TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Ludvig Leis

Diginootide meloodiaanalüsaator

Bakalaureusetöö (9 EAP)

Juhendaja: Sven Aller

Tartu 2020

Diginootide meloodiaanalüsaator

Lühikokkuvõte:

Bakalaureusetöö eesmärk oli luua tarkvaralahendus digitaalsete nootide meloodiate analüüsimiseks. Töös antakse lühiülevaade noodikirja ja helisalvestise analüüsi erinevustest. Samuti tutvustatakse diginoodi enim levinud failitüüpe ning loodud programme noodistuste analüüsimiseks. Töös tutvustatakse põhjalikult valminud tarkvaralahenduse sisu. Programmi arendusetapid kirjeldatakse detailselt lahti ja antakse ülevaade testimise tulemustest. Lõpuks pakutakse välja programmi edasiarendusvõimalused.

Võtmesõnad:

Muusika, meloodiajärgnevus, diginoot, MusicXML

CERCS: P170 Arvutiteadus, arvanalüüs, süsteemid, kontroll

Melody analyser for digital sheet music

Abstract:

The purpose of this Bachelor’s thesis was to create a melody analyser software for digital sheet music. The differences between audio and notation analysing are briefly described. Also, this thesis gives an overview of main file types for digital sheet music and works that have been done for analysing sheet music. The content of the software is thoroughly described. The developing process is shown in detail and the testing results are discussed. Finally, further development opportunities are described.

Keywords:

Music, melody sequence, digital sheet music, MusicXML

CERCS: P170 Computer science, numerical analysis, systems, control

Sisukord

Sissejuhatus

Muusikateooria ja -analüüs on muusikateaduse valdkond, mis uurib, millest muusika koosneb ning millised on muusika väljendusvahendite omavahelised seosed [1]. Muusikaanalüüsil keskendutakse peamiselt muusikalise teksti uurimisele, milleks on kas helisalvestis või noodikiri [2]. Helisalvestise puhul toimub analüüs helilainete põhjal. Selle alla kuuluvad muusika esitused, mis on kuulatavad mis tahes tehnilise vahendi või protsessi vahendusel (näiteks raadio, plaadimängija, televisioon, arvuti, internet) [3]. Noodikiri ehk notatsioon on muusika üleskirjutamiseks kasutatav märgisüsteem [4]. Noodid on tavaliselt esitatud, kas paberil või digitaalsel kujul.

Tänapäeval on võimalik arvutite abiga muusikaanalüüsi automatiseerida nii helisalvestiste kui ka noodikirja puhul. Tänu sellele saab muusika olemust, arengut ning mõju inimestele uurida efektiivsemalt ning suuremas mahus kui käsitsi analüüsimisel. Selles töös keskendutakse muusika notatsiooni uurimisele, sest see vorm sisaldab detailsemat informatsiooni muusika struktuuri elementide kohta kui helisalvestis.

Muusikateoreetilisel analüüsil on võimalik uurida mitmeid erinevaid muusika väljendusvahendeid üksikasjalikult. Üks tähtsaimaid muusika komponente on meloodia. Meloodia on kõige väljapaistvam ja meeldejäävam osa muusikast, mille järgi kuulaja eristab enamasti ühte teost teisest, mistõttu saab selle kaudu hinnata ka nende omavahelist sarnasust.

Selle bakalaureusetöö eesmärgiks on luua tarkvaralahendus, millega saab analüüsida digitaalsete nootide meloodiat. Programmi abil peaks kasutajal olema võimalus saada informatsiooni digitaalse noodi elementide kohta ning leida noodistuste vahelisi sarnaseid meloodiajärgnevusi. Samuti peaks olema võimalik programmi abil anda hinnang, kas teoste meloodiad on omavahel sarnased või mitte.

Bakalaureusetöö koosneb kuuest osast. Esimeses osas kirjeldatakse lahti tööga seotud mõisted ja terminid. Teises osas antakse ülevaade helisalvestiste ja noodikirja analüüsi peamistest erinevustest muusikateoreetilisel uurimisel. Kolmandas osas tutvustatakse levinumaid digitaalsete nootide failivorme. Neljandas osas kirjeldatakse, mida on digitaalsete nootide abil analüüsitud ning antakse ülevaade olemasolevatest diginoodi analüüsimise rakendustest. Viiendas osas keskendutakse selleks bakalaureusetööks valminud programmi tutvustamisele. Seal kirjeldatakse, mida kasutati tarkvara arendamisel, millised on kasutusvõimalused ning

kuidas toimivad programmi põhifunktsioonide algoritmid. Samuti antakse ülevaade testide tulemustest ning kirjeldatakse, millised võiksid olla programmi võimalikud edasiarendused.

1. Mõisted ja terminid

Meloodia ehk viis on ühehäälselt väljendatud muusikaline mõte [5]. See on põhiline muusikaline väljendusvahend, sest meloodia moodustavad muusika „juhtiva" viisi ehk liini, mis tuleb muusikat kuulates enim esile. Meloodia moodustavad noodid, mis omavad kindlat helikõrgust ja rütmi ning on omavahel järjestikku [6].

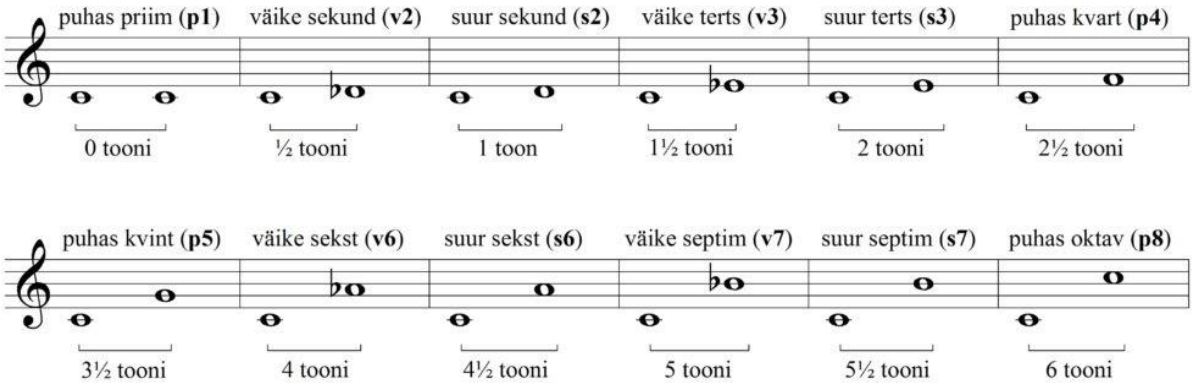
Harmoonia on helide kooskõla ehk kokkusobivus [5].

Rütmi moodustab muusikas helivältuste järgnevus. Helivältus on muusikas heli või vaikuse ajaline kestus [5].

Noot on muusikalist heli tähistav sümbol [7].

Heli kõrgendamist või madaldamist nimetatakse **altereerimiseks**. Heli kõrgendatakse poole tooni võrra dieesiga ♯, madaldatakse bemolliga ♭ ning altereerimise tühistab bekarr ♮ [7].

Nootide omavahelisi kaugusi nimetatakse **intervallideks** (joonis 1). Kõrvuti olevate nootide puhul on tegemist meloodilise intervalliga, üksteise all asetsevate nootide kaugust nimetatakse harmooniliseks intervalliks [7].



Joonis 1. Intervallid noodijoonestikul.

Akord on tertsilise ülesehitusega kolme või enama heli üheaegne kõlamine [7].

Taktimõõt koosneb kahest numbrist, mille ülemine number näitab löökide arvu taktis ja alumine number ühe löögi pikkust. **Taktiks** nimetatakse meloodia osa ühest rõhulisest helist teiseni [5].

Helilaad on helikõrguste süsteem, mis tugineb kindlale alusastmele ehk põhitoonile ning mille astmete vahel moodustuvad funktsionaalsed seosed [5, 7].

Helistik on kindlalt helilt üles ehitatud mažoorne või minoorne laad [7]. Helistik saab oma nimetuse toonika ehk põhitooni esimese astme järgi. Mažoorne laad on rõõmsakõlaline (esimese ja kolmanda astme vahel suur terts) ja minoorne laad on nukrakõlaline (esimese ja kolmanda astme vahel väike terts) [6].

Tämber ehk kõlavärv, mis sõltub heliallika materjalist, heli tekitamise ja võimendamise viisist, leviruumist ja kuuldeorgani eripärast [5].

Muusika väljendusvahendid on suhtlemisvahendid helilooja, esitaja ja kuulaja vahel, mille abil mõistame muusika sõnumit. Selle alla kuuluvad meloodia, harmoonia, rütm, meetrum, tempo, dünaamika, faktuur, laad, register ja tämber [5].

Noodistus on noodistamise tulemus ehk teose noot [8].

Enharmonism on erinimeliste helide samakõlalisus tempereeritud helisüsteemis [8].

2. Muusikateoreetilise analüüsi uurimisvormid

Selleks, et muusika ülesehitust mõista ja analüüsida, on tarvis teada muusikateooria põhialuseid. Muusikateoreetilises analüüsis keskendutakse peamiselt väljendusvahendite uurimisele. Väljendusvahendid on muusika struktuuri osad ning iga osa läbitöötamisel on võimalik avastada üha detailsemaid seoseid muusikas. Nii helisalvesti kui noodikirja kaudu saab muusikat analüüsida, kuid nende kahe vormi puhul on analüüsitav sisu erinev.

2.1 Analüüs helisalvestise põhjal

Helisalvestise puhul on tegemist helilaine analüüsimisega. Helisalvestise analüüsil on võimalik hinnata muusika tämbrit, dünaamilisust ja rütmi, tuvastada mängitavad instrumendid ning kirjeldada muusika emotsiooni. Samuti on võimalik kuulamise teel kindlaks teha näiteks helistikku, taktimõõtu ja määrata meloodianootide järgnevusi ning ka teisi muusikateoreetilisi struktuure, kuid detailsete muusikaelementide määramisel ei saa alati kindel olla, sest helisalvestis ei anna otseselt infot nende kohta. Helisalvestisi saab uurida ka erinevate programmide abil. Digitaalse audio analüüsil uuritakse helilaineid [9]. Lainekuju, ulatuse ja helisageduste kaudu on võimalik hinnata heli dünaamikat, kvaliteeti ning tuvastada muusika elemente nagu näiteks muusika tämbrit, mängivaid instrumente või korduvaid struktuurivorme. Muusika helilaineid on üritatud ka noodistada tarkvara abil, et saada sel viisil kätte informatsiooni nootide kohta. Üks tasuline programm selleks on AnthemScore[1], kuid kuna helisalvestise helilaine koosneb mitmete erinevate heliallikate helilainetest, siis noodistamine on ebatäpne [10] ning alati on tarvis noodistus käsitsi üle kontrollida ning vastavad parandused teha. Muusika struktuuri üksikasjalike elemente saab lihtsamalt ja täpsemini uurida noodikirja põhjal.

2.2 Analüüs noodikirja põhjal

Noodikirja puhul analüüsitakse muusikat kui teksti. Tänu märgisüsteemi reeglitele ning laias valikus sümbolitele on võimalik noodistada mis tahes muusikat. Kõikide partiide puhul on võimalik üksikasjalikult tuvastada kõik noodid, intervallid, akordid ja rütmijärgnevused ning nendega seotud lisad (näiteks rütmiaktsendid, dünaamikamärgid ning noodi altereeringud). Selle põhjal saab täpsemalt analüüsida muusika väljendusvahendite seosed. Noodikirja analüüsi miinused on seotud heli puudumisega ehk muusika kõla pole võimalik noodistuse kaudu uurida.

[1] https://www.lunaverus.com/

Kõik heliloojad ei noodista oma loomingut, mistõttu levivad paljude muusikastiilide teosed peamiselt helisalvestistena, mitte noodikirjana. Küll aga on olemas veebilehed (näiteks MuseScore[2]), kust on võimalik tasuta alla laadida teiste inimeste poolt loodud erinevaid muusika noodistusi nii kaasaegse muusika kui ka näiteks klassikalise muusika kohta.

