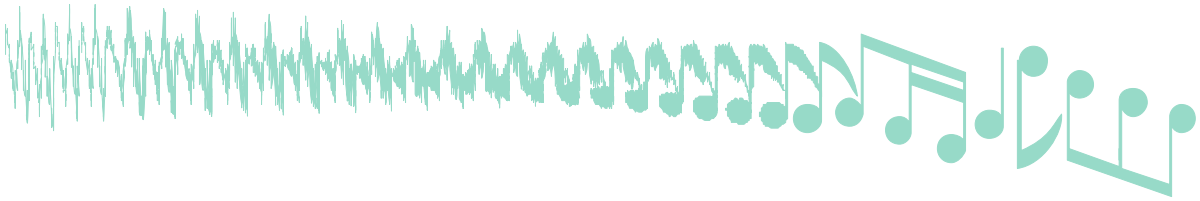
חיפוש במאגר מידע מוסיקלי בעזרת קלט קולי

פרויקט שנתי

חורף-אביב תשנ''ט



צביקה בן-חיים

ת.ז. 034986505

מנחה: גל אשור

תוכן העניינים

רשימת איורים

תקציר

בדו"ח זה תוצג תוכנה, אשר מאפשרת לאדם לבצע חיפוש במאגר מידע של מנגינות, על ידי זמזום קטע מהמנגינה אותה הוא מחפש. החיפוש מתבצע ללא צורך בידע במוסיקה או בעיבוד אותות. האדם מבצע שאילתה (query) על ידי שירה, באופן מסוים, של קטע קצר מהמנגינה אותה הוא מחפש, ומקבל כתוצאת החיפוש את שם המנגינה, שם היוצר, ופרטים אחרים אודותיה, מתוך מאגר המידע.

אופן הפעולה

השלבים הכלליים בביצוע החיפוש הינם זיהוי pitch, חלוקה לתווים בודדים (סגמנטציה), וחיפוש התווים במאגר מידע של שירים ידועים. כמו כן נבדק הצורך בסינון מקדים מעביר-נמוכים לפני זיהוי ה-pitch (פרק 2). סינון כזה מקובל ביישומי זיהוי pitch רבים, אך נמצא כי הסינון אינו משפר את איכות הזיהוי עבור הקלטות מוסיקה בקול יחיד, ובמקרים מסוימים גם עלול לפגוע בטיב הסגמנטציה.

זיהוי ה-pitch התבצע בשיטת super-resolution autocorrelation (Medan et al., 1991), לפיה ערך ה-pitch של האוטוקורלציה מתוקן על סמך שערוך לינארי (פרק 3). שיטה זו מאפשרת חישוב pitch בדיוק גבוה תוך חסכון ניכר בזמן חישוב.

הסגמנטציה התבצעה בשני שלבים (פרק 4). בשלב ראשון, ההקלטה חולקה לסגמנטים על סמך ה-volume, כאשר הסגמנטים (שהם בעלי volume גבוה) מופרדים על ידי אזורי שקט (volume נמוך). בשלב שני, מזוהה איזור בעל pitch אחיד בתוך הסגמנט, ואיזור זה נקבע להיות התו המתאים לסגנמט.

החיפוש במאגר המידע בוצע על ידי מציאת מרחק העריכה המינימלי בין תווי ההקלטה לבין הרשומות במאגר המידע (פרק 5). נבדקו מספר מרחקי עריכה: חיפוש על פי קוד Parsons (1975); חיפוש על פי תדרים יחסיים; וחיפוש המשלב תדרים יחסיים ומשכי-זמן של הצלילים.

תוצאות ומסקנות

בדיקת האלגוריתם התבצעה בשני אופנים. כל חלק של המערכת נבדק בתנאי מעבדה, על ידי סימולציה של פעילות שגרתית של שאר חלקי המערכת (פרק 6). כמו כן, נבדקה התוכנה כמערכת שלמה, על ידי ביצוע הקלטות אנושיות בתנאי אמת (פרק 7).

נמצא כי אלגוריתם קוד Parsons, אשר מקובל במספר יישומים דומים, הוא פחות חסין לרעש מאלגוריתמים חדשים שנבדקו בעבודה זו.

זמן ביצוע החיפוש תלוי בגודל מאגר המידע וכן במשך זמן ההקלטה, אך בתנאים רגילים ניתן לבצע את החיפוש, על מחשב אישי, בזמן מהיר יותר מאשר משך ההקלטה של המנגינה, עובדה המאפשרת יישום בזמן אמת.

בבדיקה בתנאי אמת בעזרת מאגר מידע של 200 שירים, נמצא כי התוכנה מזהה נכון את השאילתה ב-79% מהמקרים. לשם השוואה, בני אדם זיהו נכון את אותן הקלטות רק ב-53% מהמקרים. סימולציות מצביעות על כך שהתוכנית תפעל היטב גם עבור מאגרי מידע גדולים יותר, אם נעשה שימוש בשאילתות של כ-12 תווים.

Abstract

This report presents a software application, which allows a person to search a database of melodies by intoning part of the tune. The person, who need not have any knowledge in music or signal processing, performs a query by singing, in a certain fashion, a short segment of the tune. The program then determines the name of the melody, the name of its creator, and other details available from the database.

Internals

The general steps in performing a search are pitch detection, segmentation into distinct notes, and searching for the tune in a database of known melodies. The necessity of a low-pass prefilter was also examined (chapter 2). Although prefiltering is common practice in most pitch detection applications, it was found that such prefiltering does not improve the pitch detection quality for single-voice music recordings. In some cases, prefiltering can also reduce the accuracy of the sgmentation process.

Pitch detection was performed using the super-resolution autocorrelation method (Medan et al., 1991), whereby autocorrelation pitch-detection is augmented by linear estimation, increasing the pitch resolution (chapter 3).

Segmentation was performed as a two-stage process (chapter 4). First, the recording was divided into segments based on volume, where high-volume segments are separated by regions of lower volume. Then, a region of constant pitch is identified within each segment, and this pitch value is identified as the segment's note.

The search itself is performed by finding the minimum edit distance between the recording notes and the database records (chapter 5). Several types of edit distances were used: Parsons code matching (Parsons, 1975), relative frequency matching, and combined frequency/duration matching.

Results and Conclusions

The algorithm was tested in two different approaches. Each section of the system was tested independently under laboratory conditions, by simulating standard operation of the other segments (chapter 6). Additionally, the system as a whole was tested in real-world conditions, using human recordings (chapter 7).

The Parsons code search, which is used in several existing applications, was found to be less error-robust than the new search algorithms tested in this report.

Execution time of the search process depends on the database size and recording duration. However, on a present-day personal computer and assuming reasonable conditions, execution time has been found to be shorter than the recording duration.Real-time implementation of the program is thus possible.

In a real-world test, using a database of 200 tunes, the software correctly identified some 79% of the queries. As a comparison, humans identified only about 53% of the same recordings. Simulations indicate that the program will work well for larger databases, if queries of about 12 notes are used.

פרויקט זה לא היה מגיע למקום שבו הוא נמצא היום לולא הדחיפה, העידוד והעזרה מאנשים רבים, איתם חלקתי את רעיונותי. בעזרתם, הפרויקט קרם עור וגידים, והפך מתחביב איתו שיחקתי בשעות הפנאי לתוכנה הפועלת באופן מרשים.

בראש ובראשונה, ברצוני להודות למנחה שלי, גל אשור, שהיה האדם המעורב ביותר בכל שלבי פיתוח הפרויקט. למרות לוח הזמנים הצפוף שלו הצליח גל למצוא זמן לענות במהירות רבה על דואר אלקטרוני והיה תמיד נכון להפגש איתי. נסיונו ורעיונותיו הפנו אותי, לא פעם ולא פעמיים, אל כיוון הפעולה האופטימלי, והפכו את הרעיון הערטילאי המקורי לפרויקט בר-ביצוע. לא פחות חשוב, עצותיו בנוגע לדרך הצגת הפרויקט אפשרו לי להדגים את פעולתו בצורה אפקטיבית ומשכנעת.

הרעיון לפרויקט היה נמוג כלעומת שבא, אלמלא תמיכת משפחתי. כאשר זרקתי, כבדרך אגב, את הרעיון לאוויר, לא עלה על דעתי כלל שאני אוכל לממש אותו. אמי מרים עודדה אותי להמשיך לחשוב על הנושא, והאמינה בישימותו גם כאשר לי היו ספקות. השיחות עם אבי יעקב, בכל שלבי העבודה, אפשרו לי לבחון רעיונות חדשים, ועצותיו היוו השראה לחלק מהרעיונות בתוכנה. תודתי הכנה נתונה לשניהם.

הייתי רוצה להודות גם לשאר בני משפחתי, איתן ורפי, וכן לשותפי למעונות, אלון בידרמן, דודו כהן, פז פולז ושלומי רובננקו, שנאלצו להקשיב למנגינות ששרתי למחשב בכל שעות היום והלילה, וקיבלו אותם בהבנה רבה, יחד עם קריאות השמחה והאכזבה המזדמנות.

תודתי נתונה גם לנמרוד פלג, שקשריו ונכונותו לעזור פתרו בעיות טכניות רבות, ולכל צוות המעבדה לעיבוד אותות.

עוד אנשים רבים תרמו לתוכנה, על ידי השאלת חוברות תווים לצורך הכנסתם למאגר המידע, ועל ידי שירה לבדיקת ביצועי התוכנה. על חוברות התווים אני מודה לגל אשור, דודו כהן, נעמה לילך, דורה פומרנצבלום וריטה רזניקוב. על השקעת הזמן, וההתגברות על המבוכה שבשירה למחשב, אני מודה לאורי איילון, גל אשור, רביב ברילר, ענבל ברעם, דודו כהן, בני משפחת לויטה – נועה, אורי, מיכל, רותי, רחל וחיים, נעמה לילך, עידו עוזיאל, פז פולק, אופיר קידר, שלומי רובננקו, ניצן רוזנפלד, ובני משפחתי – רפי, איתן, מרים ויעקב. לכולם תודה מקרב לב.

