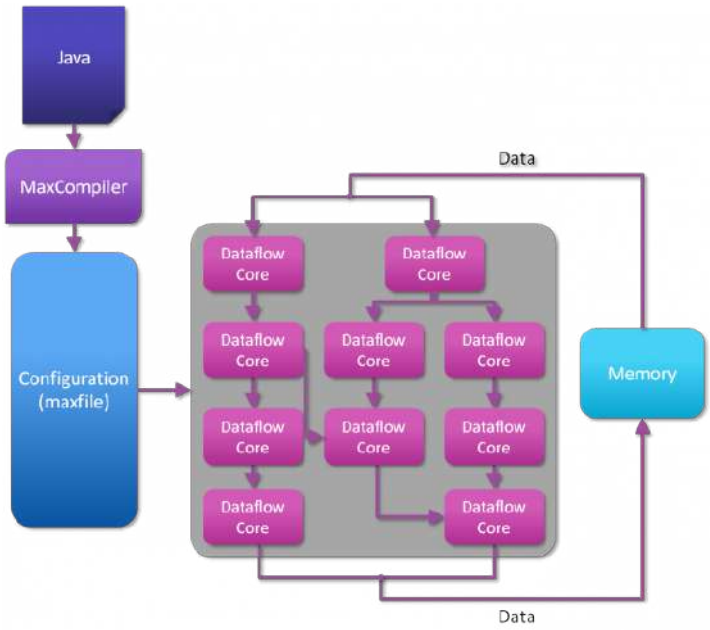


Fig. 4.3: Dataflow processing architecture [4].

Chapter 5
State of the Art of ECG Optimization Algorithms

The content of this Chapter was published at the 14th International Conference on Informatics and Information Technologies [43], 2017, International Conference on Telecommunications [46], 2018, 8th Balkan Conference in Informatics [48], 2017 and 25th Telecommunication Forum (TELFOR) [49], 2017.

5.1 ECG signal processing

Kohler et al. [82] give an overview of existing QRS detection methods. Li et.al [87], Bahoura [21], Shambi [121] and Martinez [93] have reported successful implementations of DWT-based QRS detectors.

Pan and Tompkins, [107], have presented a real time algorithm for ECG QRS detection. Their algorithm considers the slope, amplitude and with information, and adaptively adjusts to the thresholds and parameters. It uses integer arithmetic in order to operate without requiring much computation power. There are no execution times presented, though their analysis is concentrated in the quality, where their correctness rate 99.3 percent.

An efficient implementation of DWT's in Field Programmable Gate Array devices [120]. They have optimised the power consumption and throughput. Additionally, a three level DWT algorithm with 4 Daubechies length filter is presented.

Several studies in literature addressed the delineation concept of ECG signal. Alfaouri and Daqrouq [14] present algorithms which produces better quality output, however no consideration is made for the performance.

Among the most cited DSP based filtering approaches is Afonso's algorthm [10]. His study is based on using filter banks, and the reported results at a 360Hz sampling rate suggest that these types of algorithms can run very fast with promising performances.

Gusev et al. [67] have proposed a pattern matching algorithm in order to match the QRS pattern with default patterns.

An increasing amount of papers address Neural networks. Xue et.al [138] have presented such an approach with excellent performance. QRS detection is based on an Artificial Neural Network (ANN). An adaptive whitening filter is used to filter low frequencies. Whereas the QRS complex is detected with a linear matched filter, which compares the output against the high frequency signal input. high frequency signals are compared against.

Poli et.al [113] presents a solution for calculating a threshold based on genetic algorithms. QRS complexes are detected with a linear and nonlinear polynomial filter, with an applied adaptive local maxima threshold. Parameters are optimized via genertic algorithms and, are successful in decreasing detection errors.

Martinez [92], has proposed a phasor-transform based algorithm for eliminating baseline drift, on a 360Hz sampling frequency. This algorithm converts each sample into a phasor and correctly identifies feature points.

Ajdaraga and Gusev [13] have analyzed how the sampling frequency and resolution impacts ECG signals and their QRS detection. They report that even the rescaled signals obtain good performance when fine tuning the threshold parameters.

5.2 ECG mHealth Solutions

Transformation of healthcare services to solutions for mobile devices is a trending topic.It is a similar trend compared to the concept of IoT, since it enables means of securely sharing information between devices.

In the literature, there are studies concerned with the requirement elicitation for the time-critical mHealth solutions. The requirement analysis in this paper follows the concepts of the Medical Cloud architecture proposed by Tasic et. al. [131].

In their work, Gusev et. al. [68] have focussed on challenges on implementation of a mobile telemedicine application. They have analyzed the issue of Low Power Bluetooth connections, number precision, the transmission, the architecture, design decisions and optimizations.

AbuKhousa et.al. [7] have also focussed on the challenges of mHealth solutions. They have particularly investigated the rising healthcare delivery costs, sharing of information, and the shortage of medical staff. They also give insight about the issues of trust, privacy, security and some technical issues.

A recent research has been done by Patel et. al [110], addressing the issue of developing efficient algorithms. They have concentrated on developing efficient algorithms for detection of arrhythmia with low computational power. These type of algorithms are suitable for mobile devices, and can be used in the initial detection of heart abnormalities.

An Android based mHealth solution is also proposed by Leutheuser et.al [86]. They proposed using three-level hierarchical classification system in order to instantaneously discover any probable problems.

One of the functional requirements was to perform a preprocessing on the noisy ECG signal. In today's mobile technology, devices have multi-core architectures and GPU's. In case they are used efficiently, the process for early detection can be facilitated. Our previous work addresses this issue [42]. By using NVIDIA GPU's we have achieved faster codes with a scalability depending on the number of used cores.

The proof-of-concept study is proposed by Rolim et.al [116] also. Their solution is based on automation of the process for healthcare solution by using wearable sensors and mobile devices. Even though being an old dated, the issues raised by of Martin et. al [91] still exist. They have also proposed high-performance and low power DSP prototype.

The mHealth application must run as a real-time application. Any time critical application has strict rules it must follow. Stonebraker et.al. [128] has provided an outline of eight important requirements that a real-time stream processing application should met.

The scalability of a cloud environment hosting services for ECG signal analysis is presented in [108]. This study is important since it provides the challenges that end-user applications are facing.

The data managed in a cloud solution for Personal Health Records (PHR) is highly sensitive. Patient's private life can be threatened in the case of cyber attacks. That's why the study of Kaletsch et.al [79] is very important, which investigates the privacy issues. Banica and Stefan [22] have also focussed on the security layer of a cloud-based e-Health service.

Security and privacy issue of e-Health cloud architectures is also considered in the study of Ikuomola and Arowolo [75]. They have proposed a Secured e-Health System, whilst the security of the health records is ensured by using Homographic Encryption and bi-layer access control.

Part II
Parallelization of DSP Filtering

Chapter 6
CUDA DSP Filter for ECG Signals

The content of this Chapter was published at the 6th International Conference on Applied Internet and Information Technologies [40], 2016.

This chapter aims to explore whether GPU DSP filter processing of the ECG signals is faster than the CPU processing. This will be tested by measuring the speedup of the obtained GPU solution over the serial solution. Furthermore, our research also aims to determine the optimal size of threads per block to obtain the maximum speedup.

6.1 Parallelization for GPU Computing

The algorithm presented in Figure 8.1 is a sequential version of a convolution of a one dimensional input with a corresponding kernel. The complexity of the algorithm depends on the input and the kernel stream length, i.e $O(nm)$. When run on a CPU, the flow is sequential, meaning that the inner loop length depends on the kernel size.

GPU computing is a rather different computing paradigm when compared to the traditional CPUs. GPUs have a massively parallel architecture consisting of thousands of smaller, more efficient cores designed for handling multiple tasks simultaneously. Using this key feature, we have achieved a row based parallelization of convolution computation for the ECG input signal.

The sequential loop that iterates the input signal is massively parallelized by synchronously utilizing thousands of GPU cores. The proposed solution is visualized in Figure 8.2.

General purpose GPU programing on NVIDIA CUDA enables modifiable execution plans, which are specified by blocks and grids. Block is considered as a group of threads. Its important to note that threads inside the blocks can be synchronized. Threads across same block can communicate via a shared memory dedicated to the block. Besides, a grid is a group of blocks. CUDA does not provide means for synchronization between blocks. Moreover, a blocks inside a grid can communicate via the global memory.

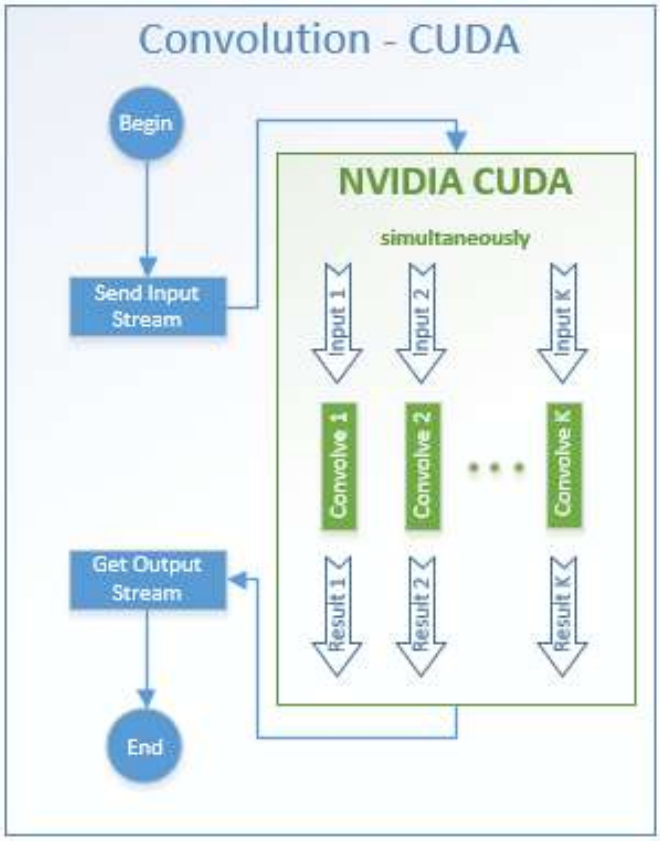


Fig. 6.1: Parallel GPU Computation algorithm.

This row version of the algorithm uses threads which size is proportional to the filter kernel. Whenever the threads are executed on a streaming multiprocessor of a GPU, a number of threads per block (TPB) is defined. We plan to test several values of TPB and determine its optimal size.

Note that this solution does not include any optimization of the CUDA code, and we are aware of a lot of concurrent memory reads. Still, the distribution of the workload on the GPU cores, makes the advantage of using 1536 cores, even the system clock is much lower.

6.2 Experimental results

This section describes the conducted experiments and presents the obtained results.

6.2.1 Testing environment

The sequential and parallelized code are tested on an Amazon EC2 - G2 2xlarge instance. It consists of a 8-core Intel(R) Xeon(R) CPU E5-2670 2.60GHz 64-bit system with a 15GB of memory. Additionally, the parallelized code is tested on a NVIDIA GRID GPU (Kepler GK104) device, having 1,536 CUDA cores with a 800Mhz system clock and 4GB RAM.

In this research we have experimented with the Hamming window and the Blackman window with length of 100 and 200 elements to obtain relatively good results.

6.2.2 Functional Verification

Several experiments are conducted to verify if the functional characteristics of the sequential and parallel algorithms are identical The input was a short stream of 1000 samples of an ECG signal with all characteristic P, Q, R, S and T waves, as presented in Figure 8.3.

We have verified both the sequential and parallelized CUDA and sequential solutions obtain identical results.

6.2.3 Test data

The measured parameters in the experiments are the time required to process the sequential algorithm T_s, and the time required to process the parallel algorithm with p cores, denoted by T_p.

The speed-up is calculated by (10.1) as a ratio of the measured times for execution of the sequential and parallel algorithms.

$$S_P = \frac{T_s}{T_p} \tag{6.1}$$

6.2.4 Results

Note that for each input signal, the speedup values are calculated by using T_s and T_p values for the same input configuration. Hence, the main reason for using these values is to compare the performance of the sequential code on CPU and the parallelized code on CUDA enabled GPU.

A total of 5 experiments were conducted, where input length varies from 10.000 to 500.000 in steps of 10.000 samples. The tested kernel length was 100, 500, 2000, 5000 and 7500.

Table 8.1 presents the running times of the sequential and parallel algorithms tested in our experiments only for selected values.

One can observe that the speedup increases as the input length is increased as it was expected. Moreover, the speedup is increasing slightly as the kernel length increases, due to a more efficient utilization of GPU cores without extra costs for allocating and dealocating a core.

Table 6.1: Speed-up analysis as the input size and kernel increase.

No	Input Size	Kernel Size	CPU Run. Time (ms)	GPU Run. Time(ms)	Sp
1	10000	100	5.3	0.4	12.9
	50000		26	1	27.8
	100000		54	2	30.5
	500000		266	6	42.8
2	10000	500	27	1	40.5
	50000		132	2	60.7
	100000		264	4	66.9
	500000		1323	16	80.9
3	10000	2000	103	1	75.9
	50000		524	6	81.8
	100000		1075	12	91.6
	500000		5288	56	95.1
4	10000	5000	248	3	89.8
	50000		1330	14	93.0
	100000		2649	28	95.7
	500000		13342	133	100.1
5	10000	7500	359	4	91.7
	50000		1955	21	93.7
	100000		3940	41	96.1
	500000		19784	197	100.2

6.2.4.1 Speedup of CUDA GPU vs CPU Solution

The conducted experiments are tested for various input and kernel length. Figure 6.2 presents the speedup of the CUDA algorithm compared to the sequential CPU version, where the input length varies from 10.000 to 500.000 samples and filter kernel lengths of 100, 1.000 and 2.500, with TPB value of 1024.

The algorithm has a steady linear speedup as the ECG signal length increases. Additionally, increasing the kernel length has a noticable effect on the speedup. This is due to the effect of allocating and deallocating a thread, which becomes negligible as kernel length increase. We observe that for some configurations a speedup of up to 100 is achieved, with a scalable nature.

Therefore the hypothesis is confirmed.

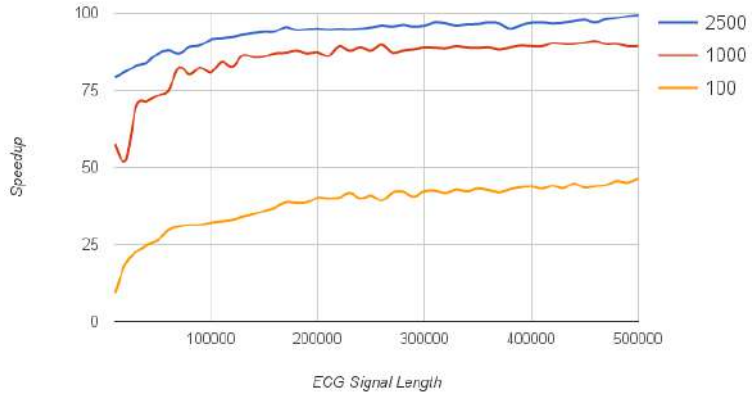


Fig. 6.2: Speedup of parallelized CUDA solution compared to sequential version.

6.2.4.2 Optimal number of Threads Per Block

In the context of our research question we have provided several additional experiments. Selecting right parameters for number of threads in a block has positive effect on the speedup. In order to find the optimal value of the TPB number, we have conducted tests for various TPB from 8 up to 1024. Figure 6.3 presents the speedup values for different TPB values.

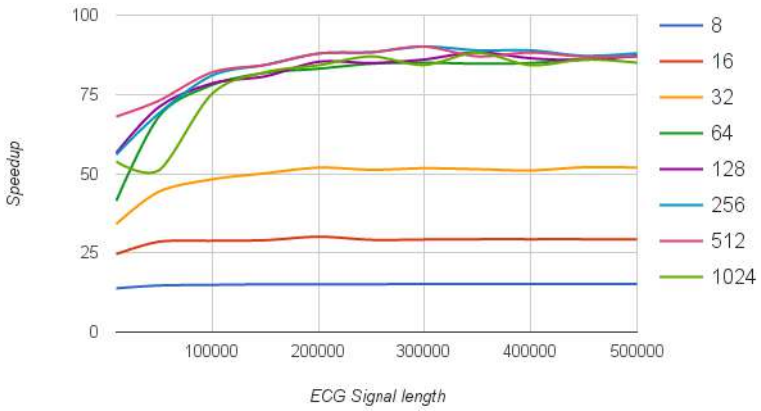


Fig. 6.3: Effect of threads per block on speedup.

Our analysis shows that, TPB should be kept as high as possible by taking into account the upper limit of threads per block being provided by the specification of the GPU.

Chapter 7
Optimizing high-performance CUDA DSP filter for ECG signals

The content of this Chapter was published at the 27th DAAAM International Symposium [42], 2016.

In this chapter, we focus on optimizing the parallel version of the filtering algorithm on graphics processing unit (GPU) cores. The goal is to find an optimized solution that speed ups the parallel CUDA solution.

We especially want to explore whether the utilization of shared and constant memories on GPU can yield faster execution times on ECG signal filtering. To test this we will measure the execution times of naive GPU solution over the optimized solution. We are also interested in determining whether loop unrolling and precision has an effect on the speedup. Moreover, we think it would be worthy to investigate the performance of the element version.

7.1 Identifying CUDA GPU obstacles for high-performance convolution

CUDA allows massive parallelism, which should be utilized in an efficient way. In order to design an optimized code, it is important to optimize memory alignment and thus accesses. Generally, most important performance consideration is the coalescing of the memory accesses. Figure 7.1 illustrates the coalescing concept.

Fig. 7.1: Coalesced access to memory - all threads access one cached line.

Moreover, utilization of shared memory is another performance improvement. Shared memory is located on-chip and generally provides much higher bandwidth and lower latency than the global memory [36]. This is valid especially when there are no bank conflicts. When multiple addresses of a memory request values of the same memory bank, the accesses are serialized, which degrades the performance significantly.

Constant memory is another CUDA improvement that enables a fast access to data. Particularly, consecutive reads of the same address do not incur any additional memory traffic. A single read from constant memory can broadcast to other nearby threads. This will ensure that no bank conflict will occur in the case of access to constant memory.

A high-performance solution will include a combination of the shared memory and constant memory. Since the filter kernel coefficients are not changing over the execution, a natural solution will be to use the shared memory for storing the input signal and constant memory for the filter kernel, as presented in Figure 7.2.

Another obstacle is the bank conflict that may occur while accessing the data in the shared memory. An example of 2-way bank conflict is illustrated in Figure 7.3.

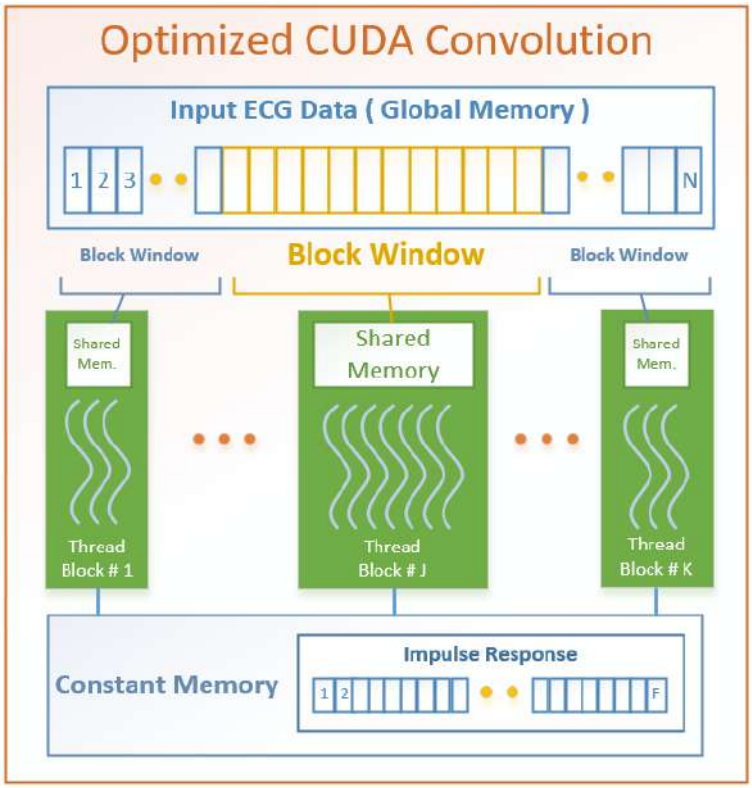


Fig. 7.2: Data flow in a solution that uses both the shared and constant memory.

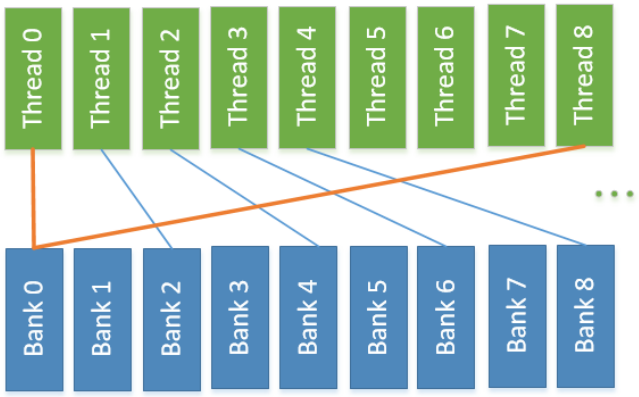


Fig. 7.3: A two-level bank conflict

The solution needs a highly accurate filter and fast processing. Therefore the FIR filter is the best candidate for the filtering. A problem will occur if the filter length is bigger than the maximum number of threads per block to eliminate propagation of partial results and introducing internal synchronization inside a block. Currently, NVIDIA devices have 1024 threads per block. In our case, to process an ECG signal, a FIR filter length of 1000 is acceptable, since it generates 30*db* attenuation with a relatively small ripple passband of 0.1db.

7.2 Optimization approaches

A sequential algorithm for performing convolution is an iterative procedure that repeats the kernel item for each new data element. The complexity of the algorithm *O(nM)* depends on the input n and the kernel stream length M. The flow on CPU is sequential, where inner loop length depends on the kernel size.

In our previous research [40], we have used a naïve parallel CUDA version of the DSP filtering algorithm running on a GPU. Since it did not include any optimization, we have faced lower performance due to lots of concurrent memory reads. In this research, we target all identified bottleneck problems by optimizing the concurrent memory reads, aligning memory accesses and reorganization of computations. The following optimization approaches will be tested:

O1 – using shared memory for the input signal,
O2a– using shared memory for the filter kernel,
O2b– using constant memory for the filter kernel,
O3 – using loop unrolling,
O4 – smaller calculation precision, and
O5 – element version.

7.2.1 Utilizing Shared Memory

The O1 optimization approach utilizes the shared memory for storing relevant segment from the actual ECG signal, although the complete input is expected to be stored on the Global Memory.

Let any thread block consist of *TPB* threads, and the filter length is *F*. Each block will eventually require an input segment of length *TPB* + *F* from the original signal. The first *TPB* threads (within a given thread block) initialize the relevant shared memory. Before each convolution operation, the block threads are synchronized by the intrinsic CUDA synchronization primitive *__syncthreads()* in order to ensure consistent data when initializing the shared memory.

Similarly, the O2a optimization approach utilizes the shared memory for storing corresponding filter kernel coefficients. In this case, O1 and O2a will both occupy the shared memory and therefore, the capacity limitation will decrease the level of performance gain.

7.2.2 Utilizing Constant Memory

The O2b optimization approach, which is complementary to the O2a approach, uses CUDA constant memory to store the read-only filter kernel (weight) coefficients. In this manner, consecutive reads of the same address do not generate any additional memory traffic. It is important to note that no bank conflict happens when using constant memory.

A combination of the O1 and O2b optimization approaches will use shared memory for the input signal segment and constant memory for impulse response coefficients. This avoids the bank conflicts that occur when utilizing only shared memory. Figure 4 presents the idea of this version of the parallel CUDA algorithm.

7.2.3 Loop unrolling

The O3 optimization approach is based on unrolling the loop in the thread. Eventually, this will eliminate instructions that increment the loop index and test if the limit is reached and combine two or more loop bodies into one loop body. The technique decreases the number of realized operations per thread.

Figure 7.4 presents a simplified code segment that loops 100 times. Figure 6a shows the basic method, and Figure 6b a code that uses twice unrolled loop. The procedure decreases the number of overall processed instructions, since instead of 100 checks for the looping condition and 100 instructions to increase x, the program is now performing those instructions only 50 times.

```
for (x=0; x<100; x++)
{
        process(x);
}
```

a) Normal loop

```
for (x=0; x<100; x+=2)
{
        process(x);
        process(x+1);
}
```

b) Unrolling by a factor of 2

Fig. 7.4: A loop unrolling example

7.2.4 Precision decrease

The naive parallel CUDA code used double precision as a base type. On devices of computing capability 2.x, each bank has a bandwidth of 32 bits every two clock cycles, and successive 32- bit words are assigned to successive banks [5]. The warp size is 32 threads accessing 32 locations in a memory bank, each with a 32-bit precision. When double precision is used, in the case of accessing 64-bit numbers, the total number of memory locations is 16. Thus, a simultaneous access of 32 threads is not possible in a warp to 32 words of 64-bit length will cause, and only a half warp can be executed.

In order to increase the performance, the O3 approach decreases the precision to 32-bit single precision mode. In such case bank capacity limitation will be eliminated since 32 threads in a warp will simultaneously access 32 words of 32-bit length. Devices with computing capability 3.x allow a bandwidth of 64 bits every clock cycle [3]. Thus, bank capacity limitation arising from using double precision will be eliminated for devices with computing capability 3.x or higher. In addition, processing a single precision instruction is faster than the double precision.

7.2.5 Element version

Moreover, the element version of convolution is designed and implemented for ECG signal. In this version, instead of rows, a thread computes multiplication and summation of elements. Threads are defined for each combination of input and kernel items. Summation, on the other hand, is performed in logarithmic time. However, this approach leads to use of a lot of synchronization, which can cause a performance decrease.

7.3 Experimental Methodology

This section describes the conducted experiments.

7.3.1 Testing Environment

The sequential and parallelized codes are tested on an Amazon EC2 - G2 2xlarge instance. It consists of an 8-core Intel(R) Xeon(R) CPU E5- 2670 2.60GHz 64-bit system with a 15GB of memory. Moreover, the parallelized code is tested on a NVIDIA GRID GPU (Kepler GK104) device, having 1,536 CUDA cores with an 800Mhz system clock and 4GB RAM. The total amount of constant memory is 64KB where each block has 48KB of shared memory. This device has a computing capability 3.x, and input data is 64-bit double precision.

7.3.2 Experiments and test cases

The experiments were defined by testing the each previously described optimization approach independently and in a combination with the other approaches. Each experiment had several test runs for various sizes of the input data stream that consist of 10.000 up to 500.000 samples of an ECG signal with incremental steps of 10.000 samples.

Since the sampling frequency is 500Hz, the input data stream present an ECG signal from 2 seconds up to 1000 seconds. The filter kernels tested consisted of M=100, 250, 500, 750 and 1000 elements.

Each test run for the experiments was tested at least five times and an average value of measured times was calculated and used for further processing. A functional verification was conducted for each test run to verify if the functional characteristics of the naïve and optimization parallel algorithm executions obtain identical results.

7.3.3 Test Data

The measured parameters in the experiments are T_{pn} as time required by executing the naïve parallel algorithm using n cores, and T_{on} as time required by the given experiment with an appropriate combination of optimization approaches. The speed-up is calculated by Equation 10.1 as a ratio of the measured times for execution of the sequential and parallel algorithms.

$$S_n = \frac{T_{pn}}{T_{on}} \tag{7.1}$$

7.4 Performance analysis

This section describes the conducted experiments and presents the obtained results.

7.4.1 Shared Memory

Figure 7.5 shows the speedup of the O1 optimization using the shared memory for the input signal. The optimized version is more efficient, and the speedup increases with the filter kernel size. An average value of 13% increase is obtained as speedup for the filter length of 100, and 78% for filter length of 1000. We observe that for each filter size, the speedup $S_n > 1$ and the greater speedup is achieved when increasing the filter length, which proves the scalability of the improvement.

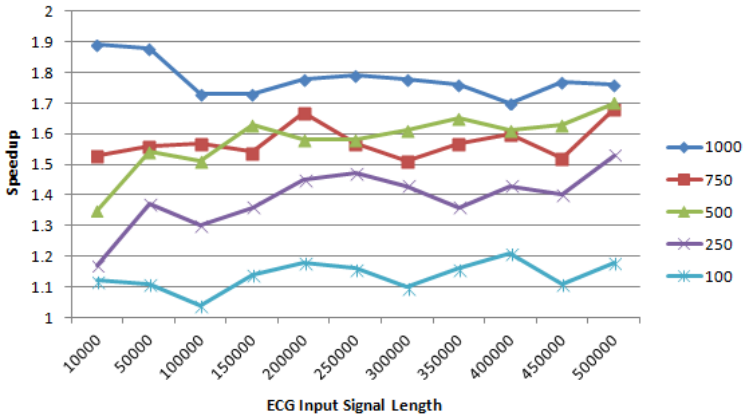


Fig. 7.5: Speedup of the O1 optimization using shared memory for the input signal in the case of different kernel sizes.

Additionally, Figure 7.6 shows the speedup of O2a optimization only using the shared memory for filter kernel. It can be noted code is optimized by roughly 10% and that speedup increases with the increasing filter kernel length. The higher the filter kernel is, the better performance gain is obtained. Better results are obtained for the usage of shared memory for storing the input signals, instead of the filter coefficients.

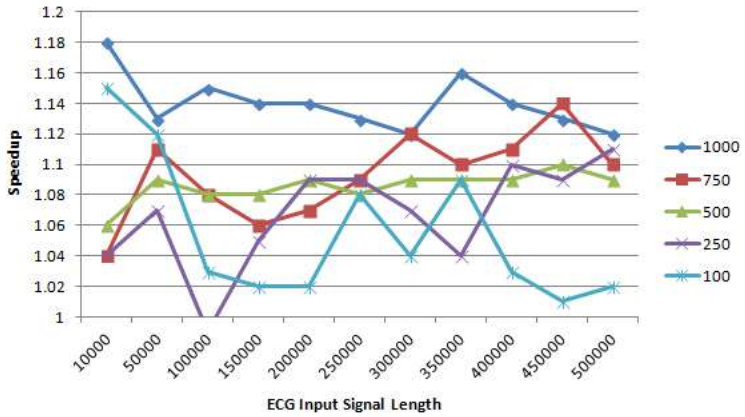


Fig. 7.6: Speedup of the O2a optimization using shared memory for different kernel sizes.

7.4.2 Constant Memory

Figure 7.7 presents the analysis made on the constant memory defined by the O2b optimization approach. One can conclude that using a constant memory for the kernel will increase the performance by average 15%. We have compared the O2a and O2b approaches, as presented in Figure 7.8. The solution using constant memory instead of shared memory for storing the filter kernels obtains a higher performance by an average speedup of 4%.

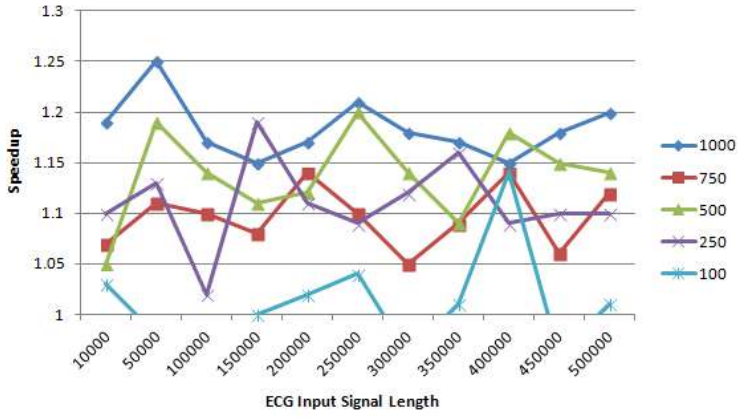


Fig. 7.7: Speedup of the O2b optimization approach using Constant Memory for the filter kernel.

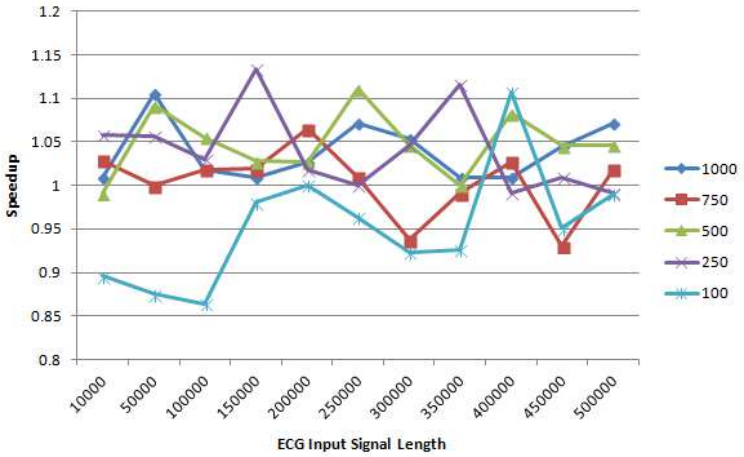


Fig. 7.8: Speedup of the O2b optimization approach compared to the O2a.

7.4.3 Loop Unrolling

The speedup obtained by loop unrolling O3 optimization approach is presented in Figure 7.9. On average, the loop unrolling optimization technique increases the performance for 1 to 5 percent. The best performance gain was obtained for the unrolling length of 64.

7.4.4 Decreasing the double to single precision

The conducted experiments generally use a 64-bit double precision for storing and calculating the convolution. However, a GPU double precision calculation is a costly operation [136]. We have tested the performance by decreasing the precision to 32-bit, since the functional testing showed difference in both approaches expressed in order of 10^{-5}. Figure 7.10 presents the effect of the decreased precision on the performance. As input size increases, the single precision version is approximately 40 to 60 percent faster than the pure parallel code.

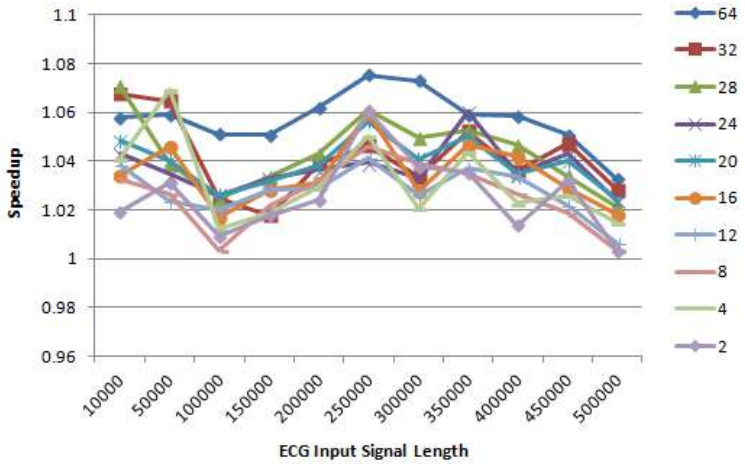


Fig. 7.9: Speedup of O3 optimization approach using loop unrolling for filter length of 1000.

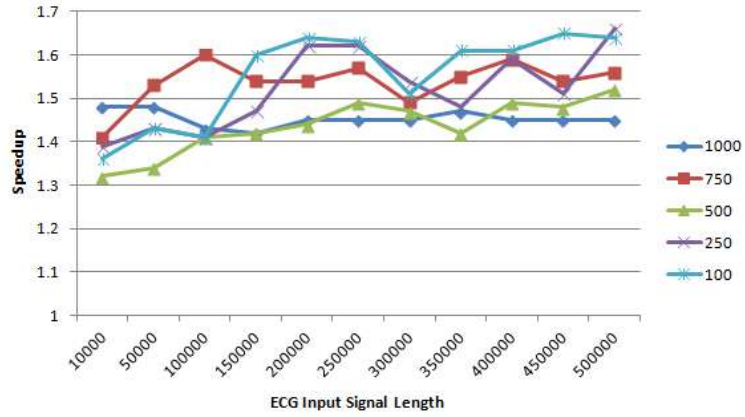


Fig. 7.10: Speedup obtained by the O4 optimization approach by decreasing the precision.