Sarnaselt helisalvestisele on ka noodikirja võimalik analüüsida tarkvara abil. Järgmistes peatükkides seletatakse lahti, millised diginoodi failiformaadid on olemas ning mida ja kuidas nende abil analüüsitakse.

[2] https://musescore.com/

3. Digitaalne noodikiri

Traditsiooniliselt on noodistusi esitatud paberkandjal, kuid notatsiooniprogrammide tekke tõttu on noodikirja käsitsi kirjutamise asemel esile kerkinud ka noodistuste digitaalne esitus [11]. Kasutusel on mitmeid erinevaid digitaalse noodi failiformaate.

3.1 Notatsiooniprogrammide failiformaadid

Tänapäeval on peamised kasutusel olevad noodikirjutamisprogrammid (sulgudes on programmi failiformaadi laiend) MuseScore[3] (.mscz), Sibelius[4] (.sib) ja Finale[5] (.musx). Igal programmil on olemas neile omane failitüüp. Nende failiformaatide plussiks on see, et neid on lihtne vastavates programmides töödelda ning nendes säilib kogu info kirjutatud noodi kohta ja ka noodi elementide graafiline paigutus. Miinuseks on see, et ühe programmi failiformaate ei saa otse lugeda ega töödelda teises notatsiooniprogrammis (selleks tuleb faile teisendada) ning eelnimetatutest on Sibelius ja Finale tasulised [12].

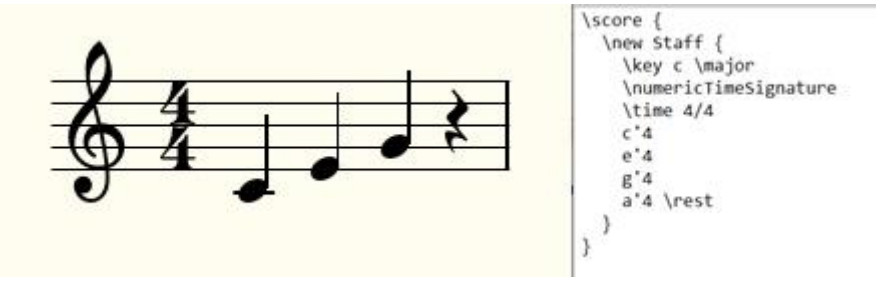
Lisaks nendele failiformaatidele on olemas ka teisi digitaalsete nootide formaate. Ühed neist on tekstil põhinevad failitüübid ehk neid on võimalik avada ning töödelda ka tekstiredaktorites.

3.2 Tekstipõhised notatsiooni failitüübid

3.2.1 LY-fail (Lilypond)

LY-fail on noodikirjutamise programmi Lilypond failitüüp noodistuste loomiseks. LY-faili struktuur on lihtsasti loetav ja töödeldav tänu elementide tasandi märgistusele ning ülesehituse kompaktsusele (joonis 2). Tänapäeval pole LY-failid enam levinud, sest enim kasutatud noodikirjutamisprogrammid ei suuda neid faile avada ega väljastada. Samuti on Lilypondi programm aegunud ja näiteks operatsioonisüsteemidele nagu Windows 10 on tarkvara toetus lõpetatud [13].

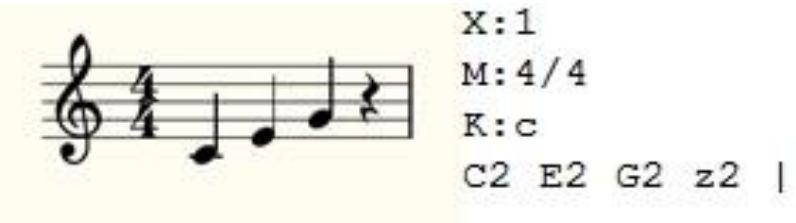
[3] https://musescore.org/
[4] https://www.avid.com/sibelius
[5] https://www.finalemusic.com/



Joonis 2. Noodikiri LY-failis.

3.2.2 ABC

ABC notatsiooni failitüüp (faililaiendiga .abc ja .abh) on mõeldud peamiselt folkloor- ja pärimusliku muusika noodikirja esitamiseks. Rahvamuusika ei ole reeglina keerulise ülesehitusega, mistõttu on ABC-fail oma vormingult üsnagi minimalistlik (joonis 3) [14]. Mitmekülgse noodistuse puhul (erinevad partiid, mitmehäälsus, ebatavalised meloodia- ja rütmikäigud) on ABC-faili lugemine keerulisem, kuna kompaktse ülesehituse tõttu pole täielikult arusaadav, kus mingi noodistuse üksikelement algab või lõpeb.



Joonis 3. Noodikiri ABC-failis.

3.3 Diginoodi universaalsed failiformaadid

3.3.1 NIFF

NIFF (Notation Interchange File Format) oli üks esimesi failiformaate, mis loodi 1994. aastal erinevate notatsiooniprogrammide vahel diginootide vahetamiseks [15]. NIFF-fail oli hea formaat noodistuse skaneerimise programmidele, sest noodi tuvastus käis selle kaudu, kus noot asus noodijoonestikul, mitte noodinime järgi [16]. See oleks pidanud saama digitaalsete noodifailide standardiks, kuid takistuseks oli NIFF-faili binaarne formaat. MusicXML-faili tulekuga ei pakutud NIFF failitüübile noodikirjutamisprogrammide poolt tuge [17] ning NIFF ei leidnud enam kasutust, sest tekstiformaadiga failitüüpi on palju kergem lugeda ja töödelda kui binaarset failitüüpi [16].

3.3.2 MusicXML

MusicXML on 2000. aastal loodud XML failitüübil põhinev formaat digitaalsete nootide esitamiseks [18, 19]. MusicXML sündis eesmärgiga luua universaalne failiformaat digitaalsete nootide vahetamiseks, lugemiseks ja kirjutamiseks erinevate noodikirjutamisprogrammide vahel ning samuti nootide digitaalseks jagamiseks ja arhiveerimiseks [20].

MusicXML sisaldab kõikvõimalikku informatsiooni noodistuse ülesehituslikest elementidest (struktuur, faktuur, taktid) kuni üksikute nootide detailideni (altereerimismärgid, rõhud, rütmisümbolid). MusicXML-fail on vormindatud sarnaselt XML-failile (Lisa I) ning neid on kahte tüüpi - pakitud (*compressed*) MusicXML-fail laiendiga .mxl ja pakkimata (*uncompressed*) fail laiendiga .musicxml [29]. See on rangelt struktuuritud tekstidokument, tänu millele on selle failitüübi kirjutamine ja lugemine toetatud nii tekstitöötlus- kui ka notatsiooniprogrammides [20]. Tänaseks on MusicXML-fail enim tuntud ja kasutatav failiformaat digitaalsete nootide töötlemiseks ning jagamiseks [15]. Eelnimetatud omadusete põhjal on MusicXML võrreldes teiste failitüüpidega sobivaim formaat diginootide töötlemiseks. Nii järgnevates tarkvara näidetes kui ka selle bakalaureusetöö raames valminud programmis on kasutatud seda failiformaati noodistuste analüüsimiseks.

4. Noodikirja analüüsimine tarkvara abil

Noodikirja on tänu tehnoloogia arengule võimalik analüüsida ka arvutitehnoloogia abil. Arvutite kasutamine võimaldab noodistusi töödelda efektiivsemalt, suuremas mahus ning avastada seoseid, mis käsitsi analüüsides võivad märkamata jääda.

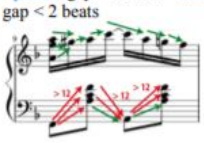
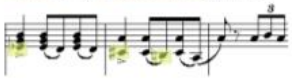
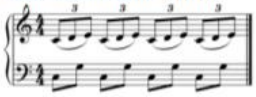
4.1 Muusika sarnasuste tuvastamine teisenduskauguse abil

Muusikapalade sarnasust on üritatud määrata teisenduskauguse abil. Paljud katsed muusika sarnasuste leidmiseks põhinevad Levensteini kauguse algoritmil, mis mõõdab kahe sõne sarnasust. Kaugus arvutatakse selle põhjal, kui palju lisamisi, kustutamisi või asendusi tuleb teha, et saada ühest sõnest teist [21]. 1990. aastal loodi teoste sarnasuste leidmisel algoritm Marcel Mongeau ja David Sankoffi poolt. Nemad kohandasid Levensteini algoritmi muusikanootidele sobivamaks, lisades juurde tükeldamise (*fragmentation*) ja ühendamise (*consolidation*). Tänu sellele said nad kindlate määratud kaaludega hinnata meloodiate sarnasust nii nootide kõrguse kui ka rütmi põhjal teisenduskauguse abil [22]. Lisaks Mongeau-Sankoffi algoritmile on Helsingi ülikoolis võrreldud meloodia järgnevusi ka intervallide toonilise suuruse järgi, mida tehakse ka selles töös [23]. Eelmainitud võrdlusviisid on olnud aluseks paljudele noodistuste analüüsimisega seotud töödele. Näiteks on teisenduskauguse põhjal üritatud tuvastada muusikaõpilaste seas diktaatides tehtud sagedasemaid vigu ning sellega automatiseerida diktaatide hindamist solfedžo õpetamisel [24].

4.2 MusicXML-failide analüüsimise programmid

4.2.1 Score Analyzer

Score Analyzer oli rakendus, mis tuvastas automaatselt noodistatud teose ettekandmise raskusastme. Noodikirja analüüsimisel kasutati MusicXML-faile, sest tänu oma loogilisele failistruktuurile ning noodielementide märgenduse olemasolule oli kerge kätte saada vajalikku informatsiooni. Noodistusi hinnati seitsme kriteeriumi järgi, lähtudes klaverist (joonis 4): mängimise kiirus, sõrmestus, käe positsioon, polüfoonia, harmoonia, ebaregulaarset rütmid ning teose pikkus. Nootide sõrmestuse määramisel kasutati eraldi algoritme, mis pakkusid võimalusi sõrmede asetsemiseks klahvidel, kuna MusicXML-failides seda informatsiooni ei leidunud. Lõpuks pidi programm väljastama arvulise hinnangu raskusastmele skaalal 1-4, kus 1 oli kõige lihtsam noodistus ning 4 kõige raskem. Kuna rakendus ei ole enam veebis kättesaadav, siis polnud võimalik selle töö raames seda testida [25].

Performance difficulty criterion	Definition	MusicXML implementation	Instruments
Playing speed	The required fingers velocity to play the piece. Depends on the tempo and the shortest significant note value (i.e. a piece presenting a high tempo may contain only long values, and conversely, a piece with a low tempo may contain groups of short notes thus increasing the required fingers agility for the players)	<note><type> elements Tempo attribute in <sound> element	All
Fingering	Fingering: choice of finger and hand position on various instruments. Different notations exist according to the instrument. (ex: in piano: 1 = thumb, 2 = index finger, 3 = middle finger, etc.) Cost functions are used on intervals to extract the general fingering difficulty level See [8][8][9] for more detail.	<fingering> element within each <note> element 	All, requires adaptations in constraints and costs functions (some instruments do not use thumbs)
Hand Displacement	Ratio of hands displacements greater than an octave (12 semitones). Depends on the duration of the interval: if the duration exceeds 2 beats (i.e. 2 quarters in 4/4, 2 eights in 6/8), the displacements is not considered as difficult. The difficulty degree of the displacement evolves with its size (in pitch), its duration and its fingering	Combined <note> elements where <pitch> gap > 12 and <duration> gap < 2 beats 	All, requires adaptations depending on the instrument morphology
Polyphony	Chords ratio (aggregate of musical pitches simultaneously attacked) Polyphonic difficulties may increase with the number of notes played at the same time and their fingerings. Simultaneous voices (in a Fugue for instance) constitute special cases of polyphonic difficulties to treat.	<chord> element 	All (except for monophonic instruments, such as the flute)
Harmony	Ratio of differences from the piece main tonality. Characterized by the amount of accidental alterations.	<alter> and <accidental> elements 	All
Irregular Rhythm	Ratio of irregular polyrhythms (simultaneous sounding of two or more independent rhythms). Example: synchronizing a triplets over duplets	<time-modification> element 	All (except for monophonic instruments)
Length	The number of pages of the score. May also be measured in bars number to avoid dependency to the page layout.	new-page attributes or <measure> elements	All

Joonis 4. Programmi Score Analyzer kriteeriumid noodi raskusastme hindamiseks.