מבוא

יום אחד, לפני כשנתיים, שמעתי ברדיו שיר צרפתי שלא הכרתי. למרות שלא הבנתי את המילים, אהבתי את המנגינה, ולא יכלתי להפסיק לזמזם אותה. למרות זאת, גיליתי שאין דרך יעילה לגלות מה שם הזמר או השיר, בלי לדעת לפחות כמה משפטים מתוכו. לבסוף הצלחתי למצוא את מילות השיר באינטרנט (Été Indien של ג'ו דסין), אך רק לאחר שעות ארוכות שבהן ניסיתי לפענח כמה מילים מהשיר, בעזרת הידע הקלוש שלי בצרפתית.

אני בטוח שמצב דומה קרה כמעט לכל אדם, ולכן נראה לי מוזר שאין שיטה מסודרת ומקובלת לחיפוש שירים על פי מנגינותיהם בצורה אוטומטית. אחרי הכל, לא נראה שזו בעייה קשה בהרבה מאשר חיפושי טקסט, ובוודאי שאינה דורשת את הגמישות הרבה הנחוצה, למשל, לחיפוש וזיהוי תמונות.

החלטתי, לכן, לנסות לכתוב תוכנה שתקבל כקלט שאילתה בצורת זמזום של קטע משיר, תזהה את התווים שלו, ותחפש אותם בתוך מאגר מידע של מנגינות ידועות. הרעיון קסם לי מכיוון שהוא נראה לי מקורי ומעניין, ושילב שני תחומים בהם אני מוצא עניין רב: עיבוד אותות ותיאוריה של מוסיקה. מעבר לכך, קיימים גם שימושים פרקטיים למערכת כזאת: למשל, שירות ללקוחות בחנות מוסיקה.

במהלך העבודה נתקלתי במספר נסיונות קודמים לביצוע חיפוש של מנגינות. המוקדם ביותר, למיטב ידיעתי, הוא ספרו של D. Parsons (1975), שהוא למעשה קטלוג של עשרות אלפי מנגינות קלאסיות. לכל יצירה רשומים 15 התווים הראשונים, ועבור כל תו נכתב יחסו לתו הקודם: גבוה יותר, נמוך יותר או זהה. לדוגמה, התווים הראשונים של השיר "יונתן הקטן" יירשמו בתור "נ-ז-ג-נ-ז-נ-ג-ג-ג-ז-ז" (ג=גבוה יותר, ז=זהה, נ=נמוך יותר). רוב האנשים מסוגלים, עם קצת אימון, לזהות עליה או ירידה של תווים, כך שאין בעיה להפוך מנגינה ידועה לקוד כזה, גם ללא ידע במוסיקה. לאחר מכן מתבצע חיפוש בספר, שבו המנגינות מסודרות לפי סדר האלפבית של הקידוד. מבחינה מסוימת, החיפוש בספר זה דומה לחיפוש במילון.

למרות שפתרון זה הינו פשוט יחסית, הוא חסר את הנוחות של חיפוש אוטומטי, וכן אינו מתקן טעויות ברישום הקוד של האדם. עובדה זו מקבילה לכך שלא ניתן למצוא מילים במילון כאשר הן כתובות עם שגיאת כתיב, בעוד שתוכנות איות אוטומטי מתקנות שגיאות כאלה.

במהלך הפרויקט נתקלתי גם במאמרים שתיארו שיטות לביצוע החיפוש בדומה לצורה עליה חשבתי אני, בצורה אוטומטית ובעזרת מאגר מידע ממוחשב (Ghias et al., 1997; Prechelt & Typke, 1999; McNab et al., 1997). ההשוואה בין הרעיונות שלי לרעיונות של אנשים אחרים, שנתקלו באותם בעיות, הלהיבה אותי לא פחות מהתכנון המקורי של הפרויקט.

נסיונות אלה דחפו אותי לחשוב מחדש על חלקים של האלגוריתם שלי, והפנו אותי לפתרונות שלא חשבתי עליהם בעצמי. בחלק מהמקרים, שיפרתי את האלגוריתם שלי על ידי שילוב הרעיונות של אנשים אחרים; במקרים אחרים, בדקתי ומצאתי שהרעיונות שלי פעלו טוב יותר בתנאי אמת.

כפי שקורה פעמים רבות, הסתבר לי עם הזמן שהגדרת הבעייה היא פשוטה הרבה יותר מאשר מציאת הפתרון האופטימלי. במהלך העבודה נתקלתי במספר קשיים, והחלטתי להרחיב את הפרויקט לפרויקט שנתי, כדי שאוכל להשלים אותו באופן שיספק אותי. קושי אחד שגיליתי, אותו לא צפיתי במלואו, הוא המגוון הרחב של הזיופים והעיוותים הקיימים בשירה אנושית, שמהם המוח מתעלם באופן אלגנטי כאשר הוא מבצע זיהוי של שיר. היות והכוונה היא שהתוכנה תהיה נגישה גם עבור אנשים חסרי כל ידע בשירה, יש צורך באלגוריתם חיפוש שיוכל להתגבר על בעיות אלה.

בעייה נוספת שהתעוררה היא חיקוי היכולת של המוח לזהות את נקודת הזמן שבהן מסתיים תו אחד ומתחיל תו שני. להערכתי, המנגנון האנושי במקרה זה מורכב למדי ומתבסס על זיהוי מספר גדול של רמזים באות הדיבור, כגון זיהוי העיצור /ל/ בתחילת התו, כאשר שרים "לָה-לָה-לָה". בפרויקט ניסיתי להצמד לפתרונות פשוטים יותר, וכתוצאה מכך נוצרו מגבלות מסוימות על האופן שבו האדם יכול לזמזם את השאילתה.

למרות הקשיים שנקלעו בדרכי, ולמרות הלילות הארוכים של debugging, נהניתי מאוד במהלך כתיבת העבודה. גם עכשיו, כאשר התוכנית רצה ופועלת, ביצועיה מפתיעים אותי כל פעם מחדש. זו הפעם הראשונה בחיי שבה היה לי רעיון טוב והיתה לי ההזדמנות והאמצעים לממש אותו, ועל כך אני מרגיש בר-מזל.

צביקה בן-חיים
ספטמבר 1999

מונחים ויחידות

עבודה זו עוסקת בקשר שבין תכונות פיסיקליות-אקוסטיות של מידע קולי מוקלט, לבין תכונות פסיכולוגיות-מוסיקליות של מידע מוסיקלי. כיוון שאלה שני נושאים שהתפתחו בנפרד באופן היסטורי, היחידות והמונחים בהם שונים ומותאמים לשימושים הנפוצים באותו תחום. בפרק קצר זה נסקור בקצרה את הקשר התיאורטי שבין שני תחומים אלה, ונתאר באופן בסיסי את הקשר שבין אקוסטיקה למוסיקה (Helmholtz, 1887; Taylor, 1992; Levarie & Levy, 1968).

מידע קולי מתואר כאות המשתנה באופן רציף עם הזמן, ומתאר את לחץ האוויר בכל רגע. האוזן רגישה לשינויים בלחץ ומעבירה מידע זה למוח, שם הוא מפוענח ומתורגם למידע שמיש, באופן שאינו מובן עדיין במלואו. עם זאת, הקשר בין אותות מחזוריים לבין צלילים מוסיקליים הוא נושא המבוסס על מתמטיקה פשוטה יחסית. לכן, על ידי התייחסות למערכת הפענוח שבמוח כאל "קופסה שחורה", ניתן לנבא באופן מדויק את התגובה הסובייקטיבית של המאזין לקלטים פשוטים, כגון תווים ואקורדים.

ניסויים פשוטים מראים כי צורות גל מחזוריות גורמות לתחושה סובייקטיבית של צליל. מסיבות היסטוריות, צורת הגל הבסיסית בה משתמשים לרוב היא גל סינוס, אם כי צורות גל אחרות יוצרות גם הן תחושה (שונה) של צליל. התכונה החשובה ביותר של גל הסינוס היא התדר שלו, וניסויים מראים שככל שתדר הגל עולה, כך עולה התחושה הסובייקטיבית של "גובה הצליל", המכונה pitch.

כמו ברוב החושים האנושיים, שינויים בתדר הגל משפיעים באופן לוגריתמי על תחושת ה-pitch. לדוגמה, השינוי הסובייקטיבי (מבחינת pitch) במעבר מתדר של 100 הרץ ל200- הרץ שקול לשינוי ה-pitch במעבר מ200- הרץ ל400- הרץ. צורה אחרת לבטא עובדה זו היא שהאוזן אינה רגישה להפרשים בין תדרים, אלא ליחסים ביניהם (אשר מכונים מרווחים מוסיקליים). שתי הדוגמאות שלמעלה הינן בעלות יחס תדרים של 1:2, ולכן התחושה הסובייקטיבית היא של מרווח זהה ביניהן.

המרווח בעל היחס 1:2, כלומר, המרווח המבטא הכפלה של התדר הבסיסי, מכונה אוקטבה. במוסיקה המערבית, האוקטבה מחולקת ל12- מרווחים שווים, אשר מכונים סמי-טונים או חצאי-טונים. סמי-טון שווה ליחס של $1:2^{1/12}$ בין תדרי הצלילים.

