Algoritmer och metoder vid sökning och identifiering av ljud

Andreas Näsman, 36156
Kandidatavhandling i Datateknik
Handledare: Jerker Björkqvist
Fakulteten för naturvetenskaper och teknik
Åbo Akademi
2016

Referat

Användningen av ljudidentifikation har ökat markant de senaste åren. Så gott som alla smarttelefoner har idag en inbyggd hjälpreda som svarar på ljudförfrågningar och många tjänster erbjuder service med system som mottar och svarar med ljud. Algoritmerna och metoderna bakom identifieringen kan vid första ögonkastet verka rätt så komplexa och svårbegripliga, men är i själva verket relativt okomplicerade och enkelt uppbyggda när man tar en närmare titt.

Den här avhandlingen behandlar i första hand hur ljudsökningsalgoritmen som uppfanns av företaget Shazam i början av 2000-talet fungerar, vilken bl.a. används vid sökning och identifikation av låtklipp. Många förbättringar har gjorts under åren för att utveckla algoritmen, och de mest markanta tilläggen tas upp i den här avhandlingen. Sökning genom att nynna är en annan metod som ofta används vid sökning av ljud, vars algoritm också tas upp i den här avhandlingen.

Hur man genom att nynna en låt kan identifiera den samt hur tal-till-text i enkelhet fungerar tas också upp. Genom att kombinera de här metoderna och funktionerna får vad företaget SoundHound erbjuder i sin Hound-app, troligen den mest avancerade och lovande ljudigenkänningstjänsten hittills.

Innehållsförteckning

1 Inledning 3
2 Användningsområden och mål 3
3 Igenkänning av musik med Shazams ljudsökningsalgoritm 4
3.1 Historia 4
3.2 Processen i korthet 5
3.3 Framställning av användbart data 6
3.4 Optimering av söktid 7
3.5 Sökning och matchning av data 9
3.6 Precision 12
3.6 Sammanfattning av Shazams ljudsökningsalgoritm 13
4 Förbättringsförslag till Shazams ljudsökningsalgoritm 14
5 Sökning genom att nynna 15
5 Sökning genom att nynna ..... **Error! Bookmark not defined.**
6 Tal-till-text ..... **Error! Bookmark not defined.**
7 Slutsatser och diskussion ..... **Error! Bookmark not defined.**
8 Sammanfattning 17
Referenser ..... **Error! Bookmark not defined.**

1 Inledning

Tal och gester är för de flesta människor det mest naturliga och effektiva sättet att kommunicera på. Det hör till människans natur och har utvecklas under en mycket lång tid. På det här sättet kan man snabbt ta reda på något man undrar över och ofta också samtidigt lära sig andra intressanta saker som man kanske inte själv tänkt på. Fastän språket inte är samma för alla människor så är ändå grundläggande läten och kroppsspråk samma i gott som hela världen. Vare sig det rör sig om att fråga vägen till närmaste bankautomat eller ta reda på vilka ingredienser en maträtt innehåller föredrar många att ställa frågan verbalt och kunna kommunicera fram svaret.

Dagens textbaserade värld kan därför ibland ses som ganska stel och livlös. Genom att implementera system som ger respons på ljud kan man således liva upp och förbättra användarens upplevelse. Det område som den här avhandlingen kommer att fokusera på är användningen och utvecklingen av ljudresponsiva system vid igenkänning av musik. Febrilt googlande av vagt uppfattade delar av en låttext har under åren ersatts med telefonappar som direkt genom att lyssna på en del av låten eller nynnande av den kan identifiera låten och på samma gång ge givande bakgrundsinformation. För att göra detta krävs dock en viss grad av pålitlighet och funktionalitet. Ljudbaserade systemen är ofta mera komplicerade än deras textbaserade motpart och har stor risk för frustration bland användare när de inte fungerar.

2 Användningsområden och mål

Ljudidentifikation är ett ljudresponsivt system som idag används för många olika ändamål. Appar som Hound [1] erbjuder igenkänning av musik, navigering och sökning av olika slag, allt baserat på olika ljud- och talförfrågningar eller kommandon. Algoritmerna som används vid sökning ger resultat snabbt, även med lite indata som kan vara av dålig kvalité med mycket bakgrundsljud och utsatt för kompression.

Andra användningsområden för algoritmerna är t.ex. spårning av illegal användning av upphovsrättsskyddade material, vilket hemsidor som t.ex. Youtube och Twitch gör för att söka igenom sina nyligen publicerade videor. Sökning av citat eller andra textutdrag för att hitta ursprungskällan bland skrifter i bibliotek, arkiv eller dylika platser har också möjliggjorts med liknande algoritmer. [2] [3]

Målet för utvecklingen av identifikation genom tal och ljud är att ersätta de

traditionella textbaserade inenheterna, så som tangentbord och pekskärmar. Ljudidentifikation skulle alltså bli det föredragna sättet för användare att hantera sin smarttelefon eller liknande enheter. Grundläggande argument för förespråkande av ljudidentifikation framom textidentifikation är bl.a. snabbare funktioner, träffsäkrare resultat och ökad tolerans för störning. Människor kommunicerar vanligen snabbare och mera precist via tal än text, då störningar som bl.a. tryckfel endast förekommer i skrift. Detta skulle t.ex. hjälpa dyslektiker att navigera sina enheter mera smärtfritt. [4]

3 Igenkänning av musik med Shazams ljudsökningsalgoritm

3.1 Historia

Shazam var ett av de första företagen som erbjöd kunder sökning av okända låtar. Företaget grundades år 2000 och deras sökningstjänst lanserades 2002 i Storbritannien, Tyskland och Finland, troligen länder där antalet mobiltelefoner per capita var hög. Tjänsten finns ännu kvar och fungerar på liknande sätt som den gjorde då; kunden gör en förfrågan med ett ca 15 sekunders klipp av låten den vill få information om, Shazam söker igenom sin databas med låtar och skickar därefter ett svar till kunden om en möjlig träff. Till skillnad från nu så ringde man då upp Shazam och kunde på så sätt få fram sin förfrågan. Svaret skickades i form av ett textmeddelande med låtnamnet och artisten. Idag räcker det med att öppna appen och låta den lyssna, varefter appen själv tar kontakt med tjänsten och visar svaret direkt på skärmen.