4.2.2 MusicXML Analyzer

MusicXML Analyzer[6] on 2015. aastal loodud veebirakenduse prototüüp, mille eesmärgiks oli analüüsida saksa pärimusmuusika meloodiamustreid [26]. Kuigi veebirakenduse nimetus vihjab, et kasutaja saab sisestada .musicxml või .mxml faililaiendiga faile, siis tegelikult võib üleslaadida digitaalse noodistuse ainult .xml laiendiga diginoote. Rakendusel on kaks peamist eesmärki. Esiteks kuvatakse graafiliselt noodistuse kvantitatiivset infot, kui palju ja milliseid noote, intervalle, akorde on kasutatud ja milline on nende jaotus. Teiseks on kasutajal võimalik luua 2 kuni 12 noodiga meloodiajärgnevusi ning otsida, kas noodistuses leidub täpselt samasuguseid järgnevusi. Meloodiate olemasolul on võimalik tuvastada nende asukoht noodistuses ning neid kuulata. Saadud infot saab salvestada CSV-faili.

[6] http://music-xml-analyzer.herokuapp.com/.

Rakendust testiti (Lisa 2) MuseScore lehelt alla laetud kolme erineva noodistuse ning töö autori enda loodud kolme noodistusega. Pärast faili üleslaadimist kuvati nootide ja intervallide jaotuse tulpdiagrammid ning noodi helistiku, taktimõõdu ja rütmivältuste sektordiagrammid (joonis 6, joonis 7). Tulemused on kuvatud joonisel 5.

Üldise statistika testimine (1)

Noodi nimetus	Total notes (kooskõlasid arvestab ühe noodina)		Total measures		Total rests		Most frequent note		Instruments	
	Programmi tulemus	Tegelik tulemus	Programmi tulemus	Tegelik tulemus	Programmi tulemus	Tegelik tulemus	Programmi tulemus	Tegelik tulemus	Programmi tulemus	Tegelik tulemus
Noot A.xml	15	15	5	5	3	6	Unknown	E(4)	Piano	Piano
Noot B.xml	3	7	4	4	9	9	Unknown	G(4)	Piano	Piano
Noot C.xml	8	9	2	2	1	1	G	C(4)	Viiul	Viiul
Für Elise.xml	704	raske hinnata	106	106	200	raske hinnata	E	raske hinnata	Piano	Piano
Alan Walker. xml	788	raske hinnata	124	124	342	raske hinnata	Unknown	raske hinnata	Klavier	Klavier
Sadness and Sorrow.xml	257	raske hinnata	56 (siin arvestas mõlema partiid)	28	5	5	G	raske hinnata	Flute, Trombone	Flute, Trombone

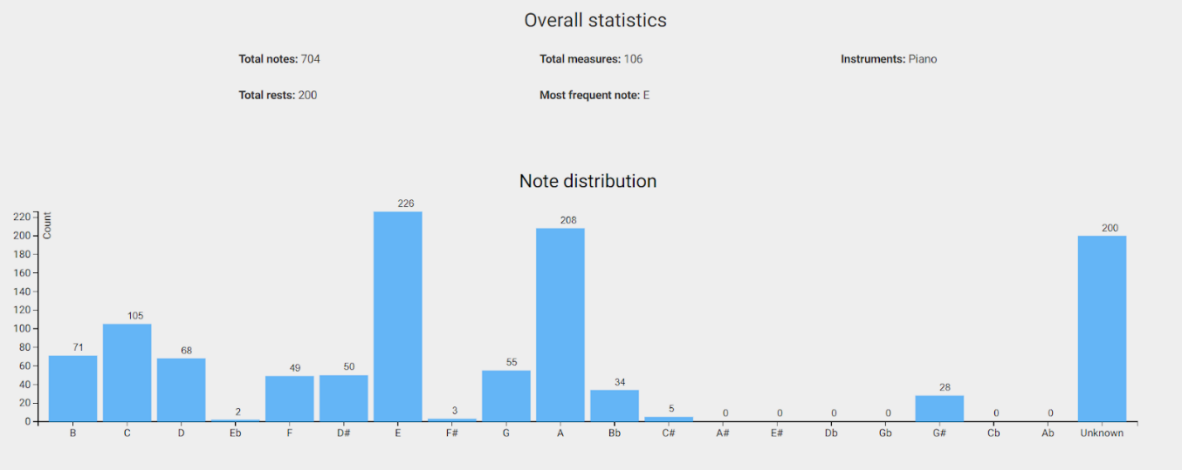
Üldise statistika testimine (2)

Noodi nimetus	Nootide tuvastus	Intervallide tuvastus	Helistik		Noodi pikkuste jaotus	Taktimõõt	
			Programmi tulemus	Tegelik tulemus		Programmi tulemus	Tegelik tulemus
Noot A.xml	Tuvastatud nootide koguarv suurem kui meloodianootide arv, esines tuvastamata elemente, jaotus vale	Tuvastatud intervallide koguarv väiksem, kui olema peaks, esines tuvastamata elemente, jaotus vale	Unknown	E-mažoor/c-minoor	Ebatäpne jaotus	4/4	4/4
Noot B.xml	Tuvastatud nootide koguarv suurem kui meloodianootide arv, esines tuvastamata elemente, jaotus vale	Tuvastatud intervallide koguarv väiksem, kui olema peaks, esines tuvastamata elemente, jaotus vale	Unknown	B-mažoor või g-moll	Ebatäpne jaotus	4/4	4/4
Noot C.xml	Tuvastatud nootide koguarv suurem kui meloodianootide arv, esines tuvastamata elemente, jaotus vale	Tuvastatud intervallide koguarv väiksem, kui olema peaks, esines tuvastamata elemente, jaotus vale	Unknown	C-mažoor/a-minoor	Ebatäpne jaotus	4/4	4/4
Für Elise.xml	Tuvastatud nootide koguarv suurem kui meloodianootide arv, esines tuvastamata elemente, jaotus vale	Tuvastatud intervallide koguarv väiksem, kui olema peaks, esines tuvastamata elemente, jaotus vale	Unknown	C-mažoor/a-minoor	Ebatäpne jaotus	3/8	3/8
Alan Walker. xml	Tuvastatud nootide koguarv suurem kui meloodianootide arv, esines tuvastamata elemente, jaotus vale	Tuvastatud intervallide koguarv väiksem, kui olema peaks, esines tuvastamata elemente, jaotus vale	Unknown	B-mažoor või g-moll	Ebatäpne jaotus	4/4	4/4
Sadness and Sorrow.xml	Tuvastatud nootide koguarv suurem kui meloodianootide arv, esines tuvastamata elemente, jaotus vale	Tuvastatud intervallide koguarv väiksem, kui olema peaks, esines tuvastamata elemente, jaotus vale	G-mažoor (66%) / E-mažoor (33%)	G-mažoor/e-minoor	Ebatäpne jaotus	4/4	4/4

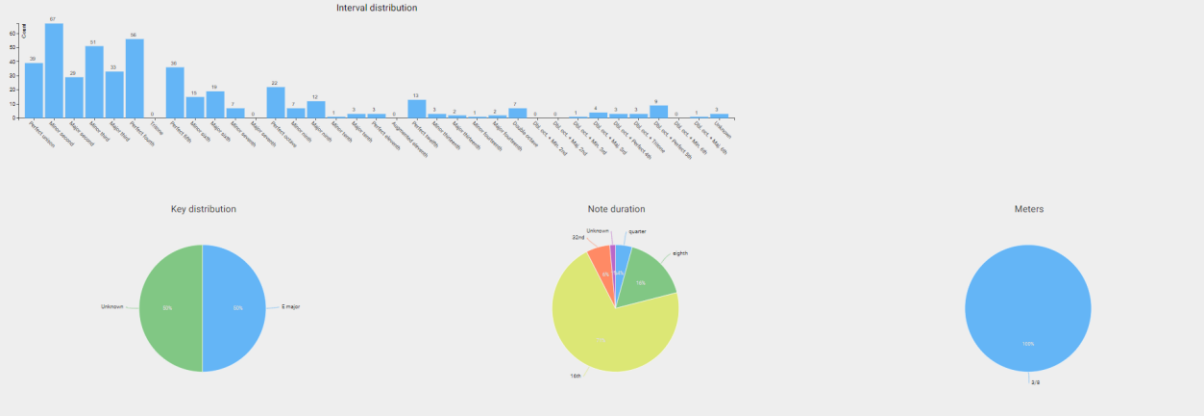
Joonis 5. Üldise statistika testi tulemused

Rakendus sai kõige paremini hakkama taktimõõdu, instrumentide ning noodistuse taktide koguarvu tuvastamisega. Nootide ja pauside koguarv oli rakendusel õigesti märgitud vahelduva eduga. Kuna nende kokku lugemine oli pikkade nootide puhul käsitsi keeruline ja aeganõudev, siis kontrolliti üle vaid lühemate noodistuste koguarvud. Rakendus ei suutnud mitte ühelgi korral õigesti tuvastada kõige sagedasemat nooti ning helistikku. Nootide ja intervallide tuvastus oli ebatäpne.

Lisaks noodianalüüsile avanes ka võimalus kuvada noodistus graafiliselt. Osade noodistuste puhul ei töötanud see funktsioon üldse, kuid oli neid, mille puhul avanes oodatava noodistuse asemele mingi teine analüüsis olev noodistus. Rakendus salvestas CSV-faile korrektselt.

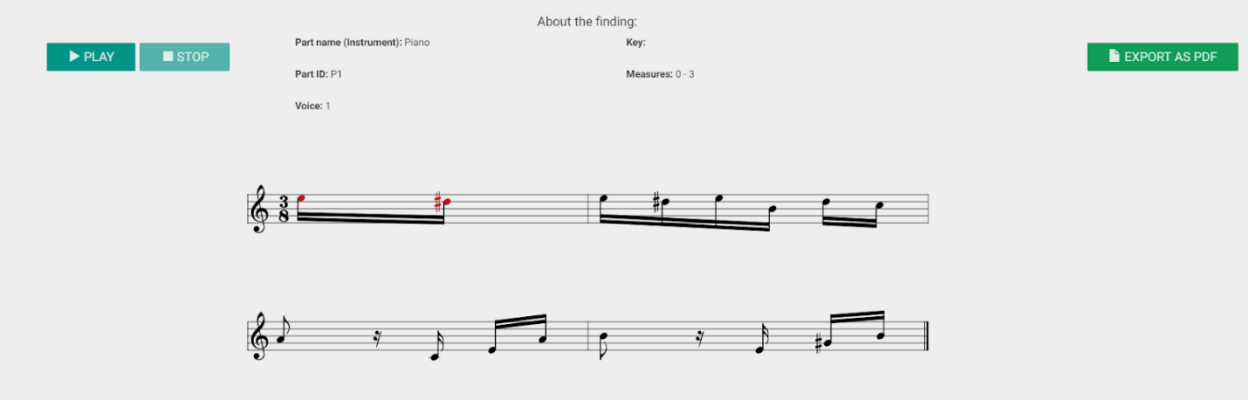


Joonis 6. MusicXML Analyzer veebirakenduse infokuva noodiga Für Elise.xml (1).



Joonis 7. MusicXML Analyzer veebirakenduse infokuva Für Elise.xml (2).

Meloodiamustri tuvastus toimis hästi. Rakendus leidis üles otsitava noodijärgnevuse ning kuvas selle noodijoonestikul ekraanile (joonis 8). Kui kasutaja oli lisanud rakendusse rohkem noodistusi, siis oli võimalik otsida loodud järgnevuse esinemisi üle kõigi mälus olevate noodistuste. Meloodia olemasolul kuvati kõik esinemiskorrad igas noodistuses graafiliselt, mis andis võimaluse tuvastada sarnaseid järgnevusi mitme noodistuse vahel. Meloodiajärgnevusi oli võimalik kuulata ning tulemused sai salvestada PDF-faili.