12 התווים הנוצרים מחלוקה זו לסמי-טונים הינם, ברוב המקרים, התווים היחידים הקיימים במוסיקה מערבית (מלבד טרנספוזיציה של תו מסוים לאוקטבה גבוהה או נמוכה יותר). ואולם, נהוג לפעמים לכוון את התווים בצורות שונות, באופן שהם אינם בעלי מרווחים זהים במדיוק (Taylor, 1992). החוקים על-פיהם מכוונים תווים בשיטה מסוימת נקראים

tempering. השיטה שבה המרווחים בין כל שני תווים הם בדיוק סמי-טון נקראת equal tempering. יתרונה העיקרי הוא היכולת לנגן, בעזרת אותם 12 תווים, יצירות בכל סולם שהוא. לעומתה, שיטות אחרות, כגון just tempering, נחשבות יפות וטהורות יותר מבחינת איכות הצליל, אך דורשות סט שונה של תווים לכל סולם. מסיבה זו, שיטות tempering אחרות אינן נמצאות כמעט בשימוש בכלים שהכיוון שלהם מסובך, כגון פסנתר.

לצורך הגדרה של שינויים דקים כאלה, נהוג להשתמש ביחידת מרווח מוסיקלי המכונה סֶנט (cent), ושווה ל0.01- סמי-טון או ליחס תדרים של $1:2^{1/1200}$. ההבדלים בין תווים זהים ב- tempers שונים הינם עד כ4- סנט. אדם חסר ידע במוסיקה אינו מבחין בשינויים כאלה, וגם בזיופים חמורים יותר ניתן להרגיש רק במקרים מיוחדים.

1. תיאור כללי

פרק זה יתאר באופן כללי את מטרת הפרוייקט, אופן פעולת המערכת והתוצאות שהתקבלו. תיאור מפורט יותר של המערכת יוצג בפרקים הבאים.

1.1 מטרת הפרויקט

מטרת הפרויקט הינה יצירת כלי תוכנה, אשר יאפשר לאדם לבצע חיפוש במאגר מידע של מנגינות, על ידי הקלטת קטע מהמנגינה אותה הוא מחפש. לצורך ביצוע החיפוש אין צורך בידע כלשהו במוסיקה או בעיבוד אותות. הפרויקט בוצע כפרויקט שנתי בהיקף 8 נקודות אקדמיות. שלד התוכנה ואלגוריתם זיהוי ה-pitch נכתבו בחלק הראשון של הפרויקט, בעוד שהסגמנטציה ומנוע החיפוש נכתבו בחלקו השני.

בסעיף זה יפורטו דרישות פעולת המערכת המלאה, תוך התייחסות לתוכנה כאל "קופסה שחורה". אופן פעולת המערכת מבחינה פנימית יתואר בסעיף הבא.

1.1.1 קלט

הקלט של התוכנה הוא מידע קולי ספרתי (PCM), אשר משמש כשאילתה (query) עבור מאגר המידע. הקלט נמסר לתוכנה על ידי הקלטה בזמן אמת, או על ידי טעינת קובץ RAW קיים. קצב הדגימה הוא 8000 Hz, עם 16 ביט לדגימה, בהקלטה מונופונית.

ההקלטה היא של קול יחיד אשר מבצע מנגינה אותה התוכנה תזהה. ביישום סביר של התוכנה, המשתמש ייצור את ההקלטה בזמן אמת, כאשר הוא רוצה לבצע שאילתה, ולרוב לא יהיה מעוניין להשתמש בהקלטה שלו לצרכים אחרים. לכן, הגיוני לבקש מהמשתמש לבצע את ההקלטה באופן מסויים, כל זמן שדרישה זו אינה פוגעת משמעותית ביכולת השירה או בנוחות ביצוע השאילתה. בבדיקות שנערכו על הקלטות שונות נמצא כי צורת הקלט האופטימלית עבור סגמנטציה מבוססת-volume היא שירה שבה כל תו מתחיל בעיצור פורץ. הסגמנטציה היא יעילה במיוחד עבור העיצור /t/. לכן, הדרישה מהמשתמש היא לבצע את השירה באמצעות הגייה של ההברות "טָה", "טִי", "טָם", או שילוב ביניהם.

מידת הרעש הסביבתי היא כזו של סביבה משרדית שקטה. כפי שניתן לצפות, מספר סוגי רעשים מחזוריים מפריעים במיוחד לזיהוי ה-pitch, ביניהם רעש של מאוורר ושל מכונות. ההקלטות בוצעו במספר אתרים שונים, וברוב המקרים לא היתה בעייה של רעשים חיצוניים.

רוב ההקלטות בוצעו בעזרת מיקרופון זול המיועד להקלטת דיבור למחשב. מיקרופון כזה קולט קולות מאיזור מצומצם, וביצועיו היו טובים מאוד בשל כך.

מספר דוגמאות להקלטות תקינות, שמורות בפורמט RAW כמתואר למעלה, נמצאות על הדיסק המצורף לעבודה.

2.1.1 פלט

הפלט של המערכת כולל רשימה של מספר שירים, המתאימים ביותר למנגינה שבהקלטה. כמו כן, מוצג מדד טיב ("אחוזי התאמה") המאפיין את מידת ההתאמה של כל אחד מהשירים המוצגים אל ההקלטה. המנגינות מוצגות בסדר התאמה יורד. ניתן גם להציג מידע נוסף על השירים, בהתאם להיקף מאגר המידע; לדוגמה, שם האמן, שנת היצירה, וכו'. במאגר המידע שהוכן לצורך בדיקת והדגמת התוכנה, המידע הנכלל כולל את שם היצירה, שם המחבר ו/או המבצע, והמילים של אותו חלק מהשיר שנמצא במאגר המידע.

בנוסף לתוצאות החיפוש עצמן, מספקת התוכנה משוב לגבי איכות ההקלטה עצמה. מידע זה כולל סיווג של טיב ההקלטה בין שלוש רמות: איכות גבוהה, איכות בינונית או איכות ירודה; בנוסף, ניתנות הערות נקודתיות במקרה הצורך, אשר מיועדות לאפשר לאדם לשפר את השאילתה, במקרה שיחליט לחזור עליה. לדוגמה, אם השירה מבוצעת במהירות רבה מדי, התוכנה עשויה להמליץ לחזור על השאילתה בשירה איטית יותר.

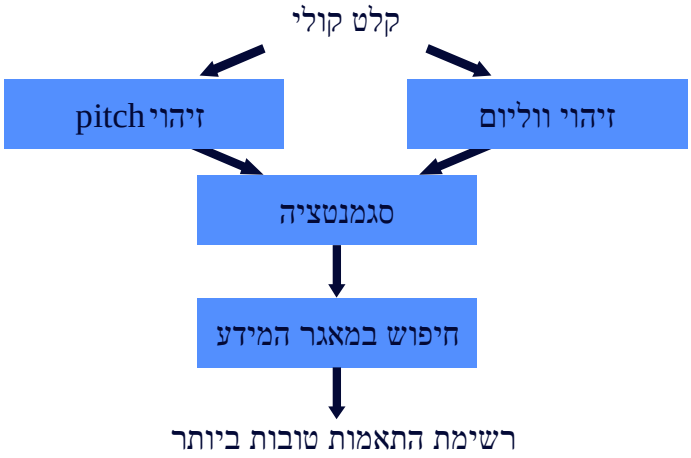
3.1.1 דרישות מערכת

התוכנה נכתבה ב-Microsoft Visual C++, גירסה 6.0. היא פועלת תחת Windows 9x או Windows NT 3.5/4, ללא צורך בחומרה נוספת מעבר לכרטיס הקלטה פשוט ומיקרופון. להפעלתה נדרשת גרסה מעודכנת של הקובץ MFC42.DLL, הנמצא על הדיסק המצורף לעבודה. התוכנה פועלת על כל מחשב Pentium, אך מומלץ להפעילה על מחשבי Pentium II ומעלה.

2.1 אופן פעולת המערכת

התוכנה מבוססת על מספר שלבי זיהוי, כמוצג באיור 1. מידע קולי מוקלט ומוכנס כקלט למערכת. בפרק 2 נבחן הצורך בסינון מקדים (prefiltering), כדוגמת סינון מעביר נמוכים. מוכח כי סינון כזה אינו תורם לזיהוי, ובמקרים מסוימים אף פוגע בו. לכן, זיהוי של התדר הבסיסי (pitch) מתבצע ישירות על המידע המוקלט. אלגוריתם זיהוי ה-pitch הינו super

resolution autocorrelation (Medan et al., 1991). אלגוריתם זה מאפשר חישוב מדויק של ה-pitch תוך חסכון ניכר בזמן חישוב. הרעיון הכללי בגישה זו הוא חישוב ה-pitch בהתבסס על שיטת האוטוקורלציה, בתוספת גורם תיקון המבוסס על שיערוך לינארי. תיאור מפורט של האלגוריתם מוצג בפרק 3. בנוסף, מתבצע זיהוי ה-volume כפונקציה של הזמן, נתון שבו משתמש אלגוריתם הסגמנטציה.



איור 1: תיאור כללי של פעולת המערכת

מתוך הנתונים (pitch ו-volume כפונקציה של הזמן), מתבצעת חלוקה למקטעים בתהליך המכונה סגמנטציה (פרק 4). המטרה בתהליך זה היא לבודד את התווים, כך שכל סגמנט מכיל תו בודד. החלוקה מתבצעת על ידי סימון קטעים בעלי volume גבוה יותר מהסביבה (סגמנטציה מבוססת-volume). בהמשך מזוהה, מתוך כל סגמנט, האזור שבו ה-pitch קבוע, ומתוך ערך pitch זה מחושב התו המתאים לאותו סגמנט. במקרה שהסגמנט אינו מכיל איזור pitch קבוע, הסגמנט מסומן כ"רעש" ומתעלמים ממנו. בסוף התהליך מתקבלת רשימה של תווים בעלי תדר ומשך-זמן ידועים.

