Szakdolgozat beszámoló

A félév alatt felépítettem a projektet és elkezdtem ismerkedni a képfeldolgozással. Először csak sima kép beolvasására volt képes a programom, ezzel megnéztem, hogy milyen tulajdonságokkal rendelkezik maga a cv2 programkönyvtár. Eljutottam addig, hogy a Tkinterrel egy sima menüt hoztam létre ahol van 3 gomb, előző kép, következő kép, és határérték változtatás névvel. Ugyanis a cv2 programkönyvtárba sok lehetőség van arra, hogy egy képen egy objektumot meg tudjunk különböztetni. Jelenleg ami számomra célszerű volt az a kék színnek az elhatárolása, a szövegtől, hiszen a mintaadathalmazom kék színnel jelzi az akkordokat, amiket majd a programom ki fog venni szövegként, megváltoztatja, majd újra képpé alakítja azt.

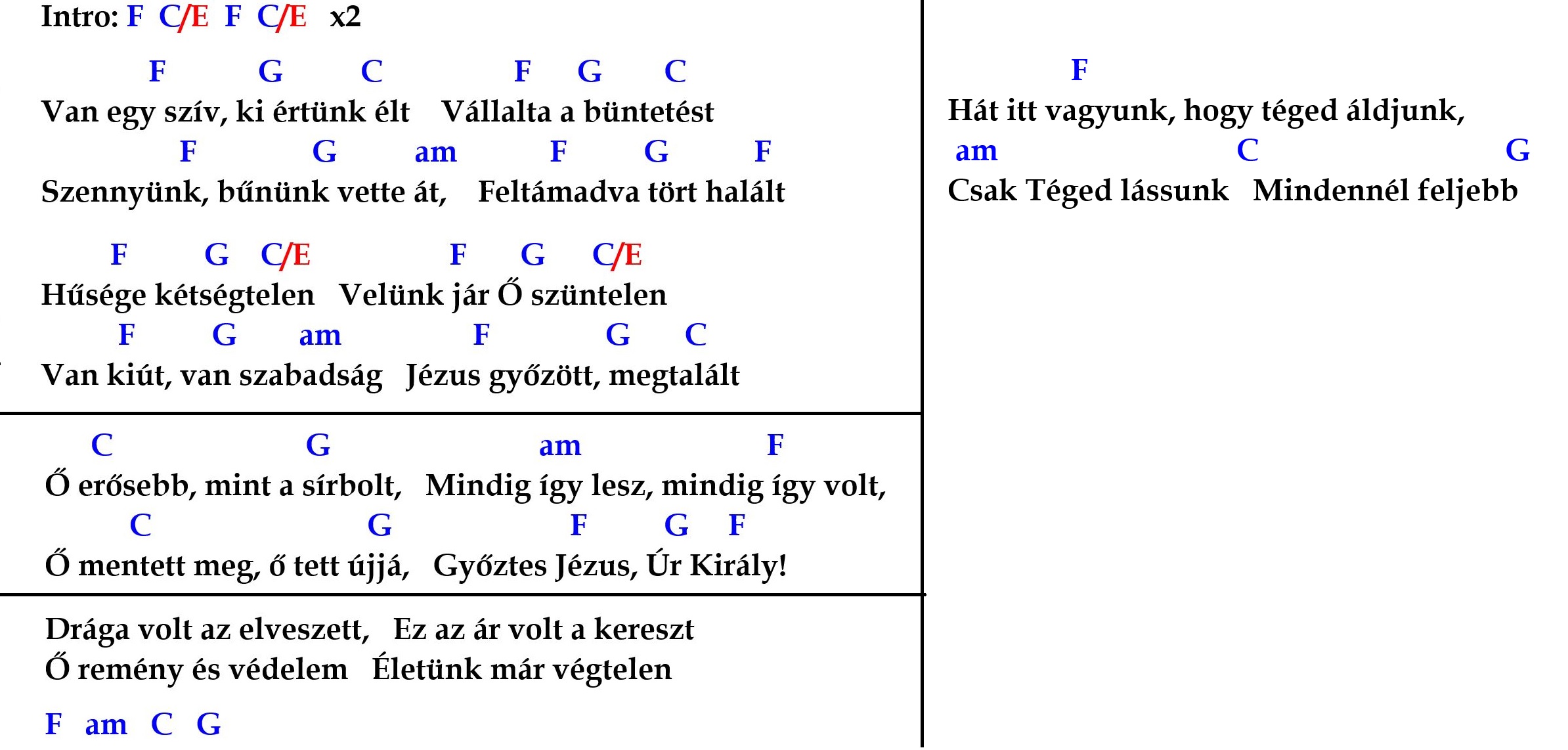
Miután megismertem a cv2, számomra hasznos metódusait, ami jelenleg a kép beolvasása, szín szerinti elhatárolás, majd azon kép mutatása, elkezdtem foglalkozni azzal, hogy miként lehet szöveggé alakítani a képen látható betűket. Ehhez használom a pytesseract nevezetű programkönyvtárat, amivel jelenleg is ismerkedek. Azt teszi lehetővé, hogy egy rendesen beolvasott képből kiszedi a szöveget az image\_to\_string metódussal, így jelenleg a programom képes arra, hogy a képet, amit megkap paraméternek a metódus átalakítsa szöveggé. Kezdetben ezt csak simán a konzolra irattam ki, de jelenleg egy külön fájlba mentem el.

