Poročilo laboratorijske naloge

Ana Polančec
Fakulteta za računalništvo in informatiko
Univerza v Ljubljani
Email: ana.polancec@gmail.com

***Abstract*—V okviru laboratorijske naloge je bila razvita mobilna aplikacija za okolje Andorid za uglaševanje glasbil v programskem jeziku Flutter. Aplikacija za obdelavo zvoka uporablja YIN algoritem. Aplikacija uporabniku pomaga uglasiti kitaro. Poročilo laboratorijske naloge zajema pregled področja, razvoj aplikacije (arhitektura aplikacije, grafični vmesnik, uporabljene metode), vrednotenje razvite rešitve ter uporabljenih algoritmov, ter pomankljivosti in predloge za možne izboljšave v prihodnosti.**

I. Uvod

Namen laboratorijske naloge je razviti mobilno aplikacijo za uglaševanje inštrumentov v programskem jeziku Flutter. V aplikaciji uporabnik lahko uglasi kitaro, izbira pa lahko med dvema najpopularnejšima načinoma uglaševanja - standardno in znižano (drop D). Aplikacija preko mikrofona mobilne naprave zaznava zvok iz okolice ter zaznanemu zvoku s pomočjo YIN algoritma določi ton. Nato uporabniku prikaže v kolikšni meri in v katero smer ton odstopa od frekvenc uglašenih strun. Ko je struna uglašena se elementi na zaslonu obarvajo in uporabnik lahko nadaljuje na uglaševanje naslednje strune. Pred samim razvojem rešitve je potreben pregled področja obdelave signala.

II. Pregled področja - Zaznavanje in obdelava zvoka

Pri zaznavanju in obdelavi zvoka gre za analizo in obdelavo vhodnega zvočnega signala. Najbolj znan algoritem za to je hitra Fourierjeva transformacija. Obstaja še nekaj drugih algoritmov za analizo in obdelavo zvočnega signala kot so na primer YIN algoritem, Harmonic Product Spectrum (HPS) in Short-Time Fourier Transform (STFT). V nadaljevanju sta podrobneje opisana hitra Fourierjeva transformacija in YIN algoritem, saj je YIN algoritem posebej zasnovan za določanje osnovne frekvence v monofonemskih signalih, kot je glasba ali govor [4].

A. FFT - Hitra Fourierjeva transformacija

Hitra Fourierjeva transformacija (FFT) je učinkovit algoritem za pretvorbo signala iz časovne domene v frekvenčno domeno. V kontekstu obdelave zvoka se FFT pogosto uporablja za analizo spektra zvoka. Algoritem omogoča hitro izračunavanje diskretne Fourierjeve transformacije (DFT) in s tem identifikacijo frekvenčnih komponent v signalu [1].

Enačba za izračun DFT je definirana kot:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi}{N}kn} \tag{1}$$

kjer je $x(n)$ vhodni signal, N število vzorcev v signalu, $X(k)$ pa predstavlja kompleksno amplitudo pri frekvenci k [1].

Postopek delovanja FFT je osnovan na deljenju problema DFT na manjše podprobleme, kar omogoča zmanjšanje števila operacij in s tem povečanje hitrosti izračunavanja. Najpogosteje se uporablja Cooley-Tukeyjev algoritem, ki izkoristi rekurzivno deljenje načina delovanja FFT. S tem se doseže kompleksnost $O(N \log N)$, kar je bistveno bolj učinkovito od direktnega izračuna DFT, ki ima kompleksnost $O(N^2)$ [1].

Pri zaznavanju tonov se uporablja analiza spektra, pridobljenega z uporabo FFT. Toni v zvoku se odražajo kot vrhovi v frekvenčnem spektru. Analiza spektra omogoča identifikacijo in lokalizacijo teh vrhov ter s tem določanje temeljnih frekvenc tonov v signalu. S pomočjo FFT je mogoče hitro pridobiti frekvenčne informacije iz zvočnih signalov, kar je ključno pri različnih aplikacijah, kot so glasba, govor in prepoznavanje zvokov [1].