לבסוף, רשימת התווים ואורכיהם מועברת למנוע החיפוש, אשר מוצא את ההתאמות הטובות ביותר לתווים המוקלטים מתוך אוסף של מנגינות מוכרות (פרק 5). מנוע החיפוש פועל בשיטת "מרחק עריכה" (edit distance), כלומר, הוא מוצא את המנגינה במאגר המידע שדורשת "עריכה" מינימלית על מנת שתהיה זהה למנגינת השאילתה. ניתן לבנות אלגוריתמי חיפוש שונים המבוססים על השוואה בין תכונות שונות של מנגינת השאילתה ומנגינות מאגר המידע. בעבודה זו נבחנו שלושה אלגוריתמים כאלה: אלגוריתם המבוסס על השוואת קודי Parsons, ונמצא בשימוש בכלים קיימים לחיפוש מוסיקה (Prechelt & Typke, 1999); אלגוריתם המבוסס על השוואת המרווחים המוסיקליים; ואלגוריתם המשלב השוואה בין המרווחים המוסיקליים ואורכי הצלילים.

3.1 תוצאות ומסקנות

מערכת הזיהוי נבדקה בשני אופנים. ראשית, נערכה סימולציה בתנאים מבוקרים של כל אחד מחלקי המערכת. סימולציה זו כללה יצירת כניסות מלאכותיות (מסונתזות) למערכת, ובדיקת נכונות התוצאות. שנית, נערכה בדיקה של התוכנה כמערכת שלמה. בדיקה זו התבצעה בתנאי אמת, ונערכה במגוון תנאים סביבתיים, על אנשים שונים ובאופני פעולה שונים.

להלן התוצאות העיקריות שהתקבלו מבדיקות אלה, והמסקנות הנובעות מהם:

- **אחוז הזיהוי:** בבדיקות בתנאי אמת נמצא כי התוכנה ביצעה זיהוי נכון של השיר בכ 79%- מהמקרים. ב10%- נוספים הזיהוי לא היה נכון, אך הופיע בין חמשת השירים בעלי הסתברות הזיהוי הגבוהה ביותר. לעומת זאת, כאשר הושמעו אותם שירים לבני אדם, הם הצליחו לזהות רק כ53%- מהשירים (הנבדקים נשאלו רק על שירים אותם הם הכירו). כלומר, ביצועי התוכנה, תחת התנאים שנבדקו, היו טובים בהרבה מהיכולת של אדם ממוצע לזהות שיר ע"פ מנגינתו.
- **שימוש בסינון מקדים:** נמצא כי סינון מקדים (prefiltering) אינו משפר את טיב זיהוי ה-pitch של קולות שירה אנושיים. זיהוי ה-pitch התבצע בצורה טובה עם או בלי סינון מקדים. במקרים מסוימים, נוצרו ארטיפקטים שהורידו את טיב הזיהוי דווקא באותות שעברו סינון מקדים. זאת בניגוד לגישה המקובלת של זיהוי pitch, שבה מתבצע סינון מעביר-נמוכים לפני הזיהוי (Medan et al., 1991).
- **אופן ביצוע השאילתה:** מבין שיטות השירה שנבדקו, נמצא כי השיטה המאפשרת זיהוי מדויק ביותר היא שירה באמצעות הגיית הברות המכילות עיצור פּוֹצֵץ בתחילתן, ובפרט ההברות "טָה" או "טִי". הברות אלה מכילות ירידה קצרה ב-volume בתחילתן, וירידה זו מאפשרת לבצע סגמנטציה יעילה ומדויקת יותר מאשר במקרים אחרים. שיטה זו מקובלת בתוכנה קיימת אחת (McNab et al., 1997), אך תוכנות אחרות משתמשות בשיטות שונות (Ghias et al., 1997; Prechelt and Typke, 1999).
- **אלגוריתם החיפוש:** בבדיקת אלגוריתמים שונים לחיפוש במאגר המידע, נמצא כי שיטת קוד Parsons (אשר נמצאת בשימוש לפחות בשתי תוכנות קיימות) יעילה פחות משיטות אחרות, שפותחו בפרויקט זה. בניגוד לשיטת Parsons, אשר מכילה כמות מינימלית של מידע על כל תו, מכילות שיטות חיפוש אלה מידע רב יותר, אך מתחשבות באפשרות שאדם יזייף. העובדה ששיטות אלה הן בעלות ביצועים טובים יותר מעידה על

כך, שאדם ממוצע שר מספיק טוב כדי להוציא מידע רב יחסית מהקלט, מידע שאינו נכלל בחיפוש Parsons.

- **מהירות החיפוש:** מהירות החיפוש תלויה בעיקר במשך זמן ההקלטה ובגודל מאגר המידע. נמצא כי על מחשב אישי בן-זמננו, ועבור גודל מאגר מידע סביר (עד 10,000 מנגינות בקירוב), זמן ביצוע החיפוש הינו קצר יותר ממשך זמן ההקלטה. עובדה זו מעידה על כך שניתן ליישם את התוכנה גם בזמן אמת, ובכל מקרה זמן ההמתנה לקבלת תשובה לשאילתה הינו סביר.

2. סינון מקדים

בעת ניתוח אות קולי, מקובל לבצע סינון מקדים (prefiltering) על האות, לפני תהליך הניתוח עצמו. הסינון מיועד לחזק מאפיינים ספקטרליים הדרושים לביצוע הניתוח, ולהנְחִית מאפיינים המפריעים לניתוח. מאפיינים אלה כוללים לא רק רעש סביבתי, אלא גם חלקים של האות שאינם דרושים לניתוח המסויים אותו אנו מעוניינים לבצע.

מכיוון שהניתוח מבוסס על זיהוי pitch, שהוא התדר הנמוך ביותר באות, הרכיבים הספקטרליים המעניינים אותנו יהיו בעיקר התדרים הנמוכים. הגיוני לצפות, לכן, שסינון מעביר נמוכים (LPF) ישפר את טיב הזיהוי, ואכן סינון מקדים כזה מקובל במערכות זיהוי pitch רבות.

ואולם, בפרק זה נביא ממצאים הרומזים על כך שבמקרה של זיהוי תווים, סינון מעביר נמוכים לא רק שאינו משפר את זיהוי ה-pitch, אלא אף גורם, במקרים מסוימים, לזיהוי פחות מדויק. תוצאות הבדיקות האלה מופיעות בסעיף 2.2 להלן. הסבר לתופעה מפתיעה זו מופיע בסעיף 3.2.

בעקבות ממצאים אלה, הוחלט שלא להשתמש בסינון מקדים בתוכנה הסופית, על אף העובדה שסינון מעביר נמוכים הוא סינון מקדים מקובל יחסית בניתוחי זיהוי pitch.

1.2 סקירת ספרות

התפתחות שיטות זיהוי pitch מתוארות בהרחבה בפרק 3. נזכיר כאן כמה עובדות הרלוונטיות לעניין הסינון המקדים.

השיטות המוקדמות ביותר לזיהוי pitch מכונות בעבודה זו "שיטות נקודתיות". שיטות אלה כוללות את שיטת ה-Zero Crossing Frequency ואת שיטת ה-Peak Detection. ייחודן היה, שהן התבססו על זיהוי תכונות נקודתיות של האות במרחב הזמן. בשיטת ה- Zero Crossing Frequency, למשל, נמדד הקצב שבו חותך האות את ציר הזמן. באות סינוס נקי, הקצב הזה הוא כפול מהתדר של הסינוס. ואולם, אם נוסיף לאות רעש בתדר גבוה, קצב החיתוך עם ציר הזמן יגדל, למרות שה-pitch לא השתנה (Gold, 1962).

השיטות האלה פועלות היטב רק במקרים של אותות מאוד נקיים, כגון אות סינוסואידלי, ולכן היה צורך בסינון להוצאת רכיבים ספקטרליים לא רצויים, ובעיקר רעשים בתדר גבוה. סינון כזה היה נפוץ מאוד באלגוריתמים לזיהוי pitch בשנות השישים והשבעים (Gold, 1962; Beauchamp, 1969; Moorer, 1975). עם זאת, לא היתה דעה אחידה בנושא תדר

הקטעון הטוב ביותר עבור מסנן זה, וגם בשאלת הצורך בסינון מרכיבי תדר נמוך מאוד, בנוסף לסינון מעביר גבוהים.

עם השיפור במהירות המחשבים של העשורים האחרונים, התאפשר מעבר לשיטות זיהוי pitch מדויקות יותר, שאינן מבוססות רק על מאפיין נקודתי של האות אלא על המחזור כולו. אלגוריתמים אלה כוללים, בין היתר, זיהוי המבוסס על אוטוקורלציה, תכונות התדר של האות, או תכונות ה-cepstrum.

גם בשיטות אלה, נהוג להשתמש בסינון מקדים כלשהו, מתוך הנחה שהוא משפר את טיב הזיהוי (Medan et al., 1991). ואולם, מכיוון ששיטות אלה אינן מסתמכות על נקודה אחת במרחב הזמן לקבלת החלטה על אורך ה-pitch, הן הרבה פחות רגישות לרעש. מכך עולה השאלה: האם אכן נחוץ לבצע סינון מקדים על האות, לפני זיהוי ה-pitch, גם בשיטות הזיהוי החדשות? והאם סינון מקדים עשוי לפגוע בטיב הזיהוי, במקרים מסוימים?

בהמשך הפרק ייבדקו שאלות אלה, ויימצאו עדויות לכך שעבור הקלטות לצורך זיהוי מוסיקה, בתנאי הקלטה סבירים, סינון מקדים אינו נחוץ לקבלת זיהוי נכון של האות, ומבחינות מסוימות אף פוגע בטיב הזיהוי.

2.2 בדיקת טיב הסינון המקדים

בסעיף זה נתאר בפירוט את הבדיקות שנעשו כדי להשוות בין זיהוי pitch עם סינון מקדים, לבין זיהוי pitch ללא סינון מקדים. סיכום קצר של הממצאים העיקריים יתואר בסעיף הבא.