**Mintaadathalmaz bemutatása:**

A mintaadathalmazom egy szöveges akkordkotta. Tehát semmilyen hangjegyek nem találhatóak benne, viszont színekkel van megkülönböztetve, hogy mi az akkord, a szöveg, a basszus akkord, vagy egy transzponálás, mind ezek a következők szerint:

* Fekete: dalszöveg
* Kék: akkord
* Piros: basszus akkord
* Zöld: transzponált akkord

A programom feladata az lesz, hogy a képet beolvasva meg tudja különböztetni a dalszöveget, az akkordoktól. Miután ez megtörtént egy listához fogja rendelni azt a kiválasztott akkordot, amit majd a felhasználó által kiválasztott hangnemhez fog társítani a megfelelő módon. Például van egy ilyen kottám:



Akkor a program megállapítja, hogy ez milyen hangnemben van, az által, hogy milyen akkordok szerepelnek benne a legtöbbször, valamint, a kezdő akkord szerint (viszont most ez megtévesztő, mert F-el kezdődik, miközben a dal C-dúrban van). Mind ez után a felhasználó kiválasztja, hogy milyen hangnembe szeretné lementeni a kottát, tegyük fel, hogy G-dúrban. Ilyenkor az F akkord-hármas C-re, a G D-re, a C G-re és az am em-ra fog változni és így ez a kotta meg lesz G-dúrban. Ahhoz, hogy ez lementhető legyen, ugye vissza kell illeszteni a megfelelő karaktereket a megfelelő színben a képhez, ez a folyamat úgy fog kinézni, hogy a kiválasztott akkordok, mikor szöveggé alakulnak azután el is fognak tűnni a képről, és majd a visszaillesztésnél, a lementett koordináták szerint kerülnek vissza a megfelelő akkordok.

**RDF bemutatása**

Subject – predicate - object

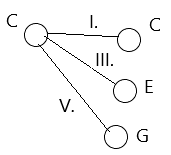
Az RDF az egy interneten levő adatokhoz tartozó standard modell. Lehetővé teszi, hogy adat összefésülést hajtson végre még abban az esetben is, mikor a séma alatt szereplő adatok hiányosak, és kifejezetten támogatja a séma fejlődését az adatok megváltoztatása nélkül.