7.4.5 Element version

The conducted experiments show that the element version actually decreases the performance. This happens due to the synchronization and bank conflicts that need to be performed in each step.

Chapter 8
Dataflow DSP Filter for ECG Signals

The content of this Chapter was published at the 13th International Conference on Informatics and Information Technologies [41], 2016.

In this chapter, we focus on parallelizing the sequential DSP filter for processing of the heart signals on dataflow cores. The DSP filter is used for preprocessing of the ECG data, in order to eliminate noise from the ECG signal. Based on the noise components of the ECG signal, several filtering methods are available, such as Low pass, High bass and Bandpass filter.

Dataflow Computing is a completely different paradigm of computing than traditional CPUs. Instructions are parallelized across the available space, rather than time. It is a revolutionary way for High Performance Computing (HPC) solutions[122, 59]. Data streams are optimized by utilizing thousands of dataflow cores, providing order of magnitude speedups. Maxeler systems are used for dataflow computing [4]. The performance of the parallelized code is compared to that of the sequential code. Our analysis shows speedups linear to the kernel size of the filter.

8.1 Parallelization for Dataflow Computing

Algorithm 1 presents the sequential version of convolution of a one dimensional input with a kernel. The complexity of the algorithm depends on the input and kernel stream length, i.e $O(nm)$. When run on a CPU, the flow is sequential, meaning that the inner loop length depends on the kernel size. This flow is visualized in Figure 8.1.

In CPU computing, iterations are parallelized across the available time, and performed sequentially.

Dataflow is a completely different computing paradigm compared to the traditional CPUs. Here, the instructions are parallelized across the available space, rather than time. Using this key feature, we have achieved parallelization of kernel computing via Dataflow cores. In this manner, iterative kernel computation is massively parallelized. Depending on the kernel size, up to thousands of dataflow cores can be

Algorithm 1 Filtering algorithm

```
1:  procedure CONVOLUTION(in, kernel, out)
2:      i ← 0
3:      while i < inputSize do
4:          sum ← 0
5:          j ← 0
6:          while j < kernelSize do
7:              sum ← sum + in[i − j] ∗ kernel[j]
8:              j ← j + 1
9:          out[i] ← sum
10:         i ← i + 1
11:     return out
```

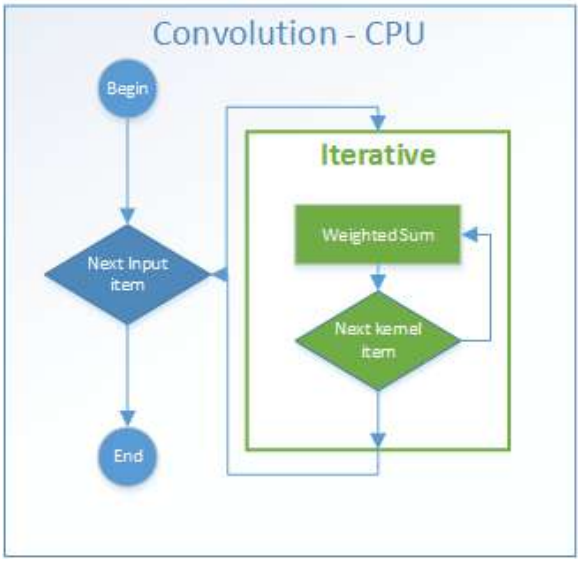


Fig. 8.1: Flow of sequential filtering algorithm.

utilized synchronously, providing a speedup with a higher order of magnitude. The proposed solution is visualized in Figure 8.2.

8.2 Tests and Results

The sequential code is tested on an 8-core Intel(R) Xeon(R) X5647, 2.93 Ghz system with 12GB of memory. On the other hand, parallelized code is tested on a Maxeler simulator. Five different kernel sizes are tested, and vice versa, for various length ECG input signals.

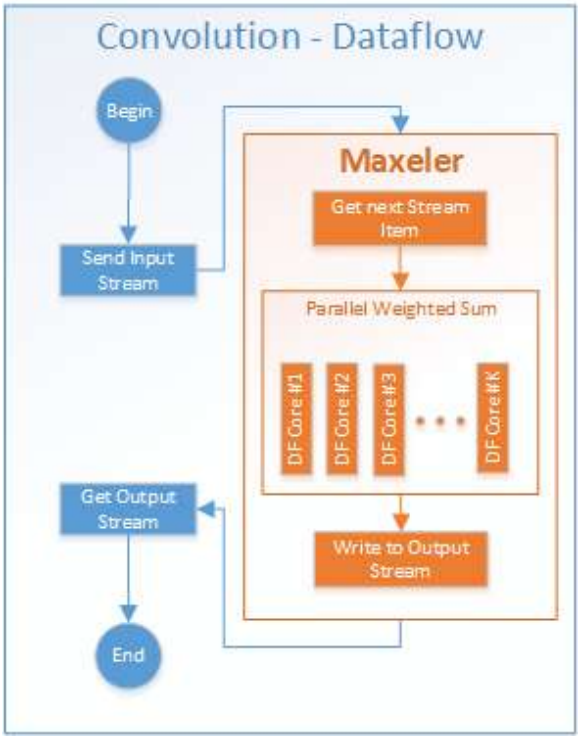


Fig. 8.2: Parallel Dataflow Computation algorithm.

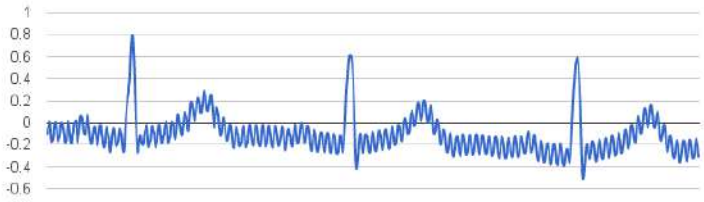


Fig. 8.3: A segment of an ECG signal with several QRS complexes.

8.2.1 Functional Verification

To verify the functional characteristics of the execution of the sequential and parallel algorithms we have provided several experiments. The input was a short sequence of 500 samples of an ECG signal with all characteristic P, Q, R, S and T waves, as presented in Figure 8.3.

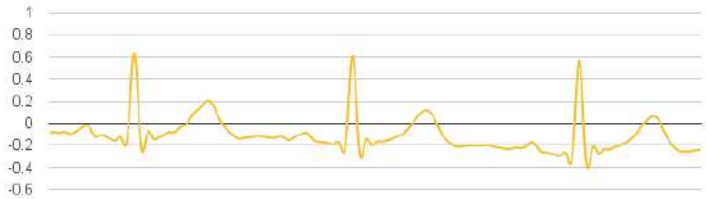


Fig. 8.4: The ECG signal filtered with a low pass filter of 30Hz.

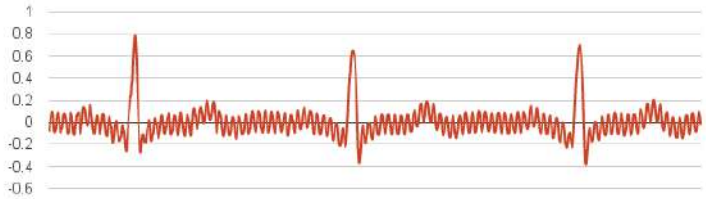


Fig. 8.5: The ECG signal filtered with a high pass filter of 0.5Hz.

Figure 8.4 presents the effect of applying a low pass filter on the ECG signal. One can notice that the 50Hz noise is eliminated.

The effect of the high pass filter is presented in Figure 8.5. The effect of this filter is elimination of the baseline drift, caused by breathing and other physical movements.

The effect of the band pass filter as a combination of a low pass and high pass filter is presented in Figure 8.6. This filter eliminates the baseline drift, caused by breathing and other physical movements and also the 50Hz noise caused by the electricity.

We have verified both the sequential and parallelized solution and obtain identical results.

8.2.2 Speed-up Analysis

Speed-up is calculated by (10.1), where T_s is the time required to process the sequential algorithm, and T_p is the time required to process the parallel algorithm with p cores. Since the system clock on the sequential machine is much higher than the system clock on the parallel machine, we will compare the number of sequential

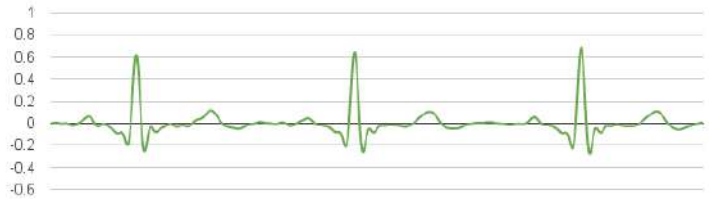


Fig. 8.6: The ECG signal filtered with a band pass filter between 0.5Hz and 30Hz.

steps N_s (operations required by the sequential algorithm) and the number of processing steps N_p (operations required by the parallel algorithm), by considering the sequential system clock C_s and parallel device's system clock C_p.

$$S_P = \frac{T_s}{T_p} = \frac{N_s}{N_p}\frac{C_p}{C_s} \tag{8.1}$$

Note that for each input signal, the speedup values are calculated by using T_s and T_P values for the same input configuration. Hence, the main reason for using these values is to compare the performance of the sequential code on CPU and the parallelized code on Maxeler Dataflow Engine (DFE).

Sequential running code has mainly two phases: the initialization and processing phases. Let the input stream contains N elements and the filter kernel M elements. On initialization, the input stream and filter kernel are transfered from the memory to the CPU by a total of $N+M$ memory access operations and the N output elements are written in the memory. Processing phase requires $N*M$ multiplications and $N*M$ additions. Assuming that each memory access, multiplication and addition requires 1 processing step, the relation that shows the total number of processing steps is presented in (8.2).

$$N_s = 2NM + 2N + M \tag{8.2}$$

The number of operations for the parallel algorithm is calculated differently. In addition to processing, the dataflow engine needs to transfer data from memory to device and return the results back, which is equal to a total of $N+N+M$ memory access operations for the input and output stream, and the filter kernel. The dataflow engine performs operations on N samples concurrently in a pipelined manner, so the processing takes N processing steps plus the pipeline length of the number of operations, which is equal to the kernel length M. Note that the summation can be realized in a tree parallel organization, which will take only $log_2 M$ steps, but since we expect that $N >> M$ it will not affect the final result. So the total processing steps is expressed by (8.3).

$$N_p = 2N + M + N + M = 3N + 2M \tag{8.3}$$

Table 8.1 presents the number of operations and calculated speedup for our experiments where the input contains 100.000 samples and kernel length was 100, 500, 2000, 5000 and 7500. The sequential machine clock was 2.93 GHz and the dataflow Maxeler device system clock 400 Mhz. The speedup increases with the length of the kernel size.

Table 8.1: Speed-up analysis as the kernel size increase.

No.	**Platform**	**Input Size**	**Kernel Size**	**Num. of Opers.**	S_P
1	CPU	100000	100	20200100	9.19
	DFE			301000	
2	CPU	100000	500	100200500	45.45
	DFE			301000	
3	CPU	100000	2000	400202000	179.72
	DFE			304000	
4	CPU	100000	5000	1000205000	440.47
	DFE			310000	
5	CPU	100000	7500	1500207500	650.18
	DFE			315000	

Chapter 9
Parallelization of Digital Wavelet Transformation of ECG Signals

The content of this Chapter was published at the 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), IEEE [44], 2017.

Wavelet Transformation is being used in many signal processing applications. It has been successful in the area of signal compression, data compression and detection of ECG characteristics. It mainly generates time-scale representation of an ECG signal, thus making it possible to accurately extract features from a non-stationary ECG signal [117, 14].

Wavelet Transformation is basically a linear operation, which decomposes a signal into various scales according to their frequency components. Each of these scales is further analyzed with a predefined resolution [9].

Wavelet Transform of a continuous signal is by definition a sum of the signal multiplied by scaled and shifted versions of the wavelet function, used to divide a continuous-time function into wavelets with the ability to construct a time-frequency representation of the signal. In many practical applications, though, Discrete Wavelet Transformation (DWT) is sufficient, as it provides only the vital information of the signal in a significantly faster manner.

This chapter aims at optimizing the sequential Discrete Wavelet Transform (DWT) used for DSP filtering and feature extraction. DWT is a highly dependent structure with numerous dependencies between data. We set a hypothesis that optimizing the DWT initialization and processing parts can yield a faster code. Our analysis shows that proposed optimization techniques provide faster code.

9.1 DWT Algorithm analysis

The DWT algorithm contains two phases determined as initialization and processing phases, as presented in Fig. 9.1. Table 9.1 presents the variables used in the algorithm.

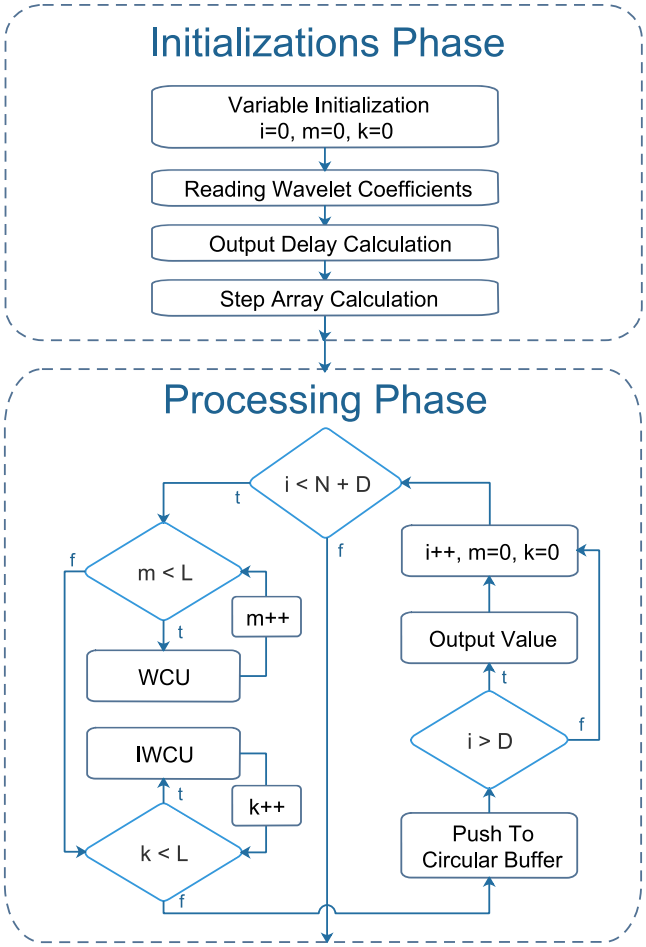


Fig. 9.1: A high-level abstraction of the DWT algorithm.

The *initialization* phase creates the base context for the processing part of the DWT. It is formed of 4 stages, which can be listed as Variable Initializations, Reading Wavelet Coefficients, Output Delay Calculation and Step Array Calculation.

Variables are declared and initialized in the first stage, whereas wavelet coefficients are read on the second. In wavelet transformation algorithms, there is a delay for the first output. This is calculated in stage 3. Finally, a vital *step* matrix for the wavelet computation is calculated.

On the other hand, the *processing* phase is responsible for the transformation itself. For each input element, the Wavelet Compute and Update (WCU) module realizes the decomposition of the ECG signal into signal approximation and detailed information.

Table 9.1: Variables used in the DWT algorithm

variables	meaning
i, m, k, K, and P	loop index variables
t and f	boolean evaluation , i.e true and false
D	delay for the first output
L	number of Wavelet levels
N	data array input size (ECG data samples)
F	filter length
WCU	wavelet compute and update
$IWCU$	inverse wavelet compute and update

The Inverse Wavelet Compute and Update (IWCU) removes the low-frequency components and regenerates the signal. The output of IWCU is pushed to a circular buffer of length D. The first baseline drift eliminated signal is actually produced after D steps [19].

Let L be the number of wavelet levels to compute. Assuming an input ECG signal using a 500 Hz sampling frequency, the highest frequency component that exists in the signal is 250 Hz (each DWT level processing divides the band in two parts). Since the ECG baseline drift removal needs a high pass filter of 0.5 Hz then the DWT algorithm requires $L = 9$ wavelet levels.

Our research is concentrated on the parallelization of DWT algorithm. Analysis shows that DWT has highly dependent structure. This in turn makes direct parallelization inconvenient. In this section, we provide the dependency analysis and propose an efficient solution for parallelizing DWT algorithm.

Profiling the code showed that execution time is mostly spent in two segments of the code. These are the *Step Array Calculation* stage and the *Processing* phase.

Algorithm 2 Step Initialization and Calculation Algoritm

```
1:  P,K ← 0
2:  while P < L do                                   ▷ Initialization
3:      while K < 2^L do
4:          Step[P][K] ← 0
5:  P ← 0
6:  while P < L do                                   ▷ Calculation
7:      M,K ← 0
8:      while K < P do                               ▷ Calculate M
9:          M ← M * 2
10:     while K < 2^L do
11:         Step[P][K] ← 0
12:         if K%M == 0 then
13:             Step[P][K] ← Step[P][K] + 1
14:         if K%(M * 2) == 0 then
15:             Step[P][K] ← Step[P][K] + 1
```

Algorithm 2 shows the operations executed for the *Step Array Calculation* stage. At a glance, it is seen that the complexity of this stage is $O(L*2^L)$. Especially on high Wavelet levels, this part becomes a serious bottleneck.

The operations start with initializing the two-dimensional $step$ array, of length L and 2^L. A 2-level loop is used, though the second level executes 2^L iterations. Once initialized, the calculations are performed on the second loop, requiring 2^L iterations for each input.

In the *Processing* phase, a loop iterates $N+D$ times. Calculation of the delay D is presented in Algorithm 3. We can conclude that D is proportional to C_L*2^L, where C_L is a constant number depending on level L. As the number of levels increases significantly, the constant C_L and N can be neglected resulting in an algorithmic complexity of $O(L*2^L)$.

Algorithm 3 Delay Calculation

```
1: D,P ← 0
2: while P < L do
3:     D ← 2*D + (F − 1)
```

From the high-level algorithm presented in Fig. 9.1, we observe that in each iteration WCU and IWCU iterate L times. Algorithm 4 presents the inner structure of the WCU. Operations start from the first level and repeatedly execute until the last level. The input to this module is the data value computed as result of the previous WCU. WCU actively performs operations with the values of the dynamic wavelet filter stored in the filter buffer.

Algorithm 4 Wavelet Compute and Update operations

```
1: Circular[Tail] ← Previous
2: Tail ← Tail − 1
3: Next,P ← 0
4: while P < L do
5:     Next ← Circular[P] * Coefficients[P]
```

Each WCU operation starts with pushing the previously calculated value by a preceding WCU to the circular buffer. The next step is to rearrange the tail pointer of the circular buffer. Finally, the coefficients are convolved with the buffer. The output of this operation is used as input to the next WCU. WCU is performed in a sequence for all wavelet levels L.

IWCU practically contains same operations except that it uses $Step$ array as an indicator whether the operations will be performed or bypassed in the current iteration. If $Step$ array is 1, then the previously computed value by a preceding IWCU is updated to 0.

From the profiled code we observed that WCU part is the most important bottleneck having a highly dependent nature.

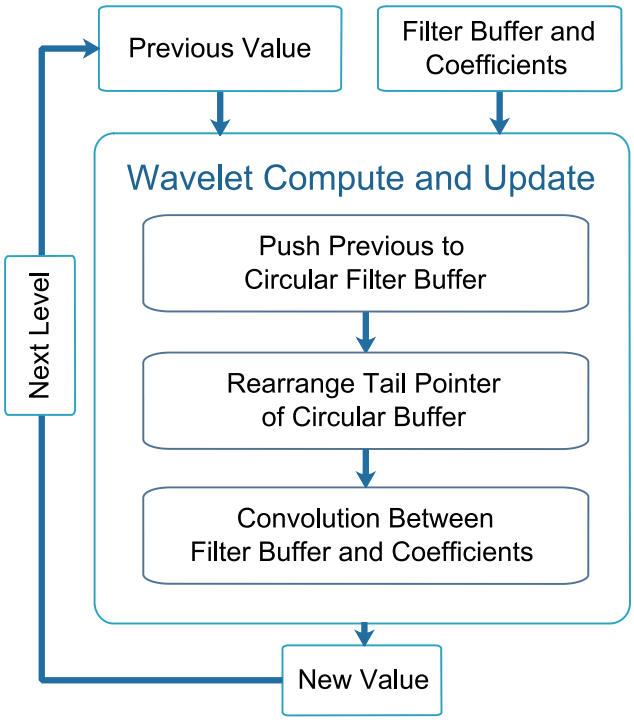


Fig. 9.2: A high-level view of Wavelet Compute and Update.

9.2 Dependency Analysis and Parallelization

The previous section presented algorithmic details, especially for the main bottlenecks which are *Step Array Calculation* stage and the *Processing* phase. This section discusses the data dependence between loop iterations.

Starting from the *Step Array Calculation* stage, it is observed that iterations are independent, which is important for efficient parallelism. On the other hand, when we analyze the *Processing* phase, we see that it has a highly dependent structure, especially due to the fact that current input to the WCU or IWCU, depends on the output of preceding WCU or ICWU computation.

To realize a visual presentation of the data dependence we will use that $A \rightarrow B$ means B depends on A. Fig. 9.3 shows the data dependency of the *Processing* phase implementation and gives an initial idea about the sequence of computations that need to be processed. It also gives an idea how to arrange computations in a parallel environment exploiting concurrent computations.

Each of the nodes presented in Fig. 9.3 stands for both WCU and IWCU operations. The result of WCU computing the input N and level L is basically the *detailed*

approximation of the signal at L'th decomposition level. This information is then transferred to the next $L+1$'th level of the N'th input signal for further processing.

The computations in each node realize a WCU operation, that is composed of several steps. The first step inserts previously computed data sample. The second step is to increment the tail of the circular buffer and lastly convolve coefficients with the buffer, where the output of the convolution is passed as input to the next WCU.

For a single input element, the detailed approximation for decomposition level L starts by computing the approximation at first level. Detailed approximation at any level is computed by performing a convolution with an orthonormal wavelet basis. This information then is transferred to the next node, which is the right node in Fig. 9.3. In this manner, by repeatedly computing and passing the detailed approximation to the right node, next level detailed approximation can be computed.

Once the level L is reached, the same procedure is applied to the next input element which is at the bottom of the starting node. In this manner, the algorithm flows from right to left when going at higher decomposition levels, and top to bottom when computing the next input elements. One can observe that the algorithm has highly dependent nature.

On the first sight, it is observable that dependency prevents direct parallelization. However, several methods can rearrange the presented structure and parallelize the execution. One possible way rearranges the nodes, such that the calculation of values for a certain node assumes that previous (left and upper) nodes are already calculated. We use the Pipeline-Parallel-Processing methodology for parallelizing DWT.

Fig. 9.4 shows the organization and flow of computations in the existing data flow arrangement of the nodes. One can observe that computation waves can flow with 45 degrees to the axes. Each wave contains independent computations and can be executed simultaneously at a given time stamp. This ensures that previous nodes (found on the left) are already calculated. Due to this pipelined structure, the first output will be ready after L iterations, which is 9 in this case.

When iteration size is relatively bigger, the overhead due to the opening and closing phases of the pipeline can be neglected. Next section will outline the implementation strategies of the following algorithm to different platforms.

In this paper, we use OpenMP library to implement the parallel algorithm. The *Step Array Calculation* stage does not have dependencies between loops, thus iterations are directly parallelized by OpenMP loop directives.

On the other hand, for the *Processing* phase, each node in Fig. 9.3 computes the WCU, by performing a set of complex operations. The idea behind is to allocate a separate thread to each of the nodes. The main consideration would be to order the execution of threads in the right manner, with the aim to produce a correct output.

The nodes (representing computations) found on the wavefront can be processed independently when the previous nodes (on the left) are calculated and results are transmitted to the neighbor. These threads are synchronized once they finish their execution, and continued to the next iteration.

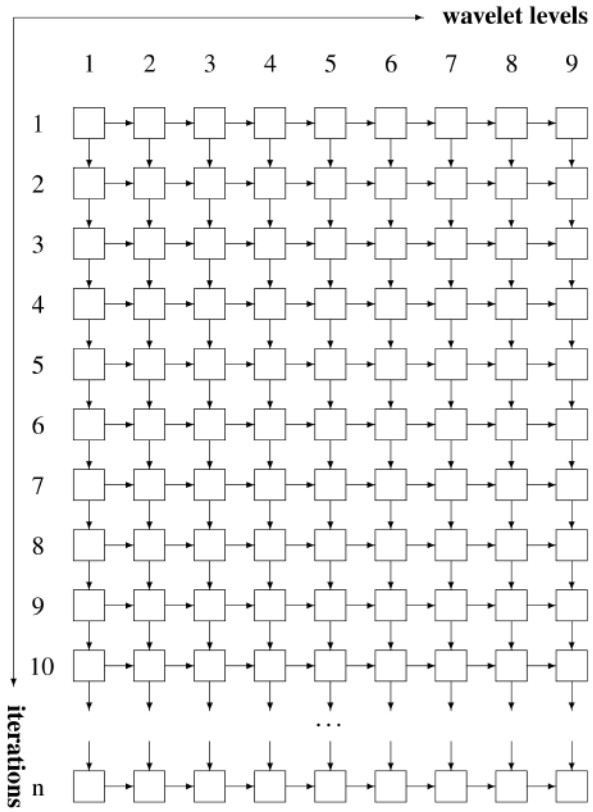


Fig. 9.3: Dependency analysis of the Sequential DWT code. Each node presents a WCU or IWCU operation.

Our analysis is based on using cores more than the maximum number of algorithm nodes that can be simultaneously executed. This is critical in order to eliminate delays. Let's consider a reverse case of having less cores than needed, such as 4 cores and decomposition level 9. Fig. 9.4 shows the numbered WCU nodes. In the first time step $t = 1$, only one thread will execute the node number 1 and other threads will be idle. The next time step $t = 2$ addresses execution of 2 threads (nodes 2 and 10). This is followed by $t = 3$, where three cores will execute the code (nodes 3, 11, 19) and only one core will be idle. Starting with the fourth time step ($t = 4$) the cores will fully execute the algorithm without idle moments. However, in the next timestamp ($t = 5$) 5 nodes should be executed simultaneously and there are however there are only 4 available cores. Thus, this will require 2 cycles in order to complete, such that nodes 5, 13, 21, 29 will be executed in one cycle simultaneously, and, only the node 37 will execute in the latter cycle, while the other cores will be idle. This

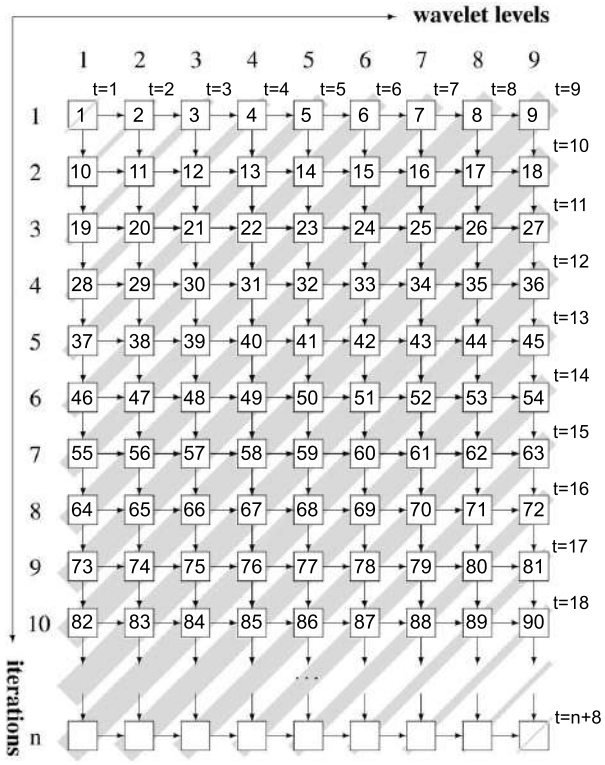


Fig. 9.4: Simultaneous Execution of nodes on the DWT code for 9 decomposition levels.

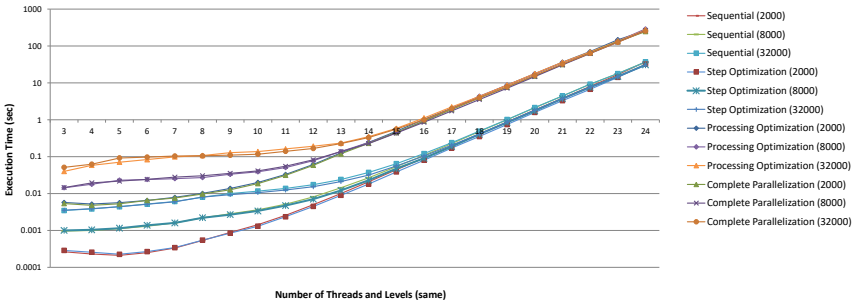


Fig. 9.5: Execution times of running the three proposed optimizations, with input size of 2000, 8000 and 32000. Number of cores are identical to number of Wavelet Levels. Values are presented on a logarithmic scale of base 10.

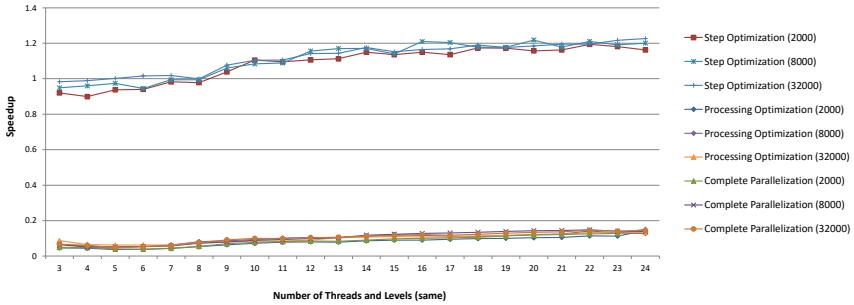


Fig. 9.6: Speedup of three proposed optimizations, with input size of 2000, 8000 and 32000. Number of cores are identical to number of Wavelet Levels.

will increase the delay, and decrease the performance of the application seriously. The best performance for decomposition level of L is to use at least L cores.

By parallelizing the both bottlenecks, theoretically, the algorithm can achieve a speedup of L on L cores. The next sections present experimental research of the proposed optimizations.

9.3 Testing methodology

Denote the response time required to process the sequential algorithm be denoted by T_s, and the response time required to process the parallel algorithm with p cores, be denoted by T_p. Then, the speedup is defined as the ratio of the execution times by (10.1).

$$S_P = \frac{T_s}{T_p} \tag{9.1}$$

The sequential and parallel code are tested on an Amazon C3 c3.8xlarge instance. It consists of a high-frequency Intel Xeon E5-2680 v2 (Ivy Bridge) Processor with 32 cores, 60GB of memory. The performance of the code is tested for various wavelet levels.

OpenMP library is used for shared memory parallelism [105] without prebuilt optimizations from the OpenMP library. The static scheduling method is used to evaluate the sequential results.

Three optimization approaches are tested:

O1: Parallelization of *Step Array Calculation*
O2: Parallelization of *Processing*
O3: Complete Parallelization

The experiments were defined by testing the each previously described optimization approaches in a combination with the other approaches. Each experiment had several test runs for various sizes of the input data stream that consist of 2.000, 8.000 and 32.000 samples of an ECG signal. The tested wavelet levels consists of 3 up to 24, with incremental steps of 1 level. Daubechies filters with length are used in the experiment.

Although being theoretically possible, it is difficult to calculate behind 24 wavelet levels practically, due to memory and architectural constraints.

Each test run for the experiments was tested at least five times and an average value of measured times was calculated and used for further processing. Moreover, functional verification was conducted to verify the functional characteristics of the sequential and optimization parallel algorithm executions obtain identical results.

Chapter 10
Optimal Parallel Wavelet ECG Signal Processing

The content of this Chapter was published at the 14th International Conference on Informatics and Information Technologies [43], 2017.

In this chapter, we investigate the dependence between the nodes in the DWT implementation (and therefore to their corresponding threads) and the available number of cores that can execute the code. This analysis leads to valuable conclusions that will allow construction of even better optimizations. We give a detailed analysis and also realize experimental testing to analyze the practical implementations. Evaluation of the results are compared with the results of previously parallel code [45].

10.1 Discrete Wavelet Transform Analysis

Our analysis on the previous study [45], showed that DWT algorithm contains two bottlenecks, exposed in the *Initialization* and the *Processing* phase.

Observation is that the former phase does not include data dependencies between iterations. This was a vital information for pure parallelization. Though, the latter phase is highly dependent, preventing direct parallelization. This is presented in Fig. 9.4 where data dependence is vizualised as A $\rightarrow$ B, with the meaning *B depends on A*.

10.2 Previous Parallel Algorithm

Our previous parallel algorithm [45] was based on optimizing both of the bottlenecks. The *Initialization* phase was parallelized by a straightforward approach. Nevertheless, high data dependency on the *Processing* phase required re-arrangement on the nodes for a concurrent computation.

Computation waves can flow with 45 degrees to axes where each wave contains independent computations and can be executed simultaneously at a given time stamp. This ensures that previous nodes (found on the left) are already calculated. Due to this pipelined structure, the first output will be ready after L iterations, where L is the number of wavelet levels.

The proposed implementation requires that each block of independent nodes to be synchronized between iterations. However, this is a costly operation and prevents theoretical speedup of L, when executed on L cores.

Next section gives further optimization strategies, in order to achieve the best efficiency through the parallel algorithm.

10.3 Optimization Approaches

The methodology for testing the parallel algorithm on the previous study [45] was based on executing both the bottlenecks on the same number of cores.

The algorithmic and storage complexity of the DWT is $O(L * 2^L)$, making it nearly hard to increase the Wavelet levels.

One interesting approach is to keep the core numbers for *Initialization* phase high. This would increase the efficiency, simply because the data independent iterations.

In the *Processing* phase the maximum available nodes that can concurrently be processed is restricted to the number of wavelet levels. Thus, increasing the core numbers, will only increase the number of idle cores. However, executing this region with less number of cores can decrease the burden of barrier synchronization.

Our previous work did not address the effect of filter length. Theoretically, increasing the filter length will directly increase the percentage of processing compared to the percentage required to synchronize iterations.

Moreover, OpenMP provides built in optimization strategies [76]. Previous study did not considered using them. It would be interesting to test their effect on the barrier synchronization.

10.4 Testing Methodology

Let the response time required to process the parallel algorithm be denoted by T_p, and the response time required to process the optimized parallel algorithm with p cores, be denoted by T_{op}. Then, the speedup is defined as the ratio of the execution times by (10.1).

$$S_{OP} = \frac{T_p}{T_{op}} \tag{10.1}$$

The proposed optimization approaches are tested on an Amazon C3 c3.8xlarge instance. It consists of a high-frequency Intel Xeon E5-2680 v2 (Ivy Bridge) Processor with 32 cores, 60GB of memory. OpenMP library is used for testing the proposed optimizations.

The following optimisation approaches will be tested:

OA1: Using more core numbers for Initialization phase.
OA2: Using less core numbers for Processing phase.
OA3: Increasing the filter length.
OA4: Using compiler optimizations.
OA5: Combined Effect.

On the previous study, we observed the effect of input size is negligible as the wavelet levels increase. Due to this, the input size will be fixated to 10.000 sample length ECG signal. Throughout the tests, wavelet levels vary from 3 up to 24, with incremental steps of 1 level.

On the test environment, the maximum number of available cores is 32. Considering this, the test configuration is presented in Table 16.1.

Table 10.1: Test Environment Setup

Optimization Approach	**Description of The Testing Methodology**
OA1	Core numbers from 2 to 30, incremental steps of 2
OA2	Core numbers from 2 to 10, incremental steps of 2
OA3	Daubechies filters of length 4, 8, 16 , 32 and 64
OA4	OpenMP's built-in O1, O2 and O3 optimizations
OA5	Combination of the most efficient approaches

Each test case was tested ten times and an average value of measured times was calculated and used for further processing. Moreover, functional verification was conducted to verify the functional characteristics of the executions obtain identical results.