1.2.2 תיאור כללי

כדי לבדוק את הנחיצות של סינון מקדים, נערכה השוואה של תוצאות זיהוי ה-pitch עבור אות כניסה זהה, בשני מצבים. במצב הראשון זיהוי ה-pitch התבצע ישירות על האות, ללא כל סינון מקדים; במצב השני התבצע סינון מקדים בעזרת מסנן מעביר-נמוכים.

זיהוי ה-pitch התבצע בשיטת super-resolution autocorrelation, שתתואר בפירוט בפרק 3. זו השיטה שנבחרה לזיהוי pitch עבור התוכנה הסופית. שיקולי הבחירה של שיטה זו על פני שיטות אחרות, כגון זיהוי תדרים במישור פורייה, מתוארים בפרק הבא.

הסינון המקדים התבצע בעזרת מסנן FIR מסדר 100 שתוכנן על ידי הכפלת תגובת ההלם האידיאלית בחלון Han. תדר הקטעון של המסנן הינו 800 הרץ. תדר זה נמצא מעל לתדרי ה-pitch הגבוהים ביותר (כ600- הרץ), על מנת לוודא שאין ניחות של מידע חשוב. עצמת

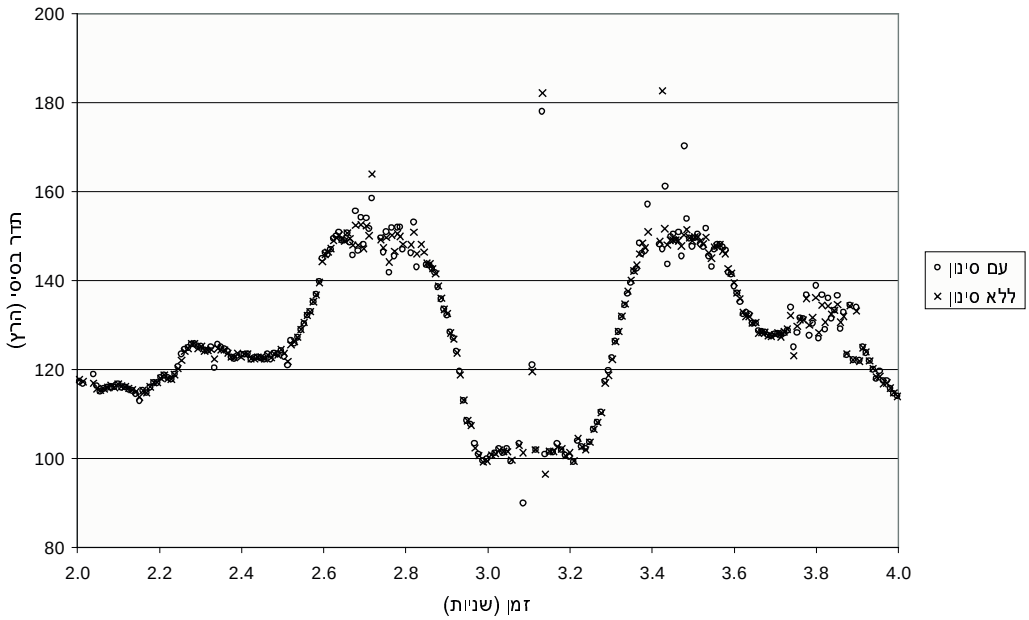
הניחות בתדרים מעל 1000 Hz הינה לפחות 30 dB ביחס לתדר DC. גליות ההעברה בתדרים נמוכים מ-700 Hz הינה לכל היותר 0.5 dB. המסנן הינו בעל פאזה לינארית והשהייתו היא 50 דגימות, או כ6- מילישניות.

לכל הקלטה בוצע זיהוי pitch בשתי שיטות: עם סינון וללא סינון. תוצאות הזיהוי נשמרו בקובץ כרצף של אורכי pitch בצירוף זמני הזיהוי. תוצאות הזיהוי לאחר סינון הוזזו 50 דגימות אחורה בזמן, על מנת לפצות על ההשהייה של המסנן.

שיטת הזיהוי, שתתואר בפרק 3, הינה pitch-synchronous tracking, ולכן תוצאות ה-pitch הקודמות משפיעות על נקודת הזמן שבה חל זיהוי ערך ה-pitch הבא. כתוצאה מכך, יש לצפות שזמני הזיהוי יהיו שונים עבור האות המסונן והלא-מסונן.

כדי שתתאפשר השוואה בין שני האותות, למרות ההבדלים בנקודות הזמן, התבצעה אינטרפולציית cubic spline, לקבלת ערכי ה-pitch של האות המסונן בנקודות הזמן של האות הלא-מסונן.

לבסוף, התקבלו שני וקטורים של ערכי pitch, באותן נקודות זמן, המתארים את ה-pitch עם סינון וללא סינון. על הווקטורים האלה התבצעו מספר השוואות, כמתואר להלן.



איור 2: השפעת סינון מקדים על זיהוי pitch

2.2.2 בדיקה על הקלטות קול אנושי

הבדיקה הראשונה התבצעה על שלוש הקלטות של שירה, אשר מכסות מספר סוגי שירה שונים. ההקלטות בוצעו על ידי שלושה אנשים שונים. קטע אחד הכיל המהום, ושני הקטעים האחרים הכילו שירה ללא מילים, באמצעות הגיית "לָה לָה" על מנת לבצע את היצירה. הקטעים היו של שלוש יצירות שונות. ההקלטות בוצעו בתנאי רעש סבירים, כמתואר בדרישות הקלט בסעיף 1.1.1. אורכם הכולל של שלושת הקטעים היה כ30- שניות.

קטע זיהוי pitch, מתוך אחד האותות האלה, מוצג באיור 2. האיור מציג את זיהוי ה-pitch בשתי השיטות (עם סינון מקדים וללא סינון מקדים). שוני בין שתי התוצאות מצביע על קיומו של הבדל בין שתי השיטות. ניתן לראות איכותית מגרף זה כי ההבדלים בין שתי השיטות הם מועטים, וזיהוי ה-pitch מתבצע בצורה טובה על שני האותות.

כדי לבדוק תוצאה זו בצורה כמותית, התבצע ניתוח ב-Matlab על אותות אלה. הניתוח כלל מספר שלבים. ראשית, הוגדרו ידנית הגבולות שבתוכם נעו הערכים האמיתיים של ה-pitch . במקרה של איור 2, גבולות אלה היו 95-155 הרץ. כל ערך שהיה מחוץ לגבולות אלה נחשב לזיהוי שגוי. תוכנית הניתוח מדדה את מספר המקרים שבהם חל זיהוי שגוי רק באחד האותות, בעוד שבאות השני הזיהוי לא היה שגוי. מספר זה מאפיין את מידת הנטייה של האלגוריתם לזהות באופן שגוי מידע שניתן היה לזהות נכונה עם אלגוריתם אחר. מקרה של זיהוי שגוי ללא סינון, שהפך לזיהוי נכון לאחר סינון, יכונה "תיקון"; מקרה הפוך, שבו זיהוי נכון ללא סינון הופך לזיהוי שגוי לאחר הסינון, יכונה "קלקול". תוצאות החישוב מתוארות בטבלה הבאה. הנתונים מוצגים באחוזים מתוך כלל ערכי ה-pitch שנמצאו עבור אותו קטע.

	"קלקולים": זיהויים שגויים לאחר סינון שהיו תקינים לפני הסינון	"תיקונים": זיהויים שגויים לפני סינון שנהיו תקינים לאחר הסינון
קטע א'	5.2%	2.4%
קטע ב'	3.2%	0.5%
קטע ג'	1.9%	1.5%

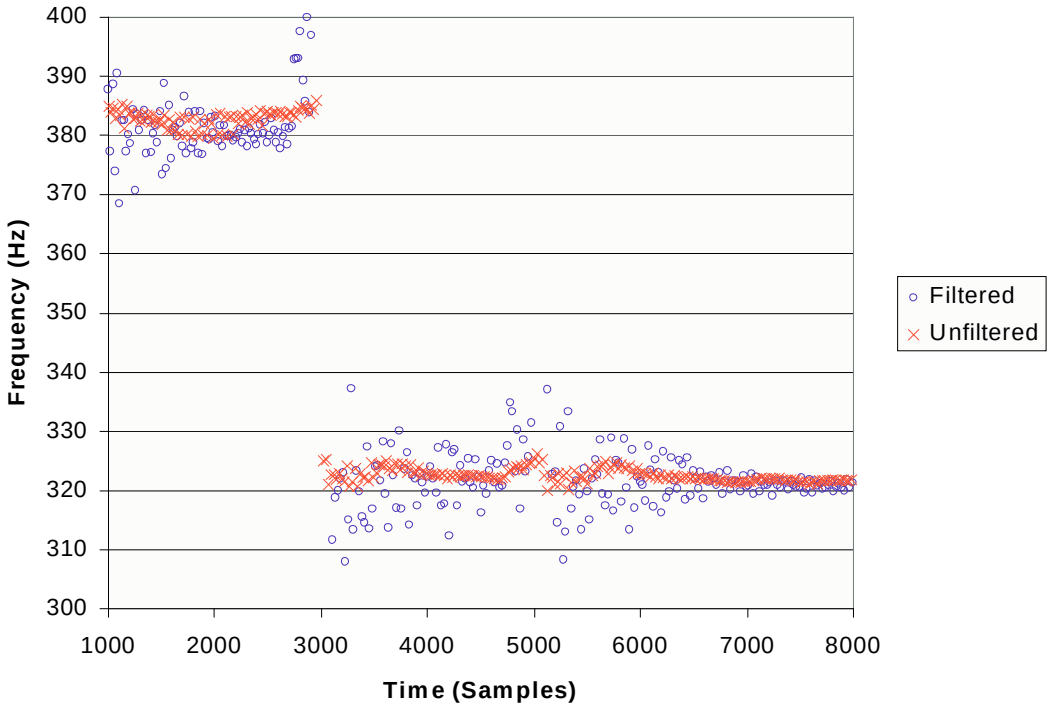
בנוסף, נמדדו הבדלים בין האלגוריתמים בנקודות שבהן לא חל זיהוי שגוי. במקרים אלה חושב וקטור ההפרש בין ערכי הזיהוי של שני האלגוריתמים. מתוך וקטור זה חושב הממוצע (אשר יכול להצביע על מגמתיות של אלגוריתם אחד ביחס לשני) וסטיית התקן (אשר מצביעה על מידת ההתאמה בין שני האלגוריתמים). תוצאות אלה מוצגות בטבלה הבאה.