Joonis 8. MusicXML Analyzer veebirakenduse meloodiajärgnevuse otsingu kuva.

4.3 Kokkuvõte varasemalt tehtud automaatse noodianalüüsi töödest

Digitaalsete nootide analüüsimise puhul ei ole olemas graafilise kasutajaliidesega tarkvara, kus oleks automatiseeritud sarnaste meloodiajärgnevuste tuvastus või Levensteini algoritmil põhinevate teisenduskauguse hinnangute kaudu noodistuste sarnasuse leidmine. MusicXML-faile on analüüsitud ning sellest leitavat infot on spetsiifiliste probleemide lahendamiseks rakendatud, kuid MusicXML-failis on veel suurel hulgal andmeid, mida saab noodistuse info ning meloodia analüüsimisel ja omavahelisel võrdlusel ära kasutada. Inspireerituna sarnaste meloodiajärgnevuste leidmise algoritmidest ning olemasolevate noodianalüüsi tarkvara puudustest valmis programm Diginootide meloodiaanalüsaator.

5. Diginootide meloodiaanalüsaator

Selles peatükis kirjeldatakse bakalaureusetöö raames valminud digitaalsete nootide analüüsi tarkvara (Lisa 2) kasutusvõimalusi ning ülesehitust. Tarkvara eesmärgiks on võrrelda kasutaja poolt valitud kahte noodistust. Esiteks on diginoote võimalik võrrelda kvantitatiivse info järgi nagu näiteks enim kasutatud noodid, akordid, intervallid, teiseks saab leida noodistuste meloodiate pikima sarnase järgnevuse ning kolmandaks püüab programm määrata kahe noodistuse meloodia vahelist sarnasust kauguse tuvastuse algoritmi järgi.

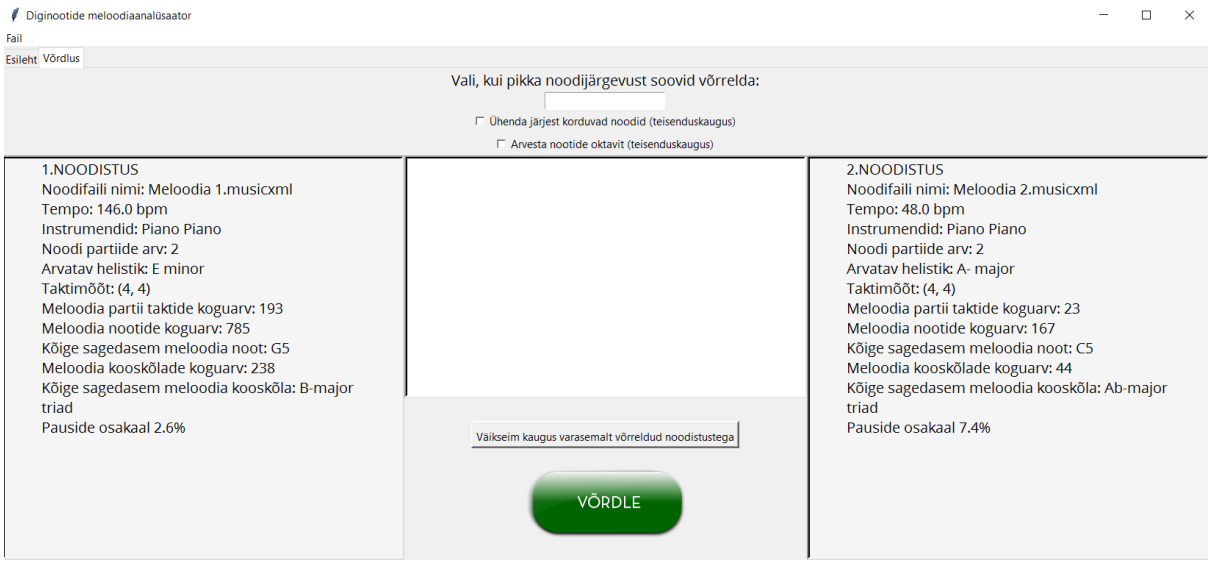
5.1 Kasutatud tehnoloogiad

Tarkvara on kirjutatud programmeerimiskeeles Python. Selle keele kasutamise plussid olid sobivate tekstitöötluse teekide olemasolu ning autori varasem kogemus. Noodikirja töötlemiseks kasutati Pythoni teeki Music21, mis on 2008. aastal loodud arvutipõhise muusikateaduse uurimiseks MIT (Massachusetts Institute of Technology) õppejõudude ja programmeerijate poolt. Music21 abil on võimalik õppida muusikateooriat ja noodikirja läbi programmeerimise, luua muusikat ning analüüsida ja töödelda olemasolevaid MusicXML-faile [27]. Tarkvara kasutajaliides on loodud Pythoni standardvarustusse kuuluva moodul Tkinteri abil [28]. Pythoni teegi Matplotlibi abil integreeriti graafikute visualiseerimine.

5.2 Tarkvara kasutamine

Rakenduse avakuval (joonis 9) on kasutajal võimalik avada mõlemat MusicXML-faili tüüpi. Seejärel ilmub võrdluse vaheleht, kus on kuvatud info esimese noodistuse kohta. Võrdluse läbiviimiseks tuleb üles laadida ka teine noodifail ning pärast seda avanevad vahelehel „Võrdlus“ (joonis 10) võimalused sarnasuste tuvastamiseks.

Joonis 9. Tarkvara avakuva.



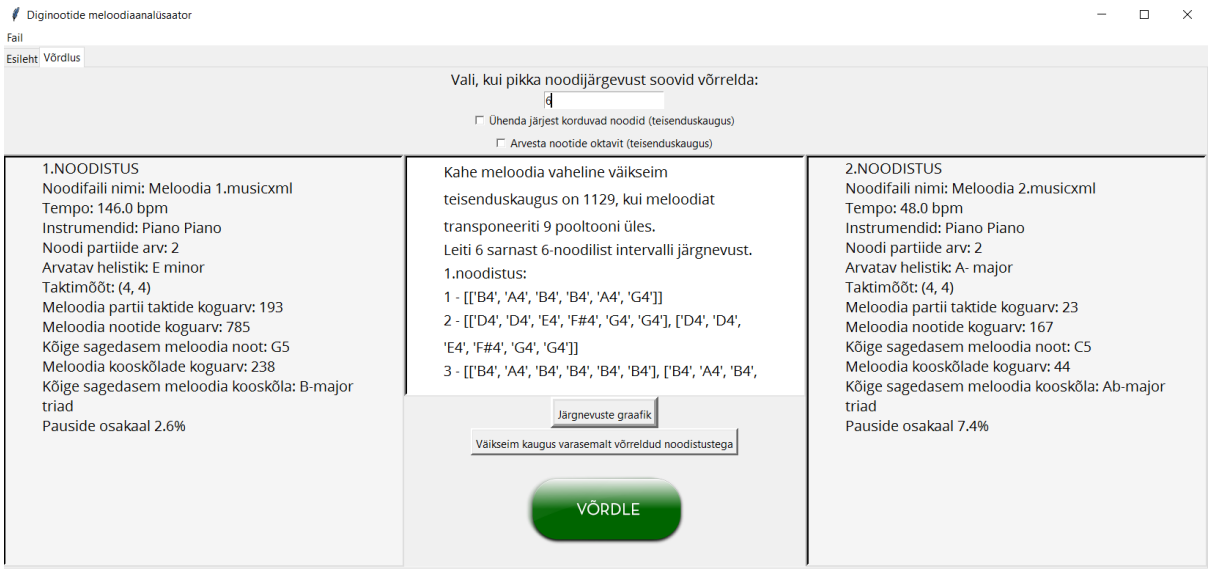
Joonis 10. Vaheleht „Võrdlus“ ja info nootide kohta.

Võrdluse vahelehel (joonis 10) on kuvatud esimese sisestatud noodistuse info vasakul ning teise noodistuse info paremal. Vahelehel on kummagi kohta järgnev info:

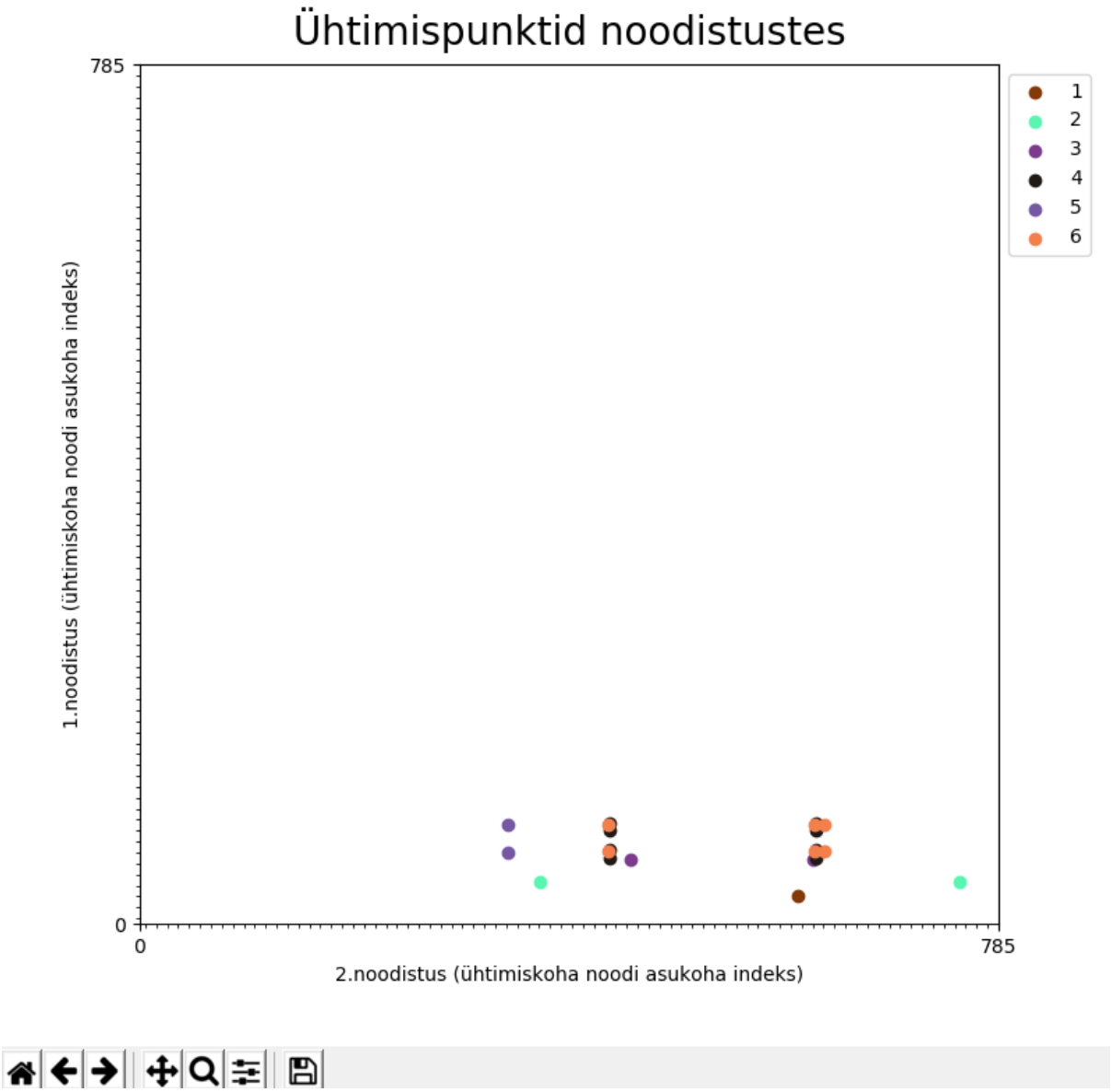
1) Noodifaili nimi.
2) Tempo – BPM ehk *beats per minute* (löökide arv minutis).
3) Instrumendid – instrumentide nimetused, mida on noodistuses kasutatud.
4) Noodistuse partiide arv.
5) Arvatav helistik – näitab, milline on noodi kõige tõenäolisem helistik. Helistikku tuvastatakse Music21 teegi enda meetodi järgi, mis põhineb Krumhansl-Schmuckler helistiku tuvastamise algoritmil [30].
6) Taktimõõt – noodistuse esialgne taktimõõt.
7) Meloodia partii taktide koguarv.
8) Meloodia nootide koguarv.
9) Kõige sagedasem meloodia noot – Kuvas on näidatud suure ladinakeelse tähega noodi nimetus ning numbriga noodi oktaav.
10) Meloodia kooskõlade arv – kahe või enama noodiga kooskõlad.
11) Kõige sagedasem meloodia kooskõla – kooskõla nimetus on inglise keeles vastavalt Music21 teegi andmetele. Kui meloodia on ühehäälne, siis kooskõla puudub.
12) Pauside osakaal – näitab, kui palju pause on kõigi partiide peale kokku. Kõikide partiide taktide kestvusest võetakse kõikide partiide pauside kestvuse summa protsent.