Genom att registrera sig på Shazams webbplats och logga in med telefonnummer och lösenord hade kunden tillgång till en lista på sina tidigare gjorda sökningar, nedladdning av ringsignaler och köp av CD-skivor av artisten kunden sökt låtar om. Den sistnämnda funktionen har med tiden blivit standard för alla liknande tjänster, troligen p.g.a. av understöd från skivbolag som sett detta som en attraktiv funktion som gynnar dem. Om man idag använder Shazam eller någon av konkurrenterna får man information om det mesta som har någon – mer eller mindre – direkt anknytning till låten man sökt. Låtens namn, artisten, albumet, låttexten, musikvideon, möjlighet att köpa och ladda ned låten, lyssna på låten på andra tjänster så som Spotify och Apple Music osv. erbjuds om den matchade låten. [2]

3.2 Processen i korthet

Förenklat fungerar Shazams ljudsökningsalgoritmen för musik på liknande sätt som sökningen av fingeravtryck i ett brottsregister. Låtars igenkänningstecken och karakteristika kan ritas upp i unika spektogram. Ett spektogram är ett diagram uppbyggt av tiden på x-axeln, frekvensen på y-axeln och intensiteten som fås via svärtningsgraden, dvs. mängden färg på ett visst område (syns som mörka fläckar i figur 1A). Spektogram görs för bekanta låtar och omvandlas till 64-bitars strukturer som finns sparade i en databas, på samma sätt som fingeravtryck för kända kriminella är sparade i en databas i ett brottsregister. När användaren matar in ett ljudklipp, t.ex. en del av en låt där namnet är okänt, skapar algoritmen ett spektogram av ljudklippet, vilket görs till en 64-bitars struktur som den försöker matcha med existerande strukturer i databasen. Om sökningen ger ett positivt resultat visas informationen för det sökta objektet där då också namnet framgår. För att få tillgång till så många låtar som möjligt att jämföra med används webbplatser där användare själva får föreslå låtar som läggs till i databasen. [5]