	ממוצע ההפרשים בין האלגוריתמים (Hz)	סטיית התקן של ההפרשים בין האלגוריתמים (Hz)
קטע א'	0.075	3.0
קטע ב'	0.012	2.6
קטע ג'	0.008-	3.1

מהתוצאות ניתן לראות כי ההפרשים בין שתי השיטות יוצרים משתנה אקראי בעל תוחלת 0 בקירוב, ובעל סטיית תקן נמוכה (כרבע טון). תוצאות אלה מעידות על כך שההבדל בין שיטות הזיהוי אינו גדול.

3.2.2 בדיקה על הקלטות מפסנתר

עד עתה בדקנו את זיהוי ה-pitch על הקלטות של קולות אנושיים. אך בקולות כאלה מידת אי-הוודאות רבה: לדוגמה, אין אנו יודעים מה ה-pitch הנכון, ואנו יכולים רק להשוות בין ערכי pitch שהתקבלו בשיטות השונות.



איור 3: זיהוי pitch מהקלטה של פסנתר

כדי לקבל השוואה שבה נוכל לשלוט על הפרמטרים בצורה מדויקת יותר, התבצעה הקלטה של פסנתר, ועליה בוצעה השוואת ערכי pitch. הקול בפסנתר נובע מתנודות במיתרים שאורכם קבוע. לכן, כאשר מנגנים תו מסוים, ה-pitch נשאר קבוע, ואינו תלוי בזמן או בעצמת הניגון. ידע מוקדם זה מאפשר לנו לדעת איזו שיטה היא המדויקת יותר, ולא רק למצוא הבדלים בין השיטות.

תוצאת ניתוח ה-pitch של קטע קצר בפסנתר מוצגת באיור 3. גרף זה מציג את ה-pitch המזוהה בשתי השיטות, כפונקציה של הזמן. קטע הזמן מכיל 3 תווים, כאשר התו השני והשלישי זהים זה לזה מבחינת pitch.

מהגרף ניתן לראות כי זיהוי ה-pitch לאחר סינון הוא הרבה פחות יציב. בעוד שאנו מצפים לקבל קו ישר כל זמן שאותו תו מנגן, מתקבלות תנודות רבות לאורך התו, ובמיוחד באיזורי המעבר בין תווים. לעומת זאת, זיהוי ה-pitch ללא סינון מקדים הינו מדויק הרבה יותר, ונשאר יציב מאוד ביחס לזיהוי המסונן.

3.2 דיון ומסקנות

1.3.2 הדמיון בין שתי שיטות הזיהוי

התוצאות המוצגות בסעיף הקודם מטילות ספק בצורך בסינון מקדים לקבלת זיהוי pitch מדויק. התבוננות בתוצאות המוצגות באיור 2 מצביעה על דמיון רב בין הזיהוי בשני המקרים. דמיון זה מתבטא גם בעובדה שממוצע ההפרשים בין האלגוריתמים קרוב מאוד לאפס, וסטיית התקן של ההפרשים הינה נמוכה (הבדל של כ2%- בין האלגוריתמים).

תופעה זו נובעת מכך שחישוב ה-pitch מתבצע על ידי מציאת מקסימום של פונקציית האוטוקורלציה, בטווח נתון של ערכי pitch אפשריים. אך פונקציית האוטוקורלציה עצמה כוללת סכום על מספר גדול של איברים, כשכל איבר הוא מכפלה של שתי דגימות. בתור שכך, יש לה אפקט החלקה של האות, הדומה למסנן מעביר נמוכים. עבור ערך pitch אופייני של 100 הרץ, מתבצע סיכום על פרק זמן של 0.01 שניה. לשם השוואה, מסנן לינארי המבצע אינטגרציה על קטע באורך 0.01 שניה הוא בעל תדר קטעון (3 dB) של 870 הרץ. לכן, סינון בעל תדר קטעון דומה לא יוריד רעשים מלבד אלה שתהליך האוטוקורלציה "מסנן" באופן פנימי.

2.3.2 ההשפעה המזיקה של סינון מקדים

למרות שבאופן בסיסי שתי השיטות פועלות היטב, נמצאו מקרים שבהם הסינון המקדים פוגם בטיב הזיהוי. עובדה זו בלטה במיוחד במקרה של זיהוי pitch של פסנתר, שבו

ידוע שה-pitch קבוע כל זמן שמנגנים תו בודד. אלגוריתם הזיהוי ללא סינון מאשר תוצאה זו, ומזהה pitch קבוע למקוטעין המשתנה באופן חד במעברים מתו אחד לשני. לעומתו, אלגוריתם הזיהוי לאחר סינון נותן תוצאות לא קבועות, במיוחד בקטעי המעבר בין תווים שונים.

תופעות "קלקול" התגלו גם במקרים נדירים בקולות שירה אנושיים. למרות שבכל שלושת הקטעים שנבדקו היו יותר מצבי "קלקול" מאשר מצבי "תיקון", שתי התופעות היו נדירות יחסית, ולכן ניתן להטיל ספק במשמעותיות של תוצאות אלה.

הסבר אפשרי לתופעות הנזק של סינון מקדים הוא הטיפול במצבי מעבר בין תדרים. במצבים אלה, מסנן מעביר נמוכים מבצע סכימה על חלון הכולל את שני התדרים, ובכך גורם ליצירת אות הכולל את שני התדרים, עובדה שעלולה לפגוע ביעילות זיהוי ה-pitch. לעומת זאת, אלגוריתם האוטוקורלציה עצמו מבצע סכימה רק על מחזור אחד של ה-pitch, ולכן השילוב של שני תדרים שונים יכול להתרחש לכל היותר במדידת pitch אחת.

3.3.2 סיכום

בפרק זה ראינו עדויות לכך שסינון מעביר-נמוכים אינו משפר את איכות זיהוי ה-pitch, ולעיתים אף מוריד את איכות הזיהוי. הפגיעה העיקרית בטיב הזיהוי מתרחשת באיזורי מעבר חדים בין תווים. כדי לקבוע בוודאות את היתרונות והחסרונות של סינון מקדים, יש צורך בבדיקה מקיפה יותר, ובעיקר במדגם רחב יותר של אותות קלט.

עם זאת, היכולת להבחין בצורה מדויקת בנקודת המעבר בין pitch אחד לבא אחריו היא תכונה חשובה של המערכת, אשר בשלבים מאוחרים יותר תצטרך לזהות מצבים כאלה כמעבר מתו אחד לשני. פגיעה בחדות הזיהוי באיזורים אלה עלולה לפגוע בטיב הסגמנטציה. לכן, בעבודה זו הוחלט שלא להשתמש בסינון מקדים לפני זיהוי ה-pitch.

3. זיהוי Pitch ו-Volume

פרק זה יעסוק בשלב הראשון במעבר ממידע קולי למידע מוסיקלי: זיהוי ה-pitch וה-volume. ה-pitch הוא התדר הנמוך ביותר בקול, והוא קובע את גובה הצליל הסובייקטיבי אותו אנו שומעים. זיהוי נכון של תדר זה הוא חיוני להצלחת תהליך הפענוח של המידע המוסיקלי, ולכן חשוב שהזיהוי יהיה מדויק ויעיל. בפרק זה ייבחנו מספר שיטות מקובלות לזיהוי pitch. יתרונותיהן וחסרונותיהן של השיטות השונות יוצגו, תוך שימת דגש על סיבוכיות חישוב, דיוק, והתמודדות עם בעיות ספציפיות. לבסוף ייבחר האלגוריתם שישמש לזיהוי pitch בתוכנה הסופית.

זיהוי pitch הינו אחת הבעיות המוקדמות ביותר בתחום עיבוד קול ממוחשב, ובתחילת שנות ה60- כבר היו מקובלות מספר שיטות זיהוי. השיטות הראשונות לזיהוי pitch נבנו על הצורך באלגוריתמים מהירים וחסכוניים, שיאפשרו זיהוי pitch בזמן סביר, גם על המחשבים של אותה תקופה. אלגוריתמים אלה יכונו כאן "אלגוריתמים נקודתיים".

מאוחר יותר, עם ההתפתחות המחשבים, חל מעבר לשיטות מתוחכמות יותר, שהיו חסינות יותר לרעשים. בין שיטות אלה נתאר כאן שיטות המבוססות על מעבר למרחב התדר ושיטות המבוססות על חישובי אוטוקורלציה.

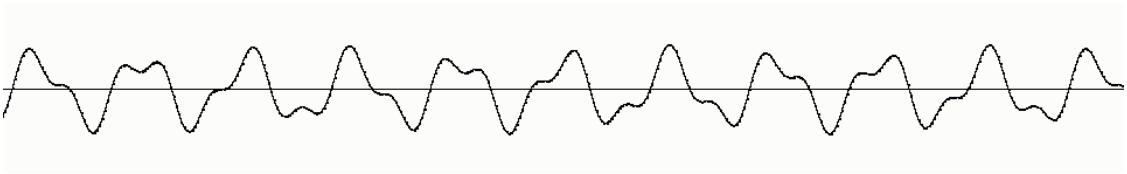
מלבד זיהוי ה-pitch, נדון בפרק זה גם באופן זיהוי ה-volume (עצמת הקול). זיהוי זה חשוב מאוד לתהליך הסגמנטציה (פרק 4). שיטת חישוב ה-volume היא פשוטה יחסית לזיהוי ה-pitch, וגם מגוון שיטות החישוב מצומצם יותר. לכן, תיאור שיטת זיהוי ה-volume יהיה קצר יותר.