Chapter 11
Overview and Related Work

The content of this Chapter was published at the 6th International Conference on Applied Internet and Information Technologies [40], 2016, 27th DAAAM International Symposium [42], 2016, 13th International Conference on Informatics and Information Technologies [41], 2016, 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), IEEE [44], 2017 and 14th International Conference on Informatics and Information Technologies [43], 2017.

11.1 Related Work of DSP Filters

Anwar et al. [18] have published a solution using shared memory, without analyzing the possibilities of the constant memory. Wefers et al. [135] have concentrated on implementing a real-time convolution on frequency domain. Herdeg et al. [96] have also concentrated on the frequency and time domain fast convolution. On time domain convolution, though, authors have not provided any information related to memory storage, access, and optimizations.

We have considered parallelizing DSP filters on a Maxeler dataflow engine [41]. We obtained promising results, with linear speedups proportional to the kernel length. We calculated speedup by measuring a number of sequential steps needed to convolve the signal, whereas here we compared execution times.

Additionally, we have focused on parallelizing DSP filtering algorithm on CUDA [40], by utilizing thousands of GPU cores. The experiments have shown linear speedups, proportional to the kernel size. Obtained results serve as a basis for this research. We have also observed that threads per block (TPB) should be kept as high as possible by considering the upper limit being provided by the specification of the GPU card.

Sava et al. [118] have developed two parallel solutions for generic Wavelet Transform for signal processing, without addressing specifically the concept of baseline drift of the signal. Their algorithm is based on pipeline processing farming. They

conclude that the performance of algorithm increases as filter length and data length increase. However, authors have not provided any information about the impact of core numbers, parallelization platform, and the scalability.

Kayhan and Ercelebi [80], proposed lifting scheme based DWT algorithm for ECG denoising. Tests were conducted with an 360Hz ECG signal with 216.000 samples. Their algorithm provided fast executions where on Daubechies filter of 8, 0.141s execution times were provided.

Rajmic and Vlach [114] have proposed a real-time algorithm via segmented wavelet transform analysis, presenting only the principle without practical implementations.

Stojanović et al. [127] have proposed optimized algorithms for biomedical signal processing. Their results are 2-4 times faster than the sequential implementation, though being incomparable with our work.

11.2 Overview of Obtained Results

11.2.1 CUDA DSP Filter for ECG Signals

This section evaluates and discusses the obtained results, and also provides an analysis of the TPB number to determine an optimal configuration of the algorithm.

This work contributes CUDA GPU parallelization for noise filtering of ECG heart signals. The provided parallel solution takes advantage of thousands of CUDA cores. Their size is increasing linearly to the input length used.

Results obtained by executing the sequential algorithm and the parallel CUDA algorithm show they are identical for low-pass, high-pass and band-pass filters.

The analysis on GPU shows that higher available numbers for threads per block produce higher speedups. Increasing the kernel length has a noticable effect on the speedup. The row version of the CUDA algorithm achieves a speedup proportional to the input and filter length.

This research is the first step of the ECG signal processing with the intention to extract hidden information.

As future work, we plan to carry out more tests on multiple GPUs tied together and compare the results to the OpenMP and similar solutions. We also plan to find a more appropriate CUDA algorithm and optimize the GPU execution if possible by an element algorithm version, where the threads are defined on element level, and not on a row level, expecting that the parallelized code will scale linearly with increasing filter size, and achieve even higher speedups by eliminating the synchronization barriers and aligning the memory access without bank conflicts.

11.2.2 Optimizing high-performance CUDA DSP filter for ECG signals

Figure 11.1 combines the optimizations compared to the double precision version of the previous code. We observe that Shared Input and Output version speedups the code by 13% and 78%. Using Constant memory version is 15% faster compared to the previous code [40]. Combining these, the maximum speedup of 2.4 is obtained. Thus we can conclude that the hypothesis we set is confirmed.

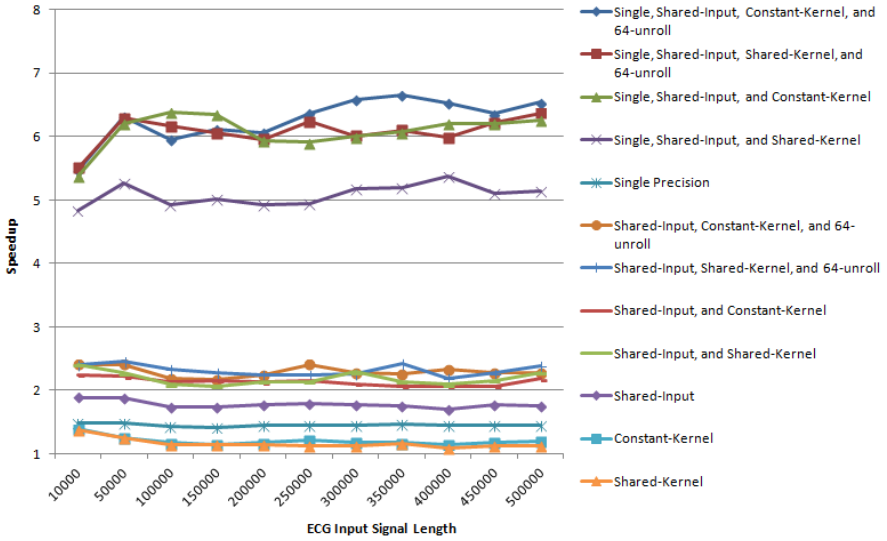


Fig. 11.1: Performance gain by a combination of optimization approaches.

We were also interested to investigate the effect of the loop unrolling. We observe that using a loop unrolling with a length of 64 introduces additional 4 % performance gain. Additionally when the precision is decreased to 32 bit, the speedup increases up to 6. An interesting case was to see whether the element version performs better. We observed that, due to the internal synchronization and bank conflicts performed on each step, the performance of the element version is not attractive.

This research was concentrated on time-domain convolution. Generally, data intensive streams are convolved on the frequency domain, by actually multiplying two signals on the frequency domain. Gained results were much faster, but the results were not accurate. This is because ECG signal is not always periodic; since the heart beat rate changes due to physical activity, emotional state, etc. So, in the case of ECG, frequency-domain convolution is not recommended.

We also investigated the usability of NVIDIA's CUBLAS library to dispatch each of the required convolution calculations as a Level 1 BLAS (Basic Linear Al-

gebra. Subprograms) caxpy (Scalar Alpha X Plus Y) operation. Caxpy computes a constant alpha times a vector x plus a vector y, and finally, overwrites the initial values of vector y. In the case of ECG signal, the impulse response has variable filter elements, making the use of this operation useless.

Moreover, we designed and implemented the element version of convolution for ECG signal context using explicit synchronization for reduction and summation. Even though the reduction operation has logarithmic complexity, our tests have shown that this approach decreases the performance. We plan to develop an optimized algorithm where multiplications will be distributed across different blocks, and the synchronization will be handled inside threads of a block. We believe that this will optimize the current version of the code.

Higher speedups are obtained when using single instead of double precision. This is directly related to the achieved GFLOPS (billions of floating point operations per second) of the device. On GPUs, double precision GFLOPS is smaller than the single precision [1], which results in faster executions when floating point numbers are used.

In this research, we have also considered using loop unrolling. Approximately 1-5 % performance gain is obtained with double precision. We have also tested to decrease the precision, and nearly doubled the performance of the algorithm.

This research contributes to the CUDA GPU optimization strategies for the noise elimination on ECG heart signals. The proposed optimizations, take advantage of intra-block shared memory and the general constant memory. By storing ECG segments on shared memories, we have optimized the memory accesses.

Additionally, we used storing of the filter coefficients on the constant memory, thus consecutive reads of the same address does not generate any additional memory traffic. We are aware of the limitations, where filter length should be smaller than the maximum number of threads per block due to the internal synchronization inside a block.

Results obtained by executing optimized algorithms show they are identical for each of the filter types. From the obtained results, we can conclude that proper usage of shared and constant memory has a positive impact on the performance. Our analysis showed that their combined effect yield 2.4 times faster executions compared to the previous code. We can, therefore, conclude that the hypothesis is confirmed.

Considering loop unrolling speeds up the code by 1-5%. Moreover, we tested the decreased precision effect on the performance and got nearly 1.5 faster code when on 1000 filter length. We observed that the element version is not effective when ported on GPU.

It is important to note that each of the proposed optimization techniques adds up to the combined speedup. We observed that the best-combined effect had a speedup of 6.

11.2.3 Dataflow DSP Filter for ECG Signals

The provided parallel solution takes advantage of thousands of Dataflow cores. Their size is increasing linearly (according to the maximum number of available space) depending on the kernel size used.

Results obtained by executing the sequential algorithm and the parallel dataflow algorithm show that the obtained results are identical for low-pass, high-pass and band-pass filters.

The analysis shows that the speedup is proportional to the filter length. In this research we have experimented with the Hamming window and the Blackman window with length of 100 and 200 elements to obtain relatively good results.

11.2.4 Parallelization of Digital Wavelet Transformation of ECG Signals

Fig. 9.5 presents the execution times on a logarithmic scale with base 10 since the algorithm complexity is $O(L*2^L)$. Additionally, Fig. 9.6 presents the speedup values for the proposed optimization approaches. Input sizes are selected as 2000, 8000 and 32000 samples of ECG activity. Wavelet levels vary from 3 to 24, with incremental steps of 1. The increase of wavelet levels demands more cores.

It can be observed that **O1** optimization approach tends to give positive results. The performance of the code is increased by roughly 20% for increased number of wavelet levels and cores.

The results with the **O2** and **O3** optimization approaches are not efficient. This is due to the barrier synchronization, used to synchronize the nodes.

This can be neglected only when the number of nodes (that can simultaneously execute wavelet levels) is relatively high.

Speedup values presented in Fig. 9.6 show that the optimization approach **O1** gives up to 20% faster code. This was the expected case since the *Step Array Calculation* phase does not contain any data dependencies between loop iterations. Considering that an input ECG signal is using a 500Hz sampling frequency, then eliminating the baseline drift will require 9 or 10 wavelet levels. In this case, the proposed optimization approach will yield $10-15\%$ faster code.

Moving forward to the optimization approach **O2**, it is clearly seen that this approach is not attractive for significantly low number of wavelet levels. As wavelet levels increase, the performance of the algorithm increases. The main reason not to obtain a higher speedup is the overhead of using barrier synchronization for synchronizing nodes.

Since using higher wavelet levels is practically not possible we conclude that this strategy is not attractive. Same arguments can be made for the optimization approach **O3**, except that it is yielding a bit faster code compared to the optimization approach **O2**.

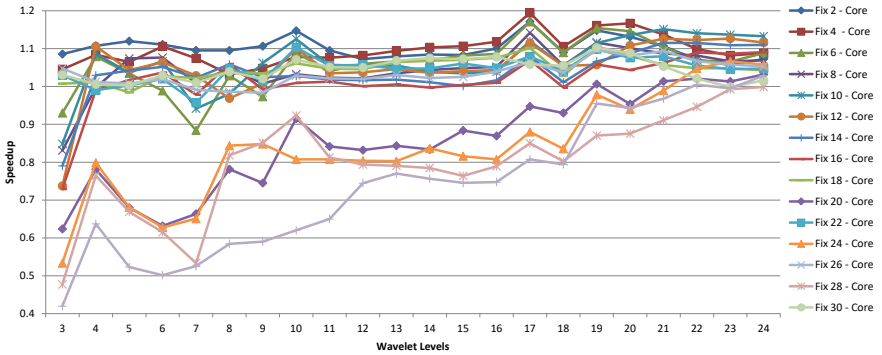


Fig. 11.2: Speedup of the paralleled Initialization phase with fixed number of cores compared to an implementation with cores equal to the Wavelet levels.

We tried using explicit synchronization strategies, without any success. Such strategies are only efficient on a high level of iterations.

Based on the results presented, we can conclude that the hypothesis set in this paper is partially confirmed. Even though the proposed parallel algorithm for the processing part was not efficient on small wavelet levels, parallelization of the initialization part resulted with a faster code.

This work contributes OpenMP parallelization for the baseline drift elimination of ECG heart signals. Three optimization strategies were provided.

Results obtained showed that parallelizing the initialization part gives a speedup of nearly 1.2. On the other hand, the parallelization approach for processing part was based on transforming loop iterations, in a manner that they become independent. Pipelined parallel algorithms were developed, and tested.

Our observation is that, on low wavelet levels, the parallel algorithm for the processing part is not efficient. This is primarily due to barrier synchronization between iterations.

We also tested the effect of input sizes. Unless the delay is bigger, input size plays a huge role in the speedup. The higher the input size is, the higher the speedup.

In both cases, it can be noted that the provided algorithm is scalable. Theoretically, if we run the algorithms on higher orders of magnitude, the achieved speedup will be higher.

11.2.5 Optimal Parallel Wavelet ECG Signal Processing

Figure 11.2 presents the speedup values when running the parallelized *Initialization* phase with fixed number of cores. These values are calculated by comparing with

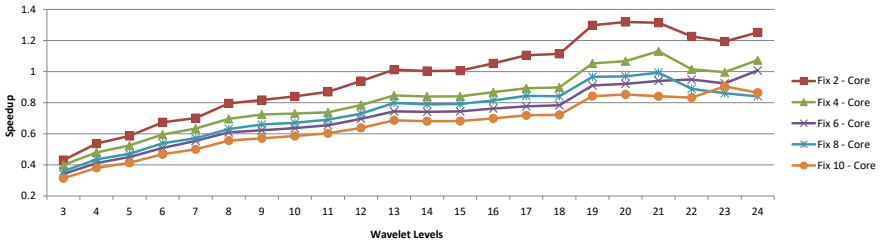


Fig. 11.3: Speedup of the paralleled Processing phase with fixed number of cores compared to an implementation with cores equal to the Wavelet levels.

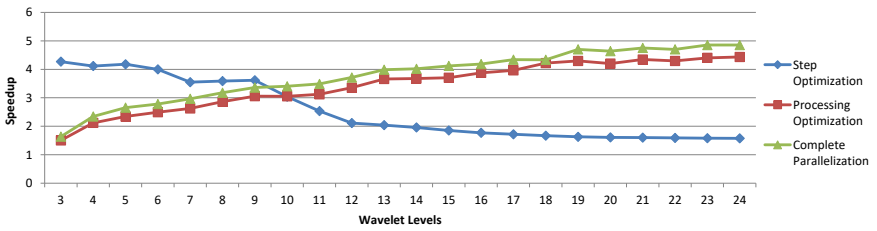


Fig. 11.4: Speedup of combining the best optimisation approaches compared to an implementation with cores equal to Wavelet levels without optimisations.

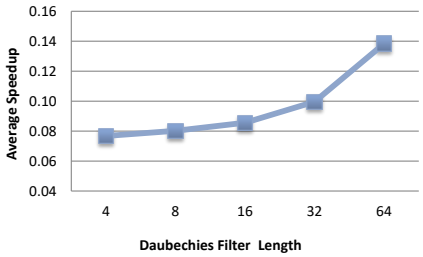


Fig. 11.5: Average speedup of the paralleled Processing phase with different filter lengths.

the case when running on core numbers equal to Wavelet levels. It is observed that, from wavelet levels ranging from 3 to 10, the fixed 2 core execution is faster by an average speedup of 12%. Similar speedup is obtained between wavelet levels 11 and 20, when running on 4 cores. On wavelet levels higher than 20, the average speedup is calculated to be 14%, though on executions with fix 10 cores.

These results indicate that, running the initialization phase on fixed number of cores yield faster code. It is observed that, as the wavelet increase, using higher number of cores becomes efficient. The speedup which is obtained is nearly 12%.

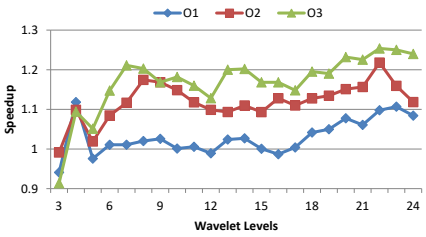


Fig. 11.6: Speedup of the parallel algorithm with using built-in OpenMP optimization flags.

Next, on the Figure 11.3, speedup values which correspond to running parallelized *Processing* phase with fixed number of cores. Again, values are calculated by comparing with the case when running on core numbers equal to Wavelet levels. It is observed that this optimization approach is effective especially when the Wavelet level is greater than 13. Fix core 2 case yields the best results. This was expected, since it decreases the negative effect of barrier synchronization. On wavelet levels higher than 13, the average speedup is calculated to be 10%, though on executions with fix 2 cores.

Figure 11.5 presents the effect of filter length. Filter length has an effect only on the *Processing* phase, thus test is conducted on the complete parallelization case. As expected, increasing the filter length has a direct effect on the speedup of the parallelization. This is primarily due to the fact that, as filter length increase, the effect of synchronization becomes less important. On filters of length 64, the completely parallel algorithm performs 2 times faster.

The effect of compiler optimizations is shown in Figure 11.6. Results indicate that the build-in **O3** optimization, fastens the completely parallel algorithm by at least 15%.

Figure 11.4 shows the combined effect of the optimization approaches, where the code is tested on the best configuration, i.e. fix 2 cores, with 64-length Daubechies filter and built-in compiler optimization flag **O3**. Observation is that on Wavelet levels less than 10, the *Step Optimization* algorithm has an average of 4 speedup. What is more interesting, when the Wavelet levels are greater than 10, the optimizations yield a speedup of 4 *Complete Parallelization*, in a scalable nature.

Observation is that proposed optimisation approaches can yield faster codes when run on proper configurations.

This work contributes OpenMP optimization for the baseline drift elimination of ECG heart signals. Totally five optimization approaches were proposed. Results indicate that, each of them can yield faster codes on proper configuration.

Approaches *OA1* and *OA2* yields speedup values of at least 10%. On the other hand, results showed that as filter length increases, the proposed parallel algorithm's efficiency increases. To be more specific, the *OA3* approach speeds up the parallel code by a factor of 2, when the filter length is increased from 4 to 64.

The effect of compiler's built-in optimization strategies were tested. The outcome of this *OA4* approach was that the *O3* flag performs best, with at least a 15% performance gain.

Lastly the combined effect was tested as the proposed *OA5* approach. Observation was that the combined effect yields a speedup of 4.

Part III
QRS Detection

Chapter 12
Optimal DSP Bandpass Filtering for QRS detection

The content of this Chapter was published at the 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), IEEE [64], 2018.

ECG signal is usually contaminated with noise, that can stem from different internal and external sources, such as environmental noise or internal body muscle movement. The next step is to reduce the search space and extract only the QRS complexes as much as possible in order to construct a good QRS detector. Analyzing the QRS structure and the properties of a typical ECG signal, one can conclude that a bandpass filter can successfully be used for these purposes. Some authors suggest using only a high-pass filter to extract the QRS complexes, mostly based on the knowledge that the intensity of QRS is larger than the intensity of high frequencies noise.

To avoid misinterpretations due to the baseline drift and other potential noise and obstacles generated by other internal waves, we continue with an analysis based on bandpass filters. Our goal in this chapter is to determine the optimal values of the bandpass filter that can successfully eliminate the noise and extract the QRS waves only.

Therefore, we will investigate the effect of the filter length in case of FIR filtering, or other characteristics relevant for IIR filters or wavelet filters, trying to find the optimal values of the central frequency, bandwidth and -3 db cutoff frequencies of the filter. The influence of the band pass filter will be evaluated by the accuracy, sensitivity, and precision. We use the standard MIT-BIH database while evaluating the algorithm.

12.1 Analysis of the ECG and QRS spectra

We have analyzed experimentally the power spectra of QRS complexes and other signals including the noise of the MIT-BIH dataset of 48 records. Here we present,

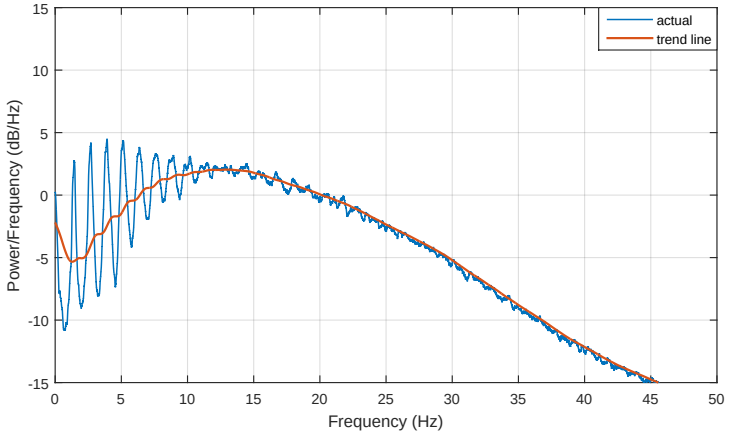


Fig. 12.1: Relative power spectra of QRS complexes (MIT-BIH 100).

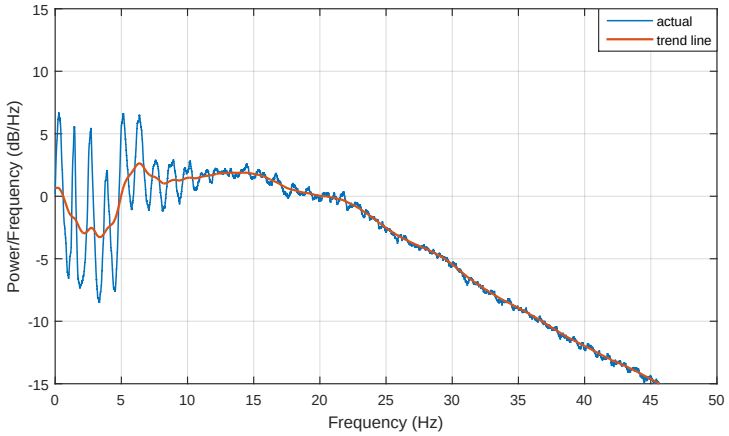


Fig. 12.2: Relative power spectra of noise, P and T waves (MIT-BIH 100).

a typical spectrum for patient ID=100 in Fig.12.1 for QRS signals only, and in Fig.12.2, the determined noise spectra and the spectra of the P and T waves.

Although the measured values mostly correspond to the earlier findings and conclusions, we can determine that there is no clear distinction of the exact QRS band. Note that the patients with pacemakers have a slightly different spectrum, mostly due to the fact that the QRS is shorter.

One can observe oscillations in the range of 0 and 10 Hz in the spectra of the QRS complexes and a typical answer is that they happen due to respiration and muscle movements. However, the respiration rate and muscle movement affect only the spectra from 0 to 1 Hz.

Oscillations in the region from 1 to 10 Hz represent a typical situation of a frequency modulation between the QRS complex waves and heart rate waves. Suppose that a typical QRS wave is a part of an ideal sine wave with a frequency of f_{QRS}. Then this wave is repeated at regular time intervals determined by the heartbeat rate with frequency f_{HR}. A composition of these two signals results in a frequency modulation of the signal with a higher frequency (f_{QRS}), also known as carrier frequency, with a signal with a smaller frequency (f_{HR}) or signal frequency. The power spectra of this composed signal will consist of these two frequencies presenting the corresponding intensities.

However, in reality, the QRS wave is not an ideal sine wave and it is a composition of several frequencies. This also applies to the heart beat since a typical heart changes its frequency almost in each heartbeat. This is reflected as a variation of the carrier frequency. The oscillations in the observed spectra, actually represent the effects of frequency modulation of these signals.

The following analysis addresses the power spectra of the QRS complexes (Fig. 12.1) in the band 0 to 10 Hz. The regular peaks that repeat each 1.25 Hz are noticeable and they represent the heart rate frequency f_{HR}, which corresponds to the average heartbeat rate of $1.25 * 60 = 75BPM$. The variations in the sine wave shape present the heart rate variations, and in higher harmonics, they are less expressed. One can notice that a low pass filter will determine the heart rate frequency (f_{HR}) and the high-pass filter, the carrier frequency (f_{QRS}).

The same frequency modulation phenomenon can be observed also for T (and P waves although they are smaller in intensity) in Fig. 12.2. One can notice the first harmonic with a value of about 3.8 Hz, and its harmonics.

In this paper, we desire to determine the spectra of the carrier frequency and detect the QRS complexes, by a theoretical interpretation of the experimental results about spectral analysis at higher frequencies.

The literature [15] points the typical values of time duration of a normal heartbeat QRS complex to be between 60 and 120 ms. The inner part of the signal that mostly looks like a half sine wave occupies approximately most of this interval. So, one can expect that the distribution is mainly around a frequency that corresponds to a full sine wave of 120 ms, which in fact is 8.33 Hz. A larger and smoother QRS complexes last 120 ms correspond to 4.17 Hz, and smaller sharper QRS complexes might last at least 30 ms, which correspond to 16.7 Hz.

So far, our analysis leads to a conclusion that filtering the signal within 4 and 20 Hz will reduce the feature space to mostly the values of those in the QRS complexes. Note that in the case of paced beats, the QRS complexes might look even sharper and last less than 30 ms, which will need filtering with frequencies higher than 33 Hz. On the other hand, in the case of premature ventricular beats, the morphology of the QRS complex is prolonged and might last more than 250 ms, which leads to frequencies lower than 4 Hz.

The problems in detection might arise by the onset of P and T waves. The typical values of P waves (half sine wave) is 110 ms, so its power spectral energy will be around 4.5 Hz. T waves are even larger than 200 ms that corresponds to frequencies below 2.5 Hz.

This analysis shows that a bandpass filter with a central frequency of approximately 8.33 Hz and cut off frequencies at 4 Hz and 20 Hz will enable feature space reduction to extract QRS complexes. However, a good QRS detector has to be implemented also on experimental results and analyze also the other sources of the power spectra, including the environmental noise, P and T waves and muscle and motion artifacts.

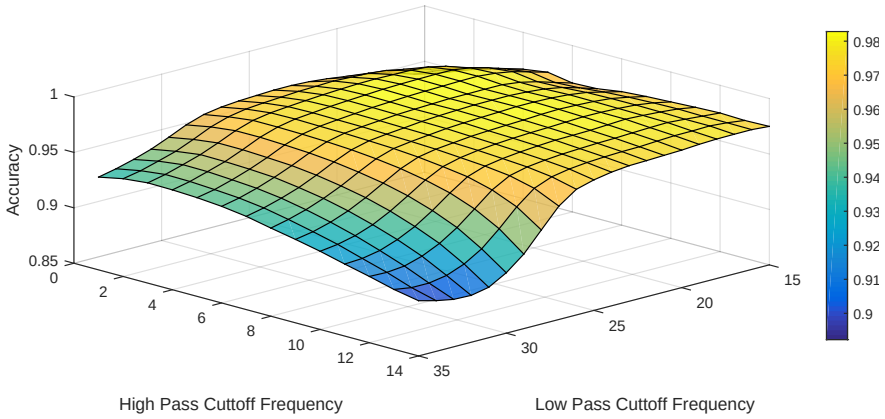


Fig. 12.3: Accuracy for different bandpass filter values for FIR filters.

12.2 Testing Methodology

There is no ideal filter design, so this is why we analyze several filters to estimate their performance not just from the feature space reduction and noise elimination aspects but also from the aspects of their computational complexity, and their implementation in real environment with real wearable sensors, such as the long-term Savvy ECG sensor [119].

For the experiments, we have used a differential approach to detect a QRS complex, based on applying filtering with DWT reconstruction db3 low and high pass filters, which actually constructs a bandpass filter. The output of this filter is compared to a threshold value, which is calculated as one-third of the average value of all output items. The experimental environment was a computer with a high-frequency Intel Core i7-3632QM CPU @ 2.2 GHz processor with 8 cores, and 12GB memory.

The dataset used was the MIT-BIH database [99]. We have conducted test cases on all 48 records with 650000 samples each corresponding to 30 minutes ECG records with a 360 Hz sampling frequency.

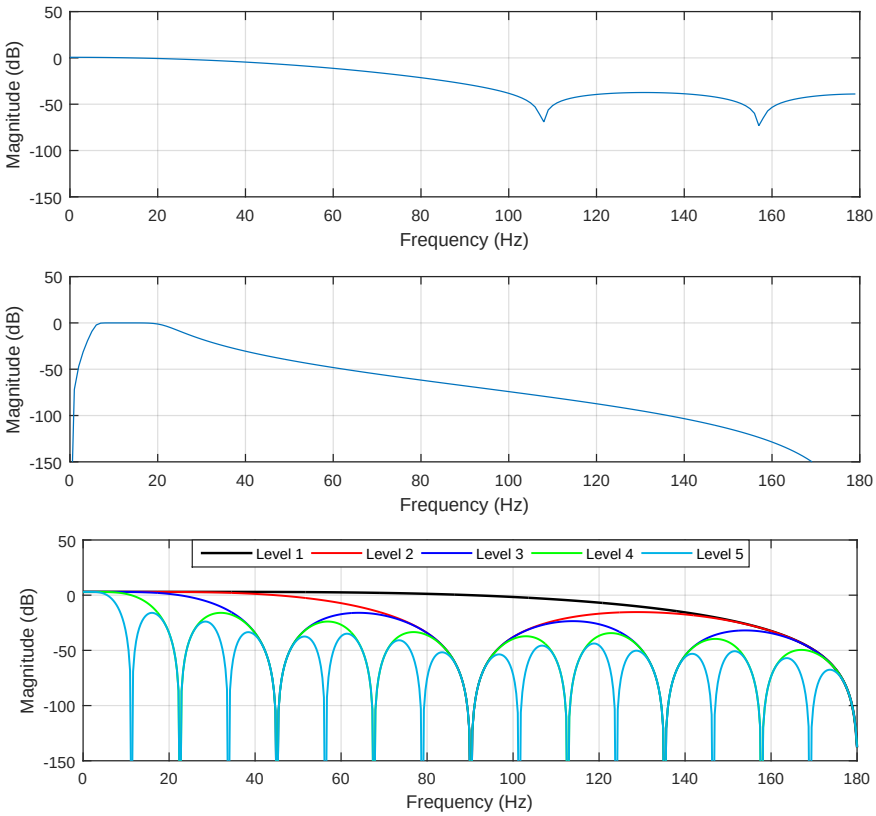


Fig. 12.4: Frequency response characteristics of used FIR, IIR and DWT filters.

The test cases included high-pass cutoff frequency ranges from 1 to 14 Hz with incremental steps of 1Hz and low pass cutoff frequency from 15 up to 35Hz, with incremental steps of 1Hz.

The test range of the band pass FIR filter length, ranges from 11 to 201 with incremental steps of 10. On the last case, the best parameters will be combined and tested. We have used FIR filters with lengths of 5 up to 201 and IIR filters with length of 5, 7, 9, 11, 13 and 15.

The signal filtered by DWT was reconstructed by using the approximation of A3 (0-22.5Hz) and deducting the A5 (0-5.625Hz) which leads to a bandpass filter with a close approximation of the desired band between 5.625 - 22.5Hz.

The test goal is to measure and optimize the *sensitivity (SEN)*, *accuracy (ACC)* and *positive predictivity rate (PR)* values, as defined by (12.1).

$$
\begin{aligned}
SEN &= \frac{TP}{TP+FN} \\
ACC &= \frac{TN+TP}{TN+TP+FN+FP} \\
PR &= \frac{TP}{TP+FP}
\end{aligned}
\tag{12.1}
$$

To calculate these performance measures we used the count of those beat annotations that appear to be correct by *true positive* and denoted by TP, and samples that were correctly annotated that are not peaks *true negative* denoted by TN. Annotations that were generated as a QRS peak, but are not real heart beats as *false positive* by FP, and those that were missed by the algorithm are *false negative* as FN. Our testing QRS detection algorithm is based on a wavelet filter using a simple algorithm based on differential approach testing the slope of the ECG curve. The performance of the algorithm is 99.63% QRS sensitivity values, without using extra filters prior to applying the QRS detector algorithm.

Table 12.1: Comparison of various filters on QRS detection sensitivity (Sens.), accuracy (Acc.) and positive predictivty rate (PR).

Filter Type	TP	TN	FP	FN	Sensitivity	Accuracy	PR
No Filter	105443	0	2874	4051	0.963	0.938	0.973
FIR (9)	106683	0	2888	2811	0.974	0.949	0.973
IIR (9)	108486	0	1141	1008	0.991	0.981	0.989
DWT (db2)	108668	0	1165	826	0.992	0.982	0.985

Chapter 13
Optimizing the Impact of Resampling on QRS Detection

The content of this Chapter was published at the International Conference on Telecommunications, Springer [65], 2018.

Hamilton [69] has created a QRS detector that works on a sampling rate of 200 Hz and tuned threshold parameters to obtain a relatively good performance, measured as QRS detection sensitivity and positive predictive rate.

Three different peaks may be classified in the QRS detection process, a real QRS beat: a pattern with identified Q, R and S points with predefined slope and amplitude; a noise peak: generated by muscles or skin, which does not follow the pattern either by the number of detected points, amplitude or length; or artifact: a peak that looks pretty much as a QRS beat, but lacks the amplitude or occurs due to muscle movements or loose contact with the electrode.

In this chapter, an experimental research is used to measure the performance of different sampling rates and find the optimal threshold values.

13.1 A QRS Detection Algorithm

QRS detection algorithms generally follow a standard procedure [106]. They start with digital signal processing (DSP) filters with the aim to reduce noise and weaken the effect of other waves on QRS detection. In the next phase the signal is matched against a threshold. Last, there is a decision layer to classify peaks as real or noise.

One of the first published and most cited paper for QRS detection is the Pan & Tomkins algorithm [107]. Apart from being a real-time algorithm, it offers the ability to adapt to noises. One of the disadvantages of this algorithm is the dependence on the sampling rate, which is configured to run at 200 Hz, and its bad performance on long records with small amplitudes. Unfortunately, there is no information about bit resolution in their work.

Hamilton's algorithm is another similar derivative-based approach [69]. It uses a preprocessor similar to the Pan & Tomkins algorithm and offers a different set of

complex rules for decision making. An open source implementation code has been released by EP Limited [70].

The algorithm starts processing the input signal with a 16 Hz low pass filter, followed by an 8 Hz high pass filter. The output is provided to a differentiation filter that calculates a difference between consecutive samples and determines the slope of the signal. Then, an absolute value is calculated and the average is calculated over an 80 *ms* window. This ends the first phase where the energy of the signal is calculated, as presented in Fig. 13.1 b) for the signal in Fig. 13.1 a).

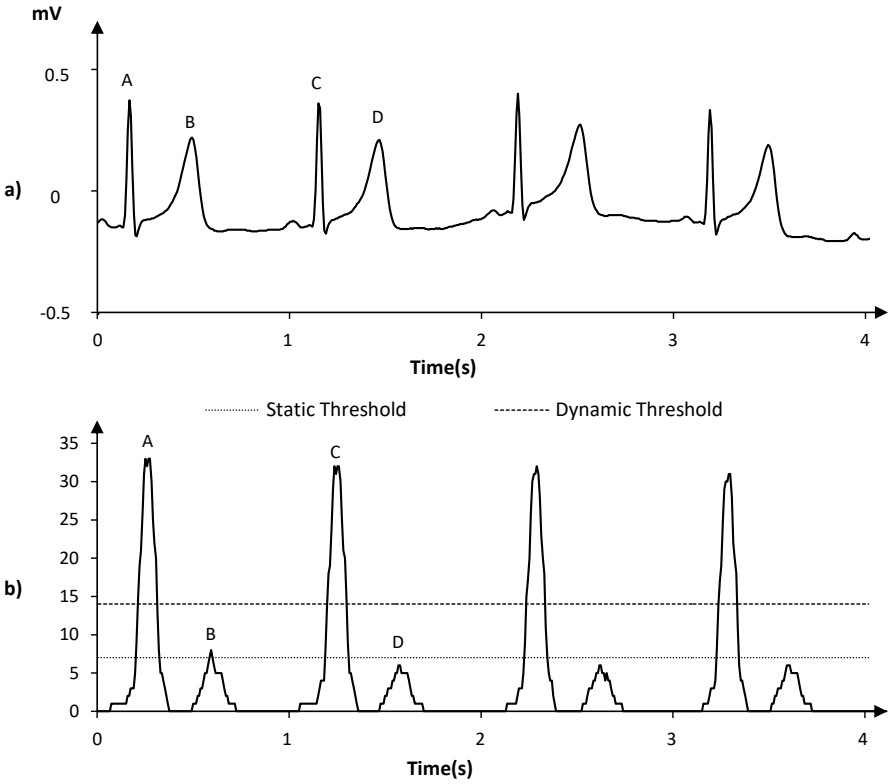


Fig. 13.1: Detecting artifacts, noise and real peaks based on values of static and dynamic thresholds in the original Hamilton's algorithm presented on signal extracts over record 113 (MIT-BIH Arrhythmia database).