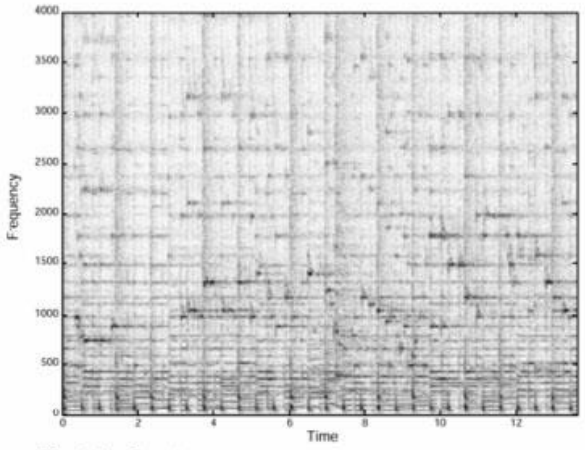


Fig. 1A - Spectrogram

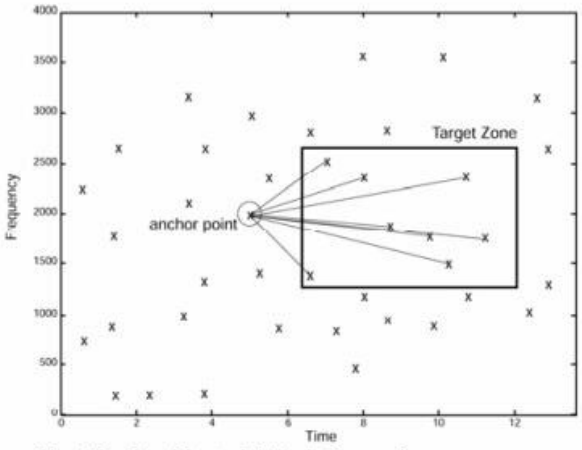


Fig. 1C - Combinatorial Hash Generation

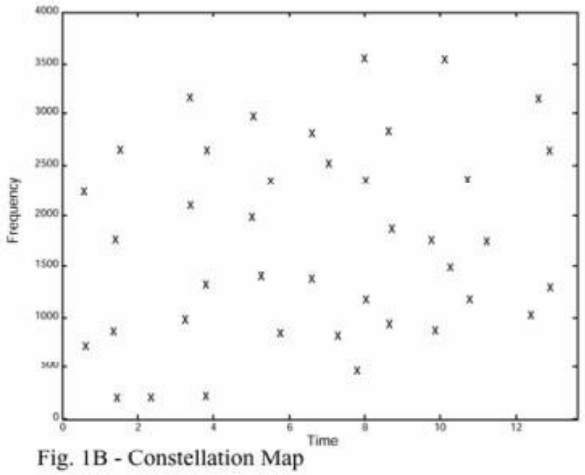


Fig. 1B - Constellation Map

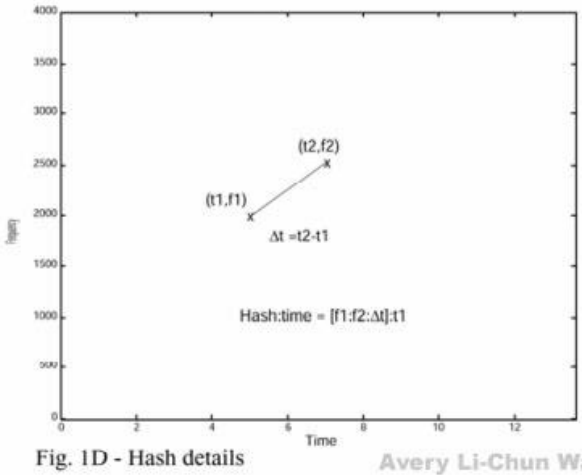


Fig. 1D - Hash details

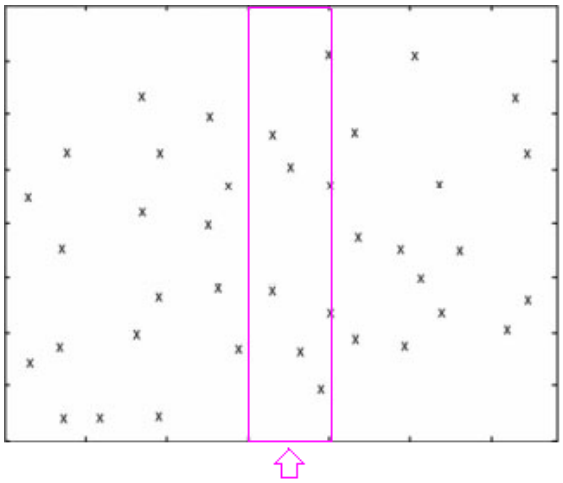
3.3 Framställning av användbart data

Algoritmen som Shazam utvecklade under 2000-talet tycks vara grundläggande för de flesta liknande appar eller tjänster idag. Wang [2] nämner i sin text att liknande företag som grundades efter Shazam använt sig av andra liknande algoritmer som *Philips' robust hashing algorithm* och *Muscle Fish algorithm*.

Orsaken till att Shazam valde att utveckla en ny algoritm till deras sökmotor var att ingen existerande algoritm uppfyllde deras krav på snabbhet, robusthet och precision. Algoritmen måste snabbt med hög sannolikhet kunna hitta en låt bland miljontals kända låtar; antalet felförutsägelser fick alltså inte vara många. Fastän data som algoritmen får in är påverkade av kompression, eko, bristfällig information p.g.a. avbrott i nätverksanslutningen, bakgrundsljud eller andra störande element, måste algoritmen fortfarande fungera med hög prestanda jämfört med optimala förhållanden.

Användningen av spektogram som förklarades i korthet förut lämpar sig väldigt bra för detta ändamål. Omvandlingen från musik till spektrogram filtrerar bort störande och onödiga element och sparar endast det ”viktigaste” av en låt i form av s.k. formanter. Formanter är lätta att upptäcka i ett spektrogram där de framhävs som mörka fläckar, vilket kan ses i figur 1A tagen ur Wangs [2] text. De är egentligen toppar i spektogrammet, tillfällen då musikens röster eller instrument har hög energi, alltså där volymskillnader eller andra förändring sker. Formanter under hög amplitud har också högst sannolikhet att upprepade gånger kunna reproduceras, även om störningar förekommer. Ofta är det just formanterna man hör om man lyckas urskilja en låt i ett gatuvimmel eller någon annan högljudd miljö.

Figur 1B är således en förenklad version av figur 1A, där endast formanterna har bevarats som kryss och resten filtrerats bort. Eftersom spektogrammen för låtar är unika, så är också delar av spektogrammen unika. Om man alltså nu tänker att spektrogrammet för en hel låt och en del av samma låt ritas ut på en bred respektive en smal bit genomskinlig plastfilm, så kommer man att hitta endast ett ställe där formanterna passar in med varandra, om man för den smala plastbiten över den större plastbiten.



Spektogram för hela låten (stor plastbit)

Spektrogram för låtdel (liten plastbit)

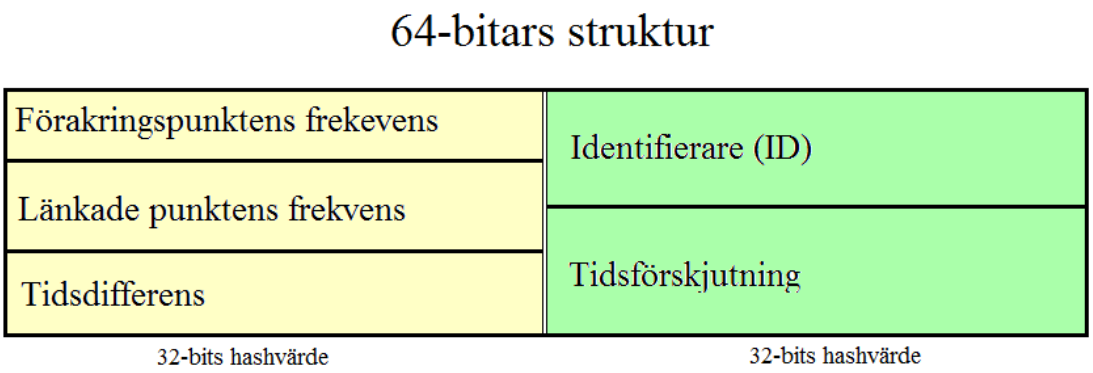
Även om vissa formanter faller bort från den smala plastbiten, eller några nya kryss ritas in så är förändring inte så värst stor och resultatet kommer fortfarande bli det samma. Systemet är alltså relativt robust och fungerar även under vissa förändringar.

3.4 Optimering av söktid

Valet av att spara kända låtar i form av spektrogram fyller alltså i stor mån kraven på robusthet och precision. För att göra sökningstiden så kort som möjligt omvandlas data från spektogrammen till mera lämplig form för att sparas i s.k. hashtabeller. Hashtabeller lämpar sig bra för det här ändamålet då man har olika nycklar att använda och kollisionsrisken är rätt så låg då varje spektogram är så unikt. Vissa punkter i spektrogrammen väljs som *anchor points,* dvs. punkter som förankras som utgångspunkt till andra punkter, vilket figur 1C visar. Denna typ av punkt kommer hädanefter att benämnas *förankringspunkt*. Varje enskild förankringspunkt länkas samman med punkter i ett målområde, som är sammankopplat med just den förankringspunkten.

Varje länk i spektogrammen sparas i 32-bitars positiva heltal (unsigned integer) – alltså i ett hashvärde – där förankringspunktens frekvens och den länkade punktens frekvens ingår, liksom tidsdifferensen mellan punkterna. För att hålla reda på länkarnas ordning och vilken låt de tillhör sparas dessa med en korresponderande 32-bitars del där låtens id och hashvärdets tidsförskjutning från början av låten ingår. Wang [2] nämner att antalet länkar och deras täthet varierar beroende på användningsområdet för algoritmen. De här två 32-bit delarna utgör tillsammans alltså en 64-bitars struktur

där all väsentlig data finns sparade. Ett låtklipp består således av flera 64-bitars strukturer, beroende på hur många förankringspunkter som bildas.