1.3 שיטות נקודתיות

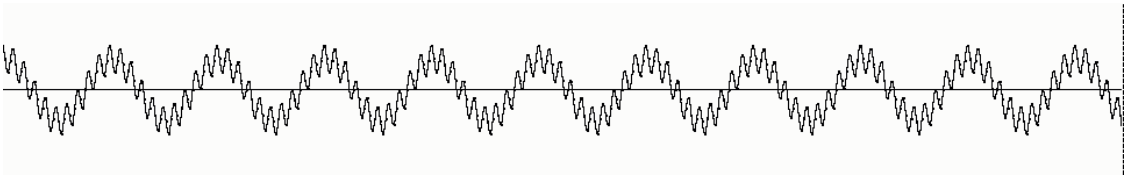
בשם הכולל "שיטות נקודתיות" נכנה כל שיטה שבה זיהוי זמן המחזור מתבסס אך ורק על מספר קטן של דגימות מתוך המחזור הכולל. מטבע הדברים, שיטות אלה הינן רגישות לרעש יותר מאשר שיטות מתוחכמות יותר, כיוון שטעות בדגימה אחת יכול לשנות לחלוטין את תוצאות הניתוח. עם זאת, מספר שיטות כאלה היו מקובלות בשנות ה60-, וזאת בגלל הסיבוכיות הנמוכה שלהן, שאיפשרה חישובים בזמן אמת גם על המחשבים של אותה תקופה. עם השיפור ביכולות המחשבים, נזנחו שיטות אלה, ולכן נסקור אותן בקצרה בלבד.

אחת השיטות הפשוטות ביותר לזיהוי pitch היא "תדר חיתוך האפס" (Zero Crossing Frequency, ZCF) (Gold, 1962; Beauchamp, 1969; Moorer, 1975). בשיטה זו, נמדד הקצב

שבו חותך האות את ציר הזמן, כלומר, מספר המעברים מאמפליטודה גבוהה לנמוכה או להיפך. עבור גל סינוס מושלם, תדר זה הוא כפול מהתדר של גל הסינוס. השיטה פועלת באופן מקורב גם עבור צורות גל פשוטות נוספות (איור 4). ואולם, עבור אותות בעלי רכיבים בתדר גבוה, ייתכן מצב שבו ה-ZCF מזהה תדר גבוה בהרבה מתדר ה-pitch, גם אם אמפליטודת רכיב התדר הגבוה היא נמוכה בהרבה מאפליטודת ה-pitch (איור 5).



איור 4: דוגמה שבה אלגוריתמים נקודתיים פועלים היטב



איור 5: דוגמה שבה אלגוריתמים נקודתיים אינם פועלים היטב

שיטה אחרת הינה זיהוי pitch על סמך נקודות קיצון של האות (Peak Detection). כל נקודה שבה האות עובר ממצב עלייה למצב ירידה, ואשר נמצאת מספיק רחוק מאמפליטודת האפס, נקראת נקודת קיצון. בגל סינוס מושלם ישנן שתי נקודות קיצון, אך הבעייה נוצרת כאשר נוספים רכיבי תדר גבוה, ואז מספר נקודות הקיצון גדל. באותות כגון איור 4 ניתן, לרוב, להתגבר על הבעיה, על ידי הגדרה נכונה של האמפליטודה הדרושה לנקודת קיצון ; אך באיור 5 יתקבל ערך pitch גבוה מהערך האמיתי.

בגלל הרגישות של אלגוריתמים אלה לרעשים בתדר גבוה, היה מקובל להשתמש במסנן מעביר נמוכים כסינון מקדים, לפני הפעלת אלגוריתם זיהוי ה-pitch. תיקונים נוספים נעשו על ידי שימוש בתכונות ידועות על מערכת הקול של האדם, כגון טווח ערכים אפשרי וקצב שינוי אפשרי של ה-pitch. למרות כל זאת, הדיוק של האלגוריתמים היה מוגבל, ולכן עם השיפור במהירות המחשבים חל מעבר לזיהוי pitch על סמך שיטות לא-נקודתיות.

2.3 שיטות מבוססות תדר

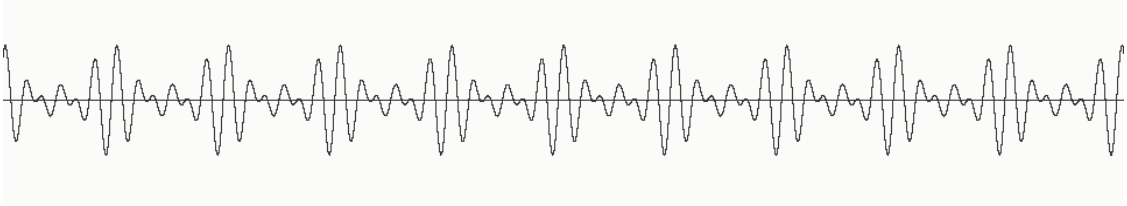
שיטה אחרת לזיהוי pitch מבוססת על זיהוי הרכיבים הספקטרליים של האות. ניתן לבצע התמרת פורייה בחלון-זמן קצר (short-time Fourier transform, STFT), ולקבל בכך מידע על הספקטרום של האות בקטע זמן קצר, שבו ה-pitch הוא קבוע בקירוב. ניתן, לכן, לזהות את ה-pitch בתור התדר הנמוך ביותר שבו יש מקסימום מקומי של העוצמה הספקטרלית (Terhardt et al., 1982).

שיטה זו הינה חסינה לרעש יחסית לשיטות דומות, מכיוון שרעשים מוקלטים הינם בד"כ בתדרים גבוהים מתדרי ה-pitch, ולכן אינם מפריעים לניתוח. יתרון נוסף של הפירוק הספקטרלי הוא שניתן להפריד בין צירופי צלילים בתדרים שונים. עובדה זו מאפשרת לזהות תווים גם כאשר הם מורכבים ממספר קולות (כגון אקורדים).

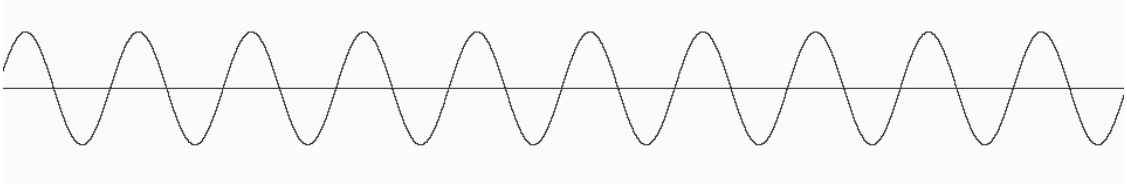
עם זאת, שיטת הזיהוי הספקטרלית יוצאת מתוך נקודת הנחה סמויה, והיא שפירוק של האות לגלי סינוס הינה הדרך האופטימלית לתאר את התחושה הסובייקטיבית של "גובה צליל", אשר מכונה במוסיקה pitch. הנחה זו השתרבבה כבר לתוך המחקרים הראשונים באקוסטיקה (Helmholtz, 1885). במחקרים אלה יוצרו אותות סינוסואידליים באמצעים מכניים, ונמצא הקשר הלוגריתמי בין תדר לצלילים בפסנתר. מכאן ועד לטענה, כי כל שילוב של גלי סינוס תיצור תחושה של שילוב של מספר צלילים, המרחק קצר.

הופעתו של הסינתיסייזר, שמאפשר שליטה מלאה על צורת הגל המושמעת, הראתה כי המציאות מורכבת הרבה יותר. באמצעות שילוב נכון של גלי סינוס בתדרים שונים ניתן ליצור תחושות גובה צליל אשר מתאימות לגל סינוס טהור בתדר אחר לגמרי. אחת הדוגמאות המוכרות של תופעה זו זכתה לשם "missing fundumental", והיא מופיעה לא רק באותות מסונתזים אלא גם בכלי נשיפה מסוימים (Taylor, 1992). במצב זה, מופיעות מספר הרמוניות של תדר מקורי f; לדוגמה, גלי סינוס בעלי תדרים $4f, 5f, 6f$. כאשר מושמעים תדרים אלה, ניתן לשמוע ברקע צליל נמוך, שתחושת הגובה הסובייקטיבית שלו זהה לזו של גל סינוס בתדר f - למרות שפירוק פורייה של אות זה אינו מכיל כלל את תדר היסוד.

לא קיימת תיאוריה יחידה אשר מנבאת בהצלחה, עבור כל אות כניסה, את תחושות ה-pitch שיתקבלו למאזין (Moorer, 1975). עם זאת, התיאוריות המוצלחות יותר מבוססות על מציאת מחזוריות באות בתחום הזמן. לדוגמה, אות בעל תדרים $4f, 5f, 6f$ הוא בעל מחזוריות של $1/f$ בתחום הזמן (איור 6) - בדיוק כמו אות בעל תדר f בלבד (איור 7).



איור 6: שילוב של 3 סינוסים (בתדרים $4f, 5f, 6f$)



איור 7: גל סינוס נקי (בתדר f)

בגלל הסיבות שהוצגו לעיל, השימוש בשיטות מבוססות תדר לזיהוי צלילים מוגבל למצבים שבהם יש צורך לזהות מספר קולות בו-זמניים. במצבים של קול יחיד, ובפרט במצבי זיהוי pitch של אותות דיבור או שירה, נעשה שימוש כמעט בלעדי של שיטות מבוססות זמן.