Kasutaja saab sisestada programmi arvu (nootide järgnevuse pikkuse) ning seejärel leitakse kõik sellise pikkusega sarnased meloodialõigud kahe noodistuse vahel. See peaks olema

positiivne täisarv ning mõistliku tulemuse saamiseks on minimaalne soovitatav järgnevuse nootide arv kolm. Kasutajal on võimalus määrata teisenduskauguse arvutamise meetod ning märkida, kas diginootide meloodiates ühendatakse samad järjestikused noodid üheks või mitte (vt ptk 5.3.2). Kui sobiv arv on sisestatud, tuleb vajutada nuppu „Võrdle“. Keskmisesse lahtrisse kuvab programm kahe noodistuse vahelise teisenduskauguse ning sarnased meloodiajärgnevused (joonis 11). Meloodiajärgnevuste paiknemist saab kuvada ka graafikul (nupp „Järgnevuste graafik“), et tuvastada kattuvate järgnevuste asetus noodistuste vahel (joonis 12). Graafiku Y-telg (vertikaalne) näitab esimese noodistuse sarnaste meloodiajärgnevuste alguse indeksit noodistuses ehk mitmendast noodist sarnane meloodia järgnevus algab ning X-teljel (horisontaalne) kuvatakse sama info teise noodistuse kohta. Ristumispunkt on mõlema noodistuse ühise sarnase meloodia järgnevuse alguspunkt. Graafiku iga värv tähistab erinevat meloodiajärgnevust. Jooniselt 12 saab välja lugeda, et meloodia sarnasused on esimese noodistuse alguses ning teise noodistuse teises pooles.

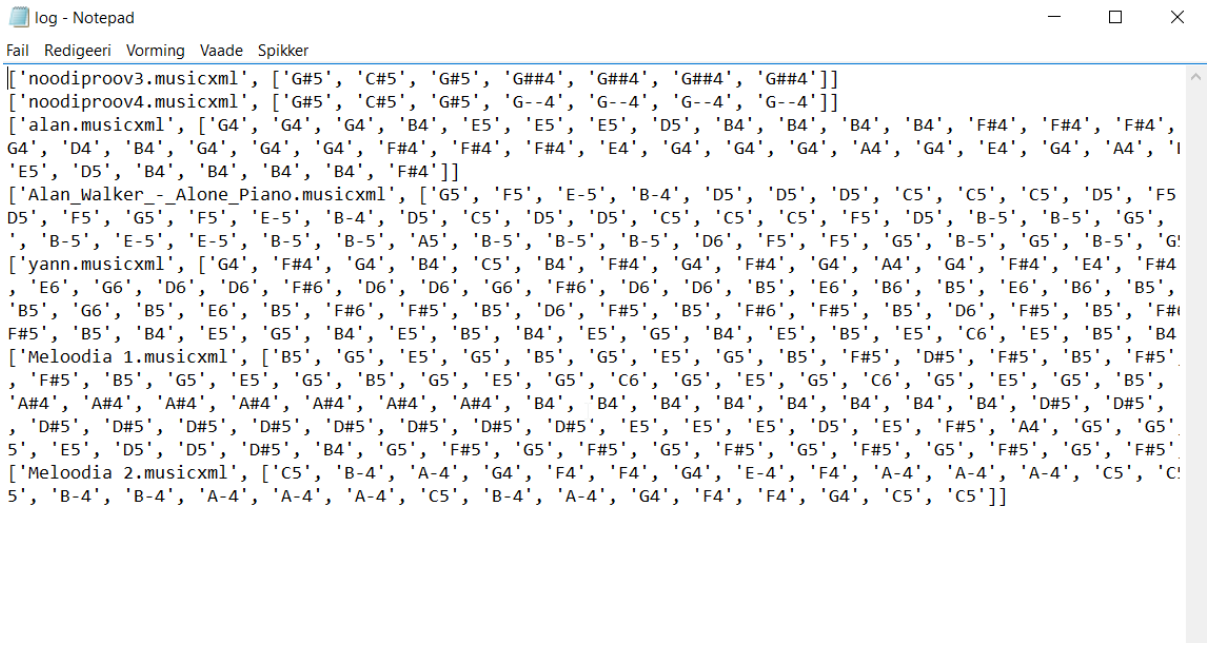


Joonis 11. Võrdluse kuva.

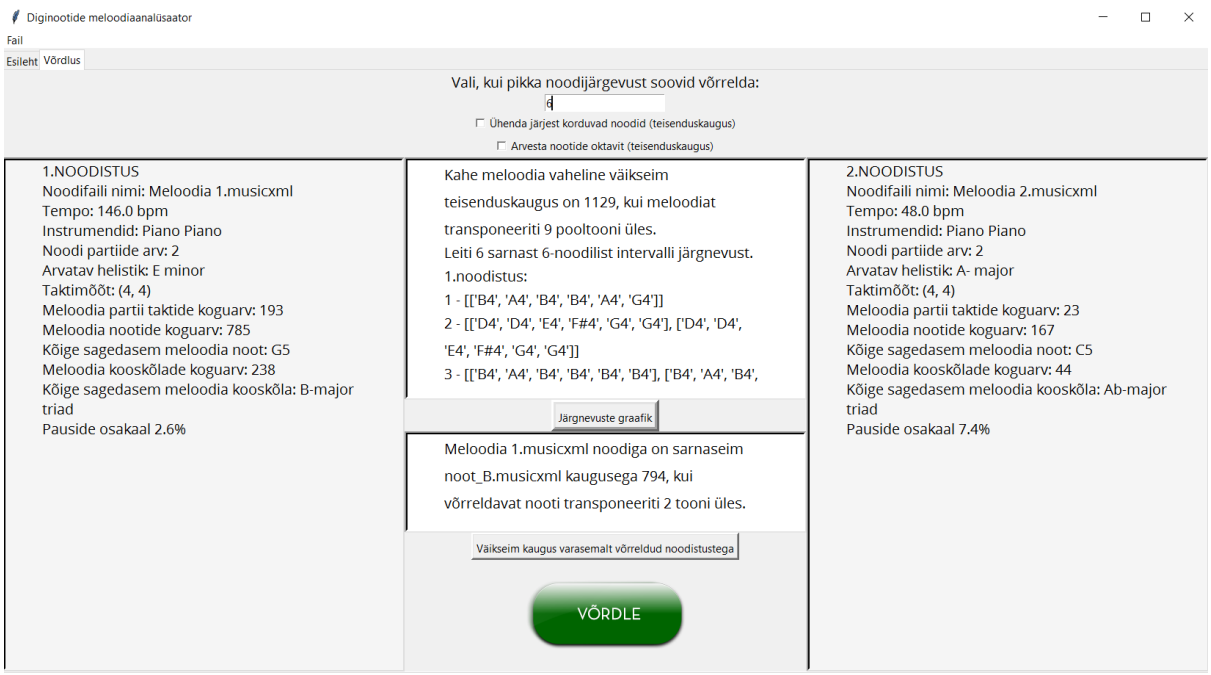


Joonis 12. Sarnaste meloodiajärgnevuste graafik.

Pärast võrdluse läbiviimist salvestatakse noodistuse failinimi ning meloodia noodid lokaalselt programmi kausta „log.txt“ faili (joonis 13). Kui kasutaja vajutab nupule „Võrdle kaugust varasemate noodistustega“, siis ilmub uus lahter, kus on kuvatud esimese sisestatud noodistuse väikseim teisenduskaugus võrreldes varasemalt analüüsitud noodistustega (joonis 14). Järjest korduvate nootide ühendamise ja meetodi valiku võimalus kehtib ka siin. Varasemate noodistustega võrdlus nõuab palju arvutusjõudlust, seega kui „log.txt“ failis on olemas üle 15 erineva kirje, siis antakse kasutajale teada, kas soovitakse faili tühjendada või jätkata kauguste arvutamisega.



Joonis 13. Näide log.txt faili sisust.



Joonis 14. Kauguse võrdlus varasemalt analüüsitud noodistustega

Programmis saadud võrdluse tulemusi on võimalik salvestada TXT-failina ning sarnaste meloodiajärgnevuste graafikut PNG-failina. Programmis on olemas ka nupp „Abi“, mis kirjeldab programmi tööd ning kuvab kasutusjuhendi.

5.3 Tarkvara arendus

Programmi kaks peamist tööpõhimõtet on sarnaste meloodiajärgnevuste tuvastamine ning meloodiate vahelise teisenduskauguse leidmine. Järgnevalt kirjeldatakse, mis etappidest koosnevad programmi algoritmid. Tarkvara arendamisel tehti ka kitsendusi, mis kehtivad mõlema funktsiooni puhul:

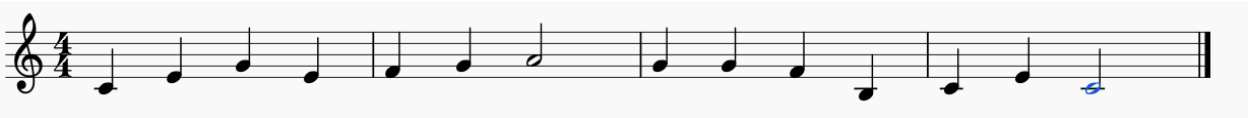
1) Noodistusest on keeruline tuvastada, millise instrumendi partii on meloodia, mistõttu meloodiana arvestatakse esimese instrumendi ülemise partii kõrgeimat häält.
2) Kui taktis esineb kooskõlasid, siis arvestab algoritm noodijärgnevuses ainult iga kooskõla ülemist nooti.

5.3.1 Sarnaste meloodiajärgnevuste tuvastus intervallide järgi

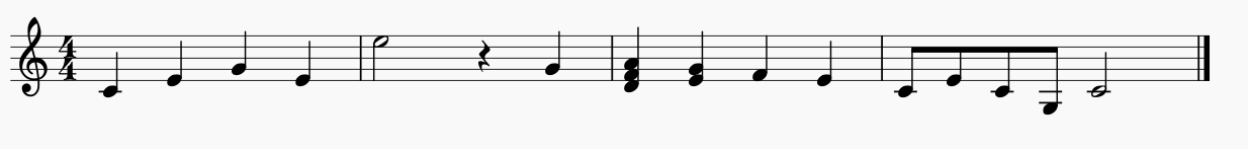
Noodifaili valimise järel tuvastatakse meloodia partii ning jagatakse selle elemendid eraldi andmestruktuuridesse. Edasi analüüsib programm läbi kõik partiis esinevad taktid, eemaldab taktidest pausid ning tekitab nootidest ühtse jada nende järjekorda muutmata. Nootide jada teisendatakse nootide vaheliste intervallide jadaks, kus iga element tähistab pooltoonide arvu antud intervallis. Programmis tuvastatakse intervalle, mis on 12 pooltooni ulatuses. Suurema ulatusega intervallide pooltoonid jagatakse 12-ga ning jääk asendatakse esialgse pooltoonide arvuga. Sel juhul on nootide vaheline intervall sama ja ulatusega vastavuses. Samuti arvestatakse ka nootide liikumist meloodias. Üles liikumist märgitakse intervalli pooltooniga, alla liikumist vastava intervalli pooltooni vastandarvuga. Seejärel teisendatakse saadud intervallide pooltoonidest koosnev arvujada ühetähelisteks sümboliteks. Pooltoonid on vahemikus -12 kuni 12 ehk teisendamiseks kasutatakse 25-elemendilist sõnastikku. Algoritm tuvastab sarnaseid meloodiajärgnevusi sõnede võrdlemise meetodil, mistõttu on tarvis teisendada andmestruktuurid üheks pikaks sõneks, kus iga sümbol tähistab ühte kindlat intervalli.