The second phase detects if the energy output is a peak, and classifies the peak as an artifact, noise peak or real beat. Peaks are determined by calculating local maximums and comparing them to a *static* and *dynamic* threshold. The static peak determines if the analyzed local maximum is an artifact or candidate peak, while the dynamic threshold classifies real peaks by extracting noise. The effect of static and dynamic threshold filters is illustrated in Fig. 13.1, where the peaks *A*, and *C* are

real beats (reaching a value over both thresholds), whereas *B* as noise peak (reaching over static threshold, but not dynamic) and *D* as an artifact (below both thresholds).

The dynamic threshold is calculated as a mean of the last eight candidate peaks. The default implementation uses a *fixed* static threshold equal to $MIN\_AMP\_PEAK = 7$. In our improved algorithm, we treat this parameter as modifiable. The experiments [46] show that decreasing this value will produce more artifacts and higher number of true beats, and vice versa.

When we tested our improved algorithm, we have found that the optimal value of the threshold parameter depends on the quality of the analog to digital convertor, i.e. on the sampling frequency and bit resolution. The optimal value needs to be selected as a compromise between the number of artifacts, and true peaks, balancing the number of beat misses and extra found peaks.

13.2 Experiments

Testing was conducted on an annotated ECG benchmark databases in order to compare the results, the MIT-BIH Arrhythmia ECG Database [99] in particular. It contains two-channels of 48 half-hour ECG recordings publicly available on the Physionet web site [63]. The original recording frequency is 360 samples per second per channel with 11-bit resolution over a 10 mV range.

13.2.1 Test Cases

In our experiments we excluded the paced beat records 102, 104, 107 and 217, and tested a total of 44 records. The input test data was only the first ECG channel.

The experiment is planned with a lot of test cases. We start the experiment with the default fixed static threshold value of 7 and measure QRS detection performance on a set of sampling frequencies starting from 100 up to 360 with an increase of 25 (last two increases are 30 instead of 25).

The next test cases consisted of the same environment and a different static threshold parameter, starting from 2 to 10, and included other threshold values (not just evaluating the fixed value of 7).

13.2.2 Test Data

We have measured the number of *true positive (TP)* detections, which correspond to correctly detected beats, the number of *false positive (FP)* errors, which is equivalent to false detection of extra peaks that are not real beats, and *false negatives (FN)* errors, which is equivalent to misses in detection of real beats. Performance

measures are evaluated through QRS sensitivity QRS positive predictive rate, correspondingly calculated as (13.1) and (13.2).

$$Q_{SE} = \frac{TP}{TP+FN} \tag{13.1}$$

$$Q_{+P} = \frac{TP}{TP+FP} \tag{13.2}$$

QRS sensitivity indicates how many of the real beats are detected compared to the total number of beats; and the QRS positive predictive rate specifies how many of the detected peaks are real beats. It means that the QRS sensitivity specifies the successfulness of detecting all real beats, whereas the positive predictive rate specifies the successfulness of detecting real beats and avoiding false detections.

Given these performance measures, we can not posit what is better: to have higher QRS sensitivity and lower positive predictive rate, or vice versa. For example, it is ambiguous to compare an algorithm with a little higher value of QRS sensitivity, but much lower value of positive predictive rate than another algorithm.

In our analysis, we give a performance advantage to those that have approximately equal values of QRS sensitivity and positive predictive rate, or a higher value of sensitivity than the positive predictive rate.

13.3 QRS Detection Performance at Different Sampling Rates

This section presents the results from the experiments. It aims to evaluate the dependence of QRS detection performance on different sampling frequencies and to find an optimal threshold parameter for achieving the best QRS detection performance.

13.3.1 Fixed threshold - Hamilton's approach

Fig. 13.2 presents the dependence of QRS sensitivity and positive predictive rate on different sampling frequencies. In the figure, x-axis represents different sampling frequencies (from 80 to 360) and y-axis the QRS sensitivity Q_{SE} and positive predictive rate rate Q_{+P}.

QRS sensitivity fluctuates between 99.53% for a sampling frequency of 100 Hz to 99.81% for 250 Hz. The average QRS sensitivity is equal to 99.74% with a standard deviation of 0.081.

QRS positive predictive rate fluctuates within a smaller range of values (99.71% and 99.79%) and achieves an average value of 99.75% with a standard deviation of 0.032.

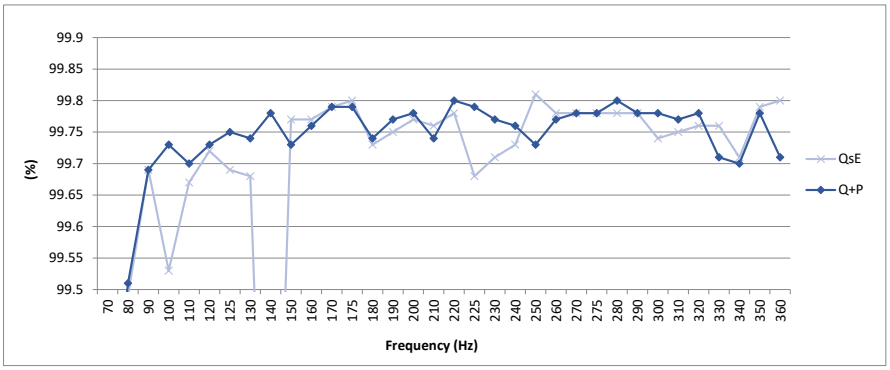


Fig. 13.2: QRS performance vs sampling rates for fixed static threshold equal to 7.

According to our measurements, the best performance is reached for a sampling frequency of 175 Hz. One can conclude that there is a discrepancy in QRS sensitivity and positive predictive rate at different sampling frequencies.

The best performance is, when the sensitivity and positive predictive rate reach almost the same values, and at the same time, their value is high. Note that the original Hamilton's algorithm was tuned for the static threshold parameter of 7 at a sampling frequency of 200 Hz.

Another interesting discrepancy is the dependence trend. The positive predictive rate reaches lower values at sampling frequencies that are modulo 50 than in the neighboring frequencies.

13.3.2 Performance Testing Different Threshold Values

To determine if there is a better performance than the one achieved for the fixed static threshold value we conducted test cases for pairs of different threshold values and sampling frequencies.

The dependence of QRS detection performance on different sampling frequencies and different threshold parameters are presented in the 3D graphs in Fig. 13.3 and Fig. 13.4. The x-axis presents the sampling frequencies, the y-axis represents different static threshold parameters and the z-axis (depth) presents the corresponding QRS sensitivity and positive predictive rate in Fig. 13.3 and Fig. 13.4.

A fixed static threshold parameter with a value of 7 reveals a relatively satisfactory performance for the QRS positive predictive rate only, even though higher static threshold values reveal higher QRS positive predictive rates. However, as we have discussed earlier, one needs to make a compromise and achieve a higher value of QRS sensitivity at the same time.

Normally, QRS sensitivity is higher for smaller static threshold parameter values.

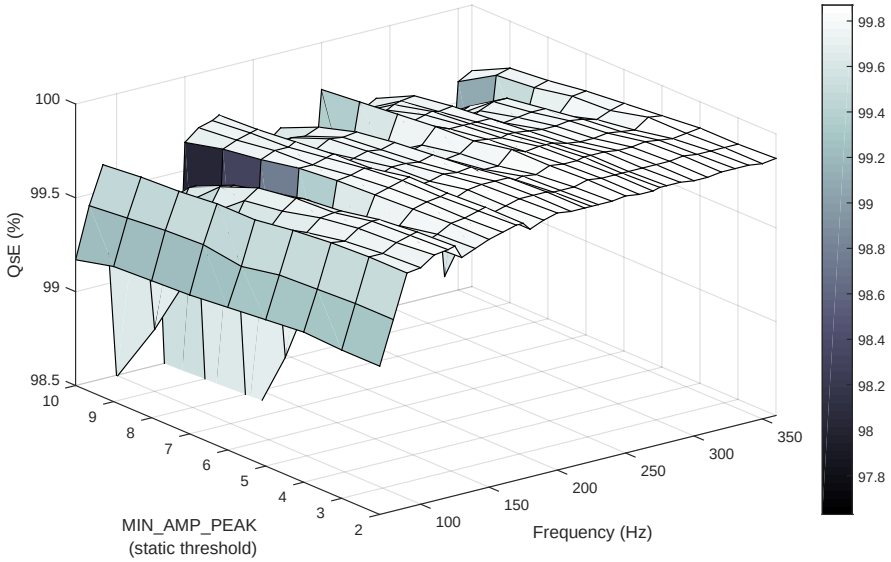


Fig. 13.3: QRS sensitivity values for different thresholds at different sampling rates.

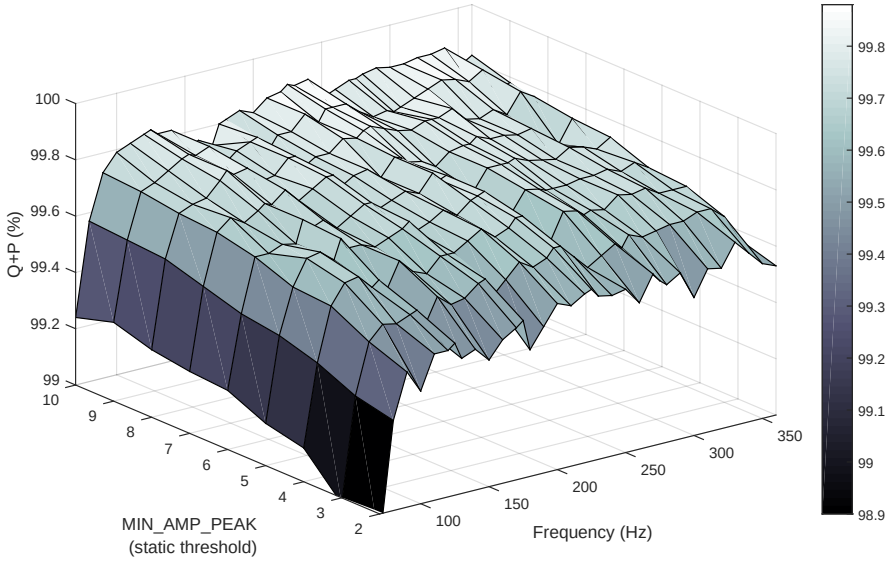


Fig. 13.4: QRS positive predictive rate for different thresholds at different sampling rates.

QRS sensitivity reaches the smallest value of 98.49% for a sampling frequency of 100 Hz and a threshold parameter of 10 and the highest value of 99.87% for

200 Hz. The average QRS sensitivity is equal to 99.71% with a standard deviation of 0.173 in all conducted test cases.

The lowest value 99.40% of the QRS positive predictive rate is reached for a sampling frequency of 100 Hz and threshold parameter 2, and the highest value of 99.88% for 225 Hz and threshold parameter 10. The average value in all test cases is 99.72% and standard deviation of 0.09.

To conclude, the threshold parameter needs to be tuned to achieve the best performance.

13.4 Discussion

This section discusses the optimal threshold parameter values at different sampling frequencies.

13.4.1 Optimal Threshold and Performance

Fig. 13.5 presents the optimal threshold values found in our experimental research, where the *x*-axis represents different sampling frequencies and the *y*-axis the optimal threshold values.

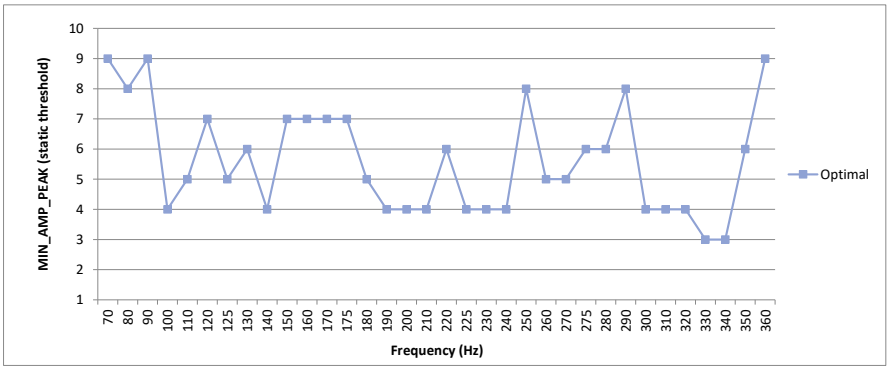


Fig. 13.5: Optimal threshold values at different sampling rates.

The performance for optimal threshold values at different sampling frequencies is presented in Fig. 13.6.

QRS sensitivity reaches higher values than those achieved for a fixed static threshold. The minimum value of 99.73% is reached instead of 99.53% for a sampling frequency of 100 Hz and maximum 99.86% (for 330 Hz) instead of 99.81% (for 250 Hz). At the same time the QRS positive predictive rate reaches a minimum

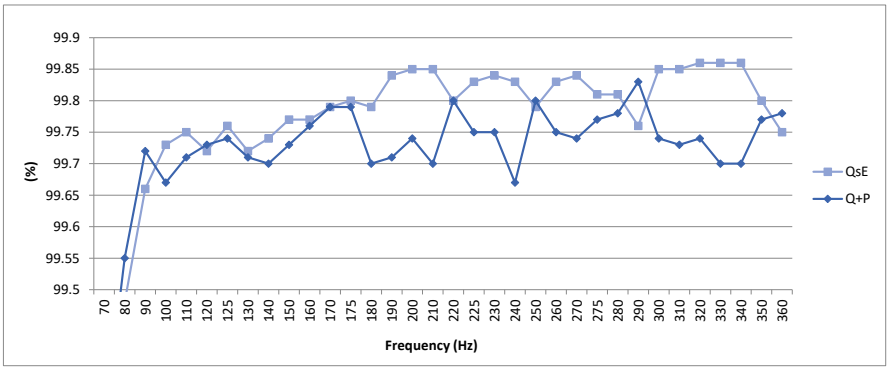


Fig. 13.6: Optimal performance values at different sampling rates.

of 99.67% (for 100 Hz) and maximum of 99.80% (for 250 Hz) instead of 99.79 Hz (for 225 Hz).

The average performance values obtained for optimal threshold values are improved to 99.80% QRS sensitivity instead of 99.74% with fixed threshold keeping the same average value of the QRS positive predictive rate. At the same time the standard deviation is reduced to 0.044 instead of 0.081 for QRS sensitivity, which is almost the same as the standard deviation of the QRS positive predictive rate.

13.4.2 Optimal vs Fixed Static Threshold Performance

Fig. 13.7 and Fig. 13.8 correspondingly compare QRS sensitivity and positive predictive rate for found optimal and fixed static threshold values at different sampling rates.

In all cases, the optimal threshold parameter achieves a better performance with an exception at the sampling frequency of 250 Hz.

Although, this conclusion is also valid for the QRS positive predictive rate, it can be noted that there are cases where the fixed threshold parameter .reaches higher values, such as for 200 and 225 Hz, or 300 and 330 Hz, but these discrepancies are very small and the selected optimal threshold parameter compensates for this to achieve higher QRS sensitivity at these frequencies.

13.4.3 Response Time and Performance Analysis of Sampling Rates

Fast response is an important issue, especially if the algorithm is to be built in wearable ECG sensing devices. Due to limited resources and energy supply, one would

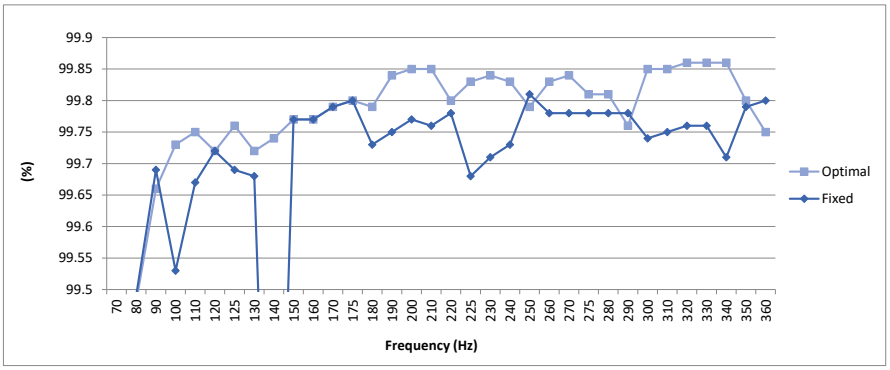


Fig. 13.7: QRS sensitivity at different sampling rates for optimal static threshold values.

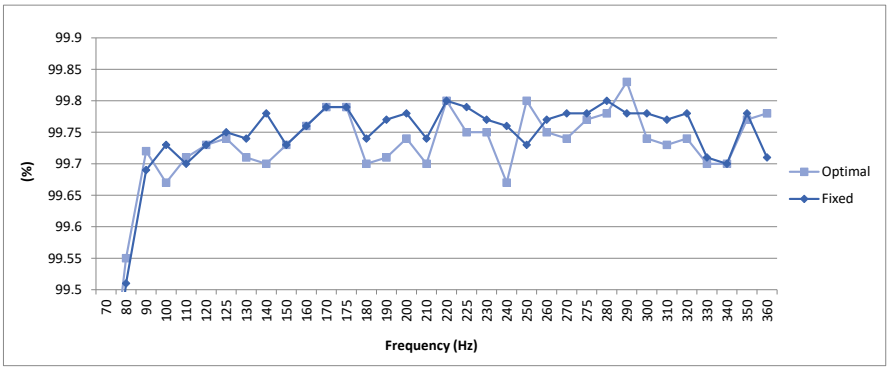


Fig. 13.8: QRS positive predictive rate at different sampling rates for optimal static threshold values.

appreciate an algorithm that achieves the best performance by executing fewer operations.

In most of the differentiation (derivation) or DSP based algorithms, the complexity of the algorithm is linear $O(n)$ meaning that it is directly dependent on the number of samples. This means that signals sampled on a sampling frequency of 125 Hz will have approximately three times less data than those sampled at 360 Hz sampling frequency.

Therefore, one would prefer a lower sampling frequency, which still achieves the best performance.

Note that using frequencies below 125 Hz generate a lot of false detections and decrease the expected QRS detection performance.

Nyquist [103] determines the minimal sampling frequency to be twice the maximum frequency of the signal to be converted into a digital stream. The highest frequency of the heart beat is 220 beats per minute, which is less than 4 Hz, the

analysis of the QRS spectra is in the range between 5 and 20 Hz. Some features, such as short QRS peaks shorter than 40 *ms* may go even beyond this limit, which corresponds to 25 Hz. Therefore, the sampling frequency should be more than 50Hz.

13.4.4 Comparative Analysis

Hamilton has measured the performance of the QRS detector at different sampling rates [70], and obtained the results presented in Table 13.1.

Table 13.1: QRS detector performance at different sample rates

	MIT BIH database		**AHA database**	
Sample Rate	Q_{SE}	Q_{+P}	Q_{SE}	Q_{+P}
100	0.996856	0.997905	0.995839	0.996423
125	0.997426	0.998257	0.996660	0.996788
150	0.997458	0.998016	0.997429	0.997652
175	0.997601	0.998093	0.997119	0.997268
200	0.997426	0.998071	0.997397	0.997588
225	0.997228	0.997994	0.997268	0.997769
250	0.997502	0.998060	0.997087	0.997300
300	0.997360	0.998016	0.997450	0.997897
325	0.997448	0.997874	0.997578	0.997684
360	0.997535	0.998038	0.997503	0.997865

The performances fluctuate in the range up to 0.08% between 99.68 and 99.76% for QRS sensitivity and a relatively smaller fluctuation difference of 0.03% between 99.79 and 99.82%, for the MIT BIH Arrhythmia ECG database. In the case of the AHA database, the obtained performances are lower, and the fluctuation difference is 0.18% for QRS sensitivity and 0.13% fluctuation difference for QRS positive predictive rate.

Ajdaraga and Gusev [13] have analyzed the accuracy of QRS detection. Their study is very precise, since a correct detection (true positive) is considered to be the one that is located at most five samples from the identified one. Therefore their reported accuracy is lower than the usual published values. In this study we use a window of 150 *ms* where the identified peak may differ in location from the real one, which is treated as a correct QRS detection. In addition, the authors use a fixed static threshold.

Malik et al. report on a conclusion from the American Heart Association Task force: [90] low sampling rates may produce a jitter and wrong QRS detection and so the sampling frequency range needs to be between 250 - 500 Hz. To achieve a better performance they suggest additional resampling interpolation that refines the signal for satisfactory QSR detection.

Berntson et al. [26] also report that sampling frequencies below 100 Hz will result with decreased performance and suggest 128 samples per second as the lowest frequency to be used. .

Ziemssen et al. [140] reported irrelevance in QRS detection performance at different sampling frequencies in the range between 100 and 500 Hz.

Ellis et al. [54] have conducted a series of experiments at sampling frequencies from 71.43 Hz up to 1000 Hz, and concluded a satisfactory performance even at the lowest analyzed sampling frequency, discrepancies above 125 Hz are especially negligible.

According to the above-mentioned papers, one might conclude that QRS performance may be satisfactory on a wide range of sampling frequencies without any problems. However, practical experiments confirm that threshold parameters need to be chosen carefully in order to obtain the optimal performance.

Chapter 14
Amplitude Rescaling Influence on QRS Detection

The content of this Chapter was published at the International Conference on Telecommunications, Springer [46], 2018.

Developing an industrial QRS detector has been a hot research topic since the late 80's. As advances in IoT rapidly continues, so do the trends in ECG processing. That is why, our primary focus in this research, is the *QRS detection* stage.

Hamilton has published a relatively good QRS detector [69]. In this chapter, we aim to optimize QRS detection performance especially on lower sample rates and amplitudes and to improve the original algorithm.

14.1 Hamilton's QRS Detection Algorithm

Algorithm details on Hamilton's QRS detector are already presented in [69]. Generally, algorithms for QRS detection publish their conceptual work, but lack implementational details.

For this particular case though, EP Limited [70] has released an open source implementation of the algorithm, which we will use to improve it. They provide a complete C-implementation for Hamilton's algorithm with different variations. It includes three different detectors and a fundamental beat classification unit.

Fig. 14.1 presents the processing steps to detect a peak and classify it as a beat. The processing is executed in two different phases, the first is *DSP Filtering* and the second *Peak Detection*. The primary aim of the first phase is to eliminate noise stemming from different sources, such as breathing, muscle or skin movements. This is important for proper beat detecting. Whereas, the latter phase aims to detect a peak, and classify it as a real beat or an artifact.

Phase I implements a total of five sequential steps: a 16 Hz low pass filter (LPF), an 8 Hz high pass filter (HPF), a differentiation filter ($\frac{d[i]}{dt}$), a filter that calculates an absolute value (ABS), and a filter that calculates an average over an 80 *ms* moving window of samples.

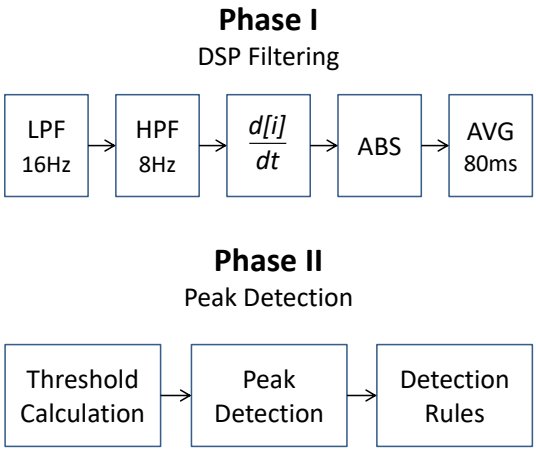


Fig. 14.1: Architectural representation of Hamilton's QRS detection algorithm.

The first two filters suppress environmental and internal noise, including baseline drift and act as a bandpass filter (BPF). The essence of the detection is the differentiation filter that aims to calculate a slope of differences. Since the slope can be negative, the ABS stage calculates only positive portions, and the AVG filter generates energy peaks that can be detected using rather simple rules in Phase II.

Phase II is responsible for peak detection by calculating a local maximum and comparing to two thresholds: *static threshold* (MIN_AMP_PEAK) with a fixed value, and *dynamic threshold* (DT) which is affected by the amplitudes of the last eight real and noise peaks. Any peak value lower than the static threshold will classify as a artifact, and any value over both the static and dynamic peaks will classify as a real peak. Otherwise it will classify as a noise peak.

The main optimization method used in this research is to choose an optimal value of the static threshold. In addition to this, we have explored the dependence of threshold values on different sampling frequencies and bit resolutions.

14.2 Testing Methodology

The experiments conducted during this study aim to find the dependence of sampling frequency and bit resolution on QRS detection performance. Particularly, we aim to find the optimal value and correct adjustment of the static threshold with scaled amplitudes to boost maximum performance. This optimization will improve the existing algorithm and reach higher performance.

14.2.1 Testing environment

The experiments were conducted on the MIT-BIH Arrhythmia database [99], which is considered as the most important benchmark for ECG monitoring. It contains two-channels of 48 half-hour ECG recordings. These recordings are publicly available on Physionet.org [63]. The sampling frequency is 360 Hz per channel, with a 11-bit resolution over a 10 mV range. Each record is accompanied by an annotation file made by physicians, and therefore enables a good-quality evaluation of the QRS detection algorithm.

Although the database has a total of 48 records, there are four records that contain paced beats, which may introduce a higher rate of errors (a paced beat followed by a QRS to be misinterpreted). Therefore, our test cases use only 44 records.

Each record contains two channels of data representation, and we used the first one which represents the ML II signal in most cases.

14.2.2 Test cases

Two experiments were conducted for the purpose of this research:

- *Exp.1*: Impact of rescaled amplitudes on the performance
- *Exp.2*: Optimal static threshold value to boost the performance

The first experiment *Exp.1* aims to determine the impact of different amplitudes on a 360 Hz sampling frequency. Test cases within this experiment start with the initial data records from the default MIT BIH Arrhythmia ECG database using an 11-bit resolution. The following test cases gradually scale the amplitudes by a factor varying from 1.00 to 0.25 with decremental steps of 0.05.

Note that if a scaling factor 1 is used, then it corresponds to the original signal records with a 11-bit resolution, and the factor 0.5 corresponds to signals with half of the amplitude, i.e. to a 10-bit resolution. Consequently, the test case with scaling factor 0.25 corresponds to a quarter of the original signal, and represents a 9-bit resolution.

The second experiment *Exp.2* addresses the optimal value of the static threshold. A crossed dependence check is performed to check the dependence of various amplitudes and the static threshold parameter on performance.

The test cases include measurements where the input is a pair of static threshold and the scaling factor of the amplitude. Static threshold values change from 2 to 10 with step 1, and amplitude scaling factor from 0.25 to 1 with step 0.05. Similar to the previous experiment, the test cases were conducted on the first channel of signals with an original sampling frequency of 360 Hz.

14.2.3 Test data

Performance evaluation of proposed algorithms are done with existing metrics based on measured correctness. Here we use the usual correctness classification:

- *Correctly detected beats* denoted as true positives *TP*;
- *Extra detected peaks* are considered as false positives *FP* meaning that the algorithm has detected a peak that is not a real beat;
- *Missed beats* identified as false negatives *FN* meaning that the algorithm has not detected real beats.

To measure the performance we will use number of errors as an indicator. The smaller the number of errors - the better the algorithm. Following the previous definition, the total number of (false) errors is equal to the sum of FP and FN, calculated by (16.2).

$$False\ Errors = FP + FN \tag{14.1}$$

Although the number of errors can be efficiently used to compare two different algorithms, we still have to compare this number with total number of QRS beats. The total number of QRS beats can be calculated as a sum of correctly detected beats and those that were missed according to (14.2)

$$Total\ QRS = TP + FN \tag{14.2}$$

So instead of number of errors one can use the performance measure called *Relative Error* (RE), that explains the relative magnitude of false errors compared to the total number of beats, which is calculated by (14.3).

$$Relative\ Error = \frac{False\ Errors}{Total\ QRS} = \frac{FP + FN}{TP + FN} \tag{14.3}$$

14.3 Evaluation of Results

This section evaluates the results from the experiments.

14.3.1 Performance Achieved on Rescaled Amplitudes

Fig. 14.4 presents the optimal performance of the execution of Hamilton's algorithm over signals sampled with 360 samples per second and 11-bit resolution. The x-axis portrays the amplitude scaling factor represented as a percentage, and the y-axis the number of false errors.

The presented results are based on measurements where Hamilton's original algorithm uses the default static threshold value MIN_AMP_PEAK $= 7$. It is obvious

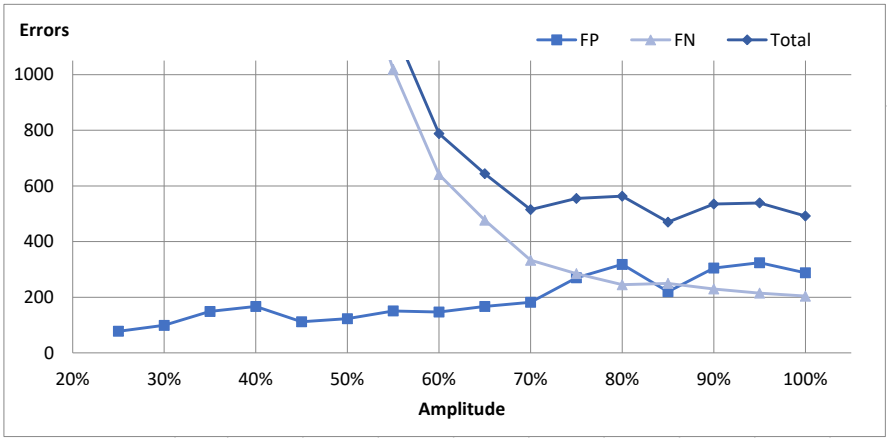


Fig. 14.2: False error detections of Hamilton's approach for MIT BIH Arrhythmia database signals

that the threshold parameter can not follow the downscaled amplitudes, and the number of generated false positives (extra generated peaks that are not real beats). Note that rescaled amplitudes with a scaling factor higher than 65% obtain a relatively good performance for the fixed static threshold parameter (the number of errors was lower than 600).

14.3.2 Optimal static threshold value to boost the performance

Presented results for the default static threshold $MIN\_AMP\_PEAK = 7$ show big discrepancies and performance fluctuations when the amplitudes are rescaled.

Second experiment defines test cases on different pairs of a static threshold parameter and amplitude scaling factor. The figures show charts where x-axis represents the amplitude scaling factor measured in %, and y-axis the static threshold parameter. The presented values are three dimensional, where the third dimension is a colored scale of relative error. The darkest squares are those with the highest performance and smallest relative error presented in %, and the lightest color is the worst performance and highest relative error.

Fig. 14.3 presents the Relative Error obtained by executing Hamilton's algorithm on different pairs of static threshold values and amplitude scaling factors.

The x-axis scale starts from 25% to 100%, and the y-axis from static threshold value 2 up to 10. The colored relative error scale starts from 0.4% (darkest) up to over 1.0% (lightest)

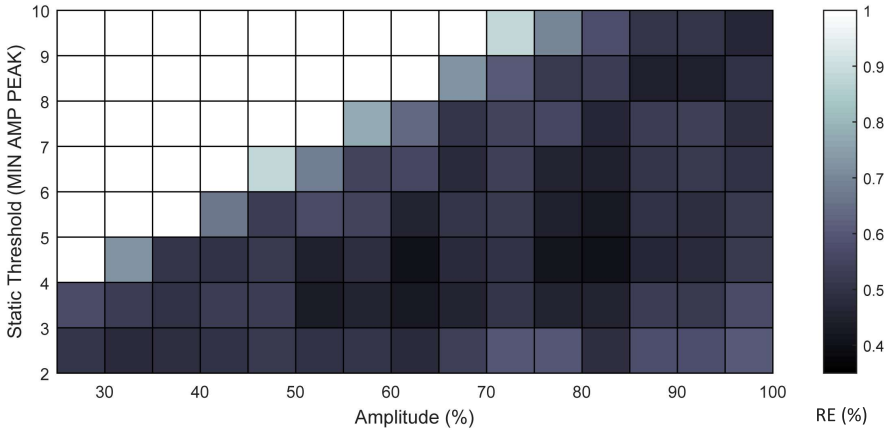


Fig. 14.3: Relative Error of Hamilton's approach with different static threshold and amplitude scaling factors.

We observe that the best configuration for original signals with a sampling frequency of 360 Hz is when the static threshold is 4 and amplitude multiplier is 65%, with a relative error of 0.402% (total false errors 405).

14.4 Discussion

14.4.1 Performance impact of rescaled amplitudes

Table 14.1 summarizes the evaluation of best and worst performance for the fixed static threshold, and gives the behavior expressed by the average number of errors and fluctuations when different amplitudes are used with the same algorithm and fixed static threshold parameter.

Table 14.1: The performance behavior for the fixed static threshold parameter equal to 7

Performance	Amplitude	Errors	Relative error
best	85%	470	0.467%
worst	25%	7085	7.033%
average	-	1726	1.713%
fluctuation	-	6615	6.566%

The fixed value of the static parameter with value equal to 7 showed fluctuation between 400 and 580 false errors in a dataset with 100733 beats. This is a relatively a high fluctuation, since it is equal to 45% of the false detections. However, the relative error in comparison to the total number of beats is between 0.578% and 0.397%.

When analyzing the fluctuations of the algorithm performance, one can conclude that the fixed static threshold parameter will only obtain good performance on selected amplitudes.

Next we will analyze which static threshold parameter achieves the best performance.

14.4.2 Selecting an Optimal Static Threshold

Table 14.2 presents the best performances from the default threshold and the best performance from different thresholds.

Table 14.2: Optimal static threshold parameters that reach the highest performance using the Hamilton's approach

Amplitude(%)	STHR	errors	$RE(\%)$
100	9	474	0.471
95	8	454	0.451
90	8	451	0.448
85	4	407	0.404
80	4	423	0.420
75	4	503	0.499
70	3	457	0.454
65	4	405	0.402
60	3	462	0.459
55	3	436	0.433
50	4	517	0.513
45	4	498	0.494
40	2	487	0.483
35	2	478	0.475
30	2	508	0.504
25	2	580	0.576

When we analyze results, we firstly observe that the default fixed static threshold does not yield the best performance. One can observe that the best static threshold is 9 instead of 7.

Interestingly, the default amplitude is not the best option either. We found that the best performance is achieved when rescaling by a scaling factor of 65% and a

static threshold of 4 achieved. In this case, the relative error is 0.402%, and the QRS sensitivity and positive predictivity rate are high (99.81% and 99.79% respectively).

Fig. 14.4 presents false error detections for the best chosen static threshold parameter from Table 14.2.

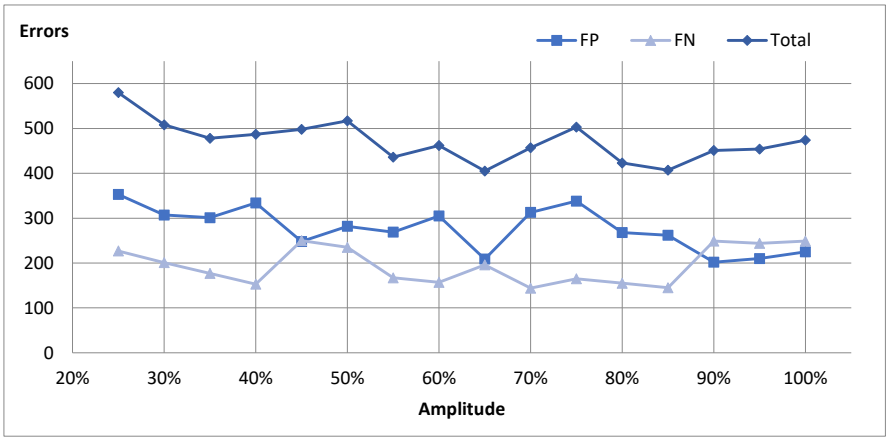


Fig. 14.4: False error detections of improved Hamilton's approach with optimal selection of threshold parameters for MIT BIH Arrhythmia database signals sampled on **360 Hz** and **11-bit** resolution.