Denna process görs för alla låtar i databasen, men också för låtklippen som ska identifieras. Genom att spara länkar mellan två punkter istället för varje punkt enskilt minskar sökningstiden för hashtabellen drastiskt. Fastän antalet element i tabellen ökar och utrymmet för att lagra data blir ungefär 10 gånger större, kommer ändå söktiden att bli ca 10 000 gånger snabbare enligt Wang [2].

Precisionen blir också en aning sämre, då detta sätt att spara spektrogrammen kräver att båda formanterna i länken är urskiljbara. Förankringspunkterna är valda så att de högst troligen går att urskilja även om störningen är stor för låtklippet som matas in. När samma hashingsprocess görs för det inmatade klippet kommer kanske länkar att försvinna då vissa formanter i förankringspunkternas målområden är för svaga för att urskiljas eller någon ny felaktig länk att skapas p.g.a. störning, men högst troligen kommer ändå majoriteten av länkarna att kunna reproduceras korrekt. Wang [2] säger att förlusten i precision är så liten i jämförelse med snabbheten som vinns att kompromissen är värd att göra.

Om man tänker på detta i praktiken blir det ganska uppenbart varför det lönar sig att spara en låts formanter i par snarare än enskilt. Förenklar man musiken till bara ett instrument, väljer en lätt igenkännlig ton av någon viss frekvens och jämför den med musik av samma instrument, kommer den troligen att hittas i de flesta jämförda fall, kanske t.o.m. flera gånger. Om man däremot jämför två toner – i musiktermer kallat ett intervall – kommer antalet träffar bara bli ett fåtal gentemot det första

scenariot. Tre toner skulle troligen ge ännu bättre resultat, men risken för att en länk med tre toner inte ska gå att reproducera blir för stor.

3.5 Sökning och matchning av data

När sökningen sker försöker Shazams algoritm matcha det inmatade data med data som finns sparade i databasen. Varje struktur som gjorts från spektogrammet för data man vill identifiera enligt processen som beskrevs tidigare, matchas nu mot strukturer sparade i databasen. Eftersom delar av en viss låt, alltså vissa av strukturerna, också förekommer i andra låtar kommer alla matchande resultat att sorteras i *lådor* (eng. *bin).* Resultaten sorteras enligt ID, så varje låda kommer alltså att representera en enskild låt. En del av lådorna kommer bara att fyllas med ett fåtal träffar, medan andra kan bli fyllda med flertal matchningar.

Då alla strukturer för låtklippet har kollats med databasen görs en scannings-process. Varje låda innehåller nu alltså träffar för länkarna som bildades ur det inmatade låtklippets formanter ur dess spektogram. För att en låda – representerande en låt – ska kunna bestämmas vara den rätta låten bör den innehålla flera träffar på rad; några träffar utspridda här och där duger alltså inte som rätt resultat. Eftersom strukturerna innehåller information om tidsförskjutningen från början av låten kan träffarna i lådorna sorteras enligt den kronologiska ordningen som de borde förekomma i det inmatade låtklippet och den ursprungliga låten i databasen.

Varje låda, eller varje potentiell låtkandidat kollas skilt och resultatet kan ritas upp i ett sambandsdiagram och ett histogram. Figur 2A och 2B visar hur resultatet ser ut för ett låtklipp och en låt i databasen med flera matchade strukturer, men utspridda över hela låten.