B. YIN algoritem

Yin algoritem je še en pomemben pristop v obdelavi zvoka, zasnovan predvsem za analizo tonov in iskanje osnovnih frekvenc v zvočnem signalu. Yin uporablja avtokorelacijsko funkcijo za oceno temeljne frekvence v signalu. Ta metoda se izkaže za učinkovito pri zaznavanju periodičnosti in harmoničnih struktur zvoka, kar je zlasti uporabno pri analizi glasbe in drugih kompleksnih zvočnih vsebin .Temelji na avtokorelacijski funkciji, ki meri podobnost signala s samim seboj pri različnih zamikih. Glavna ideja je iskanje najmanjše vrednosti v avtokorelacijski funkciji, kar kaže na periodičnost in s tem frekvenco. Algoritem poteka po naslednjih korakih [3]:

1) **Avtokorelacija:** Za dani zvočni signal $x[n]$ se izračuna avtokorelacijska funkcija $R[m]$ po formuli:

$$R[m] = \sum_{n=0}^{N-1} (x[n] - \bar{x})(x[n-m] - \bar{x}) \tag{2}$$

kjer je $\bar{x}$ povprečna vrednost signala.

2) **Normalizacija:** Avtokorelacijska funkcija se normalizira na interval $[0, 1]$ s formulo:

$$R_{\text{norm}}[m] = \frac{R[m]}{R[0]}. \tag{3}$$

3) **Izračun najmanjše vrednosti:** Iskanje najmanjše vrednosti m_0 v normalizirani avtokorelacijski funkciji, ki zadovolji $R_{\text{norm}}[m_0] <$ Treshold, kjer je prag (Treshold) določen glede na šum v signalu.

4) **Interpolacija:** Za povečanje natančnosti se izvede linearna interpolacija okoli m_0.

5) **Določitev temeljne frekvence:** Končna ocena temeljne frekvence se izračuna kot inverz periodo, tj.

$$f_0 = \frac{1}{m_0} \tag{4}$$

C. Primerjava FFT in YIN algoritma

Ob primerjavi algoritmov FFT in YIN algoritem je moč opaziti nekaj razlik. FFT je bolj splošna metoda, primerna za analizo širšega spektra zvočnih signalov, medtem ko je YIN algoritem specializiran za natančno zaznavanje tonov in frekvenc v glasbenih kontekstih. FFT je primernejši za hitro analizo velikih količin podatkov, medtem ko se YIN bolj pogosteje uporablja za natančnejše naloge, kot je določanje tonov in njihovih harmoničnih struktur. Obe metodi imata svoje prednosti in slabosti, odvisno od specifičnih zahtev aplikacij v obdelavi zvoka. Za implementacijo v okviru te naloge je bil izbran algoritem YIN.

III. Razvoj rešitve

A. Arhitektura aplikacije

Aplikacija je sestavljena iz dveh delov. V prvem delu uporabnik določi, kateri inštrument želi uglasiti, ter izbere način uglaševanja. Drugi del predstavlja samo uglaševanje. Informacija o izbranem glasbilu in želenem načinu uglaševanja se zapiše v pomnilnik. Sledi prehod na naslednji zaslon. V ozadju se menjava zaslonskih mask upravlja z Navigator razredom. Tam se glede na izbrano glasbilo in uglasitev prilagodi grafični vmesnik ter podatki o frekvencah v pomnilniku. Nato se s pritiskom na gumb prične zajem zvoka. Zvok se ob zajemanju hkrati obdeluje. YIN algoritem iz zajetega signala vrača najverjetnejšo frekvenco. Nato se ta ton primerja z vnaprej znanimi frekvencami, ki določajo izbrano uglasitev. Glede na to, ali je zaigran ton med vnaprej znanimi frekvencami ali ne (oz. v zadostni bližini ±1 Hz), uporabnik prejme povratno informacijo preko grafičnega vmesnika, v katero smer mora struno prilagoditi (”Tune up/Tune down”). Ko je struna uglašena je uporabnik o tem obveščen preko grafičnega vmesnika in lahko nadaljuje z uglaševanjem naslednje strune.

IV. Grafičen uporabniški vmesnik

Aplikacijo sestavljata dve zaslonski maski - vhodna stran za izbiro glasbila in način uglaševanja (Fig. 1), ter stran za samo uglaševanje (Fig. 5. Med stranmi se uporabnik premika s pomočjo grafične puščice.