Note, that the performance achieved with optimized static parameter (Fig. 14.4) is much better than the one obtained with a fixed static parameter (Fig. 14.4). Interestingly, signals using 65% of the amplitude reached The minimal number of errors is achieved for an amplitude scaled by a scaling factor of 65%.

In addition, we have also tested the influence rescaling had on different sampling frequencies. Fig. 14.5 presents the optimal performance of the improved Hamilton algorithm using the optimal static threshold parameter over resampled signals to a 125 Hz sampling frequency with the original 11-bit resolution.

On can conclude that the performance of the optimal chosen threshold parameter is improved even on different sampling frequencies.

Fig. 14.6 compares false error detections for typical values of bit resolutions by applying the optimal static threshold parameter from Table 14.2. Note that the performance difference of the algorithm for amplitudes using 9, 10 or 11 bits is negligible, whereas we observe big increase of the number of errors up to 2.246% for 8-bits .

Bit resolution is sometimes called sampling resolution. Note that low number of bits used by the AD converter or rescaling to an amplitude with a small scaling factor can yield an increased signal-to-quantization-noise ratio. Thus, in this case, it will produce an increased number of errors in QRS detection algorithms. Our experimental research has proven that 8-bit resolution will increase the number of errors significantly when compared to 9, 10 or 11-bit resolution.

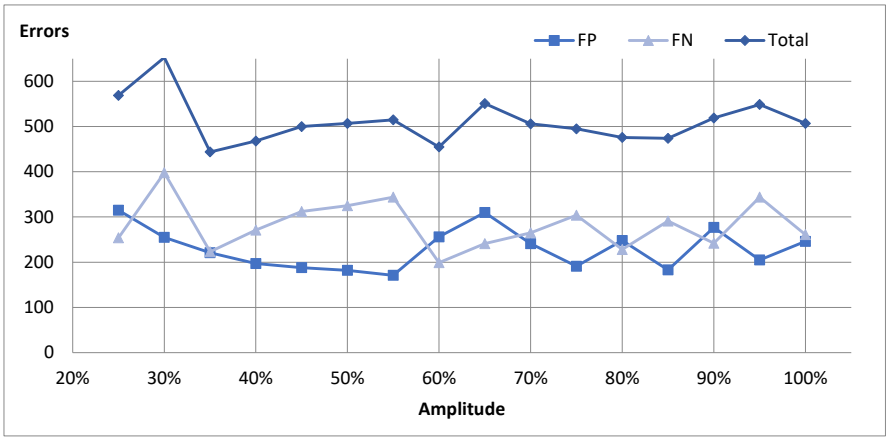


Fig. 14.5: False error detections of improved Hamilton's approach with optimal selection of threshold parameters for MIT BIH Arrhythmia database signals sampled on **125 Hz** and **11-bit** resolution.

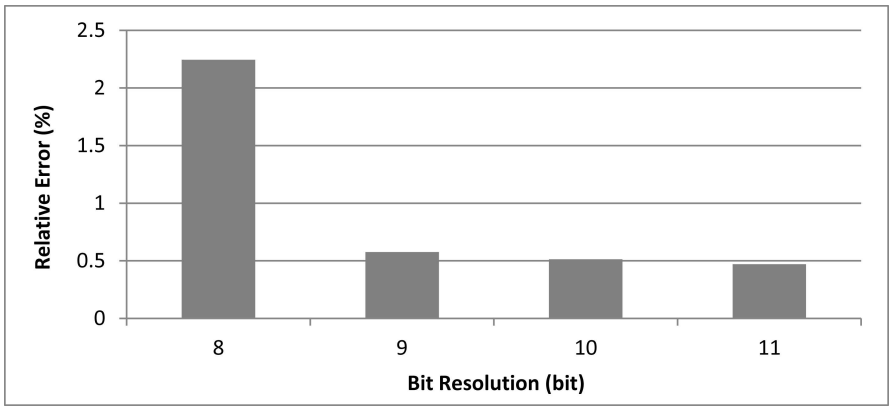


Fig. 14.6: False error detections comparison for different bit resolutions.

14.4.3 Comparison to Other Studies

Performance comparison with other approaches will be realized through conventional performance metrics for QRS sensitivity $QRS_{SE} = TP/(TP+FN)$. In addition, we also include the QRS positive predictivity rate defined by $QRS_{+P} = TP/(TP+FP)$. Sensitivity alone will not give us a good performance estimate, since one can tune threshold parameters to include as many real peaks as possible, but, at the same time, it will include extra false detections of peaks that are not real peaks. This measure shows how successful the algorithm is in capturing real beats

and avoiding false peaks. So the performance can be evaluated only if both QRS sensitivity and positive predictive rate reach higher values.

Available QRS detectors, generally focus on the original 11 bit 360Hz sampled data, though a small portion of them present results on different threshold values.

Table 14.3 gives an overview of relevant QRS detection algorithms. We selected those that report results for all 44 records without paced beats in the MIT BIH Arrhythmia database, or where the results can be verified. Some algorithms work on both signals and obviously they are not compared in this study. Note that most of the algorithms do not publish information on their performance was measured on one or two channels, and we present the algorithms that use one channel for QRS detection. In our algorithm, we apply the improved version of the Hamilton's algorithm that uses the optimal static parameter instead of the original fixed value.

Table 14.3: Performance comparison

Algorithm	**Sampling Frequency (Hz)**	**Bit Resolution (bit)**	**Total Errors**	**Relative Error[b] (%)**	QRS_{SE}	QRS_{+P}
M. Bahoura[ab] [21]	250	11	291	0.271	99.89	99.84
Our algorithm	360	11	405	0.402	99.81	99.78
Our algorithm	125	11	444	0.441	99.78	99.78
Hamilton [69]	360	11	569	0.564	99.68	99.76
Pan Tompkins[a] [107]	200	11	771	0.768	99.73	99.50
Afonso [10]	360	11	732	0.872	99.57	99.56
GQRS [63]	360	11	562	0.558	99.72	99.72
WQRS [63]	360	11	1411	1.401	99.79	98.82
SQRS [63]	360	11	1899	1.885	98.73	99.38
SQRS125 [63]	125	11	3951	3.922	96.19	99.88

[a] Values are computed according to the record-by-record tables in the referred works.
[b] QRS_{SE} and QRS_{+P} calculated by (14.3).

The most important observation is that the optimized Hamilton approach (our algorithm) produces better results than those reported in the original algorithm. More importantly, even downscaled signals will obtain good performance results if the static threshold parameter is tuned according to the provided results.

It is even more interesting to compare our results with the Pan Tompkins algorithm. Our algorithm running at 360Hz correctly finnds 375 more peaks, and makes 366 less errors.

Chapter 15
Conclusion on QRS Detection

The content of this Chapter was published at the Journal of Technology and Healthcare [47], 2019, 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), IEEE [64], International Conference on Telecommunications, Springer [65, 46], 2018.

15.1 Related Work of QRS Detection

Pahlm, and Sörnmo [106] comment that algorithms sustain a standard procedure even though their approaches differ. The steps they follow are: *Reducing noise* to an acceptable level, then applying a *Thresholding* and *Deciding* whether the peak is a beat.

In summary, published QRS detection algorithms are based on the following techniques:

- *Differentiation (derivation)*, where the difference between the current and previous samples is calculated, as a way of identifying the slope, and then it is compared to a given threshold value including the Pan & Tompkins algorithm [107], Hamilton's algorithm [69], or other relevant approaches [12, 61, 100];
- *Pure DSP algorithms*, where fundamental DSP filters with different characteristics are combined to produce a bandpass filter, with the aim to eliminate noise, and filter the signal so that a threshold will determine the beats [10, 28, 55, 58, 104];
- *Pattern Recognition algorithms*, where the signal data is matched with predefined patterns and a waveform is detected in case of similarity within given constraints of the amplitude and slopes [67, 32, 89, 126, 132];
- *Neural Network*, Multilayer Perceptron (MLP), Radial Basis Function (RBF), and Learning Vector Quantization (LVO) networks are used to adaptively predict the location of the next peak [138, 23, 39, 74, 89];

- *Digital Wavelet Transformation (DWT)*, where the signal is decomposed to a certain level of scales, and then recomposed, which effectively reduces noise. Then a threshold is applied to select proper peaks [87], [21], [121], [93], [98];
- *Genetic Algorithms* are used to optimize the preprocessing polynomial filter. The ECG signal is compared to an adaptive threshold and the parameters are optimized with a genetic optimization approach [113];
- *Hidden Markov Models (HMM)*, used to train the probability function varying according to the hidden Markov chain, and then the model predicts the current state, which could be QRS complex. P and T waves can also be computed [33, 16, 35];
- *Hilbert Transform*, where Hilbert transform of ECG signal is calculated by Fast Fourier Transform (FFT), and that is used for calculating the signal envelope [124, 102, 25]; and
- *Phasor Transformation*, where each ECG sample is converted into a phasor to correctly manage P, and T waves, by definition having lower amplitudes than an R-peak with low computational cost, and then compares it against a threshold [92].

High performance QRS detector directly affects the amount and the quality of valuable information on ECG. QRS detection is the initial step for further ECG analysis.

15.2 Overview of Obtained Results

15.2.1 Optimal DSP Bandpass Filtering for QRS detection

Fig. 12.3 presents the accuracy values for a FIR band pass filter with different values on the lower and higher cutoff frequencies. Note, that the maximal values are obtained in the band with cutoff frequencies of 4 to 20, which proves the theoretical analysis.

We do not present the measured sensitivity and precision diagrams since their diagrams are quite similar.

The frequency response of used FIR, IIR and DWT filters is presented in Fig. 12.4. To obtain a higher bandpass filtering one can use a smaller IIR length and needs a higher filter length of the FIR filter to obtain a better performance. The performance of the wavelet filter is even better.

The FIR filter is designed to reach the $-3db$ cut off value at desired frequency. The design of the IIR filter follows the same principles. While designing we have used different filter lengths of FIR and IIR filters. Filter length impacts the algorithm performance. For example, Fig. 15.1 presents the impact of FIR filter length on performance expressed by sensitivity, accuracy, and positive precision rate. One can notice that after a length of 100 the trend line stabilizes and reaches the maximal performance.

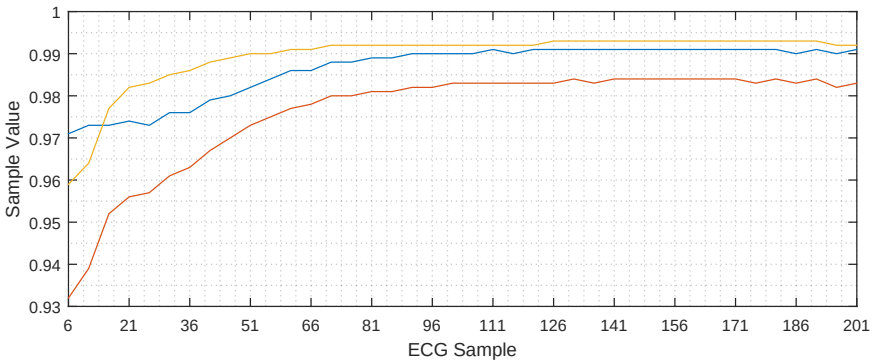


Fig. 15.1: Sensitivity, Accuracy and Precision dependency on FIR filter length.

Table 12.1 compares different filter approaches for using the filter between 5.625 and 22.5 Hz. The default configuration labeled as 'No filter' is the result of executing the our QRS detector algorithm without extra filter prior to the QRS detection. The values compared in the table are presented for FIR and IIR filters with length of 9 and DWT filter using db2.

Our algorithm is based on differential wavelet filter and the additional preprocessing bandpass filter just gives the advantage in achieved performance. The sensitivity, accuracy, and positive predictivity rate obtained in the best case for measuring overall 48 patients in the MIT-BIH database reached 0.992, 0.983 and 0.990 correspondingly. The implemented preprocessing filter actually improved the sensitivity of 3.1% and accuracy of 4.5% and reached performance values where the mistakes are less than 1.65%.

Note that some might think that wavelet transformation needs more processing power and is not as fast as the other filters. However, a recent study [98] shows that there are improved pipelined implementations that make the wavelet filtering fast.

We have concluded that QRS complexes are dispersed mostly in the range between 4 and 20 Hz. This was elaborated theoretically and proven experimentally. Therefore, adding a bandpass filter in this range will improve the QRS detection performance and reach values where the mistakes are less than 1%. Our research shows that choosing a FIR filter with a length of 101 gives sufficient performance, although IIR outperforms this filter for much smaller lengths. The wavelet-based filters are the best and the difference in obtained results is very small (less than 0.5%).

We recommend the intended implementors of ECG wearable and mobile devices to use the technique that allows stability of the filter and uses less complex operations that will enable faster processing with less resources.

15.2.2 Optimizing the Impact of Resampling on QRS Detection

The goal of this paper was to find out what happens with QRS detection when resampling the signals and using different sampling rates. For this purpose, we conducted experimental research and found that QRS detection performance at different sampling rates can be optimized if a proper static threshold value is used as opposed to a fixed value.

Our optimization of the Hamilton's algorithm results with a higher average performance of 99.86% QRS sensitivity and 99.80% QRS positive predictive rate at different sampling frequencies as opposed to 99.81% QRS sensitivity and 99.79% QRS positive predictive rate in the original Hamilton approach with a fixed static threshold.

To conclude, this optimization reveals that a good performance can be achieved at both higher and lower sampling frequencies. Our experiments show that a good quality industry QRS detector can work even with sampling frequencies of 100 Hz to achieve a performance higher than 99.80% of QRS sensitivity and QRS positive predictive rate.

In addition, lower sampling frequencies generate less data, therefore the number of operations is lower, which reduces response time and energy requirements.

This research, along with the analysis of amplitude influence on QRS detection performance, motivates us to establish a model of QRS detection behavior for resampled and rescaled ECG signals. In addition, we wish to produce a quality QRS detector independent of the sampling frequency and rescaled amplitude.

15.2.3 Amplitude Rescaling Influence on QRS Detection

In this study we primarily focused on finding the influence of the signal amplitude on QRS detection performance.

We have observed that rescaling the amplitude directly affects the performance and increases the number of false detections. The reason why is the selection of the static threshold.

The experiment showed that choosing a fixed value of the static threshold will not yield the best performance. Therefore, one needs to tune its value in order to obtain a good performance.

Our research showed which optimal values of the static threshold result with best performance, and one needs to update this value according to the input amplitude. This optimizes Hamilton's algorithm and makes for a better solution, when is compared to other approaches.

This study is useful for future designers, gives the optimal value of the threshold parameter to be used by an AD converter for satisfactory QRS detection performance. For example, we concluded that 8-bit resolution will not yield performances over 98%, while using a 9, 10 or 11-bit resolution may achieve performances of QRS_{SE} and QRS_{+P} over 99.80%.

Part IV
Improved QRS Detection and Beat Classification

Chapter 16
Improving the QRS Detection for One-channel ECG Sensor

The content of this Chapter was published at the Journal of Technology and Healthcare [47], 2019.

The performance of the QRS detector is evaluated by calculating how many real QRS peaks are found (QRS sensitivity, denoted as Q_{SE}), and how many of those detected QRS peaks are real beats (QRS positive predictive rate, denoted as Q_{+P}). The default testing database was the Massachusetts Institute of Technology - Beth Israel Hospital Arrhythmia database (MIT-BIH) [99] with 48 records of 30 minute ECG measurements. The original signals are sampled with 360 Hz and 11 bit AD conversion. Our research target is a QRS detector for signals using a sampling frequency of 125 Hz and 10 bit AD conversion.

One of the most cited papers for QRS detection built for small devices with limited resources is the Pan & Tomkins algorithm [107]. Its robustness lies in the fact that it runs fast enough to be used in real time and can cope with noisy signals. However, the performance of this algorithm depends on bit resolution in the AD conversion. In our case, when using one channel ECG sensor and smaller sampling frequencies, its performance was not satisfactory, especially for signals with smaller amplitudes. It was not a good solution for us because of these factors.

Another alternative is the Hamilton algorithm [69]. Compared to the Pan & Tompkins algorithm, it is quite similar but uses different filters and decision rules. It is a stable solution, but, still is unable to cope with small amplitudes, or variations in consecutive amplitude levels, especially, when using a smaller bit resolution in AD conversion.

Physionet.org [63] is a very comprehensive resource where one can find several QRS detection algorithms, including *Wavedet*, *gqrs*, *wqrs*, and *sqrs*. They represent simple and fast algorithms demanding a small number of resources and obtain high sensitivity and positive predictive rate values. However, they lack the beat classification, and the obtained sensitivity and the positive predictive rate are also considered to be lower than the demands of a quality industrial QRS detector for a wearable one channel ECG sensor.

After these initial efforts, the attention of researchers gradually focused on developing more sophisticated QRS detection algorithms, including Machine Learning

and other methods, as described in Section 18.1 (related work). Although some of the new approaches achieved better performance, they generally require computationally intensive algorithms, not suitable for smartphones that collect continuous ECG data from wearable sensors.

In this chapter, we improve Hamilton's algorithm [69] in order to make it efficient for industrial application. The improvement was a rather long process due to the exponential nature of the effort to improve the algorithm. The closer you are to the margin of 100%, the more effort is needed for a very small improvement of the performance. We introduced several hundred rules to cope with the identified problems in QRS detection, and several thousand tests to tune parameters, and threshold values for identified solutions. Some threshold values obtained good performance on some test file while performing badly on others. When we fine-tuned some parameters, it so happened that some of the rules did not work on other test datasets, which was even more challenging.

16.1 Background

In this section, we will explain the evaluation metrics and give an overview of the original Hamilton's QRS detector algorithm.

16.1.1 Performance measures

The benchmarks used in our testing methodology are the same used in the IEC 60601-2-47 standard for particular requirements for the safety, including an essential performance of ambulatory electrocardiographic systems, and ANSI/AAMI EC57:2012 for Testing and Reporting Performance Results of Cardiac Rhythm and ST Segment Measurement Algorithms. These standards use the MIT-BIH ECG arrhythmia database [99], and the American Heart Association's (AHA database) [72].

MIT-BIH contains half-hour ECG recordings for 48 anonymized persons, and only 44 records exclude those that contain paced beats. These recordings are publicly available on the physionet.org web site [63]. The recording frequency is 360 samples per second, per channel, with a 11-bit resolution. Even though each recording contains two-channels, we used the first channel, identified as ML II, in most of the records.

In addition, we follow the requirements according to the standard IEC 60601-2-47:2012 for medical electrical equipment, particularly, requirements for essential performance of ambulatory electrocardiographic systems. According to these requirements, any calculated peak is considered as detected if it is at most 150 *ms* away from the real beat.

A detected QRS is denoted to be True Positive (TP), if the QRS detector has found a QRS closer than 150 *ms* from the one which is annotated. A False Negative

(FN) is a missed QRS, or if the QRS detector has found a QRS outside the 150 *ms* perimeter, while a False Positive (FP) is an erroneously detected QRS (extra found).

The commonly used performance measures are sensitivity and positive predictive rate, calculated by Eq. 17.1.

$$SE = \frac{TP}{TP+FN} \qquad +P = \frac{TP}{TP+FP} \tag{16.1}$$

In addition, to find an optimal value of a parameter, we provided a lot of test experiments, and calculate the number of *Total Errors* by Eq. 16.2, as a sum of errors denoted by FP and FN. The smaller the errors are, the better performance is achieved.

$$Errors = FP+FN \tag{16.2}$$

16.1.2 Analysis of Hamilton's Algorithm

EP Limited's open source software for arrhythmia detection serves as a basis for this research [70, 69]. It has a complete C-code implementation of Hamilton's algorithm, with three different detectors, and a simple beat classifier. Two of the detectors are for general-purpose, whereas the third one is for environments with a small amount of memory.

Algorithmic details are theoretically provided in their original work [69]. Fig. 14.1 represents the conceptual level for the two phases, and the high-level steps conducted for each of them.

After eliminating the noise in the DSP filtering phase, the algorithm continues with the peak detection phase. It already has two thresholds for the AVG signal, classified as a

- *static threshold* with a fixed value, and
- *dynamic adaptive threshold* (DAT) which is affected by the amplitudes of the latest peaks.

The original algorithm sets the static threshold ($STHR$) at value *MIN_PEAK_AMP* = *7*. A general rule of thumb is that a lower value of the static threshold will find more peaks, but also detect lots of artifacts. On the other hand, a higher static threshold value yields fewer peaks, but a smaller number of artifacts.

When a new local peak is found, the *dynamic adaptive threshold* is calculated by taking the mean values for real peaks and noise peaks into account. The mean value is computed by Eq. 16.3.

$$mean = \frac{\sum_{n=1}^{8} X_n}{8} \tag{16.3}$$

Let the mean value for real beats and noise peaks be denoted as *qmean* and *nmean* respectively, and also *TH* be the constant multiplier (with a default value of 0.3125), then the DAT is calculated by Eq. 16.4.

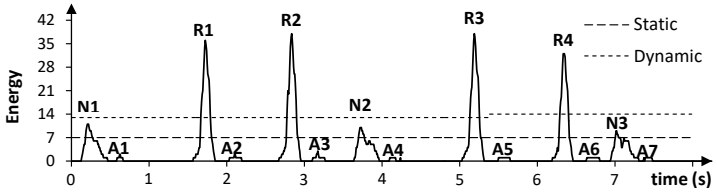


Fig. 16.1: Detecting artifacts, noise and real peaks based on values of static and dynamic thresholds in the original Hamilton's algorithm presented on signal extract over MIT-BIH record 124 (1046 sec).

$$DAT = nmean + (qmean - nmean) * TH \tag{16.4}$$

When a new local maximum is detected by calculating an AVG value, both of the thresholds are compared to this value. If the value is higher than the static threshold, then it is considered to be a *potential peak*, otherwise, it is an *artifact*. It is classified as a *noise peak* if the calculated value is lower than the dynamic peak, and as a *real beat* if it is higher than the dynamic adaptive threshold.

Fig. 16.1 presents both dynamic and static thresholds. Note, that detected peaks A1, A2, ..., A7 are categorized as artifacts (smaller than the static threshold), and R1, R2, R3 and R4 as real beats. N1, N2, and N3 are considered noise peaks since the local maxima of each label are smaller than the dynamically calculated threshold.

16.2 Identification of performance issues

A discrepancy in peak amplitudes may introduce bad detection. We have identified two cases when this happens:

- a sequence of low-amplitude peaks after an isolated high-amplitude peak;
- an isolated low-amplitude peak after a sequence of high-amplitude peaks.

Furthermore, apart from fine-tuning the threshold value, and expecting lower performance on cropped signals, we analyzed the segments where the algorithm showed lower sensitivity, and specificity even though the signal was not contaminated with noise. The conclusion was that lower performance was obtained for specific segments, and the problems can be classified as:

- a mixture of low and high-amplitude peaks;
- artifact elimination; and
- wrong R-peak location.

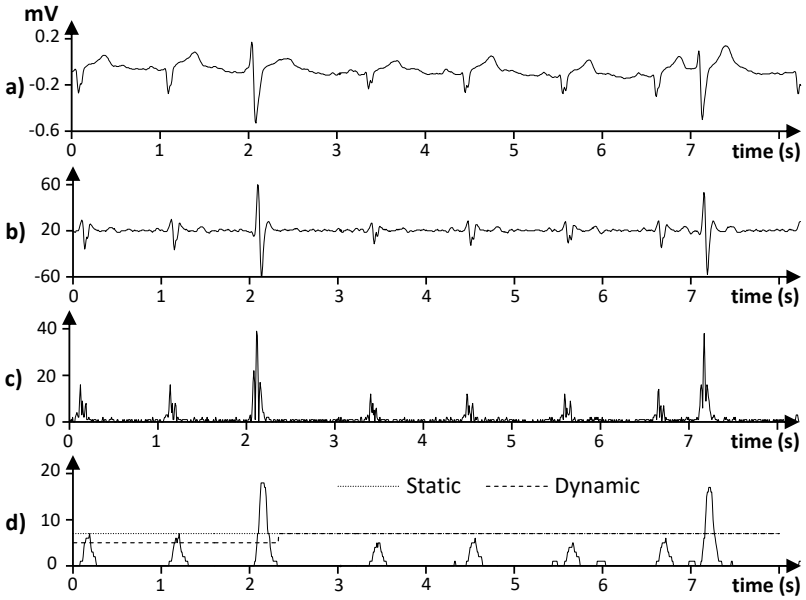


Fig. 16.2: Signal extracts of executing the Hamilton algorithm over the MIT-BIH record 114 (240 sec) a) Original ECG signal; b) Output after bandpass filtering; c) Output after differentiation and absolute calculation; d) Output after average over an 80 *ms* window.

16.2.1 Bad detection of low-amplitude peaks

Fig. 16.2 presents the case when an isolated high-amplitude peak is preceded and followed by a sequence of low-amplitude peaks. An 8-second segment extract of MIT-BIH record 114 is displayed including the original signal and outputs after executing each of the processing steps BPF, ABS, and AVG.

Fig. 16.2 d) identifies the static and dynamic thresholds and shows the case where the beats between the two high amplitude are considered as artifacts, although they should be real QRS beats.

The reason for bad detection of low-amplitude peaks after high-amplitude peaks is primarily due to the high level of static threshold. Even if one makes a correction by decreasing the static threshold value to include these peaks, there will still be a problem regardless of the fact that the peak will be treated as a candidate, and the dynamic threshold check will be applied. This is because the high-amplitude peak will increase the dynamic threshold value caused by the calculation of the mean, and so the peaks will be classified as noise peaks.

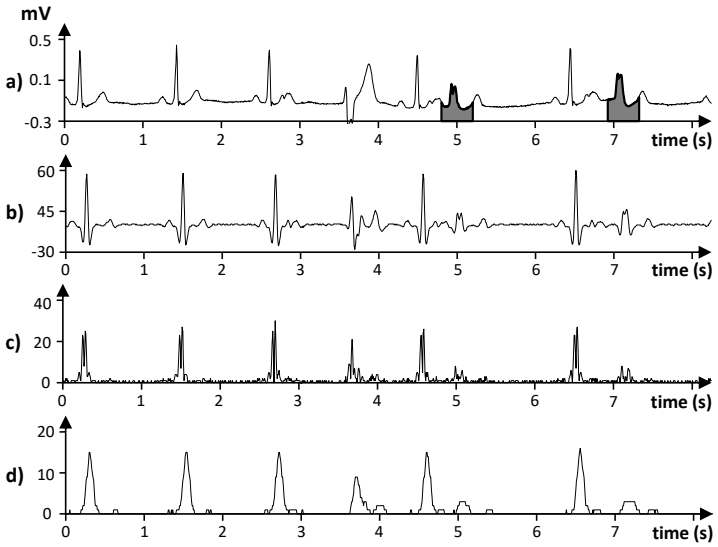


Fig. 16.3: Signal extracts of executing the Hamilton algorithm over the MIT-BIH record 201 (424 sec) a) Original ECG signal; b) Output after bandpass filtering; c) Output after differentiation and absolute calculation; d) Output after average over an 80 *ms* window.

16.2.2 Isolated peaks after sequences of high-amplitude peaks

High-amplitude peaks directly affect the calculation of *dynamic adaptive threshold*. It is recalculated each time a new local maximum is found with an amplitude higher than the *static threshold*. In this case, it is considered as a potential peak, and the values lower than the dynamic threshold are considered as a noise peak, while the others as real QRS peaks. The original algorithm buffers the latest eight peak amplitudes and calculates a DAT value by Eq. 16.4.

An extensive analysis of the MIT-BIH record 201 shows too many misses, especially in cases of aberrated atrial premature beats (classified as a beat) as illustrated in Fig. 16.3. Two of the beats highlighted by default cannot be captured due to the *dynamic adaptive threshold* and mean calculation, since most of the latest beats have a high amplitude. In this case, neither the static nor the dynamic adaptive threshold will work. The example is highlighted in Fig. 16.4 by peaks *C* and *D*, which should be classified as QRS peaks, but they are detected as artifacts because their value is lower than the static threshold.

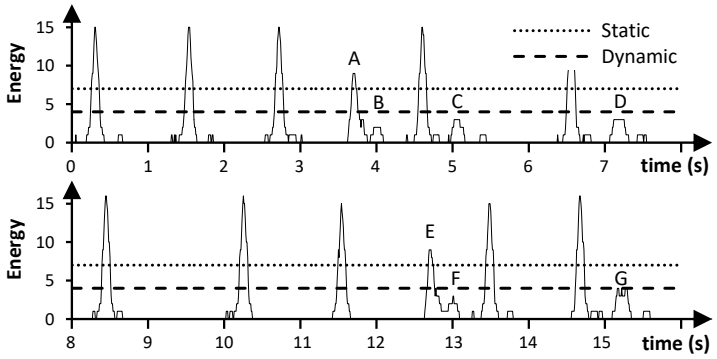


Fig. 16.4: Static and dynamic threshold values on the output after average over an 80 *ms* window on MIT-BIH record 201 (424 sec).

16.2.3 Classification of artifacts

Dynamic adaptive thresholding identifies noise and real peaks. Although in most cases, the dynamic adaptive threshold reacts properly, there are still cases where a noise peak is incorrectly calculated as real. Correct classification of artifacts is of primary importance for a quality industrial QRS detector.

Increasing the *static threshold* directly decreases the number of artifacts, however, this drastically increases the missed beats. This also applies to the *dynamic adaptive threshold*. To find an optimum of the static and dynamic peaks means to search for a comprise that would reach high values of both sensitivity, and positive predictive rate.

An example is illustrated in Fig.16.4. Peaks labeled as A, C, D, E and G are low amplitude peaks. However, peaks labelled as B and F are artifacts. With the default threshold, A and E peaks are considered as candidates for QRS, whereas the rest are considered as artifacts. For this particular case, decreasing the static threshold to 3 would catch all of the real peaks, but, will consider artifact peak F as a peak. On the other hand, keeping the static threshold at 4 will only detect G as a peak, and the other peaks will remain artifacts again.

16.2.4 Calculation of R-peak location

One of the issues in executing the original Hamilton algorithm is the proper detection of a QRS peak. Fig. 17.8 illustrates such a case, where local maxima are labeled with A, B, C, D, and E. Another peak appears in the output after bandpass filter denoted by F, as seen in Fig. 17.8 b). The proximity of marked peaks B, C, and F cause two local peaks on the output of 80 *ms* average window, marked as B and C on Fig. 17.8 c). When static thresholding is applied to the time average over

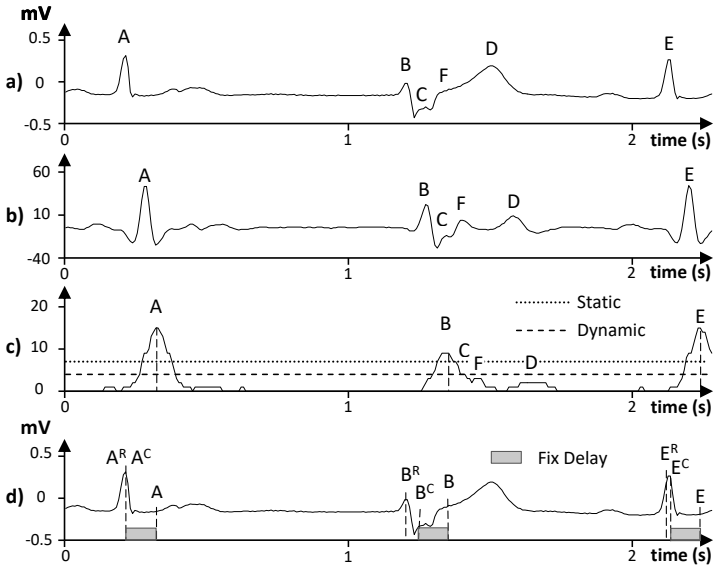


Fig. 16.5: Signal extracts and R-peak detection over the MIT-BIH record 201 (426.4 sec): a) Original signal and local peaks A, B, C, D and E; b) Output after bandpass filter; c) Output after average over an 80 *ms* window; d) Real and calculated R-peak locations with constant delay introduced by the filter.

an 80 *ms* window, C, F, and D peaks are detected to be artifacts, whereas A, B, and E are identified as potential peaks. Since the dynamic threshold is below the static, these beats are classified as real.

Note that the filter makes a constant delay, which is deducted from the location of peak values (as displayed on the signal average output). It correctly determines properly the previous QRS peak A, and the next QRS peak E, but makes a mistake in determining the peak B.

The original Hamilton's algorithm detects the location of R-peaks to be at points A^C, B^C, and E^C, even though they have different real peak locations A^R, B^R, and E^R.

Hamilton's algorithm detects a peak based on a calculated amount of delay [70]. Although filters produce a fixed delay, the author has pointed out that the detection delay can easily vary from 395 *ms* to 1 *sec*, depending on the heart rate and detection rule. It is important to note that the *search back detection* method [70] *can produce a fixed delay* if the search back algorithm does not report any local peaks. Thus, we can conclude that the original Hamilton's algorithm provides the best likely position of the R-peak, however, it is not always exact. This problem can particularly increase the total number of FP's. This particular case is observed in almost all MIT-BIH records. Even though the difference is not so big, there are cases where the difference between the real and detected location is higher.

16.3 Algorithm improvement

Increasing the algorithm performance is directly related to fine-tuning thresholds and algorithm improvements.

16.3.1 Improving the detection of low-amplitude peaks

A careful analysis shows that the step (ABS) executed prior to time average, will not be able to cope with the bad performance in detection of low-amplitude peaks. This is especially crucial in cases when the signal is a mixture of one high amplitude beat, and then followed by several beats with low amplitudes. We used the idea introduced in the Pan Tompkins algorithm [107] to square the signal, instead of calculating the absolute value.

Fig. 16.6 presents a case where a combination of a square mode and the optimized static threshold will improve the detection of low energy peaks. The peaks labelled as A, C, E, F, H, and I are real R peaks, whereas B and D are artifacts. The original algorithm, which uses the calculation of an absolute value and average along with the static threshold, is not able to classify F, and I as real peaks. However, the square mode and signal average in combination with a new (smaller) static threshold, is able to detect that there is sufficient energy for a potential peak. The square average signal also marks B, and D as potential peaks. Such peaks can be reduced with the dynamic threshold or by introducing rules for artifact detection.

Nevertheless, there are side effects, especially in the calculation of the dynamic adaptive threshold. This threshold increases due to increased amplitudes, and it becomes slightly difficult to adapt to sudden changes in the amplitudes.

This operation behaves as an important amplifier especially if it followed by a calculation of an average over an 80 *ms* moving window. It shows that this copes better with the identified problems. However, this is not enough, since this algorithm cannot perform with static threshold values, and needs dynamic calculation by other rules.

We have conducted several tests to experiment with threshold values $STHR$ from 2 to 50, to find the optimal threshold value for which the number of errors is minimal. The left part of Fig. 16.11 presents the false detections for the conducted experiment. The performance of the algorithm gradually decreases as the threshold increases, mainly due to the high values of FP. The best performance is obtained for $STHR = 2$. We noticed a decreased number of FN, which gives an idea for how to get better performance if we decrease the number of FPs through other methods.