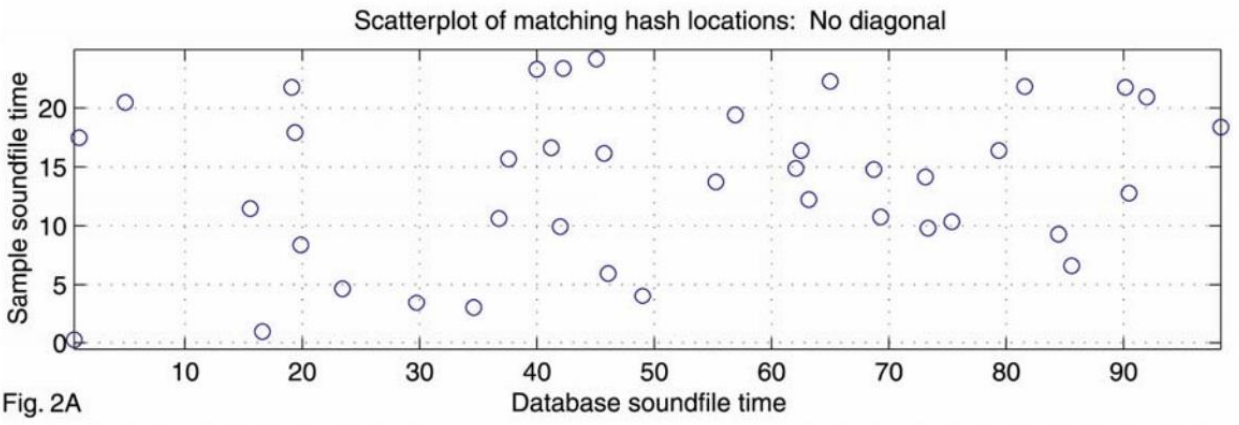


Fig. 2A

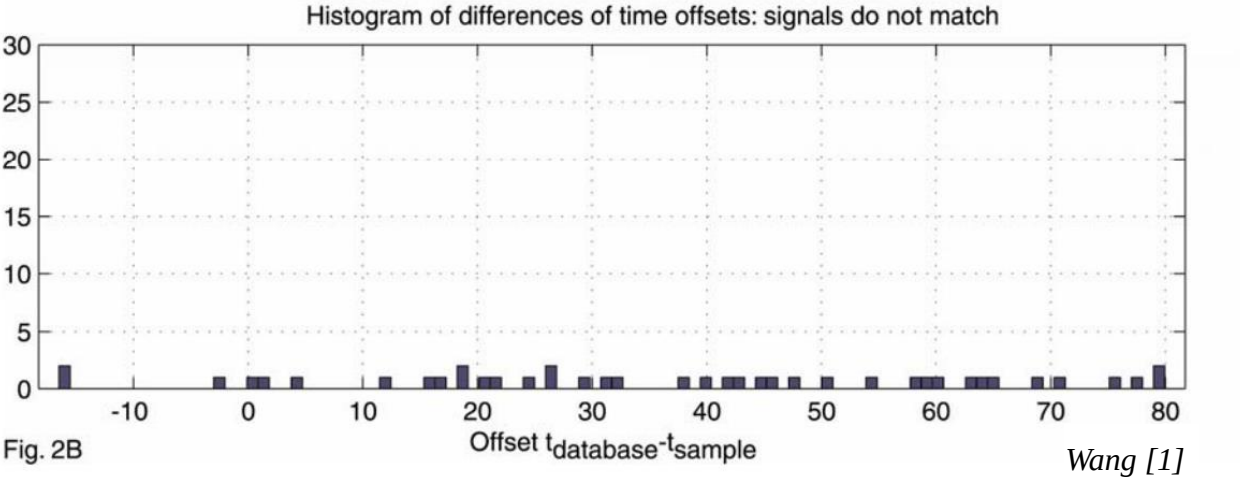


Fig. 2B

Wang [1]

X-axeln i det övre sambandsdiagrammet representerar fortskriden tid för låten i databasen och y-axeln fortskriden tid för det inmatade låtklippet. Det nedre histogrammet visar hur många träffar det finns med en viss tidsdifferens mellan inmatade data och databasens data. Träffarna (ritade som bollar i sambandsdiagrammet) är utspridda och bildar ingen diagonal någonstans i sambandsdiagrammet eller stapel i histogrammet, vilket är kännetecknande för rätt resultat. Figur 2A och 2B visar alltså hur det ser ut när ett låtklipp inte matchar en potentiell kandidat i databasen.

Figur 3A och 3B visar däremot hur det ser ut när en matchning har lyckats. En tydlig diagonal kan ses i sambandsdiagrammet, vilket i sin tur omvandlas till en stapel i histogrammet. I det här fallet har tidsdifferensen för 40 sekunder i histogrammet över 25 träffar, då resten har en eller två. Figur 4A och 4B visar hur man lättare kan visualisera omvandlingen från sambandsdiagrammet till histogrammet. Värt att notera är att diagonalens vinkel är förutbestämt och hålls konstant.

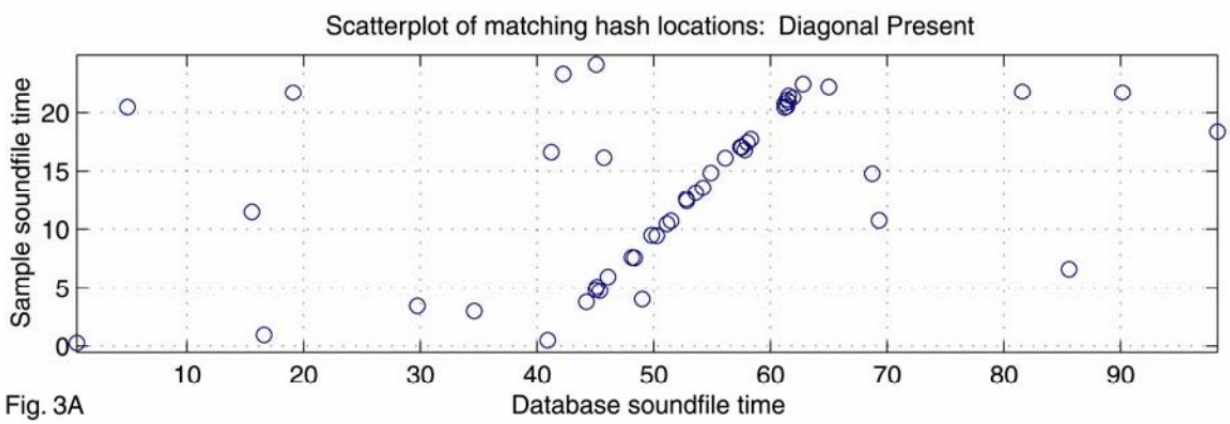


Fig. 3A

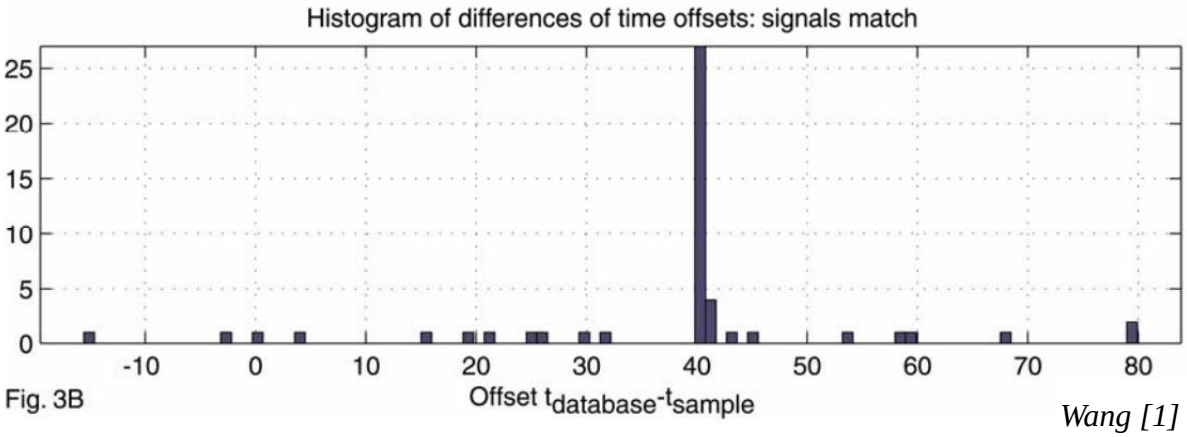


Fig. 3B

Wang [1]