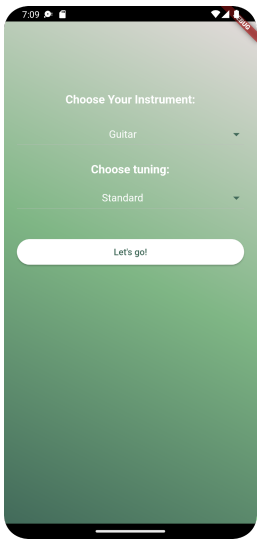

Fig. 1: Zaslonska maska vhodne strani

Uporabnik lahko iz spustnih menijev (Fig. 2) izbira med različnimi načini uglaševanja za inštrument (npr. standardno EADGBE ali drop D uglaševanje za kitaro).

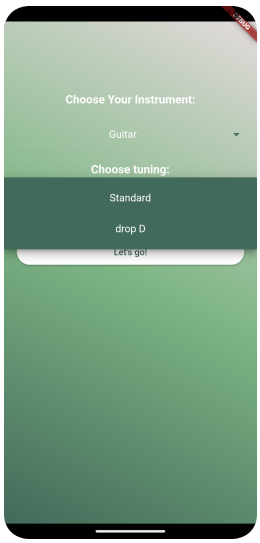

Fig. 2: Prikaz spustnega menija za izbiro načina uglaševanja

Pred zajemanjem in obdelavo signala aplikacija uporabnika prosi za dovoljenje za dostop do mikrofona, v kolikor to še ni bilo dodeljeno (Fig. 3.

Zaslonska maska strani za uglaševanje prikazuje kateri inštrument trenutno uglašujemo in na katero uglasitev želimo inštrument uglasiti. (Fig. 5). Prikazuje število strun in imena

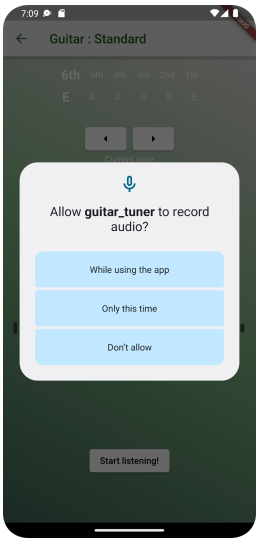

Fig. 3: Prikaz prošnje za dostop do mikrofona

tonov, na katere želimo te struno uglasiti. Na zaslonski maski sta dva gumba s puščicama, s katerimi se uporabnik premika med strunami. Struna, na kateri se trenutno nahajamo, je odebeljena. Sledi frekvenca, ki jo aplikacija trenutno zaznava in frekvenca, ki jo želimo doseči pri uglaševanju izbrane strune. Prav tako prikazuje gumb, ki ga uporabnik pritisne za začetek uglaševanja ter grafiko zvočnih valov. Med samim uglaševanjem je grafika aktivna (gif). Ista grafika je uporabljena tudi za launch-icon - ikono ob zagonu aplikacije ter ikono na domačem zaslonu mobilnika (Fig. 4).

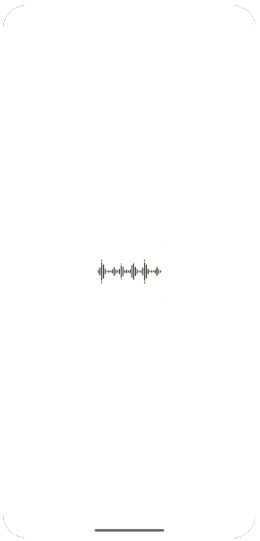

Fig. 4: Zaslon med nalaganjem aplikacije

Med samim uglaševanjem je uporabniku prikazana tudi povratna informacijo - ali je struna uglašena, ali je ton potrebno zvišati ali znižati. Ko je struna uglašena se gumba s puščicama obarvata zeleno in uporabnik lahko nadaljuje na naslednjo struno (Slika 6 - opomba: na sliki je pirkazan ton A (220Hz) kot ustrezna uglasitev, čeprav to ne drži. Temu je tako zgolj zaradi tehničnih težav pri testiranju - razdelek Vrednotenje).