Kasutaja ette antud arvu järgi võrreldakse omavahel väiksema nootide arvuga noodistust suuremaga. See tähendab seda, et kui otsitakse nelja noodi pikkust järgnevust, siis algoritm leiab väiksema noodistuse kõik nelja noodi pikkused meloodiajärgnevused (alustades esimesest neljast noodist ja lõpetades viimase nelja noodiga) ning võrdleb neid pikema noodistusega. Noodijärgnevusi tuvastatakse regulaaravaldise abil. Järgnevalt otsib programm välja kõikide leitud järgnevuste asukohad noodistuses ning seejärel teisendab saadud ühetäheliste sümbolite jadad tagasi esialgseteks noodinimedeks, et oleks arusaadav, millised noodijärgnevused on sarnased. Ekraanile kuvatakse kõik sisestatud pikkusega järgnevused, mis

esinevad mõlema noodistuse meloodias. Nende meloodiajärgnevuste asukohad on kuvatud ka graafikul. Järgnevalt on näidatud sarnaste meloodiate leidmine etapi kaupa (tabel 3).



Joonis 15. Meloodia A.



Joonis 16. Meloodia B.

Tabel 3. Nelja noodi pikkuste meloodiajärgnevuste leidmise etapid

Meloodia nimetus	A (joonis 15)	B (joonis 16)
Nootide järgnevus	C, E, G, E, F, G, A, G, G, F, B, C, E, C	C, E, G, E, E, G, A, G, F, E, C, E, C, G, C
Intervallide järgnevus	s3, v3, v3, v2, s2, s2, s2, p1, s2, >5, v2, s3, s3	s3, v3, v3, p8, s6, s2, s2, s2, v2, s3, s3, s3, p4, p4
Pooltoonideks teisendatud intervallid	4, 3, 3, 1, 2, 2, 2, 0, 2, 6, 1, 4, 4	4, 3, 3, 12, 9, 2, 2, 2, 2, 1, 4, 4, 4, 5, 5.
Intervalli järgnevus pooltoonide järgi, kus arvestatakse ka meloodia liikumist	4, 3, -3, 1, 2, 2, -2, 0, -2, -6, 1, 4, -4	4, 3, -3, 12, -9, 2, -2, -2, -1, -4, 4, -4, -5, 5

Intervallide pooltoonidele sõnastiku järgi ühekohalise sümboli määramine[7]	EDPBCCOAOSBEQ	EDPMUCOONQEQRF
Leiame sarnase järgnevuse ja teisendame tagasi nootideks	EDP → 4, 3, -3 → s3, v3, v3 → **C, E, G, E**	EDP → 4, 3, -3 → s3, v3, v3 → **C, E, G, E**

5.3.2 Takistused meloodia sarnasuste leidmisel teisenduskauguse abil

Programmi arendamisel oli idee hinnata meloodiate sarnasust Levensteini kauguse algoritmi abil. Esmalt tuli välja selgitada, millistel alustel on meloodiaid võimalik üldse võrrelda. Sarnaselt meloodiajärgnevuste tuvastamisele prooviti teisenduskaugust välja arvutada kõigepealt intervallide kaudu. Kahe noodistuse meloodianoodid teisendati intervallide jadaks ning see omakorda sõnedeks. Kõik Levensteini algoritmi teisendused olid hinnaga 1. Esimese katsetuse tulemuseks oli teisenduskaugus meloodia A (joonis 15) ja meloodia B (joonis 16) vahel 7. Protsessi analüüsimisel leiti mitu probleemi.

Esiteks tekkis küsimus, kas intervallide kaudu on üldse õige hinnata meloodiate sarnasust, sest intervallide asenduse puhul võib noodistuse meloodia muutuda ning nii lisamise või kustutamise protsessiga muutuks esialgne nootide arv. Intervallide puhul poleks saanud kaasata ka tükeldamise või ühendamise protsesse (vt. ptk. 4.1).

Teine probleem oli Levensteini algoritmi hindades. Selge oli see, et kui kaks intervalli on samad, siis on kaugus 0, kuid kui üks intervall on näiteks puhas priim ning teine on puhas kvart, siis tekkis küsimus, et kuidas saab täpsemalt määrata nende vahelist kaugust. Selle lahendamiseks üritati Levensteini algoritmis teisendada sõne sümbolid vastavate intervallide toonilisteks suurusteks ning selle kaudu määrata meloodiate kaugust. See tekitas uue takistuse, kuna nüüd eelistas algoritm asendustele kas lisamist või kustutamist, sest nende hind oli endiselt 1. Sellise kauguse määramise meetodil ei suudetud hindade suurusi tasakaalustada.

Järgmise etapina prooviti Levensteini algoritm läbi ka nootide kaudu. Sarnaselt intervallidele teisendati meloodianoodi sümbolite jadaks ning võrreldi neid sõnedena. Endiselt jäi lahendamata probleem, kuidas tasakaalustada hindu, et nii lisamine, kustutamine kui ka

[7] Kasutatud sõnastik: {0: "A", 1:"B", 2:"C", 3:"D", 4:"E", 5: "F", 6:"G", 7:"H", 8:"I", 9:"J", 10:"K", 11:"L", 12: "M",-1:"N", -2:"O", -3:"P", -4:"Q", -5: "R", -6:"S", -7:"Z", -8:"T", -9:"U", -10:"V", -11:"W", -12: "X"}

asendamine oleksid kooskõlas Levensteini algoritmi põhimõtetele. Selle probleemi lahendamine jäi bakalaureusetöö ulatusest välja. Töös tehtid teisenduskauguse arvutamise lihtsustatud variant, mis kirjeldatakse lahti järgmiselt.

5.3.3 Meloodia sarnasuse leidmise arendusprotsess kasutades teisenduskaugust

Kahe meloodia vaheline sarnasus leitakse nootide kaudu. Selleks on mõlema noodistuse kogu meloodia eraldatud nootide jadana andmestruktuuri. Noodid teisendatakse ümber arvude paarideks, kus paari esimene arv on noodinimele vastav heliklassi tähistav arv [31] (joonis 17) ja teine arv on vastava noodi oktavi number. Heliklassi tähiste määramisel arvestatakse ka enharmonismi. Meloodia rütmist või erinevatest interpretatsioonidest tingituna võib nootides esineda üleliigseid kordusi, mis tegelikult ei mõjuta meloodiajoonist ning järgnevust (joonis 18). Kasutajal on võimalus programmis noodistusi võrrelda esialgsel kujul kui ka ühendada meloodias järjest korduvad noodid üheks. Teise valikuna on kasutajal võimalus valida kahe teisenduskauguse arvutamise meetodi vahel. Esimesel juhul võrreldakse meloodiatevahelist kaugust täpselt nii, nagu noodistuse kirjapilt näitab (edaspidi meetod 1). Teisel juhul võrreldakse mõlema meloodia noote väiksema võimaliku vahemaa (edaspidi meetod 2) järgi, muutes nootide oktavit. Järgnevalt kirjeldatakse täpsemalt lahti mõlema viisi arvutusmeetodid. Mõlema meloodianootide andmestruktuuridest valitakse arvupaarid järjest välja ning võrreldakse esialgu omavahel. Järgnevalt on kirjeldatud teisenduskauguse arvutamise etapid:

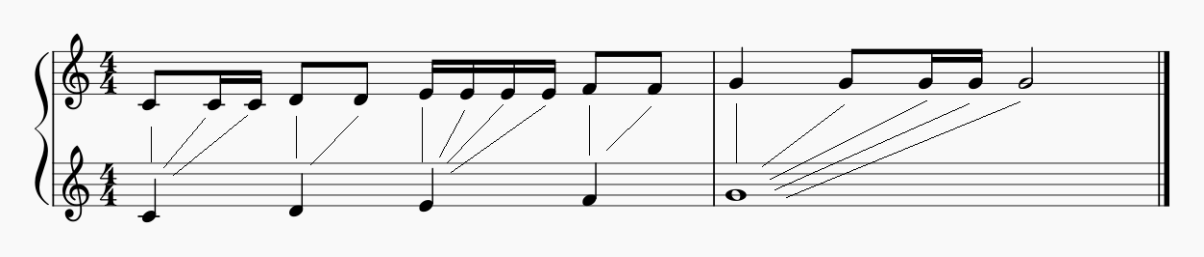
1) Kui mõlema noodi heliklassi tähis ja oktavi number on võrdne, siis on nende vaheline kaugus 0, sest tegemist on sama oktavi samade nootidega.
2) Kui mõlemad noodid on samas oktavis, aga heliklassi arv on erinev, siis leitakse nende nootide vaheline kaugus, lahutades ühe noodi heliklassi tähistavast arvust teise noodi oma. Kauguse arvutamise tulemusest võetakse absoluutväärtus.
3) Kui mõlemad noodid on sama heliklassi arvuga, aga erinevas oktavis, siis arvutatakse kaugus välja vastavalt teisenduskauguse leidmise meetodile. Meetod 1 puhul tuvastatakse esimese meloodia noodi kaugus teises meloodia noodist. Meetod 2 puhul on tulemus 0, sest noodid on samad olenemata oktavist.
4) Ülejäänud juhud arvutatakse meetodis 1 välja eraldi valemitega (joonis 19). Meetodis 2 leitakse nootide oktaveid muutes lähim tee, kas esimesest noodist teise või vastupidi.

Nootidevahelised teisenduskaugused summeeritakse ning nendele liidetakse juurde andmestruktuuri pikkuste vahe. Kõige täpsema tulemuse leidmiseks läbib algoritm sama protseduuri vastavalt kasutaja valitud meetodile kõigis helistikes ehk võrreldava noodistuse

meloodiat transponeeritakse vahemikus 1 kuni 11 pooltooni võrra üles, sest 12. pooltoon annab meile esialgse helistiku. Programmi võrdluse aknasse kuvatakse saadud tulemustest väikseim arv ning transponeerimiskaugus (joonis 9). Sarnaselt käib teisenduskauguse arvutamine ka juba analüüsitud diginootidega „log.txt“ faili sisu olemasolul.

Heliklassi tähistav arv	Vastav noot
0	C
1	C♯, D♭
2	D
3	D♯, E♭
4	E
5	F
6	F♯, G♭
7	G
8	G♯, A♭
9	A
10	A♯, B♭
11	B

Joonis 17. Heliklasse tähistavad arvud ja vastavad noodid.



Joonis 18. Järjest korduvate nootide ühendamine.

$$S = \begin{cases} (h1 - h2) + 12 * (o1 - o2), & h1 > h2,\ o1 > o2 \\ (h2 - h1) + 12 * (o2 - o1), & h1 < h2,\ o1 < o2 \\ 12 * (o1 - o2) - (h2 - h1), & h1 < h2,\ o1 > o2 \\ 12 * (o2 - o1) - (h1 - h2), & h1 > h2,\ o1 < o2 \end{cases}$$

S on kaugus, h1 ja h2 on nootide heliklasside arvud ning o1,o2 nootide oktavi arvud.

Joonis 19. Teisenduskauguse arvutamine meetod 1 puhul.