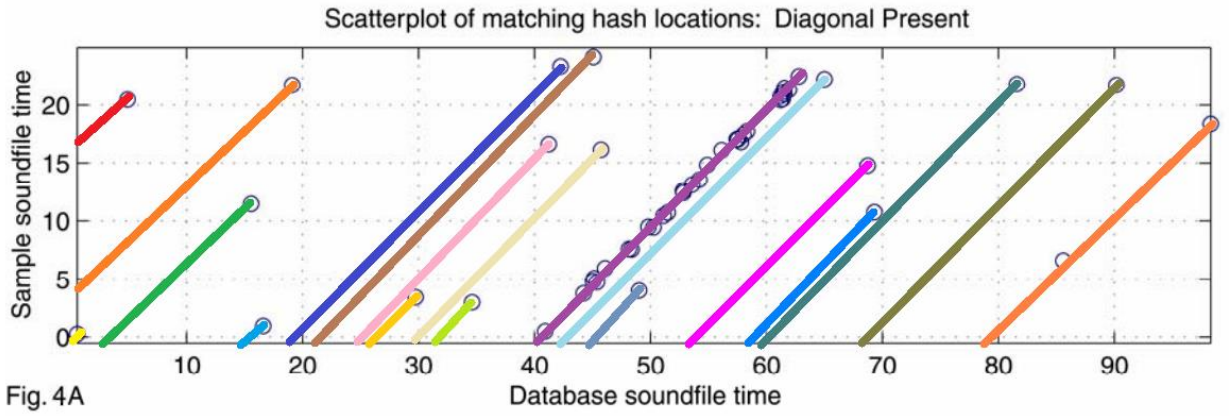


Fig. 4A

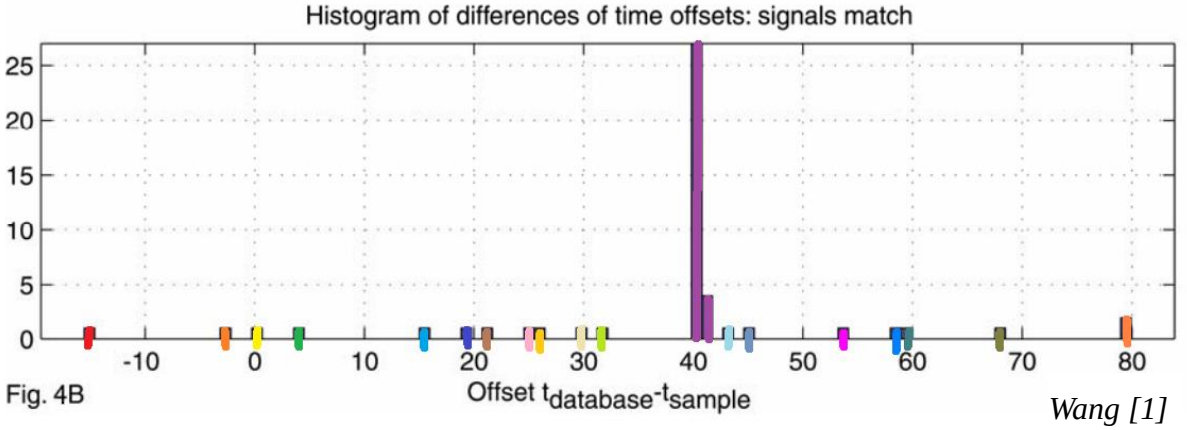


Fig. 4B

Wang [1]

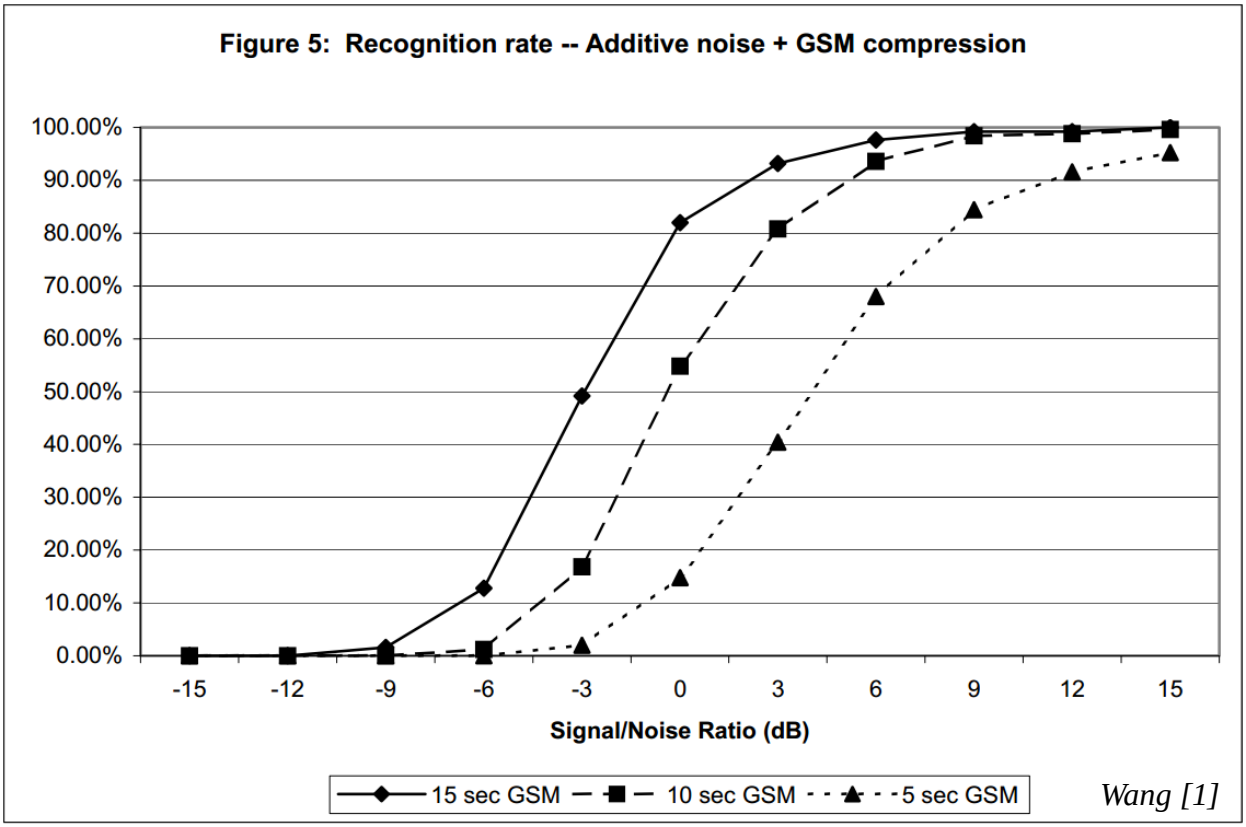
Med andra ord kan man säga att figur 2A och 2B samt 3A och 3B visar hur många träffar på rad som hittats. Finns det många träffar, men de är utspridda över hela diagrammet som 2:ans figurer visar, kan det inte vara rätt låt som man sökt efter. 3:ans figurer visar däremot flera matchande träffar på rad och rätt låt har då högst troligen blivit hittad. Histogrammens staplar är det som Shazams algoritm använder sig av för att poängsätta hur bra ett resultat är. Finns det ett kluster av staplar, vilket är samma som många träffar på rad, blir resultat högt poängsatt och är då troligen rätt låt. [2]