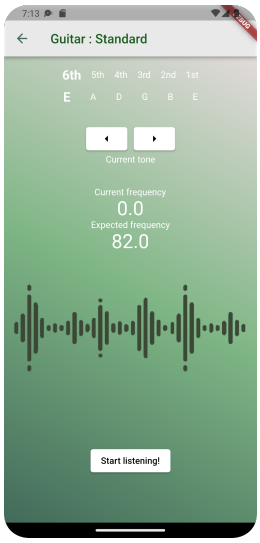

Fig. 5: Prikaz strani za uglaševanje pred začetkom

.

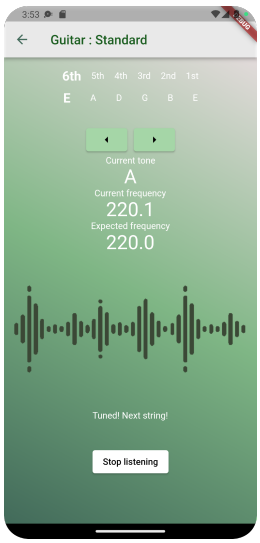

Fig. 6: Prikaz strani med uglaševanjem

A. *Implementacija*

Celotna koda projekta je dostopna v Github repozitoriju. Aplikacija je implementirana v programskem jeziku Flutter, testirana pa je v Android Studio Android emulatorju. Programski jezik Flutter zagotavlja brezšiven prenos tudi na iOS okolja, vendar je aplikacija zaradi pomankanja tehnične opreme testirana zgolj za Android okolje. Za razvoj je uporabljena naslednja programska oprema: Visual Studio Code, Android Studio.

Flutter knjižnjice, uporabljene pri razvoju aplikacije:

- pitch_detector_dart: 0.0.2

- pitchupdart: 0.0.2
- flutter_audio_capture: 1.0.0
- cupertino_icons: 1.0.2
- permission_handler: 11.0.1
- flutter_launcher_icons: 0.13.1

Specifikacije vseh knjižnic so dostopne na spletu [5]. Pri samem razvoju so bili v pomoč nekateri tutoriali, dostopni na spletu npr. [6]. Grafični element valovanja zvoka bil pridobljen na spletu [2].

Proces implementacije se je začel z osnovno postavitvijo grafičnega vmesnika - dveh zaslonskih mask: domače strani in strani za uglaševanje. Nato je bila razvita logika zajema in obdelave signala. Na zaslonski maski se nahaja gumb Star recording. Ob pritisku na ta gumb se prične snemanje zvočnega vhoda. Ta zvočni vhod se potem obdela s pomočjo YIN algoritma. Pri tem so bile uporabljene zgoraj naštete knjižnice. S pomočjo knjižnjice pitch_detector_dart je bila iz zajetega signala izluščena frekvenca. Ta frekvenca je bila nato primerjana z želeno frekvenco. Le-ta je odvisna od trenutno izbrane strune. V kolikor je frekvenca dostopala za ±1 Hz je sporočilo uporabniku ustrezno prilagojeno (”Tune up/Tune down”). Sledila je izboljšava grafičnega vmesnika druge zaslonske maske, da je vsebovala tudi povratno informacijo o tonu za uporabnika.

V. Vrednotenje

Aplikacija je bila v Android Studio okolju testirana na virtualni napravi Google Pixel 5 z API 33 Android sistemom. Za testiranje je bil za vhod uporabljen generičen zvočni vhod, ki pripada emulatorju (220Hz, ton A). Aplikacija je bila prav tako testirana na OnePlus 8 Pro mobilni napravi. Za testiranje inštrumenta je bila uporabljena kitara, saj je bila edini dostopni inštrument v času testiranja. Sam razvoj je bil zaradi omejenosti fizične opreme nekoliko težaven. Ker je testiranje v Andorid Studi okolju za zvočne učinke lahko težavno, je bilo v zadnjem delu razvoja (testiranje podajanja povratne informacije uporabniku) potrebno testiranje na fizični napravi, ki je bila redko dostopna.