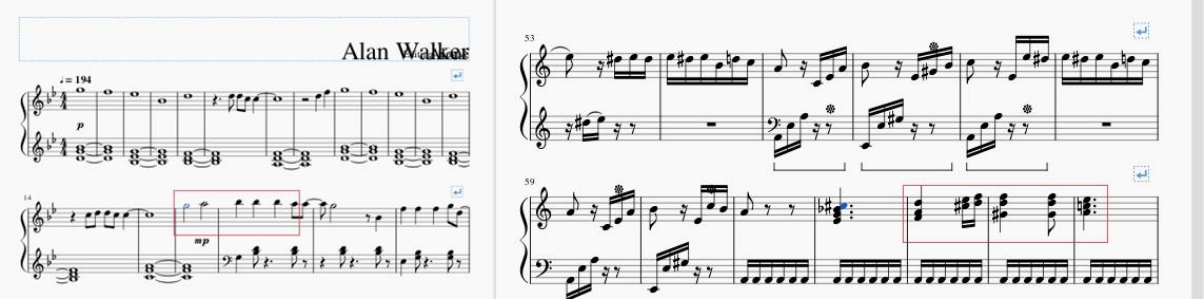
Tabel 4. Teisenduskauguse arvutamise näide meetod 2 puhul.

Meloodia nimetus	A (joonis 15)	B (joonis 16)
Nootide järgnevus	C4, E4, G4, E4, F4, G4, A4, G4, G4, F4, B3, C4, E4, C4	C4, E4, G4, E4, E5, G4, A4, G4, F4, E4, C4, E4, C4, G3, C4
Nootide teisendamine paarideks sõnastiku järgi	[(0, 4), (4, 4), (7, 4), (4, 4), (5, 4), (7, 4), (9, 4), (7, 4), (7, 4), (5, 4), (11, 3), (0, 4), (4, 4), (0, 4)] Noote kokku: 14	[(0, 4), (4, 4), (7, 4), (4, 4), (4, 5), (7, 4), (9, 4), (7, 4), (5, 4), (4, 4), (0, 4), (4, 4), (0, 4), (7, 3), (0, 4)] Noote kokku: 15
Juhul kui on märgitud märkeruut „Ühenda järjest korduvad noodid“	[(0, 4), (4, 4), (7, 4), (4, 4), (5, 4), (7, 4), (9, 4), (7, 4), (5, 4), (11, 3), (0, 4), (4, 4), (0, 4)] Noote kokku: 13	[(0, 4), (4, 4), (7, 4), (4, 4), (4, 5), (7, 4), (9, 4), (7, 4), (5, 4), (4, 4), (0, 4), (4, 4), (0, 4), (7, 3), (0, 4)] Noote kokku: 15
Meloodia sarnasuse leidmine	Ühendamata nootide puhul on meloodia A teisenduskaugus B-st: (0+0+0+0+1+0+0+0+2+1+1+4+4+5)+1 = 19 Ühendatud nootide puhul on meloodia A teisenduskaugus B-st: (0+0+0+0+1+0+0+0+0+5+0+0+0)+2= 8	

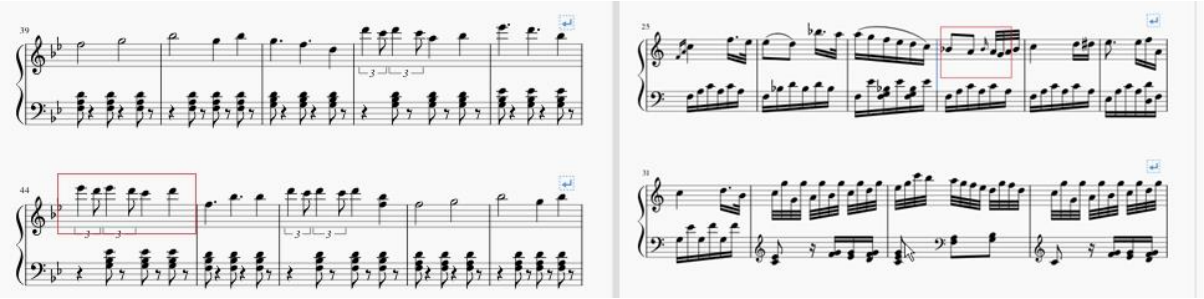
5.4 Tarkvara testimine ja tulemused

Tarkvara testiti selle kolme põhifunktsiooni kaudu. Esimese kahe funktsiooni testimiseks kasutati samu noodistusi, millega testiti ka MusicXML Analyzer veebirakendust (vt. ptk. 4.2.2). Noodistuse info oli diginootide puhul kuvatud korrektselt. Kui mingit elementi

programm ei tuvastanud, siis oli ka vastav kirje selle kohta olemas. Sarnaste meloodiajärgnevuste leidmine oli samuti täpne. Näiteks võrreldi teoste Ludwig van Beethoven „Für Elise" ning Alan Walker „Alone“ noodistusi ning otsingu tulemusena leiti kaks kuuenoodilist järgnevust, mis olid ka pikimad sarnased järgnevused. Käsitsi kontrollimisel tuvastati, et näiteks üks neist järgnevustest algas esimeses noodistuses alates 61. taktist ning teisel noodistusel 16. taktist (joonis 20). Teine järgnevus esines „Alone“ meloodia 44. taktis ning „Für Elise“ noodistuse 28. taktis (joonis 21).



Joonis 20. Esimene sarnane meloodiajärgnevus. „Alone“ vasakul ja „Für Elise“ paremal.



Joonis 21. Teine sarnane meloodiajärgnevus. „Alone“ vasakul ja „Für Elise“ paremal.

Programmi teisenduskauguse leidmise funktsiooni testiti kümne erineva vabalt valitud noodistusega. Aluseks võeti Fur Elise.musicxml noot ning uuriti, milline ülejäänud üheksast diginoodi meloodiast on sellega kõige sarnasem. Teisenduskaugused arvutati läbi neljal viisil:

a) Meetod 1 ja ühendamata noodid
b) Meetod 1 ja ühendatud noodid
c) Meetod 2 ja ühendamata noodid
d) Meetod 2 ja ühendatud noodid

Saadud tulemuste põhjal koostati programmiväliselt eraldi tabel, mis näitab noodistuste jaotust ning teisenduskaugusi võrreldes aluseks võetud diginoodiga (joonis 22).

Noodistuste teisenduskaugused võrreldes Fur elise.musicxml noodistusega				
Noodistuse failinimi	**a) Meetod 1 ja ühendamata**	**b) Meetod 1 ja ühendatud**	**c) Meetod 2 ja ühendamata**	**d) Meetod 2 ja ühendatud**
Beethoven_Symphony_No._5_1st_movement_Piano_solo.mxl	4278 (0 pooltooni üles)	3915 (0 pooltooni üles)	2422 (9 pooltooni üles)	1919 (10 pooltooni üles)
Ed Sheeran.musicxml	996 (1 pooltoon üles)	906 (1 pooltoon üles)	774 (4 pooltooni üles)	670 (4 pooltooni üles)
Feeling_Good_-_Muse.musicxml	3979 (0 pooltooni üles)	2360 (0 pooltooni üles)	1405 (2 pooltooni üles)	1075 (7 pooltooni üles)
Piano_Requiem_in_D_Minor_Dies_Irae_By_W._A._Mozart_.mxl	3244 (0 pooltooni üles)	2745 (0 pooltooni üles)	1378 (3 pooltooni üles)	1213 (10 pooltooni üles)
Yann_Tiersen_-_Comptine_Dun_Autre_Ete__Lapres.musicxml	4524 (0 pooltooni üles)	4388 (0 pooltooni üles)	1570 (10 pooltooni üles)	1562 (10 pooltooni üles)

Joonis 22. Teisenduskauguse testimise tulemused.

Testimise tulemuse põhjal sarnaneb Fur Elise.musicxml noodistuse meloodia kõige rohkem Ed Sheeran.musicxml noodistuse meloodiaga. Teisenduskauguse arvutamine viisil a) andis kõige suuremad tulemused, sest võrreldavaid noote oli rohkem ning variatiivsus oli väikem. Kõige väiksemate tulemustega oli juhul, kui kasutusel oli meetod 2 ning järjestikused korduvad noodid olid ühendatud. Meetod 2 puhul leiti väikseim teisenduskaugus transponeeritud meloodiate hulgast, meetod 1 tulemusete puhul tuvastati väikseimad kaugused, kui meloodiad olid peamiselt esialgses helistikus.

Testimise käigus selgus, et ka see meetod pole kõige parem kahe erineva noodistuse meloodiate hindamiseks, kuna algoritm ei teosta nootidevahelisi lisamisi ning kustutamisi. Meloodiate võrdlemisel ei tuvastata minimaalset kaugust, vaid pigem antakse vastavalt meetodile arvuline hinnang meloodiate sarnasuse kohta. Autori hinnangul on välja arendatud meetoditest kõige parem variant d), sest selle puhul ei arvestata korduvaid noote ning võrreldavate nootide vahel leitakse vähimad omavahelised kaugused.

5.5 Edasiarendusvõimalused

Digitaalsete nootide analüsaator on prototüüp. Programmi on võimalik mitmel viisil edasi arendada ning lisada ka uusi funktsioone.

5.5.1 Üldised arendused

Üldiste arenduste all on mõeldud neid, mis pole seotud programmi olemasolevate põhifunktsioonide algoritmide täiendustega. Praegune programm võiks aktsepteerida ka teisi diginoodi failitüüpe lisaks MusicXML-failile ning kasutaja võiks saada samal ajal analüüsida rohkem kui kahte noodistust. Selle arenduse lisamisel oleks võimalik uurida ühe kindla helilooja teoseid korraga, loomingu sisu ja selle muutusi ajas või siis analüüsida erinevaid noote vastavalt žanrile või ajastule. Näiteks saaks välja arendada süsteemi, kus kõik

võrreldavad noodistused lisatakse nimekirja ning analüüsitakse mingi teise noodistusega järjest läbi. See annaks võimaluse lihtsalt läbi töödelda suurel hulgal noodistusi. Sel juhul tuleks tagada programmikoodi töö efektiivsus, et nootide töötlemine ei oleks liiga ajamahukas. Praegust lähtekoodi tuleks analüüsida, milline on muutujate mälukasutus ja kas oleks võimalik koodi optimeerida. Näiteks saaks kasutada paremini ära rohkem Pythoni sisse ehitatud funktsioone.

Programm ei ole hetkel täielikult eestikeelne. Nootide infokuval on arvatav helistik, instrumentide ning kooskõlade nimetused inglise keelsed. Programmi saaks juurde lisada käsitsi tõlgitud sõnastikud või kasutada Pythoni tõlketeeke, et ingliskeelsed muusikaterminid muuta eestikeelseteks.

Sarnaselt MusicXML Analyzer programmile (vt. ptk. 5.2.2) võiks programmile juurde lisada rohkem graafikuid. Näiteks võiks olla võimalus kuvada tulp- või sektordiagramme noodistuse kvantitatiivse info kohta (milline on nootide, kooskõlade jaotus, pauside osakaal). Graafikuid võiks saada kuvada nii ühe kui ka mitme valitud noodistuse kohta.

5.5.2 Meloodiatevahelise sarnasuse leidmise arendused

Kasutajal võiks olla võimalus valida noodistuse partiide vahel, millise partii nootide järgnevust soovitakse omavahel võrrelda. See annaks kasutajale võimaluse määrata enne võrdlust, milline on meloodia või harmoonia partii, ning seejärel analüüsida ka sarnaseid harmooniaid noodistuste vahel.

Sarnaste meloodiajärgnevuste tuvastamisel ning teisenduskauguse mõõtmisel ei arvestata praeguses versioonis nootide rütmiga. Samuti arvatakse võrdlusest välja kõik pausid. Edasiarendusena võiks analüüsimisel olla võimalus arvestada nii noodi rütmi kui ka pausidega ning seejärel hinnata, kuidas tulemused muutuvad, millised seosed tekivad ning kas nende kaasamine analüüsi on määrava tähtsusega.

Meloodiajärgnevuste kuvamist saaks visualiseerida paremini. Esiteks võiks olla võimalus tuvastada järgnevusi graafiliselt noodijoonestikul ning nende vastavates asukohtades noodistuses. Samuti saaks juurde lisada võimaluse leitud järgnevusi kas kuulata või salvestada näiteks MIDI- või mõne helifailina, et tuvastada sarnasust ka kuulmise teel.