Az RDF az URI-val nevezi meg azt a kapcsolatot, amivel a link működik az interneten, hogy egyik pontból a másikba elnavigáljon. (Ezt nevezzük hármasoknak) Ezzel az egyszerű modellel lehetővé válik az, hogy struktúrált vagy félig struktúrált adathalmazok is keverhetővé, közzétehetővé és megoszthatóvá váljanak.

Ez a kapcsolati struktúra úgy működik, hogy a két forrás adja a csomópontokat, és az él pedig a kapcsolat nevét. Lényegében ez egy gráf, a csomópont a forrás az él pedig a predikátum, így ha két webhelyet szeretnék összekötni, akkor a kiinduló webhely lesz a téma(subject) a link megnevezés, ami átirányítja a másik csomópontba, az lesz a predikátum(predicate) és ahova érkezik pedig az objektum(object).

**RDF használata a programban (szemantikailag)**

Az esetünkben felhasznált adatok ábrázolására a következőképpen nézne ki az RDF modell. Az akkord reprezentálná a hármasok közül a témát, vagyis a subject-et, mivel azt kellene először definiálni. Az akkord, ami másnéven hármas hangzat, alapvetően három hangból épül fel. Van a fő hang, ami tegyük fel, ha C-dúr akkordot akarunk modellezni, az a C hang lesz, erre épül a dúr esetében a nagy terc, ami az E hangot fogja adni, valamint a C hanghoz igazított kvint távolságra levő G hang. Tehát a C-dúr akkordot a C E és G hangokból lehet összeállítani, ezek egyben fokok is, amiket predikátumként fogunk jelölni, hogy melyik hang az akkord hanyadik foka. Ezt a következő ábrán lehet látni.



Az ábrán látható, hogy egy akkord meghatározásához 3 él vezet kifelé, ugye ezek jelzik, hogy melyik fokon található a C-dúr akkord hangjai (rendre I., III. és V. fok ) és az objektumok maguk a hangok, ahova az él mutat. Ebben az esetben meg is lenne oldva az akkordok reprezentálása RDF modell szerint, viszont a kottába való visszahelyezéskor a programnak szükséges tudnia azt is, hogy pontosan hova illessze vissza a megfelelő akkordot egy transzponálás esetében, mikor is a hangnem változik, és a kottába szereplő akkordok kicserélődnek.

Ilyenkor szintén használva a gráf elméletét az akkord alatti szövegrész lesz az objektum az él pedig elnevezve a pozíció lesz. Ugyanis a kottában szereplő szöveg nem változik ugye csak az akkordok, ha bármilyen művelet kerül végrehajtásra. Több féle probléma merülhet fel, abban az esetben, ha statikusan akarom használni a doboz-t amit az akkordhoz kivág a program a szöveggel együtt, akkor annak a méretét egy kalkulált értékből kell kapnia, ami illszkedik a megfelelő felbontáshoz, mert hogy a felbontás különböző egyes kották képeinél.

**OOP szemantikus modell**

Osztály

metódusok

property-k

private public properties

metódusok:

read, attach, show

properties:

public: source, destination, relation

**XML modell**

hangnem

akkord

I. fok

II. fok

III. fok

pozíció (szövegrész)

**MusicXML**

<?xml version="1.1" encoding="UTF-8" standalone="no"?>

<!DOCTYPE score-partwise PUBLIC

"-//Recordare//DTD MusicXML 1.1 Partwise//EN"

"http://www.musicxml.org/dtds/partwise.dtd">

<score-partwise>

<part-list>

<score-part id="P1">

<part-name>Music</part-name>

</score-part>

</part-list>

<part id="P1">

<measure number="1">

<attributes>

<divisions>1</divisions>

<key>

<fifths>0</fifths>

</key>

<beats>4</beats>

<beat-type>4</beat-type>

<clef>

<sign>G</sign>

<line>2</line>

</clef>

</attributes>

<note>

<pitch>

<step>C</step>

<octave>4</octave>

</pitch>

<duration>4</duration>

<type>whole</type>

</note>

</measure>

</part>

</score-partwise>

Ez a MusicXML egyik példája, a kotta tárolására. Mivel a minta adat, tehát az itt jelenlevő kották másképpen épülnek fel, ahogy az feljebb is említésre került, így egy saját xml valahogy így nézne ki