A. Vrednotenje programske rešitve

Aplikacija je bila v praksi preizkušena za uglaševanje kitare. Preizkušeno je bilo standardno in drop D uglaševanje. Med testiranjem aplikacije ni prišlo do nobenih tehničnih težav. Natančnost razvite aplikacije je bila preverjena s pomočjo drugih podobnih rešitev (aplikacija GuitarTuna) ter digitalnega uglaševalca. Hkrati je bila zaigrana struna na kitari, nato pa je bilo zabeležen odziv razvite aplikacije, druge rešitve ter digitalnega uglaševalca. Razvita aplikacija je delovala enako hitro kot ostale pri razpoznavi zvoka. Uglaševanje z aplikacijo GuitarTuna je bilo prijetnejša izkušnja, saj je grafični vmesnik v večji meri prilagojen uporabniku ter naravi uporabe aplikacije. Vsebuje več grafičnih elementov, ki uporabniku olajšujejo natančno uglaševanje. Kljub temu je razvita aplikacija podala primerljivo dobre rezultate.

B. Vrednotenje uporabljenih algoritmov

Pri razvoju aplikacije je bil uporabljen algoritem YIN. Algoritem je v zadovoljivem času (primerljivo z ostalimi rešitvami na trgu) zagotavljal pravilne rezultate.

VI. Uporabnost razvite rešitve in možne izboljšave

V prihodnosti bi bilo aplikacijo moč nagraditi na različne načine. Najbolj uporabno bi bilo dodati samodejno zaznavanje zvočnega vhoda. Tako bi uglaševanje inštrumenta potekalo bolj gladko. Delovanje aplikacije bi prav tako lahko razširili na še več inštrumentov, kot npr. ukulele in violina. Prav tako bi ob ustrezni opremi lahko aplikacijo prenesli tudi na iOS okolje. Razvita rešitev je delujoča in uporabna, vendar je zgolj ena izmed mnogih na trgu mobilnih aplikacij za uglaševanje kitare. Namen razvite aplikacije je bil spoznavanje z zajemom in obdelavo zvoka v okolju Android in spoznavanje z YIN algoritmom kot alternativi za Fourierjevo transformacijo. Ta namen je bil dosežen.

VII. Zaključki

V okviru laboratorijske naloge je bila razvita mobilna aplikacija za Android okolje v programskem jeziku Flutter. Aplikacija služi uglaševanju izbranega inštrumenta. Aplikacija za obdelavo zvoka uporablja YIN algoritem. Razvoj aplikacije je zajemal pregled področja zaznavanja in obdelave zvoka, načrtovanje arhitekture aplikacije, implementacija grafičnega vmesnika in logike delovanja aplikacije. Aplikacija omogoča uporabnikom izbiro inštrumenta ter načina uglaševanja. Med samim procesom uglaševanja uporabnik prejme vizualno povratno informacijo o tem, ali je struna ustrezno uglašena, ali pa je potrebno ton prilagoditi. Kljub omejenim možnostim testiranja zaradi pomanjkanja fizične opreme je aplikacija v praksi dosegla pričakovane rezultate. V prihodnosti bi bilo mogoče aplikacijo izboljšati z dodatnimi funkcionalnostmi, kot je samodejno zaznavanje zvoka, razširitev na več inštrumentov ter prenos na iOS platformo.

References

[1] Wikipedia. Fast Fourier transform. Pridobljeno iz https://en.wikipedia.org/wiki/Fast_Fourier_transform (Dostopano dne: 30 november 2023).

[2] Grafika valovanja. Pridobljeno iz https://giphy.com/stickers/ByMissPluis-podcast-audio-bymisspluis-VAUTWXhF3eSMwHpOsi(Dostopano dne: 8 december 2023)

[3] Cheveigné, A., & Kawahara, H. (2002). YIN, A fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, **111**(5), 1917-1930. https://doi.org/10.1121/1.1458024

[4] Wang, D., Wei, Y., Wang, Y. & Wang, J. (2022) .A robust and low computational cost pitch estimation method. *Sensors (Basel)*. **22**

[5] Pub dev. Pridobljeno iz https://pub.dev/ (Nazadnje dostopano dne: 6 januar 2024).

[6] Techpotatoes. Implementing a guitar tuner app in Dart/Flutter. Pridobljeno iz https://techpotatoes.com/2021/07/27/implementing-a-guitar-tuner-app-in-dart-flutter/ (Dostopano dne: 28 oktober 2023).