3.6 Precision

Wang [2] hävdar att då antalet träffar i varje låda är så få tar scanningsprocessen bara någon mikrosekund per låda, även om det inte skulle röra sig om ljudklipp som man undersöker. Om algoritmen hittar vad den tror är den rätta låten, kommer den ändå att fortsätta sökandet för att vidare säkra sig om resultatet. Wang säger att om flera låtar har sammansatts till en kan algoritmen ändå urskilja varje låt enskilt. Om algoritmen däremot misslyckas kommer den inte ge resultat om någon låt som bäst passade in på sökningen. Man kan alltså säga att algoritmen följer en allt-eller-inget princip.

Beroende på hur stor databasen är som man jämför emot kommer toleransen för fel att variera. Toleransen kan också variera i andra ändamål än ljudidentifikation för låtar. Wang säger också att även om endast 1-2% av strukturerna skulle gå att reproducera från ett inmatat låtklipp skulle algoritmen ändå i bästa fall kunna ge resultat.

I figur 5 ser man att igenkänningsnivån stiger ju längre låtklipp som matas in eller då störningen minskar. X-axeln visar störningsnivån medan y-axelns visar sannolikheten att algoritmen ska lyckas vid igenkänningen. Signalen är här bl.a. utsatt för störningar från en pub-miljö, dvs. högljutt tal, ljudet från glas osv. och GSM-kompression.



Wang [1]

3.6 Sammanfattning av Shazams ljudsökningsalgoritm

Shazams ljudsökningsalgoritm har som tidigare sagt stått som grund får många liknande algoritmer, troligen p.g.a. dess relativt lätta uppbyggnad och funktion, men också dess egenskaper så som snabbhet och precision. Algoritmen är dock inte felfri och många utvecklingar och förbättringar har gjorts under åren sen Wang lanserade sin text [2]. Brister som algoritmen har är t.ex. förmågan att känna igen live-versioner av en låt och tolka låtar som är utsatta för tidsstretchning eller transponering (när en låt spelas i en annan tonart). Snabbheten och förmågan att söka bland en ännu större mängd data har också förbättras. Vissa av de här utvecklings- och förbättringsalternativen kommer att tas upp i den här avhandlingens följande kapitel.

4 Förbättringsförslag till Shazams ljudsökningsalgoritm

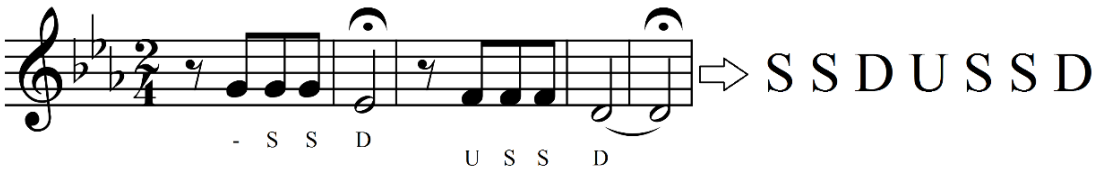
5 Sökning genom att nynna

Ett för människor mest naturliga sätten att framföra en låt i sin enkelhet är att nynna den. Att nynna är i princip en förenklad form av att sjunga som vem som helst kan göra; ingen text används och precisionen är inte lika stor. Många appar idag som t.ex. SoundHound erbjuder ljudsökning genom att nynna endast nynna en låt. Liksom Shazams ljudsökningsalgoritm har många förbättringar gjorts under åren, men i den här avhandlingen tas endast den fundamentala idén upp.