<?xml version="1.1" encoding="UTF-8" standalone="no"?>

<sheet>

<key>

<note>C</note>

<chord>major</chord>

</key>

<segment id="1">

<note>C</note>

<text>Tied a dicsőség, </text>

<position>őség</position>

</segment>

<segment id="2">

<note>em7</note>

<text>és imádás,\n</text>

<position>im</position>

</segment>

<segment id="3">

<note>F</note>

<text>Felemeljük </text>

<position>elj</position>

</segment>

<segment id="4">

<note>dm</note>

<text>kezeinket\n</text>

<position>in</position>

</segment>

<segment id="5">

<note>F/G</note>

<text>És dicsérjük </text>

<position>csé</position>

</segment>

<segment id="6">

<note>G</note>

<text>szent neved!\n</text>

<position>ved</position>

</segment>

<segment id="7">

<note>G7</note>

<text>Ó, Hatal</text>

<position>Hat</position>

</segment>

<segment id="8">

<note>C</note>

<text>mas, </text>

<position>mas</position>

</segment>

<segment id="9">

<note>am</note>

<text>Keze nagy csodákat tesz,\n</text>

<position>esz,</position>

</segment>

<segment id="10">

<note>F</note>

<text>Vele senki nem ér fel</text>

<position>fel</position>

</segment>

<segment id="11">

<note>dm7</note>

<text>, Vele </text>

<position>, Ve</position>

</segment>

<segment id="12">

<note>G</note>

<text>senki nem ér </text>

<position>se</position>

</segment>

<segment id="13">

<note>G7</note>

<text>fel.\n</text>

<position>fel</position>

</segment>

<segment id="14">

<note>G7</note>

<text>Ó, Hatal</text>

<position>Hat</position>

</segment>

<segment id="15">

<note>C</note>

<text>mas, </text>

<position>mas</position>

</segment>

<segment id="16">

<note>am</note>

<text>ó, Hatalmas, </text>

<position>mas</position>

</segment>

<segment id="17">

<note>F</note>

<text>ó, Hatalmas,</text>

<position>mas</position>

</segment>

<segment id="18">

<note>G7</note>

<text> ó, Hatal</text>

<position> ó</position>

</segment>

<segment id="19">

<note>C</note>

<text>mas!</text>

<position>mas</position>

</segment>

</sheet>

Szegmensekre van bontva az adott akkord és a hozzátartozó szöveg. Minden szegmens kap egy id-t ami a sortöréshez szükséges az első verzió szerint. A szegmensen belül a note tárolja az akkordot, a text a blokkhoz tartozó szöveget, a position pedig tárolja az akkord elhelyezkedését a szöveg felett. Python-ba ehhez készült egy szemléltető program, ami ezt az xml-t jeleníti meg egyenlőre konzolon.

Megfigyelések:

1. A minta adatok hiányosságai

A jelen példában 3 részre van bontva a kotta, szöveg, akkord, pozíció. Ahhoz, hogy ebből tudjon működni a konzolra kiiratás a megfelelő pozíciókkal, az indexelést tettem lista adatstruktúrába, hogy meglegyen mindegyik akkordnak a megfelelő pozíciója. Kétfajta adat kerül tárolásra, az egyik mindenképpen az, hogy az xml-ben megadott string alapján melyik indexű karakternél található a szegmens szövegében a string maga, mert az jelöli az akkord helyét. A másik pedig maga a string ami kiadja a kotta véglegesítését, a konkatenált akkord, - és szóközszámmal, valamint a kotta szövegével.

**Opencv**

--todo