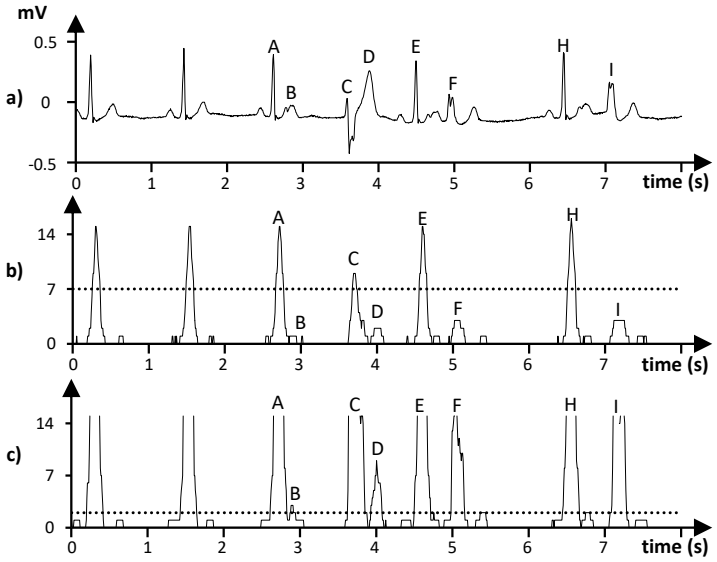


Fig. 16.6: Signal extracts of executing the Hamilton's algorithm over the MIT-BIH record 201 (424 sec): a) Original signal; b) Output after average over a 80 *ms* window using the original Hamilton's algorithm; c) Output after square average over a 80 *ms* window with the optimized static threshold.

16.3.2 Improving the calculation of the R-peak location

Since the Hamilton algorithm only gives an approximation where the peak is, one way to reduce the number of FP's that occur due to the determination of a proper R peak location, is to search the real peak in near proximity. The idea is to search the most convenient local maximum by analyzing the noise-eliminated output after bandpass filtering.

The original Hamilton's algorithm will determine the best approximate for the location of the R-peak. This is the starting point for the search of the local maximum, within a range, found *SearchL ms* to the left, and *SearchR ms* to the right of the approximated R-peak location. Once the local maximum is found on the output of the bandpass filter, we continue to find the local maximum on the actual signal, though the range for searching will be limited to 48 *ms*.

Fig. 16.7 illustrates the basic steps of this improvement algorithm in the example presented in Fig. 17.8. The corresponding search segments are marked on the original signal. The original Hamilton's algorithm detects the local peak B^C, and our improvement algorithm finds B^{RC} to be the real location.

We realized another experiment to locate the optimal values for *SearchL*, and *SearchR*. False detections for thresholding values are plotted on a surface graph presented in Fig. 16.8, for different values for *SearchL*, and *SearchR*, using the static

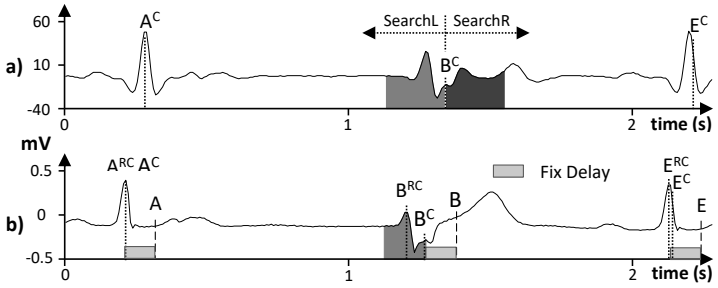


Fig. 16.7: Proper calculation of the R-peak location on the MIT-BIH record 201 (426.4 sec): a) Calculated R-peak location B^C and the search intervals over the output of bandpass filter; b) The re-calculated R-peak location B^{RC} on the actual signal.

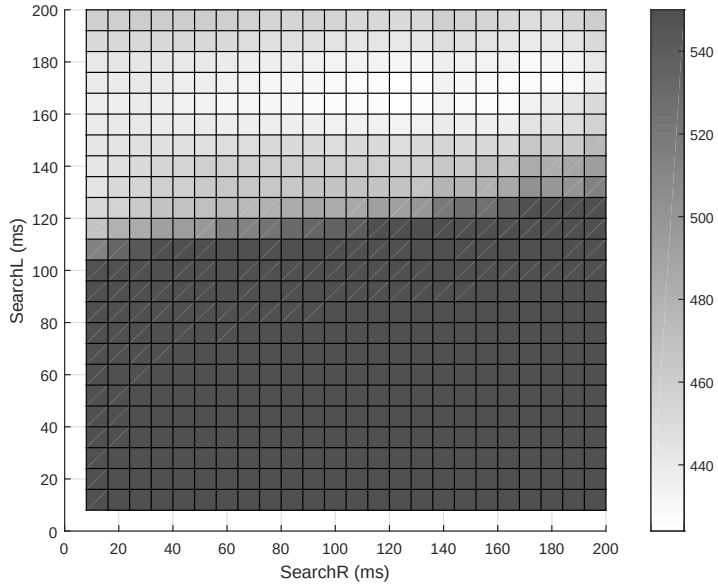


Fig. 16.8: False detections improving the calculation of the R-peak location.

threshold value $STHR = 4$. The best results are obtained when *SearchL*=160 *ms*, and *SearchR*=120 *ms*.

16.3.3 Improving the detection of low-amplitude peaks after sequences of high-amplitude peaks

Two functions, denoted as *mean* (Eq. 16.3) and *thresh* (Eq. 16.4), play a crucial role in the original QRS detection algorithm.

We proposed a change in the mean function in order to alleviate the effect of high amplitude complex proceeded by a significantly lower one. Instead of calculating the mean of the last 8 peaks, our algorithm considers only half of this value when the peak amplitude is higher than the dynamic threshold ($DTHR$) and the exact value in all other times, as defined by Eq. 16.5. This prevents a linear increase in the threshold especially for high amplitude signals and solves the identified problem.

$$mean = \frac{\Sigma_{n=1}^{8} \begin{cases} X_n, & \text{if } X_n < DTHR. \\ \frac{X_n}{2}, & \text{otherwise.} \end{cases}}{8} \tag{16.5}$$

In addition, we changed the *thresh* method. Previously, the calculated threshold was multiplied by the constant $TH = 0.3125$. Since the square mode filter is used, we updated the multiplication constant by its square, i.e $TH * TH$. Thus the new DAT calculation is defined by Eq. 16.6. Both these interventions enabled the detection of such beats.

$$DAT = nmean + (qmean - nmean) * TH^2 \tag{16.6}$$

A good performance is achieved on both cases with:

- a sequence of low-amplitude peaks after an isolated high-amplitude peak;
- an isolated low-amplitude peak after a sequence of high-amplitude peaks.

Fig. 16.9 illustrates the improvement idea for the example presented in Fig. 16.2, where a sequence of low-amplitude peaks is followed by a high-amplitude peak, which will increase the dynamic threshold to a value where all consequent low-amplitude peaks are marked as noise peaks.

The figure demonstrates the output of an average of an 80 *ms* window realized on squared (not absolute) values along with the static, and dynamic thresholds. Note that the detected potential peaks and their absolute values presented in Fig. 16.2 are smaller than the original dynamic threshold value.

The effect of applying the new way to calculate the dynamic threshold can be also observed in Fig. 16.10, presenting isolated, low-amplitude peaks after a sequence of high-amplitude peaks. The new minimum threshold detects 17 candidate peaks, whereas the original algorithm with default dynamic threshold is not able to capture the peaks labeled as C, D and G. The newly optimized threshold is able to capture all 15 peaks correctly, and also classify B, and F peaks as noise.

We have conducted a lot of experiments to determine the optimized value for the $DTHR$ threshold value. The test cases included testing the threshold values of 100 up to 400. The middle part of Fig. 16.11 presents false detection as a function of

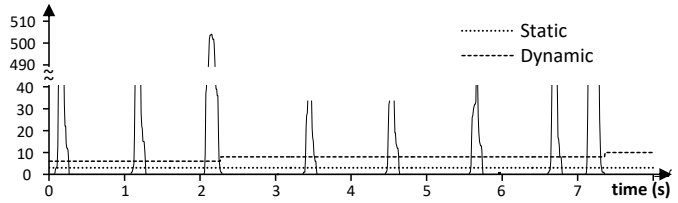


Fig. 16.9: Effect of a dynamic threshold to detect peaks after average of squared values over an 80 *ms* window over the MIT-BIH record 114 (240 sec).

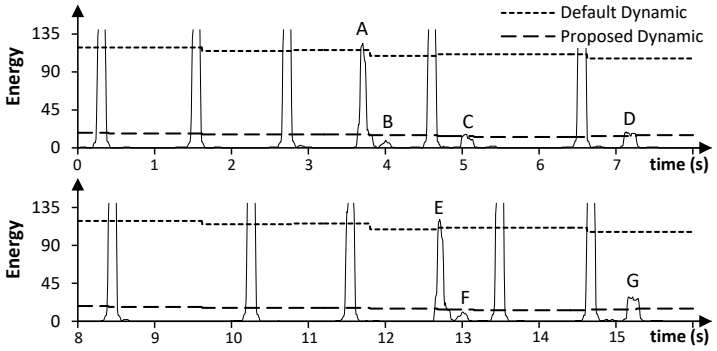


Fig. 16.10: Effect of a dynamic threshold to detect peaks after average of squared values over an 80 *ms* window over the MIT-BIH record 201 (424 sec).

the threshold values. Threshold values near 200 are the best candidates for the most effective performance.

16.3.4 Improvement of Artifact Elimination

In the final step, we introduce a new *Classification* phase. The aim of this is to classify whether the calculated beat is real or an artifact. Three important decision rules decide whether the peak is an artifact. If none of these rules are satisfied, the beat is calculated as a real peak.

From the preliminary analysis, we observed that artifacts generally follow a real beat and are closer than 320 *ms* away. The second important issue is that an artifact obviously has lower energy when compared to the previously detected beat. The original Hamilton's algorithm eliminates artifacts closer than 195 *ms*. Our findings show that this value can also be optimized. Table 16.1 describes some parameters used in our optimization approaches. We introduce the following optimization rules for artifact elimination:

$C = Artifact$ if

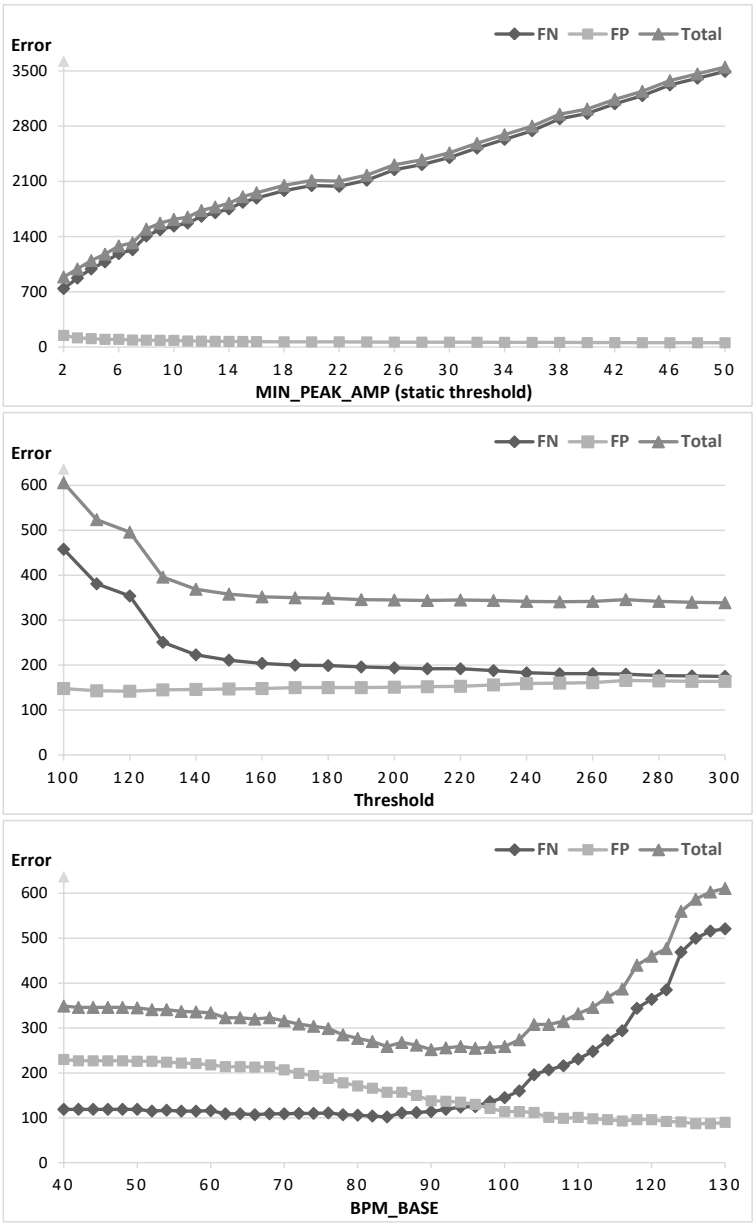


Fig. 16.11: False detections of optimization approaches for low-amplitude peaks (left); sequences of high-amplitude peaks (middle), beat rate impact (right).

A0: $RR \leq TA0 = MS250$
A1: $RR \leq TA1 = MS260 \;\&\; PH/CH > THRA1$

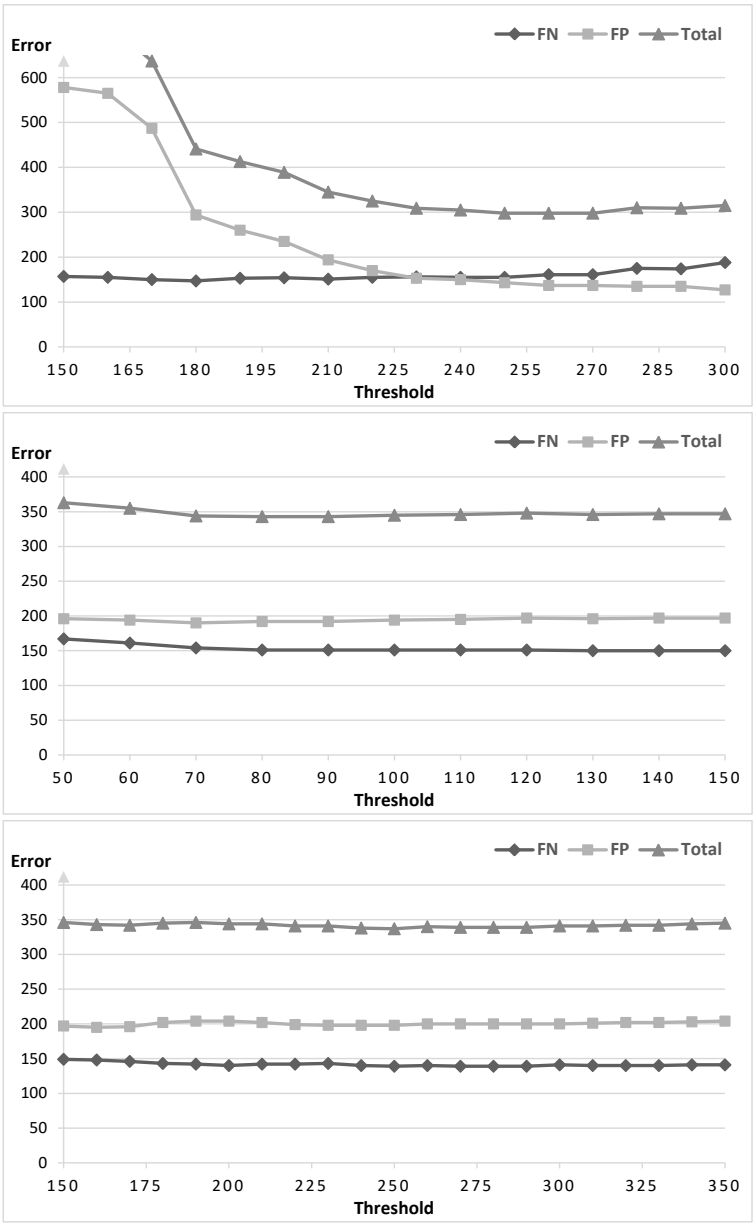


Fig. 16.12: False detections of optimization approaches A0 (left), A1 (middle) and A2 (right).

A2: $RR \leq TA2 = MS320 \;\&\; PH/CH >= THRA2$

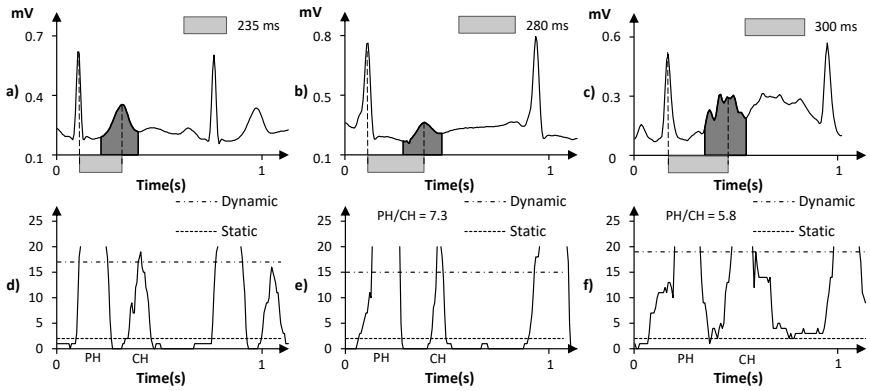


Fig. 16.13: Signal extracts and outputs after squared average over 80 *ms* window of executing our algorithm over MIT-BIH: a) signal and d) output for A0 type artifact in record 103 (1304.4 sec); b) signal and e) output for A1 type artifact in record 124 (413.3 sec); c) signal and f) output for A2 type artifact in record 101 (132.2 sec).

Table 16.1: Parameter List for Artifact Detection Rules.

Parameter	Description
C	Current Detected Peak
P	Previous Detected Beat
CH	Current Peak Time Average Height
PH	Previous Beat Time Average Height
RR	Current beat to peak interval in *ms*
TAx	Time in *ms* optimizing Ax, $x \in \{0,1,2\}$
THRAx	Parameter optimizing Ax, $x \in \{1,2\}$
MSxxx	xxx *ms* interval

Examples of different types of detected artifacts are presented in Fig. 16.13. The identified segments demonstrate that the detected peaks are artifacts, since their energy is higher than static, and dynamic thresholds, but satisfies one of the A0, A1, and A2 rules. Otherwise, if none of these rules are satisfied, then the detected beat C is not an artifact.

The results of the experiment using the A0 optimization approach, are presented in the left part of Fig. 16.12. A threshold value of $TA0 = MS250 = 250$ *ms* yields promising results, due to the lowest level of false detections.

The impact of threshold values on the A1 approach is demonstrated in the middle part of Fig. 16.12. The x-axis denotes values which are multiplied by 100 and the optimized value 80 for $THRA1$ corresponds to $80/100 = 0.8$.

The right part of Fig. 16.12 shows how the threshold parameter impacts the performance of the A2 optimization method. The x-axis denotes values that are multiplied by 100, and the optimized value 250 corresponds to $250/100 = 2.5$ for $THRA2$.

16.3.5 Beat rate impact on artifacts

Our experiments have shown that the beat rate affects the intervals in the A0-A2 optimization approaches. Let RR_{avg} be the average value of the last R to R intervals, and f_s the sampling frequency. Then beat rate BPM is measured by beats per minute, and calculated by Eq. 16.7.

$$BPM = \frac{60}{(RR_{avg}/f_s)} = \frac{60 f_s}{RR_{avg}} \tag{16.7}$$

To classify whether a peak is an artifact, we have set three time constants $TA0 = 250\ ms$, $TA1 = 260\ ms$ and $TA2 = 320\ ms$. Usually, the peaks closer than 250 *ms* are considered as peaks (corresponds to a beat rate higher of 240 BPM). The second and third thresholds correspondingly check if the peak is closer than 260 *ms* (corresponds to 230 BPM) or 320 *ms* (corresponds to 188 BPM).

However, our analysis has shown that premature beats might appear closer than these values. This is why we introduced a scaling factor to the previous improvement, and use the time thresholds calculated by Eq. 16.8, which are multiplied by the scaling factor BPM_BASE and heart rate BPM.

$$RR \leq TAx \frac{BPM\_BASE}{BPM} \quad x \in \{0,1,2\} \tag{16.8}$$

The right part of Fig. 16.11 shows how the scaling factor BPM_BASE impacts the performance. The x-axis shows the values of the scaling factor BPM_BASE in a range from 40 to 130, with increments of 2, and y-axis the number of errors. We observe that a value of $BPM\_BASE = 90$ minimizes the errors.

Chapter 17
Improving ECG Beat Classification Algorithm

In this chapter we propose an approach based on a set of decision rules to classify detected QRS complexes. It requires relatively simple operations and is a pipelined solution, processing beat by beat without extensive memory or processing requirements. Thus it can also run on mobile devices.

Our goal was to apply the IEC 60601-2-47 standard for ambulatory electrocardiographic medical devices [77] that classifies three most frequent QRS types: Normal (N), Ventricular (V), Suprabentricular (S) and Fusion of a ventricular and normal beat (F). We observe that having high sensitivity and positive predictive rate of QRS detection also affects the performance of the algorithm. Our improved version of Hamilton's [47] algorithm is used as a QRS detector in this research.

17.1 Definition of QRS classes

A typical ECG is a periodical signal, with repeating patterns of P wave, QRS complex and T wave. Figure 17.1 presents characteristic points of an ECG wave, the baseline, ST, QT and RR intervals.

Our goal is to classify most frequent QRS complex types that can be detected by analyzing single channel ECG recording and build a good quality rhythm detector.

17.1.1 Feature space

Beat classification is applied after the QRS detection phase. The technology to classify beats depends on definition of a specific feature as follows:

- **F1** *QRS width*, that can be (normal) narrow ($\leq 110ms$), or wide ($> 110ms$) [129];
- **F2** *QRS morphology*, determined by the QRS complex type, taking one of the following shapes qRs, Rs, rsR', rS, rs, Q, QS, qR, QR, according to Fig. 17.2,
- **F3** *context behavior*, determined by the last five beats,

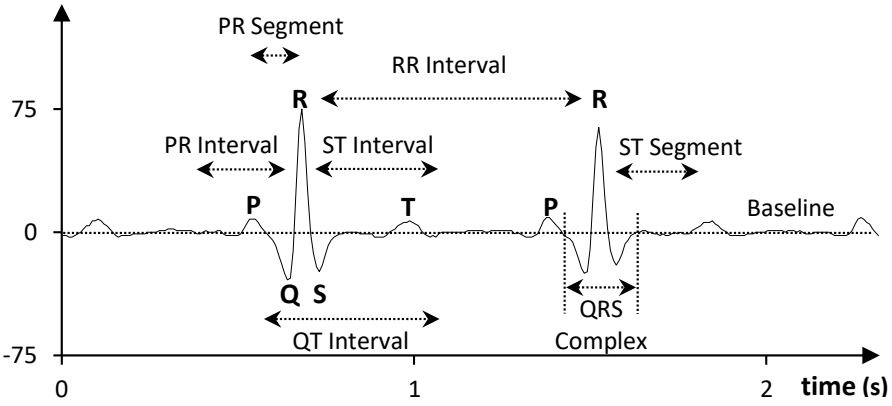


Fig. 17.1: Characteristics of an ECG wave on MIT-BIH Arrythmia record 101 (79.6 sec).

- **F4** *rhythm type*, classified as regular and irregular, determined by the last five beats,
- **F5** *heartbeat location*, determined as premature, delayed or following the underlined rhythm;
- **F6** *heartbeat rate*, classified as low (bradycardia), normal between 60 and 100 BPM for resting heart and high (tachycardia);
- **F7** *T wave morphology*, determined as normal (positive), inverse (negative), or biphasic (positive and negative part) [134];
- **F8** *elevation of the ST segment*, that can be normal, elevated or depressed [134];
- **F9** *QT interval* as normal, short, long or unmeasurable [134].
- **F10** *PR segment* as line between P-wave and the start of QRS complex.

In general, QRS morphology can have eight different shapes, namely qRs, Rs, rsR', rS, rs, Q, QS, qR, QR as portrayed in Fig. 17.2.

We have degraded it to four different shapes, particularly NOR, INV, EQN and EQI. Their illustration though are provided in Fig. 17.3.

Technical details for derived QRS types and R point calculation are provided in Algorithm 5 and Table 17.2. Please note that the values are actually energies of the ECG signal. Additionally, Table 17.1 lists the equivalencies between the primary and derived QRS complex types.

Experienced doctors use additional features, to establish a more precise diagnosis, such as

- *morphology of the P wave*, as normal, inverse, absent, abnormal, or saw;
- *PR interval*, determined as normal, unmeasurable, short, or prolonged;
- *sequence of P waves* prior to a QRS complex;
- *missing P wave* prior to a QRS complex;
- *missing QRS complex* in the rhythm sequence (different degree of AV block);
- *missing T wave* between two QRS complexes;

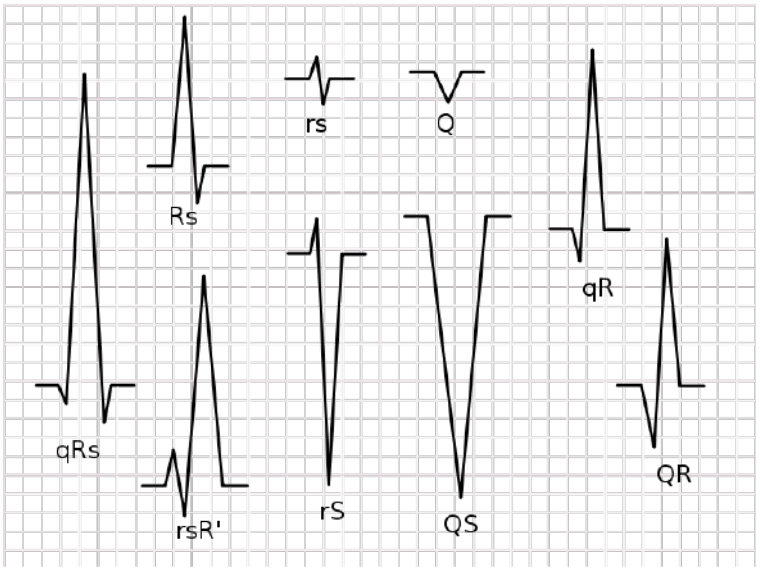


Fig. 17.2: QRS complex types[34].

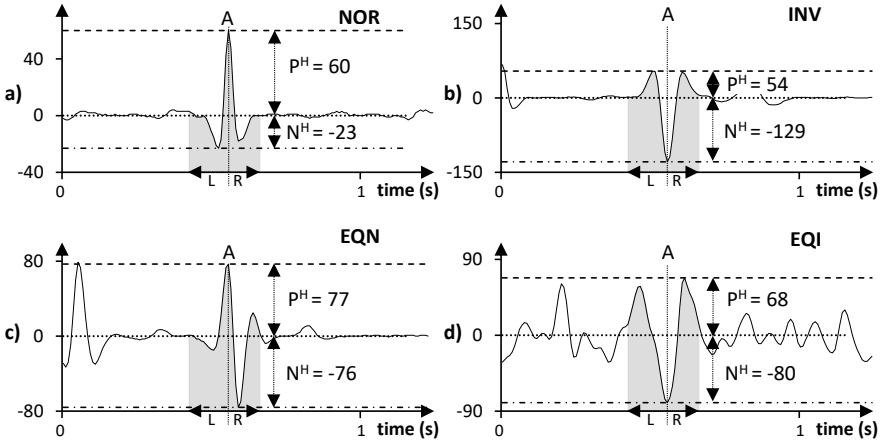


Fig. 17.3: Determination of different QRS types.

However, due to small amplitude level of P waves, the algorithms cannot be efficient in determination of the P wave, especially dependent on the position of the one-channel wearable ECG sensor and those features are omitted from our analysis.

Table 17.1: Euivalecies between primary and derived QRS complex types.

Primary QRS Complex Type	Derived QRS Complex Type
qRs	*NOR*
Rs	*NOR*
rs	*EQN, EQI*
Q	*EQI*
qR	*NOR*
rsR'	*NOR*
rS	*INV*
QS	*INV*
QR	*NOR,EQN*

Algorithm 5 Detect R point and QRS Morphology

procedure RANDQRSMORPHOLOGY($Index$, HPF)
  $Class = NONE, R_{ind} = 0$
  $PH = 0, PH_{ind} = 0$
  $NH = 0, NH_{ind} = 0$
  for $i = Index - 12$; $i < Index + 8$; i ++ **do**
    if $HPF(i) \geq 0$ & $PH \leq HPF(i)$ **then**
      $PH \leftarrow HPF(i)$
      $PH_{ind} \leftarrow i$
    else
      if $HPF(i) < 0$ & $NH \geq HPF(i)$ **then**
        $NH \leftarrow HPF(i)$
        $NH_{ind} \leftarrow i$
  $RH = \frac{ABS(PH - NH)}{MIN(PH, NH)}$
  if $RH < 0.2$ || $(PH < 8$ & $NH < 8)$ **then**
    if $PH \geq NH$ **then**
      $Class = EQN$
      $R_{ind} = PH_{ind}$
    else
      $Class = EQI$
      $R_{ind} = NH_{ind}$
  else
    if $PH \geq NH$ **then**
      $Class = NOR$
      $R_{ind} = PH_{ind}$
    else
      $Class = INV$
      $R_{ind} = NH_{ind}$
  Return $R_{ind}, Class$

17.1.2 QRS classes

The QRS complex represents depolarization of ventricular muscle cells and it is normally initiated by an electrical signal with origin in the sinoatrial (SA) node

Table 17.2: Specification of derived QRS complex types.

Derived QRS Complex Type	RH	PH	NH	PH/NH
NOR	*>=0.2*	>=8	>=8	>=1
INV	*>=0.2*	>=8	>=8	<1
EQN	*<0.2*	-	-	>=1
EQN	-	<8	<8	>=1
EQI	*<0.2*	-	-	<1
EQI	-	<8	<8	<1

[24]. Our intention in this research is to follow the *IEC 60601-2-47:2012* standard for medical electrical equipment as it it defines particular requirements for the basic safety and essential performance of ambulatory electrocardiographic systems, covering the one-channel wearable ECG sensors. According to this standard, the following beat types are to be classified into one of the following five classes:

- *S*, a supraventricular ectopic beat (sveb): an atrial or nodal (junctional) premature or escape beat, or an aberrant atrial premature beat;
- *V*, a ventricular ectopic beat (veb): a ventricular premature beat, an R-on-T ventricular premature beat, or a ventricular escape beat;
- *Q*, a paced beat, a fusion of a paced and a normal beat, or a beat that cannot be classified.
- *F*, a fusion of a ventricular and a normal beat;
- *N*, any beat that does not fall into the S, V, F, or Q categories described below (a normal beat or a bundle branch block beat);

The rough distinction of these beat types is made upon the QRS complex morphology, which differs mainly due to the origin of the electrical impulse or due to the conduction disturbances of a normally initiated beat. This initiates classification of a detected beat as:

- *Normal (N)* beat, when the QRS complex is initiated by SA node impulse;
- *Supraventricular/Atrial (S)* beat that result from impulses originating in atrial or AV nodal (supraventricular) areas outside the SA node;
- *Ventricular (V)* beat, which originates in the ventricles instead of SA node. They occur when electrical impulses depolarize the myocardium using a different pathway from normal impulses.

For testing purposes we have used the extensive Physionet ECG bank [63], where each QRS complex is labelled by an appropriate annotation. Table 17.3 presents how our classification follows the IEC 60601-2-47:2012 standard [77].

Table 17.3: Beat annotations from Physionet ECG bank [63] classified according to the IEC 60601-2-47:2012 standard [77] and used in our algorithm.

Physionet Beat type	Description	IEC 60601-2-47 Class
N	Normal beat	*N*
L	Left bundle branch block beat	
R	Right bundle branch block beat	
B	Bundle branch block beat (unspecified)	
/	Paced beat	
V	Premature ventricular contraction	*V*
r	R-on-T premature ventricular contraction	
E	Ventricular escape beat	
F	Fusion of ventricular and normal beat	*F*
A	Atrial premature beat	*S*
S	Supraventricular ectopic beat	
a	Aberrated atrial premature beat	
e	Atrial escape beat	
J	Nodal (junctional) premature beat	
j	Nodal (junctional) escape beat	
n	Supraventricular escape beat	
Q	Unclassifiable beat	*Q*

17.1.3 Premature and Prolonged Contractions

Premature contractions are those beats that appear prior to the expected beat as determined by a regular rhythm. These contractions are generally classified into three categories: *Ventricular*, *Atrial* and *Junctional* (*Supraventricular*), depending on an impulse origin - ventricular tissue, atrial tissue or atrioventricular (AV) junction correspondingly.

Fig. 17.6 and 17.7 present premature ventricular and atrial contractions correspondingly.

In our analysis, a beat is considered to be *Premature* if current RR interval differs for at least 16% of the average RR interval of the last 6 beats. This value is selected based on the study reported by Gusev et.al [66] and proven to be successful.

Postponed contractions are those beats that appear later to the expected beat as determined by a regular rhythm. Usually, they are late due to compensatory pause from a previous premature contraction, or they can appear as an *escape* beat.

17.1.4 Normal Beats

We classify a detected QRS complex as *N* (*Normal Beat*) under the conditions presented in Table 17.4 [129].

Table 17.4: Determination of a normal beat.

Parameter	Characteristics	Values
QRS complex width	Normal	60-110 (ms)
QRS morphology	qRs, Rs, rsR', rS, rs, qR, QR	Fig. 17.2
Context behavior	Similar to the last beat	
Rhythm type	Regular	
Heartbeat location	Follow the underlined rate	
Heartbeat rate	Normal	60 - 100 BPM
T wave morphology	Normal	
ST segment	Normal	
QT segment	Normal	

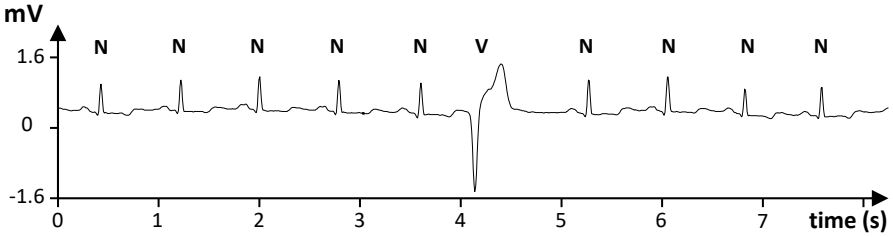


Fig. 17.4: Illustration of N beats and one V beat on MIT-BIH Arrhythmia record 100 (1514.8 sec).

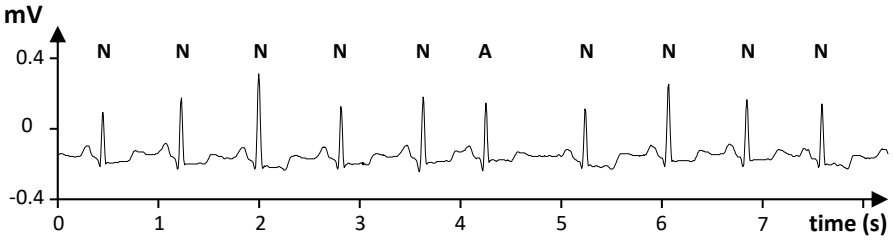


Fig. 17.5: Illustration of N beats and one A beat on MIT-BIH Arrhythmia record 100 (470.0 sec).

Fig. 17.6 and 17.7 display a series of normal beats identified by N, along with occurrence of S (A) and V beats.

17.1.5 Supraventricular Beats

Supraventricular beats arise from an ectopic focus within the atria or from AV node. QRS morphology of a supraventrucular beat is similar (same) to a normal beat due to the same pathway of conduction through the AV node, and same pattern of ven-

tricular depolarization, however its location is different, so it can be premature or prolonged.

QRS width of normal and supraventricular beats is between 60 and 110 ms. Table 17.5 presents the rules to determine its type determined by the irregular rhythm and beat location.

Table 17.5: Determination of a supraventricular and normal beats.

Parameter	Premature (A, S)	Normal (N)	Escape (e, j)
Location	Premature	Normal	Postponed
Rate	Normal	Normal	Normal
T wave	Normal	Normal	Normal

An example of a premature atrial contraction (PAC) is listed in Fig. 17.7, denoted as *A*.