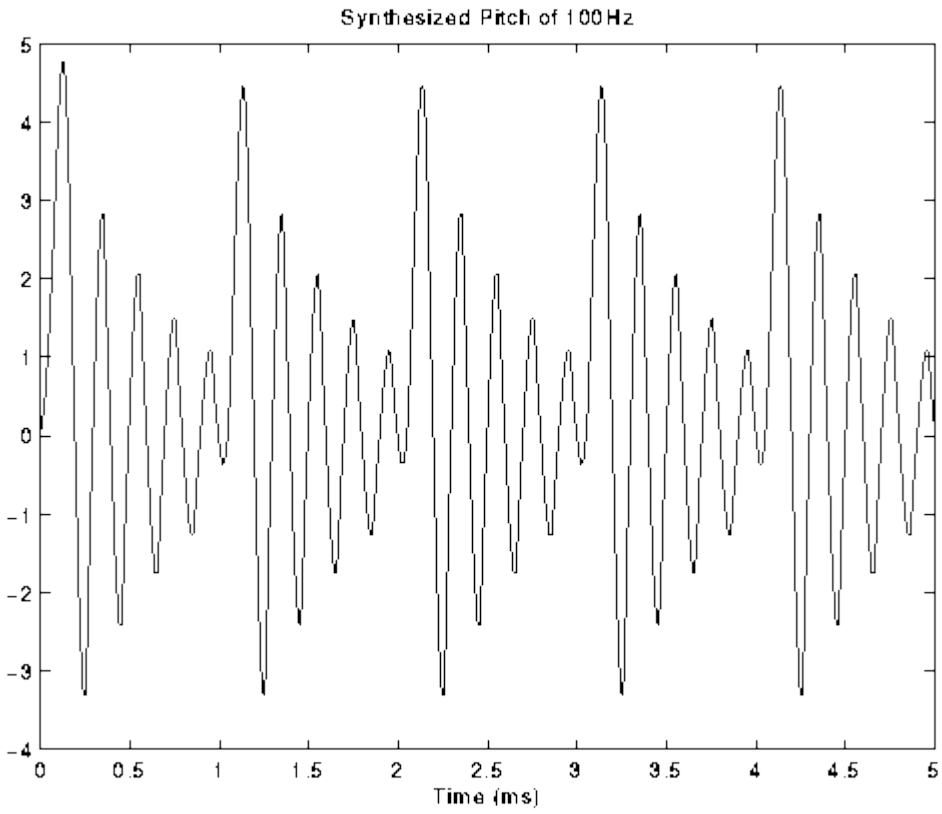
5.1 Grundläggande idén

Ändringar i tonhöjd och rytm är de två grundläggande sakerna som skiljer två melodier från varandra. Även om rytmen är det starkaste igenkänningstecknet för en melodi [6] så lämpar det sig bättre att använda tonhöjden när det gäller igenkänning av en låt via att nynna. Orsaken är att människor nynnande ofta följer tonhöjdens kontur rätt så bra, medan rytmen bara är ungefärlig. Med tonhöjdens kontur menas att tonhöjdens exakta frekvens inte tas i beaktande, utan bara om en melodis toner går uppåt, nedåt eller hålls lika.

Den mest förenklade versionen av det här systemet är att se på varje meloditon i relation till föregående meloditon, och gruppera hopp uppåt i en melodi som U (från eng. *up*), nedåt som D (från eng. *down*) och upprepning av samma ton som S (från eng. *same*). Således skulle början av Beethovens kända femte symfoni kunna ritas upp enligt figuren nedan. Notera att första tonen inte har något värde, då man alltid jämför med föregående ton, vilket den första tonen då uppenbart saknar.



För att bestämma vilken tonerna är när man nynnar används formanter på liknande sätt som i Shazams ljudsökningsalgoritm. Formanter är som tidigare sagts toppar i en ljudkurva, ställen där energinivån är högst. En enskild ton om 100Hz kan då konverteras till en ljudkurva med tiden i x-axelns och amplituden i y-axlen.



En melodislinga som man nynnar kommer då att se ut på liknande sätt med varierande frekvens och amplitud. Nästa steg i processen är att jämföra tonerna parvis och konvertera ljudkurvan till en bokstavssträng av U, D och S, på samma sätt som bilden av Beethovens femte symfoni visade. Hur stor frekvensskillnaden ska vara för att det ska räknas som ett hopp upp eller ner (U eller D) eller om det är samma ton (S) är beroende på frekvensskillnaderna mellan olika toner i den västerländska 12-tonsskalan. I [7] nämns att om en ton är inom en fjärdedels tonsteg från föregående ton, räknas det som samma ton och får då alltså bokstavsbeteckningen S.

För att skapa en databas att kunna jämföra emot kan man t.ex. använda sig av och konvertera MIDI-filer. MIDI-filer lämpar sig bra för detta ändamål då de är digitala signaler som man lätt kan konvertera till bokstavssträngar att jämföra med. Vid jämförelse och sökning måste man ta i beaktande de vanligaste felen som människor gör när de nynnar, bl.a. bortlämnande av vissa toner eller upprepningsfel, alltså där någon ton upprepas för många gånger.

Många olika sökalgoritmer kan användas för att hitta rätt låt det egentligen bara handlar om att matcha strängar med varandra. I [7] nämns att deras algoritm kan ta i beaktande ett visst antal fel och kan hitta rätt låt, även om det man nynnat är från mitten

eller slutet av en låt. Mot en databas med 200st bokstavssträngar gjorda av låtar klarar algoritmen att med 10-12 nynnande toner hitta rätt låt med en precision om ca 90 %.

6 Sammanfattning

Ljudsökningsalgoritmer har förbättras mycket under de senaste 20 åren och har idag också många andra ändamål. Även om de endast uppfyller ett krav av att få information på ett behändigt sätt, så har utveckling av metoderna som gör det möjligt hjälpt med snabbhet, precision och robusthet vid sökning av annan information. Att få veta vilken låt som spelas på radion kan verka som ganska obefintligt men konceptet om omvandlingen av komplex data till lätthanterligt data hjälper också i områden där data som hanteras kan ses som mera viktigt och relevant.

References

[1] ”SoundHound Hound,” SoundHound, 5 4 2016. [Online]. Available: http://www.soundhound.com/hound. [Använd 5 4 2016].

[2] A. L.-C. Wang, ”An Industrial-Strength Audio Search Algorithm,” Shazam Entertainment, Ltd., London, 2003.

[3] M. Sharifi, A. Oztaskent och Y. Volovich, ”Detection of creative works on broadcast media”. USA Patent US 8,433,577 B2, 30 April 2013.

[4] ”Walden Venture Capital - Sprout Stage Investors,” 4 Augusti 2009. [Online]. Available: http://wvcblog.blogspot.fi/search/label/sound2sound%20search%20science. [Använd 1 Mars 2016].

[5] B. Jacobs, ”Gizmodo,” 24 September 2010. [Online]. Available: http://gizmodo.com/5647458/how-shazam-works-to-identify-nearly-every-song-you-throw-at-it. [Använd 1 Mars 2016].

[6] S. Henderson, Kompositör, *Melodic Phrasing.* [Ljudupptagning]. Alfred Pub Co. 1992.

[7] A. Ghias, J. Logan, D. Chamberlin och B. C. Smith, ”Query By Humming -- Musical Information Retrieval in an Audio Database,” ACM Multimedia 95 - Electronic Proceedings, San Francisco, California, 1995.