Meloodiate teisenduskauguse tuvastamisel algoritmi tuleks täiendada ning leida viis, kuidas tasakaalustada Levensteini kauguse hindu, et need oleksid vastavuses muusikateooriaga. Samuti võiks noodistuste vaheliseks hinnanguks olla protsentuaalne suhe, kui sarnane on ühe

noodistuse meloodia teisega. Selle abil oleks võimalik paremini ühte noodistust võrrelda teistega ning lõpuks kuvada sarnasuse graafik, kus rohkem sarnasemad on võrreldavast noodistusest graafikul lähemal ning vähem sarnasemad kaugemal. Teisenduskauguse algoritmi võiks saada rakendada ka sarnaste harmooniate leidmisel või hinnata kogu noodistust tervikuna. Nii võiks tulevikus programmist kasu olla ka muusika plagiaadi tuvastamisel.

Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli luua digitaalsete nootide meloodiaanalüüsi tarkvara. Programmis on noodistusi võimalik võrrelda kvantitatiivse info järgi, tuvastada sarnaseid meloodiajärgnevusi ning hinnata meloodiate sarnasust.

Muusikaanalüüsi üks uurimisvorme on muusikateoreetiline analüüs. See analüüs toetub, kas helisalvestiste või noodikirja uurimisele. Helisalvestise puhul uuritakse helilaineid, noodikirja puhul analüüsitakse muusikat kui teksti. Noodikirjast on võimalik üksikasjalikumalt välja lugeda muusika väljendusvahendite sisu ning omavahelisi seoseid.

Tänu noodistuse digitaliseerimisele on võimalik noodikirja analüüsida ka tarkvara abil. Loodud on erinevaid diginoodi failiformaate, kuid kõige standardsemaks on saanud MusicXML. See failiformaat sisaldab endas kõikvõimaliku informatsiooni noodistuse ülesehituse ning elementide kohta.

Diginoote on analüüsitud erinevatel eesmärkidel. Muusikateoste vahel on üritatud leida omavahelisi sarnasusi teisenduskauguse ja intervallide tooniliste suuruste kaudu. Samuti on loodud rakendusi, mis töötlevad MusicXML-faili sisu ning kuvavad informatsiooni noodistuse elementide kohta. Sel viisil on määratud näiteks klaveriteoste noodistuste raskusastmeid ning tuvastatud erinevate noodistuste vahel sarnaseid meloodiajärgnevusi.

Selle bakalaureusetöö raames valmis diginootide meloodiaanalüsaatori prototüüp. Töös kirjeldati lahti programmi kasutusvõimalused ning algoritmide tööetapid. Programmi testiti erinevate diginootidega, et veenduda programmi töö korrektsuses. Sarnaste meloodiate tuvastamise algoritmi tuleks edasi arendada, et tulemused oleksid täpsemad ning vastavuses minimaalse teisenduskauguse leidmise reeglitega. Samuti saab täiendada programmi tööfunktsioone, et korraga võrreldavate nootide hulk oleks suurem ning analüüsitud tulemused oleks laialdasemalt visualiseeritud.

Viidatud kirjandus

[1] Eesti Muusikateaduse Selts. http://www.muusikateadus.ee (14.04.2020)

[2] Ross J., Maimets K. Mõeldes muusikast. 2004. https://www.digar.ee/viewer/et/nlib-digar:347505/303566/page/449 (14.04.2020)

[3] Eesti Fonogrammitootjate Ühing. https://www.efy.ee (28.04.2020)

[4] Piispea M. Noodikirja põhielemendid. 2012. http://eller.tmk.ee/~kaie.magimets/noodikiri1/noodijoonestik.html (14.04.2020)

[5] Kangro E., Leppoja K. Muusikaõpetuse mõisted gümnaasiumile. Tartu Ülikooli Kirjastus. 2009.

[6] Ainsalu E. Muusika põhiõpetus. Tallinn: Valgus. 1968.

[7] EMTA muusikateooria õpik. http://mt.ema.edu.ee/ (14.04.2020)

[8] Sõnaveeb. https://sonaveeb.ee/ (05.05.2020)

[9] Brown A.R. An Introduction to Music Analysis with Computer. 1999. https://eprints.qut.edu.au/6807/1/6807.pdf (04.05.2020)

[10] Hainsworth S. W., Macleod M. D. The Automated Music Transcription Problem. 2004. https://www.researchgate.net/publication/2875163_The_Automated_Music_Transcription_Problem (04.05.2020)

[11] Spreadbury D. The Evolution of Music Notation. 2017. https://www.linkedin.com/pulse/evolution-music-notation-daniel-spreadbury (17.04.2020)

[12] Azoulay A. All About Digital Music Formats. 2017. https://newzik.com/project/all-about-digital-music-formats/ (14.04.2020)

[13] Lilypond. https://lilypond.org/index.html (14.04.2020)

[14] ABC Notation. http://abcnotation.com/ (14.04.2020)

[15] Cunningham S. Suitability of MusicXML as a Format for Computer Music Notation and Interchange. 2004. https://www.researchgate.net/publication/228705496_Suitability_of_MusicXML_as_a_Format_for_Computer_Music_Notation_and_Interchange (17.04.2020)

[16] MusicXML FAQ. https://www.musicxml.com/tutorial/faq/. (17.04.2020)

[17] NIFF 6a.3. http://neume.sourceforge.net/niff/ (17.04.2020)

[18] Good M. MusicXML for Notation and Analysis | Songs and Schemas. 2019. http://michaelgood.info/publications/music/musicxml-for-notation-and-analysis/ (14.04.2020)

[19] Good, M. Beyond PDF – Exchange and Publish Scores with MusicXML. 2013. https://wpmedia.musicxml.com/wp-content/uploads/2013/04/MusicXML-Musikmesse-2013.pdf?_ga=2.229419228.1948622820.1547121805-2137806144.1539945775 (14.04.2020)

[20] Gavin B. What Is An XML file (And How Do I Open One?). 2018. https://www.howtogeek.com/357092/what-is-an-xml-file-and-how-do-i-open-one/ (14.04.2020)

[21] Gilleland M. Levenshtein Distance, in Three Flavours. 2006. http://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Fall2006/Assignments/editdistance/Levenshtein%20Distance.htm (14.04.2020)

[22] Mongeau M., Sankoff D. Comparison of Musical Sequences. 1991. http://albuquerque.bioinformatics.uottawa.ca/Papers/JournalPublication/1990_Mongeau_Sankoff.pdf (23.04.2020)

[23] Lemström K., Ukkonen E. Including Interval Encoding into Edit Distance Based Music Comparison and Retrieval. 2000. https://pdfs.semanticscholar.org/2815/e20634c32981403a1a818fdb13ce8d29bba0.pdf (14.04.2020)

[24] Tremblay G. Chapagne F. Automatic Marking of Musical Dictations By Applying the Edit DistanceAlgorithm On a Symbolic Music Representation. 2002. https://www.researchgate.net/publication/228943422_Automatic_marking_of_musical_dictations_by_applying_the_edit_distance_algorithm_on_a_symbolic_music_representation (14.04.2020)

[25] Sébastien V., Ralambondrainy H., Sébastien O., Conruyt N. Score Analyzer: automatically determining scores difficulty level for instrumental e-Learning. 2012. https://hal.univ-reunion.fr/hal-01187227/document (23.04.2020)

[26] Burghardt, M., Lamm, L., Lechler, D., Schneider, M., Semmelmann, T. MusicXML Analyzer: An analysis tool for computer-aided Identification of Melody Patterns. 2015. https://www.academia.edu/34055548/MusicXML_Analyzer._Ein_Analysewerkzeug_f%C3%BCr_die_computergest%C3%BCtzte_Identifikation_von_Melodie-Patterns (14.04.2020)

[27] What is Music21? https://web.mit.edu/music21/doc/about/what.html (14.04.2020)

[28] Tkinter. https://docs.python.org/3/library/tkinter.html (14.04.2020)

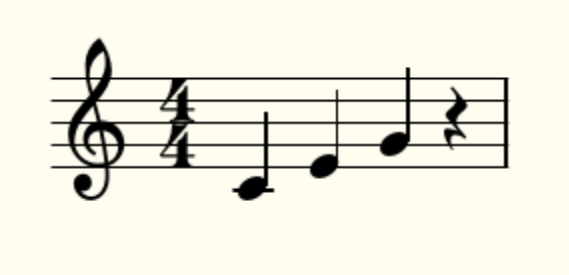
[29] musicXML. Suffixes and Media Types. https://www.musicxml.com/tutorial/compressed-mxl-files/suffixes-media-types/ (14.04.2020)

[30] Key-finding algorithm. http://rnhart.net/articles/key-finding/ (17.04.2020)

[31] Kulma. D. Pitch-class Integers. https://davidkulma.com/musictheory/integers (04.05.2020)

Lisad

I MusicXML-faili esitus



```
<?xml version="1.0" encoding="UTF-8"?>
<score-partwise version="3.1">
  <identification>
    <encoding>
      <software>MuseScore 3.0.1</software>
      <supports element="accidental" type="yes"/>
      <supports element="beam" type="yes"/>
      <supports element="print" attribute="new-page" type="yes" value="yes"/>
      <supports element="print" attribute="new-system" type="yes" value="yes"/>
      <supports element="stem" type="yes"/>
      </encoding>
    </identification>
  <defaults>
    <scaling>
      <millimeters>7.05556</millimeters>
      <tenths>40</tenths>
      </scaling>
    <word-font font-family="FreeSerif" font-size="10"/>
    <lyric-font font-family="FreeSerif" font-size="11"/>
    </defaults>
  <part-list>
    <score-part id="P1">
      <part-name>Piano</part-name>
      <part-abbreviation>Pno.</part-abbreviation>
      <score-instrument id="P1-I1">
        <instrument-name>Piano</instrument-name>
        </score-instrument>
      <midi-device id="P1-I1" port="1"></midi-device>
      <midi-instrument id="P1-I1">
        <midi-channel>1</midi-channel>
        <midi-program>1</midi-program>
        <volume>78.7402</volume>
        <pan>0</pan>
        </midi-instrument>
      </score-part>
    </part-list>
```

```
<part id="P1">
  <measure number="1" width="357.98">
    <print>
      <system-layout>
        <system-margins>
          <left-margin>0.00</left-margin>
          <right-margin>719.52</right-margin>
          </system-margins>
        <top-system-distance>70.00</top-system-distance>
        </system-layout>
      </print>
    <attributes>
      <divisions>1</divisions>
      <key>
        <fifths>0</fifths>
        </key>
      <time>
        <beats>4</beats>
        <beat-type>4</beat-type>
        </time>
      <clef>
        <sign>G</sign>
        <line>2</line>
        </clef>
      </attributes>
    <note default-x="83.07" default-y="-50.00">
      <pitch>
        <step>C</step>
        <octave>4</octave>
        </pitch>
      <duration>1</duration>
      <voice>1</voice>
      <type>quarter</type>
      <stem>up</stem>
      </note>
    <note default-x="149.15" default-y="-40.00">
      <pitch>
        <step>E</step>
        <octave>4</octave>
        </pitch>
      <duration>1</duration>
      <voice>1</voice>
      <type>quarter</type>
      <stem>up</stem>
      </note>
```

```
    <note default-x="215.22" default-y="-30.00">
      <pitch>
        <step>G</step>
        <octave>4</octave>
        </pitch>
      <duration>1</duration>
      <voice>1</voice>
      <type>quarter</type>
      <stem>up</stem>
      </note>
    <note>
      <rest/>
      <duration>1</duration>
      <voice>1</voice>
      <type>quarter</type>
      </note>
    <barline location="right">
      <bar-style>light-heavy</bar-style>
      </barline>
    </measure>
  </part>
</score-partwise>
```

II Diginootide meloodiaanalüsaator

Valminud tarkvaralahenduse programmikood ning vajalikud elemendid (k.a. testitud diginoodid) programmi töö katsetamiseks on kättesaadavad aadressilt: https://gitlab.com/LudvigL/loput-

III. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Ludvig Leis,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

Diginootide meloodiaanalüsaator,

mille juhendaja on Sven Aller,

reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Ludvig Leis

08.05.2020