17.1.6 Ventricular Beats

A ventricular beat can be detected by the abrnormal morphology of the QRS complex, such that the QRS duration is longer than or equal to 120 ms. The irregularity of the rhythm, rate and the beat location determine more specifically its type. According to Table 17.3 we classify four types of ventricular beats: a *premature ventricular beat (PVC)*, an *R-on-T ventricular premature beat (r)*, a *ventricular escape beat (E)* and *fusion of a ventricular and a normal beat (F)*. Table 17.6 defines the differences more precisely, and, therefore, our determination rules.

Table 17.6: Determination of a ventricular beat.

Parameter	PVC (V)	RonT (r)	Fusion (F)	Escape (E)
Location	Premature	Premature	Normal	Postponed
Rate	Normal	$>$ 150 BPM	Normal	Normal
T wave	Opposite	Normal	Normal	Opposite

An example of a premature ventricular beat is already illustrated in Fig. 17.6, where a *PVC* ventricular beat has an obvious diversity and its location is premature.

17.2 Definition of Features

We use an improved version of Hamilton's QRS detection algorithm [47] as illustrated in Fig. 17.8. Q,R and S points of a complex are calculated by this algorithm.

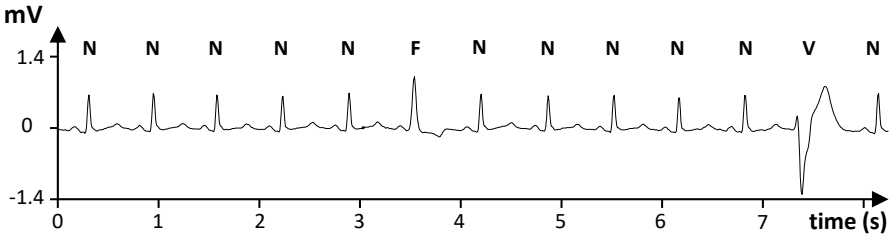


Fig. 17.6: Illustration of N beats and one V and F beat on MIT-BIH Arrhythmia record 205 (815.0 sec).

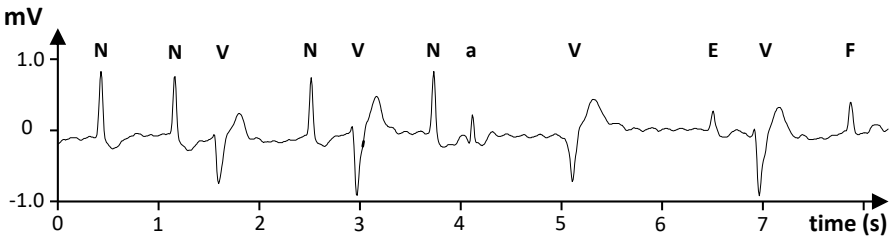


Fig. 17.7: Illustration of N and V beats, one a, F and E beat on MIT-BIH Arrhythmia record 210 (1755.0 sec).

Original Hamilton calculates the R-peak locations to be at A^C,B^C and C^C points. However, these locations are slightly different from the real R-peaks. That is why, in our previous study [47] we introduced a *SearchL* and *SearchR* intervals, and experimentally found that best results are when they are 160 *ms* and 120 *ms* respectively.We consider baseline to be the value 0 on over the bandbass filterred signal.

Based on these parameters, Algorithm 5 is used to find R-locations and QRS complex type. If we have *NOR* or *EQN* complex type, then we search on the negative y-axis for Q and S locations. For both of them, we try to find the point that is farthest from the baseline, on the left and right side of R for Q and S respectivelly. In case of *INV* or *EQI* complex type, the only difference is that we search positive y-axis for Q and S locations.

As a result of our algorithm, R-locations are found to be at R^A,R^B and R^C, with corresponding Q and S locations.

Fig. 17.9 visualizes $fQSc$ parameter for R^A,R^B and R^C peaks. Algorithm finds the maxium energy, then gets 5% of it to mark the start and end points of Q and S on signal output after average over an 80 *ms* window. This algorithm calculates that for this segment particularly fQSc is 128ms. $fQSa$ and $fQSd$ are calculated based on this parameter.

Fig. 17.10 visualizes *SB* and *QBh* parameter for R^A,R^B and R^C peaks. Algorithm calculates the time required from S to the baseline on signal after bandpass filter, and absolute energy difference between Q and baseline.

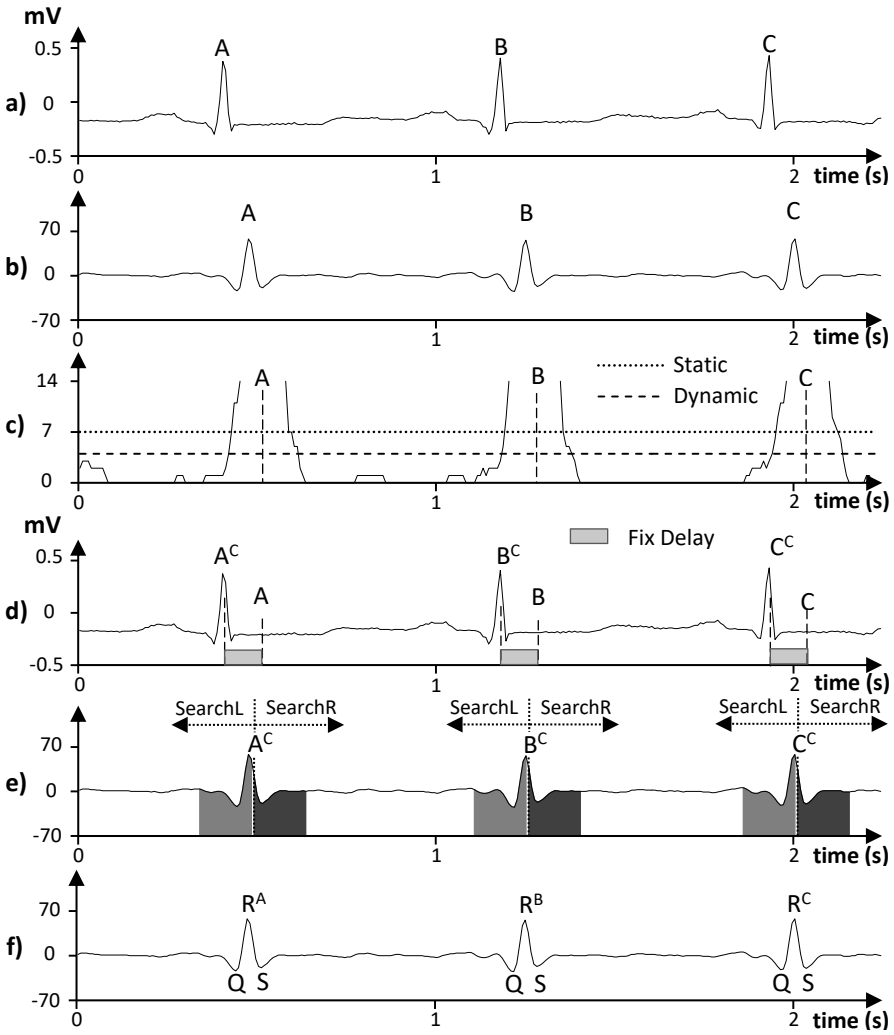


Fig. 17.8: A signal extract and R-peak,Q and S point detection over the MIT-BIH Arrhythmia record 100 (426.4 sec): a) Original signal and local peaks A, B and C; b) Output after bandpass filter; c) Output after average over an 80 *ms* window; d) Calculated R-peak locations with constant delay introduced by the filter. e) Calculated R-peak locations A^C,B^C and C^C, and the search intervals over the output of bandpass filter. f) Calculated R-peak locations R^A,R^B and R^C and characteristic Q and S locations.

Table 17.7: Feature space detection parameters.

ID	Feature	Description
F5	Q,R,S	Detected Q,R and S point.
F1	$fQSc$	Width between two points on squared average signal over an 80 ms window. Start and End points are calculated as 5% of the max height, on 600ms interval from the detected maxima.
F1	$fQSa$	Average of fQSc over last 5 peaks.
F1	$fQSd$	Current difference of fQSc and fQSa.
F1	SB	Distance from the detected S point until the baseline on high pass filterred signal.
F10	QBh	Distance from the detected Q point until the baseline on high pass filterred signal.
F8	ST	Let R1, R2 and R3 be the latest R points, T3 be the current T point, AvgRR3 the average value from R1 to R3, and AvgRT3 be the average value between R3 and T3. ST is calculated as the difference of AvgRT3 and AvgRR3.
F1	QSe	Average of input values between detected Q and S point, subtracted to the minimum on that section.
F1	QSc	Width between Q and S calculated on the high pass filterred signal.
F1	QSa	Average of QSc over last 5 values.
F1	$QSdiff$	Current difference of QSc and QSa.
F2	$TriSim$	Triangular similarity of current and previous beat where Q,R,S are the points of the triangle.
F3	$(of)QRca$	Deviation of current QR height from the average of 20 QR's, calculated on the original or filterred signal
F3	$(of)RSca$	Deviation of current RS height from the average of 20 RS's, calculated on the original or filterred signal
F3	$(of)QRpc$	Deviation of current QR height from the previous QR, calculated on the original or filterred signal
F3	$(of)RSpc$	Deviation of current RS height from the previous RS, calculated on the original or filterred signal
F6	RA	Average of RRl values over last 5 values
F4	RA/RRl	Ratio of RA over current RR
F4	RA/RRa	Ratio of RA over Rra
F6	$RARRld$	Deviation of RA over RRl
F6	RRa	Average of RRl values over last 5 values being inside the interval 80% and 115% of the previous average.
F6	RRl	Current RR interval
F6	RRr	Ratio of RRa over RRl
F6	$RRld$	Difference of current and previous RR interval, over the previous.
F2	$TYPEc$	Current type of QRS complex. One of NOR, EQ_N, INV or EQ_I
F2	$TYPEp$	Previous type of QRS complex.
F2	$TYPEl$	Type of latest normal QRS complex.
F2	$LAST$	Detected class of last QRS complex.

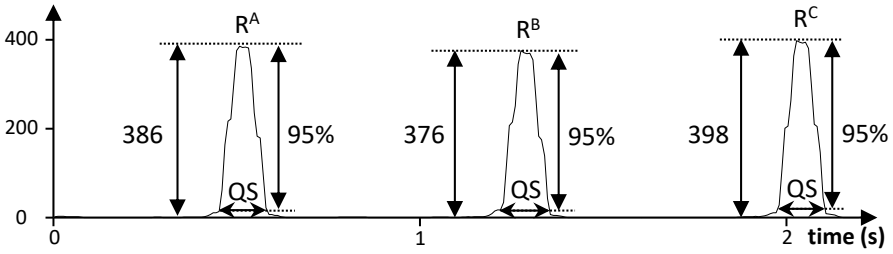


Fig. 17.9: A signal extract to visualize $fQSc$ feature over the output after average over an 80 *ms* window on MIT-BIH Arrhythmia record 100 (426.4 sec).

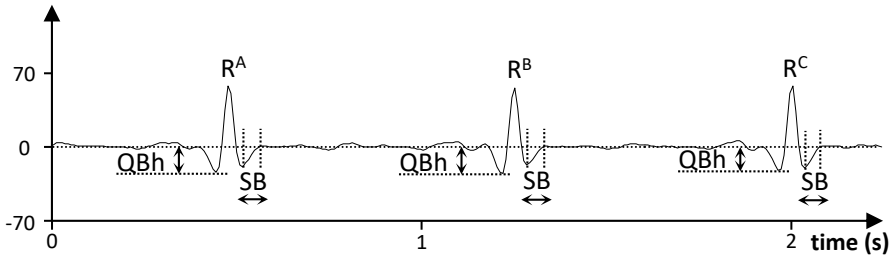


Fig. 17.10: A signal extract to visualize *SB* and *QBh* feature over the output after bandpass filter on MIT-BIH Arrhythmia record 100 (426.4 sec).

Fig. 17.11 visualizes ST parameter for $R1$,$R2$, $R3$ peaks and $T3$ location. Let the average value between $R1$ and $R3$ be $AvgRR3$, and average value between $R3$ and $T3$ be $AvgRT3$, then ST is calculated as $AvgRT3 - AvgRR3$.

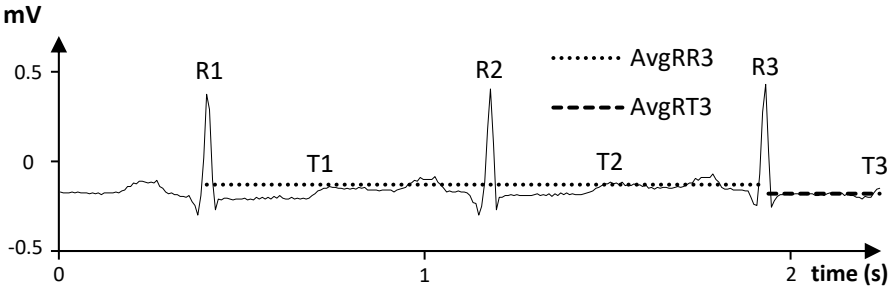


Fig. 17.11: A signal extract to visualize ST feature over the original signal on MIT-BIH Arrhythmia record 100 (426.4 sec).

Fig. 17.12 visualizes QSe parameter for $R1$,$R2$, $R3$ peaks and corresponding Q and S locations. Let the average value between Q and S for R be $AvgQS$ and $MinQS$ be the minimum value of Q and S, then QSe is defined as $AvgQS - MinQS$.

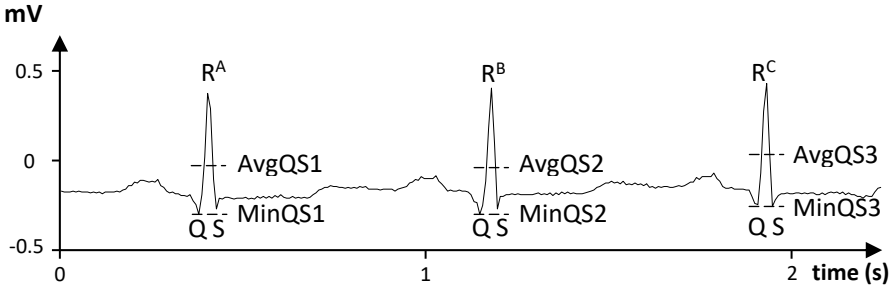


Fig. 17.12: A signal extract to visualize *QSe* feature over the original signal on MIT-BIH Arrhythmia record 100 (426.4 sec).

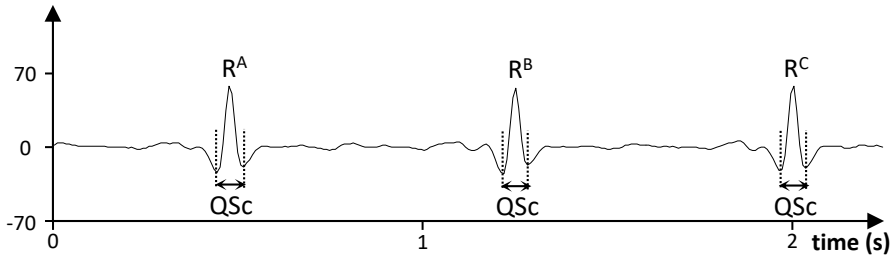


Fig. 17.13: A signal extract to visualize *QSc* feature over the the output after band-pass filter on MIT-BIH Arrhythmia record 100 (426.4 sec).

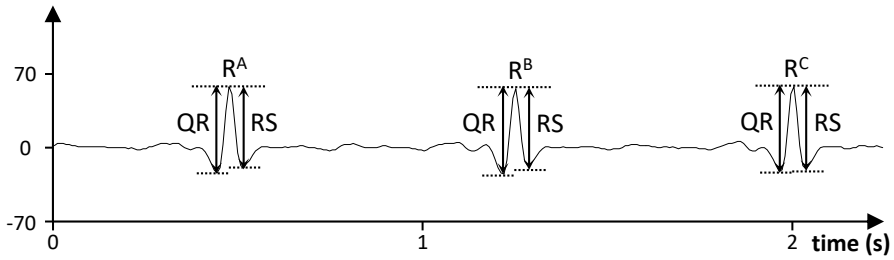


Fig. 17.14: A signal extract to visualize *QR* and *RS* heights over the the output after bandpass filter on MIT-BIH Arrhythmia record 100 (426.4 sec).

Fig. 17.13 visualizes *QSc* parameter for *R*1,*R*2, *R*3 peaks and corresponding *Q* and *S* locations.*QSc* is defined as the time elapsed from Q to S point. *QSa* and $QSdiff$ are calculated based on this parameter.

Fig. 17.14 visualizes *QR* and *RS* heights for *R*1,*R*2, *R*3 peaks. Based on this parameters, feature $fQSca$ can be calculated as the deviation of current *QR* from the average. The same applies to $fRSca$. Calculation of $(of)QRca$, $(of)RSca$, $(of)QRpc$ and $(of)RSpc$ are done in the same manner.

On the other hand, $TriSim$ feature is calculated with QR and RS heights, and QSc width. Let the previous heights be $QR1$ and $RS1$, and the width be $QSc1$, whereas current ones be $QR2$,$RS2$ and $QSc2$. If we consider QRS complex as a triangle, then we can conclude that they are similar if both of the following conditions are valid at the same time. Note that $SIM\_THR = 15\%$, which is the similarity threshold.

1. $(QR1/RS1) * (RS2/QR2) <= SIM\_THR$
2. $(QSc1/QR1) * (QR2/QSc2) <= SIM\_THR$

17.3 Set of Decision Rules

In this research, we have introduced more than 500 decision rules. In the process of introducing new rules and features, we have been working closely with cardiology experts. Fig. 17.15 gives presents a high evel overview for decision of Normal, PVC and PAC beats.

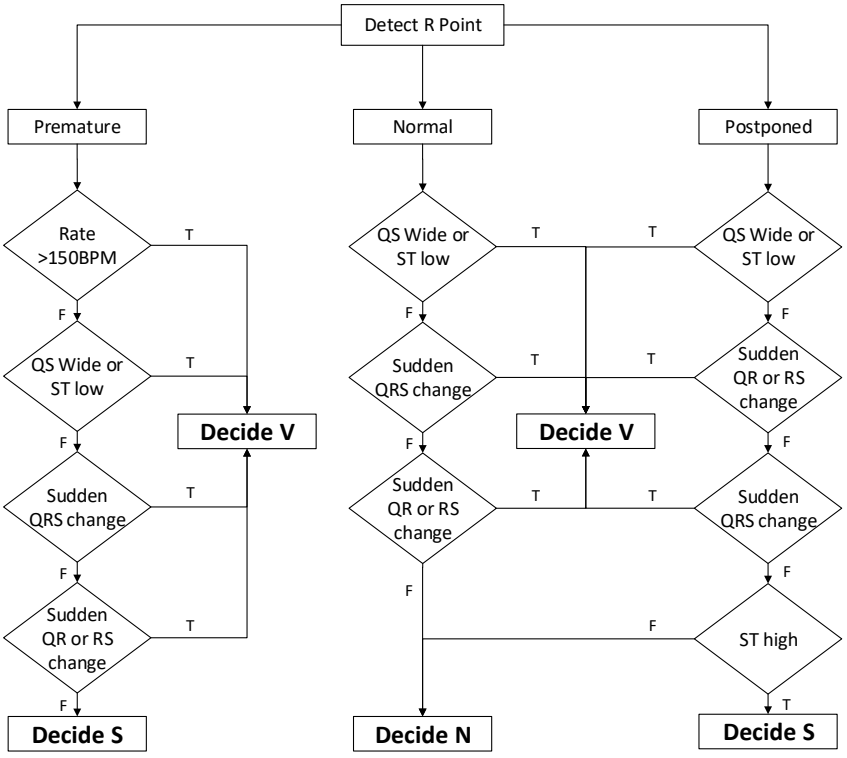


Fig. 17.15: High level view of the decision process.

Table 17.8: Top 10 Decision Rules and the Criterias used for Normal Beat.

Feature	Decision Rules and Values									
	P030	**P034**	**P036**	**P038**	**P051**	**P084**	**P204**	**R001**	**R098**	**R999**
fQSc	<17	<17	<17	<17	<20	-	-	>21— <18	<29	-
fQSa	-	-	-	-	<19.3	-	-	-	>15	-
ST	-	-	-	-	-	-	-	>-27 <16	<-6	-
SB	-	-	-	-	-	-	-	>0 <5	>4	-
QBh	-	-	>-55 <-31	>-55 <-32	<-16	-	-	-	-	-
Qse	-	>20	-	-	<40	-	-	>-35 <28	>39	-
QSc	<12	>9	-	-	<14	-	-	>13	>13	-
QSa	-	>9	-	-	-	-	-	>8.5 <12	-	-
QSdiff	-	-	-	-	<1	-	-	-	-	-
TriSim	-	<0.16	<0.16	<0.16	<0.16	<0.16	<0.05	-	-	-
fQRca	-	-	-	-	-	-	-	-	-	-
RRl	>78	-	-	-	>48	-	-	-	-	-
RRr	<1.33	<1.76	-	-	-	-	-	<1.18	-	-
RRa	>89	-	-	-	-	-	-	-	-	-
RA/Rra	-	-	>1	<1	!=1	-	-	-	-	-
RRld	>-0.15 <0.57	-	-	>-0.54 <0.16	-	-	-	-	-	-
TYPEc	!INV	-	-	-	-	NOR	-	-	-	-
LAST	-	! PVC	! PVC	! PVC	! PVC	-	-	-	-	-
Total Det.	71	145	37	39	85	257	71	47	549	87520

17.3.1 Normal Beat

There are totally 114 decision rules for classifying beat as normal which are activated on MIT-BIH Arrhythmia, one of which has no criteria and applies when none of the rules are effective. Table 17.8 presents top 10 of them, based on the number of correctly classified normal beats. These rules in fact decide for 99.33% of the existing normal beats in MIT-BIH Arrythmia database. An important note is that, we use only 18 features for the top 10 rules, in order to classify normal beat.

17.3.2 Ventricular Beats

MIT-BIH Arrhytmia database activate 141 decision rules for Preature Ventricular Cotraction. The most important 10 rules accounts for 54.26 % of the total classified

PVC's, where the details for them are presented in Table 17.9. We obseve that 18 features are generally used in these mostly used decision rules.

Table 17.9: Top 10 Decision Rules and the Criterias used for PVC Beat.

Feature	**Decision Rules and Values**									
	P022	**P029**	**P096**	**R008**	**R011**	**R012**	**R014**	**R033**	**R040**	**R083**
fQSc	<34 >3	<32	-	>16	-	-	>16	-	>18 <27	-
fQSa	<26.6	-	-	-	-	-	-	-	-	<35
fQSd	<-0.6	-	-	-	-	-	-	-	>0.4	-
ST	>-71 <8	-	-	-	-	-	<0	-	-	>-66
SB	-	>6	>5	-	>5 <11	>5	>5	>6	-	<17
QBh	>-71	-	-	-	-	-	-	-	-	>5
Qse	-	-	-	-	<-64	-	-	-	>-125 <-22	>-220
QSc	>15	-	-	>15	>10	>9	>10	>10	>8 <23	>10 <25
QSa	-	-	-	-	-	-	-	-	>10	<20.5
QSdiff	>=1	>0	-	>2	-	-	-	-	>-1.5	-
oQRca	-	-	>0.45	-	-	>0.65	>0.65	<-2.0	>-3 <-2	-
oRSca	-	-	-	-	-	-	>0.65	<-1.0	-	-
RRl	-	-	-	-	-	-	-	-	>50 <80	>40 <149
RRr	-	<1.8	-	-	-	-	-	-	>0.93	
RRa	-	-	-	-	-	-	-	-	-	>40 127
TYPEc	-	-	-	NOR	EQI	EQN	-	EQI	-	-
TYPEp	-	-	-	-	-	-	-	-	-	NOR
TYPEl	-	-	-	-	NOR	NOR	-	NOR	-	-
Total Det.	515	351	332	731	478	208	213	192	275	371

17.3.3 Atrial Beats

On the other hand, 119 rule for Atrial Premature Contraction are activated on MIT-BIH Arrhytmia databas. 64.66% of the total APC's are actually classified by the top 10 classification rules. Table 17.10 presents them, where we observe that total of 20 features are used. Iterestingly we see that high percentage of features are regarding the RR interval. This is an expected situation, since APC peak resambles a lot to a normal, and *RR* interval is one of the differentiative factors.

Table 17.10: Top 10 Decision Rules and the Criterias used for APC Beat.

Feature	**Decision Rules and Values**									
	P044	**P052**	**R147**	**R170**	**R171**	**R172**	**R173**	**R174**	**R175**	**R196**
fQSc	<17	>22	>11 <17	-	>14 <20	>16 <21	>16 <20	>16 <25	>16 <24	>17 <25
fQSa	-	>21.8	-	-	-	>16 <19.5	>16.8 <18.4	>17.4 <19.6	-	-
fQSd	-	-	-	-	-	>-3 <1.4	>-0.6 <1.2	-	-	-
ST	-	-	-	>-40	>-23 <20	-	<2	>-15 <2	>-14 <8	-
SB	-	>3 <10	<10	-	>4 <11	>6 <10	>5 <9	>5 <10	>5 <11	-
QBh	-	-	>-27 <-8	-	>-16 <-2	>-16 <-4	>-14 <6	>-16 <-4	>-15 <-2	-
Qse	-	<40	-	-	>6 <39	>11 <34	>16 <26	>12 <24	>8 <34	>-45 <-30
QSc	-	<14	-	<12	>7 <13	>8 <11	>8 <11	>9 <12	>7 <14	>11
QSa	-	-	-	<11.6	>9.4 <10.2	>9.4 <9.9	>9.4 <9.9	>9.4 <9.9	>9.4 <10.2	>12.4 <13.8
QSdiff	-	<1	-	-	-	>-0.9 <0.6	>-0.9 <0.6	-	-	-
TriSim	<0.16	<0.16	-	-	-	-	-	-	-	-
RRl	-	>61 <86	-	>56 <80	>80 <96	>87 <99	>89 <96	>89 <106	>89 <106	>56 <69
RRa	-	-	-	-	>85 <96	>88 <95	>87 <96	>88 <95	-	-
RRr	-	>1.2	-	>1.4	>0.96 <1.1	-	>0.96 <1.04	<1.01	>0.9	-
RA	-	-	-	-	-	-	-	-	>96.1	>59.4 <64.6
RA/RRl	-	-	-	-	>1.21	>0.95 <1.03	-	-	>1.17	-
RA/RRa	-	-	-	>1.4	>1.17	1	>1.12	-	-	-
RARRld	-	>0.2	-	>0.4	-	-	-	-	-	-
RRld	-	-	-	<-0.4	-	>-0.06 <0.06	>-0.07 <0.06	-	-	-
LAST	!PVC	!PVC	-	-	-	-	-	-	-	-
Total Det.	356	63	62	256	403	238	75	95	133	63

17.4 Exerimental Setup

In this paper, we consider using the following banchmark databases:

- MIT-BIH arrhythmia database [99],
- American Heart Association's (AHA) database [72],
- European ST-T database [130]

MIT-BIH arrhytmia database consists of total 48 records each of 30 minutes length, with a default frequency of 360Hz and 11 bit resolution. In this research, we use resampled data of 125Hz and 10 bits resolution instead. We also exclude paced beats, which are found in records labelled as 102, 104, 107 and 217.

We additionaly consider AHA database for the sake of this research. Even though this database contains 3 hour long recordings, we consider only the last hour since it is the only part annotated. 2202 and 8205 labelled records have paced beats, thats why they are removed. We also have dowsampled data from 250 to 125Hz and decreased the resolution to 10 bits instead of the default 12 bits.

European ST-database is the last benchmark database we considered. It already has 90 records of 2 hour long, sampled on 250 hertz with a resolution of 12 bits. We consider using all of the records with a dowsampled rate of 125 and 10 bits.

Two testing environments were used for testing:

- Intel i7-3632QM CPU, 2.2 GHz system with 12 GB of memory and Windows 10 and,
- Intel i5 CPU 2.7GHz with 8 GB 1333 MHz DDR3 with MacOS Sierra.

All algorithms are compiled by a standard gcc compiler without any compiler optimization flags.

Improved version of Hamilton's [47] algorithm having a 99.91% QRS sensitivity and 99.90% QRS positive predictivity rate is used as a QRS detector.

17.5 Evaluation of Results

We use sensitivity and positive predictivity a performance metrics. Eq. 17.1 presents the formulas for calculating them, where TP , FP and FN are the true positive, false positive and false negative values.

$$SENS = \frac{TP}{TP+FN} \qquad PPV = \frac{TP}{TP+FP} \tag{17.1}$$

Based on these metrics, results for Normal, PVC and APC beats are listed on Tables 18.2, 18.3 and 18.4 respectively.

Chapter 18
Overview and Related Work

The content of this Chapter was published at the Journal of Technology and Healthcare [47], 2019.

18.1 Related Work

Several approaches have been proposed for ECG beat classification realized by the following approaches based on: genetic algorithms, decision trees, neural networks, fuzzy rules, wavelets, or other machine learning algorithms.

Decision based approaches that were proposed in earlier times revealed reasonable results. Some of these studies are [6, 81, 101, 71]. The main disadvantage in these algorithm is in the inability to adapt to temporary changes in the morphology of beats.

Dokur and Ölmez [38] propose hybrid neural network structure analyzing a total of eight features. They use genetic algorithms to train the hybrid structure and classify ten different QRS types: N, L, R, P, p, a, E, V, F and f. A Total Classification Accuracy (TCA) of 95.7% is reported for MIT-BIH arrhythmia database. The main disadvantage lies in the fact that their algorithm needs to be trained enough and personalized in order to produce highly accurate classification results.

Engin [56] proposes a Fuzzy-Hybrid neural network algorithm for beat classification. He uses three features, in order to classify four different beat classes: the N, V, R and x (a non-beat annotation according to PhysioNet [63]). The reported results are 98% mean efficiency for only four records of MIT-BIH arrhythmia database.

Hu et.al [73] proposes MOE (mixture of experts) approach in order to implement patient-adaptable beat classifier. Their algorithm classifies four types of beats: N, V, F and Q. They exclude total of 15 records from MIT-BIH arrhythmia database, that have paced beats and that do not have PVC beats. Their algorithm produces 95.9% classification rate with using *LE* classifiers.

Wieben et.al [137] proposed an algorithm to classify PVC beats using a combination of filter banks, decision trees and a fuzzy-rule based system. Decision tree based

algorithm produces sensitivity of 85.3% and a positive predictive rate of 85.2%, whereas the fuzzy rule based system produces worse results of 81.3% sensitivity and 80.6% positive predictive rate.

Afonso et.al [11] proposes a solely filter bank based ECG classifier, where a specific machine learning method automatically creates the rules. The classifier is able to distinguish paced and non-paced beats with a sensitivity of 87.64% and a positive predictive rate of 90.97%.

A novel decision-tree approach was presented by Exarchos et.al. [57]. Their algorithm has three stages, extraction of rules, transformation to fuzzy model and optimizing the parameters. This algorithm classifies four types of beats: ventricular flutter beats (noted as non-beat annotation according to PhysioNet [63]), V, N and beats of second degree heart block; producing a 96% accuracy using the MIT-BIH arrhythmia database.

Zhang et.al. proposes a combination of Wavelet Transformation and Decision tree classification. This algorithm can classify N, L, R, P, V and A beats with 96.31% accuracy on data obtained from MIT-BIH arrhythmia database.

An efficient approach was presented by Khoureich [78], by only using waveform similarity and RR interval, in order to classify N, A, /, V, L and R beats. Experiments on 46 out of 48 records of the MIT-BIH arrhythmia database had revealed a classification rate of 97.52%.

An exclusively RR-interval based approach was also reported by Tsipouras [133], for detecting N, V, ventricular flutter/fibrillation and second degree heart block producing a 98% accuracy on MIT-BIH arrhythmia database. Furthermore, this information is used to classify six rhythm types.

We observe that, each of the presented studies use different number of features and various algorithm approaches.

18.2 Overview of Obtained Results

18.2.1 Improving the QRS Detection for One-channel ECG Sensor

The following parameter combination achieves the best overall performance: $STHR$ = 2, $SearchL$ = 152 ms, $SearchR$ = 56 ms, $DTHR$ = 200, $TA0 = 250$ ms, $TA1 = 260$ ms, $TA2 = 320$ ms, $THRA1$ = 0.8, $THRA2$ = 2.5, and BPM_BASE = 90.

Table 18.1 gives an overview of the obtained results and shows that our algorithm has reached better-combined values of sensitivity and positive predictive rate.

One can face several problems comparing any QRS detection method to other published papers as summarized below:

- no source code provided to check other approaches;
- no info about positive predictive rate; or
- no info about the number of errors.

Table 18.1: Comparison of algorithm performance over MIT-BIH Arrhythmia database.

MIT-BIH Arrhythmia database								**all 48 records**			**no paced records (44)**		
Algorithm	f_S (Hz)	**Bits**	**Scale (mv)**	**Total Peaks**	TP	FP	FN	**Tot Err**	Q_{SE}	Q_{+P}	**Tot Err**	Q_{SE}	Q_{+P}
Our Work	125	10	6	109494	109382	110	112	222	99.90	99.90	194	99.91	99.90
Ghaffari [62]	360	11	10	109428	109327	129	101	230	99.91	99.88	N/A	N/A	N/A
Bahouraa[21]	250	11	10	116137	109625	133	174	307	99.83	99.88	303	99.82	99.88
Elgendi [53]	360	11	10	109985	109775	82	247	329	99.78	99.92	322	99.76	99.92
Martinez [92]	360	11	10	109428	109111	35	317	352	99.71	99.97	N/A	N/A	N/A
J.Martinez [93]	360	11	5	109428	109208	153	220	373	99.80	99.86	N/A	N/A	N/A
Cvikl [37]	250	N/A	N/A	109494	109294	200	200	400	99.82	99.82	373	99.81	99.82
Chiarugi [29]	360	11	10	109494	109228	210	266	476	99.76	99.81	443	99.75	99.81
J.Lee [85]	N/A	N/A	N/A	109481	109146	137	335	472	99.69	99.87	459	99.68	99.87
Zidemal [139]	360	N/A	N/A	109494	109101	193	393	586	99.64	99.82	540	99.64	99.83
Hamilton [69]	360	11	10	109267	108927	248	340	588	99.69	99.77	569	99.68	99.76
Choi [30]	360	11	N/A	109494	109118	218	376	594	99.66	99.80	561	99.65	99.79
GQRS [63]	360	11	10	109494	109196	302	298	600	99.73	99.72	562	99.72	99.72
Christov [31]	360	N/A	5	109855	109615	386	288	674	99.74	99.65	670	99.72	99.62
Arzeno [20]	360	11	10	N/A	109099	405	354	759	99.68	99.63	N/A	N/A	N/A
Tompkins[107]	200	11	10	116137	109532	507	277	784	99.75	99.54	771	99.73	99.50
Paoletti [109]	360	11	10	109809	109430	565	379	944	99.65	99.49	924	99.64	99.45
Poli [113]	120	11	10	109963	109522	545	441	986	99.60	99.51	N/A	N/A	N/A
Elgendi [52]	360	11	10	109493	109397	97	1715	1812	98.31	99.92	1798	98.33	99.91

When analyzing the performance, only a small number of papers give information on the achieved positive predictive rate and they usually target achieving higher sensitivity values. However, it is very easy to achieve a higher sensitivity value and capture most of the results you would like to include in your algorithm by relaxing the constraints on the optimization parameters, but, at the same time, this will produce many extra generated peaks that do not represent a QRS peak. This is why it is very important to address both the sensitivity and positive predictive to evaluate performance.

This means that there is no direct comparison method. To cope with this problem, we have analyzed the number of errors as a performance measure (as defined by $Errors$ in Eq. 16.2). Although this performance measure can be achieved by calculation the sum of FP and FN, it can also be calculated through the harmonic mean (HM) of the QRS sensitivity and positive predictive rate by Eq. 18.1.

$$Errors = TotalQRS * \left(\frac{1}{Q_{SE}} + \frac{1}{Q_{+P}} - 2 \right) \tag{18.1}$$

We used Eq. 18.2 to evaluate this relation. In addition, we assumed that the number of extra detected (false negative) peaks is a lot smaller than the number of correctly detected QRS peaks, i.e. $TP >> FN$, leading to $TotalQRS \approx TP$.

$$\frac{1}{Q_{SE}} + \frac{1}{Q_{+P}} = \frac{TP+FN}{TP} + \frac{TP+FP}{TP} = 2 + \frac{FP+FN}{TP} \tag{18.2}$$

A number of discrepancies were found while analyzing related work. Table 1 of [107] shows that the number of errors is 782 although it is 784. The calculation of total beats is the most unambiguous. For example, TP + FN is greater than TB in [52]. Lee publishes two papers [84] and [85] providing correspondingly 109486 and 109481 total beats. The former one has 6 additional beats, which are from files *118, 201, 205, 220, 221* and *233*, whereas the latter one has 1 additional beat at record 114.

We found that various authors have used a different number of detected peaks. They should use the total number of detected beats, since the total number of peaks includes also non-beat annotations, for example, locations where there is a rhythm change. That is why we use the total number of annotated beats in the MIT-BIH Arrhythmia database 109494.

Table 18.1 also shows that researchers focusing on QRS detection algorithms mostly tend to use the original sampling frequencies, and resolution of reference ECG databases. Running the algorithm on 125Hz means that nearly 3 times less data is feed to the QRS detector, thus the execution times are deduced accordingly. Moreover, the adjustments that we proposed increased the performance metrics, which in total yields a better QRS detector intended for a small wearable one channel ECG sensor.

We have introduced several methods to improve the QRS detection in differentiation -based algorithms. Even though our approach is demonstrated on Hamilton's QRS detection algorithm, it can also be implemented in other algorithms. The results show superior performance over other published results. The improvements were efficiently built in on an industrial QRS detector, for a wearable ECG monitor, where the sampling frequency is 125 Hz, with 10-bit resolution of the AD converter.

Tuning the threshold values might increase the performance, but one needs to develop new relations to generate the thresholds, such as finding

- how many previous beats might be analyzed to estimate the mean value,
- which peaks will be classified as QRS beats since they are very similar in shape, and
- the impact of the beat rate on classifying the artifacts.

For this purpose, we have introduced several new rules for 1) threshold calculation to better classify QRS peaks; 2) elimination of QRS-like artifacts, and 3) artifact elimination based on beat rate.

Due to rescaling, loose ECG contacts, and noise caused by muscles, 62 beats cannot be detected by analyzing the first ECG channel, without the analysis of a second channel. This gives a higher performance of the QRS sensitivity to 99.96%, and our algorithm has reached a 99.90% QRS sensitivity, with 99.90% positive predictive rate when all records are analyzed in the MIT-BIH, and 99.91% QRS sensitivity for 44 records without paced beats.

Generally, the published papers on QRS detection algorithms do not offer their source codes, and only some of them are validated with referenced ECG databases, such as the MIT-BIH Arrhythmia database. Most of the algorithms give a brief description of design issues without implementation details, addressing only theoretical related results. This is why one cannot directly compare results. Different approaches generally do not achieve results as the ones achieved in a real implementation.

This study can facilitate possible QRS detection algorithms to consider rescaling, and resampling on reference ECG databases in return for better performances. Our findings show that the adjusted version of Hamilton's QRS detection algorithm yields better results with 125Hz data on nearly three times shorter execution times.

18.2.2 Improving ECG Beat Classification Algorithm

From the presented results for normal beat on Table 18.2 , we conclude that algorithm produces excelent results for Normal beat detection.

Table 18.2: Results for Normal beat detection.

Database	**TP**	**FP**	**FN**	**SENS**	**PPV**
MIT-BIH	100638	99	95	99.91	99.90
AHA	174343	477	432	99.73	99.76
EUR-ST	788103	2462	1970	99.69	99.75

When we analyze the results for Premature Vetricular Contraction listed on Table 18.3, we observe 94.55% sensitivity and 94.44% positive predictivity values. These results tend to be strong enough on AHA database also. Algorithm also produces a strong sensitivity value of 84.75% EUR-ST.

Table 18.3: Results for PVC detection.

Database	**TP**	**FP**	**FN**	**SENS**	**PPV**
MIT-BIH	6627	390	382	94.55	94.44
AHA	14273	942	2023	87.59	93.81
EUR-ST	3786	1555	681	84.75	70.89

Table 18.4 lists the results for Atrial Premature Contraction. 89.23% sensitivity and 97.37% positive predictivity rate is observed for database MIT-BIH. AHA database does not have APC beats, whereas on EUR-ST algorithm produces 42.45% sensitivity and 44.18 % positive predictivity values. This is primarily due to the low amplitude-energy signal.

Table 18.4: Results for APC beat detection.

Database	**TP**	**FP**	**FN**	**SENS**	**PPV**
MIT-BIH	2700	73	326	89.23	97.37
AHA	-	-	-	-	-
EUR-ST	467	590	633	42.45	44.18

These findings suggest that the proposed algorithm classifies the peaks with a very good performance. We also conclude that it can run on mobile devices, without extensive memory or processing requirements.

Part V
Concluding Remarks

Chapter 19
Conclusions and Future Work

This PhD research was mainly focussed on the end-two-end optimization of digital processing of ECG signals. This comes with lots of published results and conclusions.

Our findings during this PhD research are already integrated to the Cloud Based Remote ECG Monitoring portal ECGAlert, supported by the Macedonian Fund for Innovations. Main outcomes and future direction is provided in the next concluding sections.

19.1 Conclusions

Achievements of this PhD research can be summarized in four categories.

19.1.1 Requirement Analysis of Time-critical mHealth Solutions

This is the first and one of the important outcomes. Requirements elicitation of mHealth solution is documented for the time-critical medical monitoring applications. We develop and present the most important functional and non-functional requirements. 2 scientific papers have been published, which not only serves as a basis for this thesis, but also serves a basis for possible implementations of time-critical mHealth solutions.

19.1.2 Optimizing the ECG Data Pre-processing Phase

Along with this thesis, 3 scientific papers have been published on first phase of ECG processing. Primary aim of them was to obtain superliniar speed-up on DSP

filters, i.e mainly the convolution algorithm. CPU, GPU and Dataflow engines were utilized. A comparative analysis is done to find the best parallelisation platform. Our findings indicate that, a combination of multi engines can fascilitate the convolution by yielding a superliniar speedup.

19.1.3 Optimizing the ECG Feature Space Reduction Phase

In this phase, our primary aim was to optimize algorithms for detecting QRS complexes. Any further analysis and extraction of hidden information from ECG data is directly related to the quality and quantity of found QRS complexes. Aligned with this, we have published 6 scientific papers. The results show superior performance over other published results. The improvements were efficiently built in on an industrial QRS detector, for a wearable ECG monitor, where the sampling frequency is 125 Hz, with 10-bit resolution of the AD converter.

19.1.4 Optimizing the ECG Feature Extraction Phase

As a last step, we have also optimized the third step of ECG processing. We introduced a Decision-Tree based model for classification of Normal, Premature Ventricular Contraction and Atrial Premature Contrtaction beats. It requires relatively simple operations and is a pipelined solution, processing beat by beat without extensive memory or processing requirements. Our achievements here is that the proposed algorithm can also run on mobile devices.

19.2 Future Work

This thesis's primary focus was on the first three phases of ECG signal processing. As a future work we see that there are two important directions as stated below.

19.2.1 Classification of QRS detection errors

To analyze differentiation-based algorithms although the methods can be implemented in all other methods. The main goal is to classify the QRS detection errors, and analyze if it is possible to solve them. We believe that this will give an insight to the quality of the QRS database to evaluate the performance of the QRS detection algorithms.

19.2.2 Optimizing ECG Rhythm detection algorithms

Hidden information from ECG can only be extracted by proper detection and classification of QRS complexes. An important future work is to improve ECG Rhythm detection algorithms so that they can run on mobile devices and extract high level of hidden health information.

References

1. Amd radeon and nvidia geforce fp32/fp64 gflops table. https://bit.ly/2Tr1cAY, last visited on 28.05.2016.
2. Cuda c programming guide. http://docs.nvidia.com/cuda/cuda-c-programming-guide, licensed under the Creative Commons Attribution 3.0 Unported licene, last visited on 28.04.2016.
3. The cuda zone. https://developer.nvidia.com/cuda-zone, last visited on 25.04.2016.
4. Maxeler dataflow computing. https://www.maxeler.com/technology/dataflow-computing/, last visited on 26.03.2016.
5. Tuning cuda applications for maxwell. http://docs.nvidia.com/cuda/maxwell-tuning-guide, last visited on 28.05.2016.
6. John P Abenstein. Algorithms for real-time ambulatory ecg monitoring. *Biomedical sciences instrumentation*, 14:73, 1978.
7. Eman AbuKhousa, Nader Mohamed, and Jameela Al-Jaroodi. e-health cloud: opportunities and challenges. *Future Internet*, 4(3):621–645, 2012.
8. Rajendra Acharya, Shankar M Krishnan, Jos AE Spaan, and Jasjit S Suri. *Advances in cardiac signal processing*. Springer, 2007.
9. Paul S Addison. Wavelet transforms and the ECG: a review. *Physiological measurement*, 26(5):R155, 2005.
10. Valtino X Afonso, Willis J Tompkins, Truong Q Nguyen, and Shen Luo. ECG beat detection using filter banks. *IEEE transactions on biomedical engineering*, 46(2):192–202, 1999.
11. Valtino X Afonso, Oliver Wieben, Willis J Tompkins, Truong Q Nguyen, and Shen Luo. Filter bank-based ecg beat classification. In *Engineering in Medicine and Biology society, 1997. Proceedings of the 19th Annual International Conference of the IEEE*, volume 1, pages 97–100. IEEE, 1997.
12. Mark L Ahlstrom and Willis J Tompkins. Automated high-speed analysis of Holter tapes with microcomputers. *IEEE Transactions on Biomedical Engineering*, (10):651–657, 1983.
13. Era Ajdaraga and Marjan Gusev. Analysis of sampling frequency and resolution in ECG signals. In *Telecommunication Forum (TELFOR), 2017 25th*, pages 1–4. IEEE, 11 2017.
14. Mikhled Alfaouri and Khaled Daqrouq. ECG signal denoising by wavelet transform thresholding. *American Journal of applied sciences*, 5(3):276–281, 2008.
15. Jeffrey L Anderson, Cynthia D Adams, Elliott M Antman, Charles R Bridges, Robert M Califf, Donald E Casey, William E Chavey, Francis M Fesmire, Judith S Hochman, Thomas N Levin, et al. ACC/AHA 2007 guidelines for the management of patients with unstable angina/non–ST-elevation myocardial infarction. *Journal of the American College of Cardiology*, 50(7):e1–e157, 2007.

16. Rodrigo Varejão Andreão, Bernadette Dorizzi, and Jérôme Boudy. ECG signal analysis through hidden Markov models. *IEEE Transactions on Biomedical engineering*, 53(8):1541–1549, 2006.
17. Andreas Antoniou. *Digital signal processing*. McGraw-Hill Toronto, Canada:, 2006.
18. Sajid Anwar and Wonyong Sung. Digital signal processing filtering with gpu. 2009.
19. Iffat Ara, Md Najmul Hossain, and SM Yahea Mahbub. Baseline drift removal and de-noising of the ECG signal using wavelet transform. *International Journal of Computer Applications*, 95(16), 2014.
20. Natalia M Arzeno, Zhi-De Deng, and Chi-Sang Poon. Analysis of first-derivative based QRS detection algorithms. *IEEE Transactions on Biomedical Engineering*, 55(2):478–484, 2008.
21. M Bahoura, M Hassani, and M Hubin. DSP implementation of wavelet transform for real time ECG wave forms detection and heart rate analysis. *Computer methods and programs in biomedicine*, 52(1):35–44, 1997.
22. Logica Banica, Liviu Cristian Stefan, et al. Cloud-powered e-health. *Scientific Bulletin-Economic Sciences*, 12(1):26–38, 2013.
23. S Barro, M Fernandez-Delgado, JA Vila-Sobrino, CV Regueiro, and E Sanchez. Classifying multichannel ECG patterns with an adaptive neural network. *IEEE Engineering in Medicine and Biology Magazine*, 17(1):45–55, 1998.
24. Daniel E Becker. Fundamentals of electrocardiography interpretation. *Anesthesia progress*, 53(2):53–64, 2006.
25. D Benitez, PA Gaydecki, A Zaidi, and AP Fitzpatrick. The use of the Hilbert transform in ECG signal analysis. *Computers in biology and medicine*, 31(5):399–406, 2001.
26. Gary G Berntson, Karen S Quigley, Jaye F Jang, and Sarah T Boysen. An approach to artifact identification: Application to heart period data. *Psychophysiology*, 27(5):586–598, 1990.
27. Anna Böhm, Christoph Brüser, and Steffen Leonhardt Leonhardt. A novel bcg sensor-array for unobtrusive cardiac monitoring. *Acta Polytechnica*, 53(6), 2013.
28. Per Ola Borjesson, Olle Pahlm, Leif Sornmo, and Mats-Erik Nygards. Adaptive QRS detection based on maximum a posteriori estimation. *IEEE Transactions on Biomedical Engineering*, (5):341–351, 1982.
29. F Chiarugi, V Sakkalis, D Emmanouilidou, T Krontiris, M Varanini, and I Tollis. Adaptive threshold QRS detector with best channel selection based on a noise rating system. In *Computers in Cardiology, 2007*, pages 157–160. IEEE, 2007.
30. Samjin Choi, Mourad Adnane, Gi-Ja Lee, Hoyoung Jang, Zhongwei Jiang, and Hun-Kuk Park. Development of ECG beat segmentation method by combining lowpass filter and irregular R–R interval checkup strategy. *Expert Systems with Applications*, 37(7):5208–5218, 2010.

31. Ivaylo I Christov. Real time electrocardiogram QRS detection using combined adaptive threshold. *Biomedical engineering online*, 3(1):28, 2004.
32. EJ Ciaccio, SM Dunn, and M Akay. Biosignal pattern recognition and interpretation systems. *IEEE Engineering in Medicine and Biology Magazine*, 12(3):89–95, 1993.
33. Douglas A Coast, Richard M Stern, Gerald G Cano, and Stanley A Briller. An approach to cardiac arrhythmia analysis using hidden Markov models. *IEEE Transactions on biomedical Engineering*, 37(9):826–836, 1990.
34. Wikimedia Commons. File:qrs nomenclature.svg — wikimedia commons, the free media repository, 2018. last visited on 10.03.2019.
35. AA Cost and Gerald G Cano. QRS detection based on hidden Markov modeling. In *Engineering in Medicine and Biology Society, 1989. Images of the Twenty-First Century., Proceedings of the Annual International Conference of the IEEE Engineering in*, pages 34–35. IEEE, 1989.
36. C Cuda. Best practices guide. *Nvidia Corporation*, 2012.
37. Matej Cvikl, Franc Jager, and Andrej Zemva. Hardware implementation of a modified delay-coordinate mapping-based QRS complex detection algorithm. *EURASIP Journal on Applied Signal Processing*, 2007(1):104–104, 2007.
38. Zümray Dokur and Tamer Ölmez. Ecg beat classification by a novel hybrid neural network. *Computer methods and programs in biomedicine*, 66(2-3):167–181, 2001.
39. Zümray Dokur, Tamer Ölmez, Ertugrul Yazgan, and Okan K Ersoy. Detection of ECG waveforms by neural networks. *Medical Engineering and physics*, 19(8):738–741, 1997.
40. E. Domazet, M. Gusev, and S. Ristov. CUDA DSP filter for ECG signals. In *6th International Conference on Applied Internet and Information Technologies*, Bitola, Macedonia, 2016.
41. E. Domazet, M. Gusev, and S. Ristov. Dataflow DSP filter for ECG signals. In *13th International Conference on Informatics and Information Technologies*, Bitola, Macedonia, 2016.
42. E. Domazet, M. Gusev, and S. Ristov. Optimizing high-performance CUDA DSP filter for ECG signals. In *27th DAAAM International Symposium*, Mostar, Bosnia and Herzegovina, 2016. DAAAM International Vienna.
43. Ervin Domazet and Marjan Gusev. Optimal parallel wavelet ecg signal processing. 2017.
44. Ervin Domazet and Marjan Gusev. Parallelization of digital wavelet transformation of ecg signals. In *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 318–323. IEEE, 2017.
45. Ervin Domazet and Marjan Gusev. Parallelization of digital wavelet transformation of ECG signals. In *MIPRO, 2017 Proceedings of the 40th Jubilee International Convention*, Opatija, Croatia, 2017. IEEE.
46. Ervin Domazet and Marjan Gusev. Amplitude rescaling influence on qrs detection. In *International Conference on Telecommunications*, pages 259–272. Springer, 2018.

47. Ervin Domazet and Marjan Gusev. Improving the QRS detection for one-channel ecg sensor. *Technology and Healthcare*, page in press, 2019.
48. Ervin Domazet, Marjan Gusev, and Ljupcho Antovski. A time-critical mobile application based on ecg medical monitoring. In *Proceedings of the 8th Balkan Conference in Informatics*, page 3. ACM, 2017.
49. Ervin Domazet, Marjan Gusev, and Ljupco Antovski. Design specification of an ecg mobile application. In *2017 25th Telecommunication Forum (TELFOR)*, pages 1–4. IEEE, 2017.
50. Ervin Domazet, Can Ozturan, Lale Önsel, and Eltutar Zeynep. Parallelization of a glass furnace mathematical model for multi-core systems. In *2015 Proceedings of the Fourth International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering*. Civil-Comp Press, 2015.
51. Jose Duato, Antonio J Pena, Federico Silla, Rafael Mayo, and Enrique S Quintana-Orti. Performance of cuda virtualized remote gpus in high performance clusters. In *Parallel Processing (ICPP), 2011 International Conference on*, pages 365–374. IEEE, 2011.
52. Mohamed Elgendi, Mirjam Jonkman, and Friso DeBoer. Frequency bands effects on QRS detection. *Pan*, 5:15Hz, 2010.
53. Mohamed Elgendi, Amr Mohamed, and Rabab Ward. Efficient ECG compression and QRS detection for e-health applications. *Scientific reports*, 7(1):459, 2017.
54. Robert J Ellis, Bilei Zhu, Julian Koenig, Julian F Thayer, and Ye Wang. A careful look at ECG sampling frequency and r-peak interpolation on short-term measures of heart rate variability. *Physiological measurement*, 36(9):1827, 2015.
55. WAH Engelse and C Zeelenberg. A single scan algorithm for QRS-detection and feature extraction. *Computers in cardiology*, 6(1979):37–42, 1979.
56. Mehmet Engin. Ecg beat classification using neuro-fuzzy network. *Pattern Recognition Letters*, 25(15):1715–1722, 2004.
57. Themis P Exarchos, Markos G Tsipouras, Costas P Exarchos, Costas Papaloukas, Dimitrios I Fotiadis, and Lampros K Michalis. A methodology for the automated creation of fuzzy expert systems for ischaemic and arrhythmic beat classification based on a set of rules obtained by a decision tree. *Artificial Intelligence in medicine*, 40(3):187–200, 2007.
58. Terrill Fancott and David H Wong. A minicomputer system for direct high speed analysis of cardiac arrhythmia in 24 h ambulatory ECG tape recordings. *IEEE Transactions on Biomedical Engineering*, (12):685–693, 1980.
59. Michael J Flynn, Oliver Pell, and Oskar Mencer. Dataflow supercomputing. In *Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on*, pages 1–3. IEEE, 2012.
60. Centers for Disease Control, Prevention, National Center for Health Statistics, et al. Underlying cause of death 1999-2011 on cdc wonder online database, released 2014. data are from the multiple cause of death files, 1999-2011, as compiled from data provided by the 57 vital statistics jurisdictions through the vital statistics cooperative program, 2014.

61. J Fraden and MR Neuman. QRS wave detection. *Medical and Biological Engineering and computing*, 18(2):125–132, 1980.
62. A Ghaffari, MR Homaeinezhad, M Akraminia, M Atarod, and M Daevaeiha. A robust wavelet-based multi-lead electrocardiogram delineation algorithm. *Medical Engineering and Physics*, 31(10):1219–1227, 2009.
63. Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet. *Circulation*, 101(23):e215–e220, 2000.
64. Marjan Gusev and Ervin Domazet. Optimal dsp bandpass filtering for qrs detection. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 0303–0308. IEEE, 2018.
65. Marjan Gusev and Ervin Domazet. Optimizing the impact of resampling on qrs detection. In *International Conference on Telecommunications*, pages 107–119. Springer, 2018.
66. Marjan Gusev, Aleksandar Ristovski, and Ana Guseva. Distribution of premature heartbeats. In *Telecommunications Forum (TELFOR), 2016 24th*, pages 1–4. IEEE, 2016.
67. Marjan Gusev, Aleksandar Ristovski, and Ana Guseva. Pattern recognition of a digital ECG. In *International Conference on ICT Innovations*, pages 93–102. Springer, 2016.
68. Marjan Gusev, Aleksandar Stojmenski, and Ivan Chorbev. Challenges for development of an ECG m-health solution. *Journal of Emerging Research and Solutions in ICT*, 1(2):25–38, 2016.
69. Patrick S Hamilton and Willis J Tompkins. Quantitative investigation of QRS detection rules using the MIT-BIH arrhythmia database. *IEEE transactions on biomedical engineering*, (12):1157–1165, 1986.
70. PS Hamilton. Open source ECG analysis software documentation, 2002.
71. C Holzmann, U Hasseldieck, E Rosselot, P Estevez, A Andrade, and G Acuna. Interpretation module for screening normal ecg. *Medical progress through technology*, 16(3):163–171, 1990.
72. http://www.heart.org. American health association, AHA database.
73. Yu Hen Hu, Surekha Palreddy, and Willis J Tompkins. A patient-adaptable ecg beat classifier using a mixture of experts approach. *IEEE transactions on biomedical engineering*, 44(9):891–900, 1997.
74. Yu Hen Hu, Willis J Tompkins, Jose L Urrusti, and Valtino X Afonso. Applications of artificial neural networks for ECG signal detection and classification. *Journal of electrocardiology*, 26:66–73, 1993.
75. Aderonke Justina Ikuomola and Oluremi O Arowolo. Securing patient privacy in e-health cloud using homomorphic encryption and access control. *International Journal of Computer Networks and Communications Security*, 2(1):15–21, 2014.

76. Intel. Quick-reference guide to optimization with intel compilers version 12, Jan. 2010. https://software.intel.com/sites/default/files/compiler_qrg12.pdf.
77. International Electrotechnical Commission. IEC 60601-2-47:2012 medical electrical equipment - part 2-47: Particular requirements for the basic safety and essential performance of ambulatory electrocardiographic systems.
78. Ahmad Khoureich Ka. Ecg beats classification using waveform similarity and rr interval. *arXiv preprint arXiv:1101.1836*, 2011.
79. Alexander Kaletsch and Ali Sunyaev. Privacy engineering: personal health records in cloud computing environments. 2011.
80. Sema Kayhan and Ergun Ercelebi. ECG denoising on bivariate shrinkage function exploiting interscale dependency of wavelet coefficients. *TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES*, 19(3):495–511, 2011.
81. P Kinias and HA Fozzard. Rapid ecg analysis and arrhythmia detection. *Computer Techniques in Cardiology*, pages 98–122, 1979.
82. B-U Kohler, Carsten Hennig, and Reinhold Orglmeister. The principles of software QRS detection. *IEEE Engineering in Medicine and Biology Magazine*, 21(1):42–57, 2002.
83. P Laguna, Nitish V Thakor, P Caminal, R Jane, Hyung-Ro Yoon, A Bayés de Luna, V Marti, and Josep Guindo. New algorithm for qt interval analysis in 24-hour Holter ECG: performance and applications. *Medical and Biological Engineering and Computing*, 28(1):67–73, 1990.
84. Jeong-Whan Lee, Kyeong-Seop Kim, Bongsoo Lee, Byungchae Lee, and Myoung-Ho Lee. A real time QRS detection using delay-coordinate mapping for the microcontroller implementation. *Annals of biomedical Engineering*, 30(9):1140–1151, 2002.
85. Jeongwhan Lee, Keesam Jeong, Jiyoung Yoon, and Myoungho Lee. A simple real-time QRS detection algorithm. In *Engineering in Medicine and Biology Society, 1996. Bridging Disciplines for Biomedicine. Proceedings of the 18th Annual International Conference of the IEEE*, volume 4, pages 1396–1398. IEEE, 1996.
86. Heike Leutheuser, Tristan Gottschalk, Lars Anneken, Matthias Struck, Albert Heuberger, Martin Arnold, Stephan Achenbach, and Bjoern M Eskofier. Automatic ECG arrhythmia detection in real-time on android-based mobile devices. In *Proceedings of the MobileMed 2014 conference, November 20th-21st, Prague, Czech Republic*, 2014.
87. Cuiwei Li, Chongxun Zheng, and Changfeng Tai. Detection of ECG characteristic points using wavelet transforms. *IEEE Transactions on biomedical Engineering*, 42(1):21–28, 1995.
88. Tatiana S Lugovaya. Biometric human identification based on ECG, 2005.
89. Nicos Maglaveras, Telemachos Stamkopoulos, Konstantinos Diamantaras, Costas Pappas, and Michael Strintzis. ECG pattern recognition and classification using non-linear transformations and neural networks: a review. *International journal of medical informatics*, 52(1-3):191–208, 1998.

90. M Malik. Task force of the european society of cardiology and the north american society of pacing and electrophysiology. heart rate variability. standards of measurement, physiological interpretation, and clinical use. *Eur Heart J.*, 17:354–381, 1996.
91. Thomas Martin, Emil Jovanov, and Dejan Raskovic. Issues in wearable computing for medical monitoring applications: a case study of a wearable ECG monitoring device. In *Wearable Computers, The Fourth International Symposium on*, pages 43–49. IEEE, 2000.
92. Arturo Martínez, Raúl Alcaraz, and José Joaquín Rieta. Application of the phasor transform for automatic delineation of single-lead ECG fiducial points. *Physiological measurement*, 31(11):1467, 2010.
93. Juan Pablo Martínez, Rute Almeida, Salvador Olmos, Ana Paula Rocha, and Pablo Laguna. A wavelet-based ECG delineator: evaluation on standard databases. *IEEE transactions on biomedical engineering*, 51(4):570–581, 2004.
94. Joseph Mattai and Mathai Joseph. *Real-Time Systems: specification, verification, and analysis*. Prentice Hall PTR, 1995.
95. Friedemann Mattern and Christian Floerkemeier. From the internet of computers to the internet of things. In *From active data management to event-based systems and more*, pages 242–259. Springer, 2010.
96. Michael P McGraw-Herdeg, Douglas P Enright, and B Scott Michel. Benchmarking the nvidia 8800gtx with the cuda development platform. *HPEC 2007 Proceedings*, 2007.
97. Priyanka Mehta and Monika Kumari. QRS complex detection of ECG signal using wavelet transform. *International Journal of Applied Engineering Research*, 7(11):1889–1893, 2012.
98. Aleksandar Milchevski and Marjan Gusev. Improved pipelined wavelet implementation for filtering ECG signals. *Pattern Recognition Letters*, 95:85–90, 2017.
99. George B Moody and Roger G Mark. The impact of the MIT-BIH arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine*, 20(3):45–50, 2001.
100. P Morizet-Mahoudeaux, C Moreau, D Moreau, and JJ Quarante. Simple microprocessor-based system for on-line ECG arrhythmia analysis. *Medical and Biological Engineering and Computing*, 19(4):497–500, 1981.
101. FM Nolle, RW Bowser, FK Badura, JM Catlett, RR Guadapati, TT Hee, AN Mooss, and MH Sketch. Evaluation of a frequency-domain algorithm to detect ventricular fibrillation in the surface electrocardiogram. In *Computers in Cardiology, 1988. Proceedings.*, pages 337–340. IEEE, 1988.
102. M-E Nygårds and L Sörnmo. Delineation of the QRS complex using the envelope of the ECG. *Medical and Biological Engineering and Computing*, 21(5):538–547, 1983.
103. Harry Nyquist. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 47(2):617–644, 1928.

104. Masahiko Okada. A digital filter for the QRS complex detection. *IEEE Transactions on Biomedical Engineering*, (12):700–703, 1979.
105. OpenMP. Openmp application program interface version 3.0. 2008, Jan. 2014]. `http://www.openmp.org/mp-documents/spec30.pdf`.
106. O Pahlm and L Sörnmo. Software QRS detection in ambulatory monitoring: A review. *Medical and Biological Engineering and Computing*, 22(4):289–297, 1984.
107. Jiapu Pan and Willis J Tompkins. A real-time QRS detection algorithm. *IEEE transactions on biomedical engineering*, (3):230–236, 1985.
108. Suraj Pandey, William Voorsluys, Sheng Niu, Ahsan Khandoker, and Rajkumar Buyya. An autonomic cloud environment for hosting ECG data analysis services. *Future Generation Computer Systems*, 28(1):147–154, 2012.
109. Matteo Paoletti and Carlo Marchesi. Discovering dangerous patterns in long-term ambulatory ECG recordings using a fast QRS detection algorithm and explorative data analysis. *Computer Methods and programs in biomedicine*, 82(1):20–30, 2006.
110. Abhilasha M Patel, Pankaj K Gakare, and AN Cheeran. Real time ECG feature extraction and arrhythmia detection on a mobile platform. *Int. J. Comput. Appl*, 44(23):40–45, 2012.
111. P.D.Khandait, N.G. Bawane, and S.S.Limaye. Article: Features extraction of ECG signal for detection of cardiac arrhythmias. *IJCA Proceedings on National Conference on Innovative Paradigms in Engineering and Technology (NCIPET 2012)*, ncipet(8):6–10, March 2012. Full text available.
112. Everett H Phillips and Massimiliano Fatica. Implementing the himeno benchmark with cuda on gpu clusters. In *Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, pages 1–10. IEEE, 2010.
113. Riccardo Poli, Stefano Cagnoni, and Guido Valli. Genetic design of optimum linear and nonlinear QRS detectors. *IEEE Transactions on Biomedical Engineering*, 42(11):1137–1141, 1995.
114. Pavel Rajmic and Jan Vlach. Real-time audio processing via segmented wavelet transform. In *Proc. of the 10th Int. Conference on Digital Audio Effects (DAFx-07), Bordeaux, France*. Citeseer, 2007.
115. Anders P. Ravn, Hans Rischel, and Kirsten Mark Hansen. Specifying and verifying requirements of real-time systems. *IEEE Transactions on Software Engineering*, 19(1):41–55, 1993.
116. Carlos Oberdan Rolim, Fernando Luiz Koch, Carlos Becker Westphall, Jorge Werner, Armando Fracalossi, and Giovanni Schmitt Salvador. A cloud computing solution for patient's data collection in health care institutions. In *eHealth, Telemedicine, and Social Medicine, 2010. ETELEMED'10. Second International Conference on*, pages 95–99. IEEE, 2010.
117. C Saritha, V Sukanya, and Y Narasingh Murty. ECG signal analysis using wavelet transformation. *BulgJ Physics, pp-68-77,(35)*, 2008.
118. H Sava, M Fleury, AC Downton, and AF Clark. Parallel pipeline implementation of wavelet transforms. *IEE Proceedings-Vision, Image and Signal Processing*, 144(6):355–359, 1997.

119. Saving. Savvy ECG biosensor, 2017.
120. DU Shah and CH Vithlani. Efficient implementations of discrete wavelet transforms using fpgas. *International Journal of Advances in Engineering & Technology*, 1(4):100–111, 2011.
121. JS Shambi, SN Tandon, and RKP Bhatt. Using wavelet transforms for ECG characterization. *IEEE Engineering in Medicine and Biology*, pages 77–83, 1997.
122. John A Sharp. *Data flow computing: theory and practice*. Intellect Books, 1992.
123. Steven W Smith. *Digital signal processing: a practical guide for engineers and scientists*. Newnes, 2003.
124. Zhou Song-Kai, Wang Jian-Tao, and Xu Jun-Rong. The real-time detection of QRS-complex using the envelope of ECG. In *Engineering in Medicine and Biology Society, 1988. Proceedings of the Annual International Conference of the IEEE*, pages 38–vol. IEEE, 1988.
125. Dieter Speidel and Mithun Sridharan. Quality assurance in the age of mobile healthcare. *The Journal of mHealth*, 01(5):42–46, 2014.
126. CA Steinberg, S Abraham, and CA Caceres. Pattern recognition in the clinical electrocardiogram. *IRE Transactions on Bio-Medical Electronics*, 9(1):23–30, 1962.
127. Radovan Stojanović, Saša Knežević, Dejan Karadaglić, and Goran Devedžić. Optimization and implementation of the wavelet based algorithms for embedded biomedical signal processing. *Computer Science and Information Systems*, 10(1):503–523, 2013.
128. Michael Stonebraker, Uğur Çetintemel, and Stan Zdonik. The 8 requirements of real-time stream processing. *ACM SIGMOD Record*, 34(4):42–47, 2005.
129. Borys Surawicz, Rory Childers, Barbara J Deal, and Leonard S Gettes. Aha/accf/hrs recommendations for the standardization and interpretation of the electrocardiogram: part iii: intraventricular conduction disturbances a scientific statement from the american heart association electrocardiography and arrhythmias committee, council on clinical cardiology; the american college of cardiology foundation; and the heart rhythm society endorsed by the international society for computerized electrocardiology. *Journal of the American College of Cardiology*, 53(11):976–981, 2009.
130. A Taddei, G Distante, M Emdin, P Pisani, GB Moody, C Zeelenberg, and C Marchesi. The European ST-T database: standard for evaluating systems for the analysis of ST-T changes in ambulatory electrocardiography. *European heart journal*, 13(9):1164–1172, 1992.
131. Jurij Tasic, Marjan Gusev, and Sasko Ristov. A medical cloud. In *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2016 39th International Convention on*, pages 400–405. IEEE, 2016.
132. Panagiotis Trahanias and Emmanuel Skordalakis. Syntactic pattern recognition of the ECG. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):648–657, 1990.

133. Markos G Tsipouras, Dimitrios I Fotiadis, and D Sideris. An arrhythmia classification system based on the rr-interval signal. *Artificial intelligence in medicine*, 33(3):237–250, 2005.
134. Galen S Wagner, Peter Macfarlane, Hein Wellens, Mark Josephson, Anton Gorgels, David M Mirvis, Olle Pahlm, Borys Surawicz, Paul Kligfield, Rory Childers, et al. Aha/accf/hrs recommendations for the standardization and interpretation of the electrocardiogram: part vi: acute ischemia/infarction a scientific statement from the american heart association electrocardiography and arrhythmias committee, council on clinical cardiology; the american college of cardiology foundation; and the heart rhythm society endorsed by the international society for computerized electrocardiology. *Journal of the American College of Cardiology*, 53(11):1003–1011, 2009.
135. Frank Wefers and Jan Berg. High-performance real-time fir-filtering using fast convolution on graphics hardware. In *Proc. of the 13th Conference on Digital Audio Effects*, 2010.
136. Nathan Whitehead and Alex Fit-Florea. Precision & performance: Floating point and ieee 754 compliance for nvidia gpus. *rn (A+ B)*, 21(1):18749–19424, 2011.
137. O Wieben, VX Afonso, and WJ Tompkins. Classification of premature ventricular complexes using filter bank features, induction of decision trees and a fuzzy rule-based system. *Medical & biological engineering & computing*, 37(5):560–565, 1999.
138. Qiuzhen Xue, Yu Hen Hu, and Willis J Tompkins. Neural-network-based adaptive matched filtering for QRS detection. *IEEE Transactions on Biomedical Engineering*, 39(4):317–329, 1992.
139. Zahia Zidelmal, Ahmed Amirou, Mourad Adnane, and Adel Belouchrani. QRS detection based on wavelet coefficients. *Computer methods and programs in biomedicine*, 107(3):490–496, 2012.
140. Tjalf Ziemssen, Julia Gasch, and Heinz Ruediger. Influence of ECG sampling frequency on spectral analysis of rr intervals and baroreflex sensitivity using the eurobavar data set. *Journal of clinical monitoring and computing*, 22(2):159–168, 2008.