3.3 אוטוקורלציה

1.3.3 תיאור האלגוריתם

מרבית שיטות הזיהוי המודרניות המסתמכות על עיבוד בציר הזמן מבוססות על חישוב פונקציית האוטוקורלציה של קטעים מהאות. ישנן מספר הגדרות שונות לפונקציה זו, ונובעים מהם אלגוריתמים שונים במקצת. נציג כאן אלגוריתם המכונה pitch-synchronous tracking, בעקבות Medan et al., 1991.

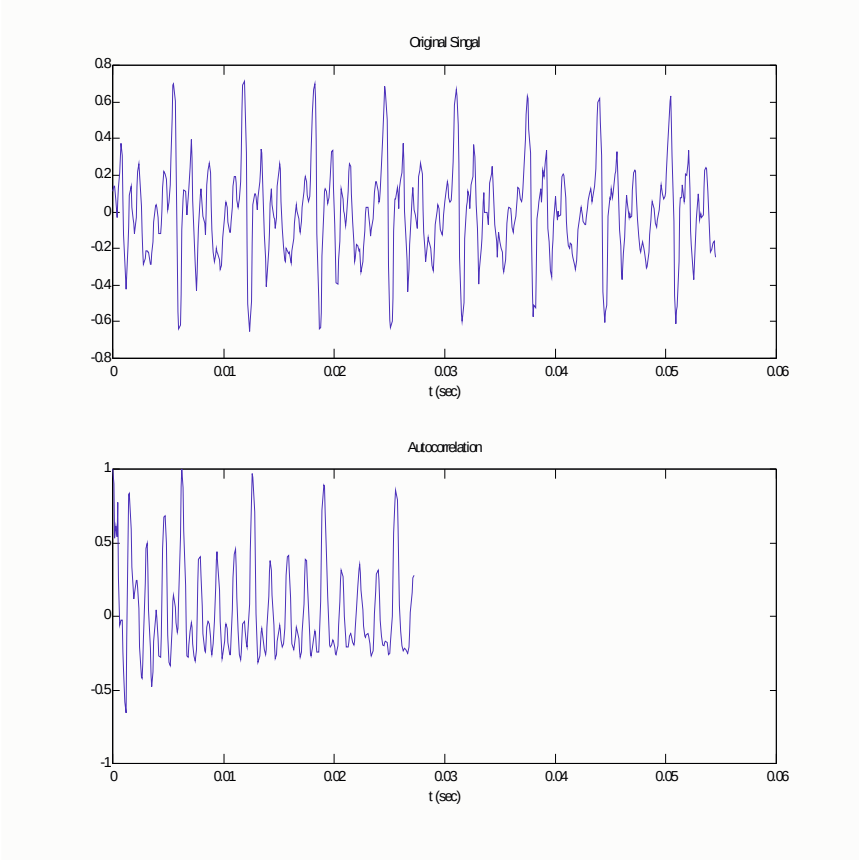
נסמן ב- $x[n]$ אות דגום בעל תוחלת אפס, עבורו רוצים למצוא את ה-pitch. אנו מניחים כי האות הוא קווזי-סטציונרי, כלומר האות הוא סטציונרי בקירוב עבור קטעי זמן קצרים. נגדיר את האוטוקורלציה במרווח p כלשהו (מספר שלם של דגימות) באופן הבא:

$$r[p] = \frac{\sum_{n=1}^{p} x[n] \cdot x[n+p]}{\sqrt{\sum_{n=1}^{p} x^2[n]} \sqrt{\sum_{n=1}^{p} x^2[n+p]}}$$

הנוסחה משווה בין שני קטעים עוקבים של האות: $x[1,\ldots p]$ ו- $x[p+1,\ldots 2p]$. הביטוי במכנה הינו גורם נירמול המבטיח כי התוצאה תהיה בין -1 לבין 1.

האוטוקורלציה היא 1 אם ורק אם ישנו קשר לינארי בין שני הקטעים, והמקדם המקשר ביניהם הוא חיובי, כלומר $x[n+p]=\alpha\cdot x[n],\quad \alpha>0,\quad 1\le n<p$. האוטוקורלציה היא -1 אם ורק אם ישנו קשר לינארי בין האותות עם מקדם שלילי. בכל מקרה אחר תתקבל תוצאה בין -1 לבין 1, כאשר ערך קרוב לאפס מציין שהקטעים הם בלתי-תלויים לינארית, וערך קרוב ל 1- (בערך מוחלט) מציין קשר כמעט לינארי.

חישוב pitch בעזרת אוטוקורלציה מסתמך על העובדה שעצמת תדר ה-pitch היא גבוהה בהרבה מעצמת התדרים הגבוהים יותר, כך שברוב המקרים מתקבל אות מחזורי בקירוב. עבור אות כזה, יש קשר לינארי בקירוב בין חלקים עוקבים של האות, אם ורק אם המרווח ביניהם הוא כפולה שלמה של זמן המחזור (שהוא אורך ה-pitch). לכן, השיאים הגבוהים ביותר בגרף של $r[p]$ כפונקציה של p מעידים על ערך ה-pitch (איור 8).



איור 8: דוגמה לחישוב pitch באמצעות אוטוקורלציה

2.3.3 בעיות יישום ופתרונן

כדי לזהות את ה-pitch יש לזהות את השיא בגרף האוטוקורלציה, אשר מציין את תדר המחזוריות ה"מובהק" ביותר. בעת הנסיון להגדיר אלגוריתם לבחירת המחזוריות, נוצרות שתי בעיות עיקריות:

"**זיהוי הרמוניות**": אות הקלט מורכב מתדר בסיסי (pitch) בתוספת תדרים גבוהים יותר. אם אמפליטודת התדרים הגבוהים אינה זניחה, תדרים אלה עלולים להופיע בגרף האוטוקורלציה בתור שיאים בעלי מרווחים קצרים מרווח ה-pitch (כמודגם באיור 8). במקרים כאלה עלול להיות זיהוי של תדרים גבוהים מתדר ה-pitch האמיתי.

"**זיהוי שברים**": מכיוון שכל כפולה של אורך ה-pitch הינה בעלת מחזוריות חזקה, נוצרים שיאים חזקים בערכי p שהם כפולות שלמות של אורך ה-pitch. ברוב המקרים, האוטוקורלציה דועכת באיטיות עם הגידול ב-p (לדוגמה, איור 8), אך ייתכנו מקרים שבהם האוטוקורלציה בכפולה השניה או השלישית הינם גבוהים במקצת מהאוטוקורלציה בערך ה-pitch האמיתי. כתוצאה ממקרים כאלה נוצר זיהוי אורך pitch גבוה מדי, כלומר - תדר נמוך מדי.

שתי בעיות מנוגדות אלה מהוות את עיקר המכשול בפני יישום מדויק של שיטות אוטוקורלציה. אין פתרון אחיד לבעיות אלה. הפתרונות המוצעים הינם טכניים בעיקרם, ומבוססים על נסיון וטעייה עם קבצי הקלטות, יותר מאשר על ידע תיאורטי בנוגע לתכונות המידע הקולי. מוצג כאן הפתרון ששימש לבניית התוכנה הסופית. הפתרון מבוסס על מאמר (Medan et al., 1991), עם מספר שינויים שהוכנסו לאחר בדיקת המערכת עם הקלטות בתנאי אמת.

בעיית זיהוי ההרמוניות ניתנת לפתרון אם נשים לב לכך שהקורלציה במרווח ההרמוניות גבוהה רק לטווח קצר. אם נסכם על טווח גדול יותר, תופיע השפעה חזקה יותר של תדר ה-pitch, אשר תבטל את השפעת ההרמוניות. כדי לנצל תכונה זו, נגדיר את הקורלציה לטווח ארוך

$$r_L[p] = \frac{\sum_{n=1}^{N} x[n] \cdot x[n+p]}{\sqrt{\sum_{n=1}^{N} x^2[n]} \sqrt{\sum_{n=1}^{N} x^2[n+p]}}$$

כאשר יש לשים לב לכך שתחום הסכימה הוא כעת בין 1 ל-N, שהוא מספר גדול במידת מה מה-pitch הצפוי. (בהגדרת האוטוקורלציה המקורית הסכימה התבצעה על p דגימות בלבד.) כתוצאה מכך שהסיכום הוא ארוך-טווח, השפעת התדרים הגבוהים חלשה מאוד, וגרף הקורלציה הוא הרבה יותר חלק. חישוב זה הוא, לכן, חסין לרעש יותר מהחישוב קצר-הטווח.

עם זאת, נמנעים מלהשתמש בו, הן בגלל סיבוכיות החישוב הגדולה יותר, והן בגלל העובדה שהוא כולל חלון זמן ארוך יותר - ולכן הנחת הסטציונריות היא פחות מדויקת.

כפשרה בין האלגוריתמים לטווח קצר ולטווח ארוך, נעשה שימוש בשיטת ה"מועמדים" (candidates) לבחירת ה-pitch (Medan et al., 1991). הרעיון בשיטה זו הוא לחשב אוטוקורלציה לטווח קצר עבור כל ערך pitch אפשרי, ולבחור את ערכי ה-pitch המבטיחים ביותר, עליהם תחושב גם קורלציה לטווח ארוך.

האלגוריתם הספציפי שאיתו נעשה שימוש בתוכנה הסופית נבחר על סמך ניסוי וטעייה עם קולות מוקלטים, והוא מבוצע באופן הבא: גרף האוטוקורלציה מחושב כרגיל בעזרת אוטוקורלציה לטווח קצר. מתוכו נלקחים כמועמדים לערכי pitch כל הערכים שהם נקודות מקסימום מקומי. מקסימום מקומי מוגדר במקרה זה כנקודה על הגרף שהינה בעלת אוטוקורלציה גבוהה מ6- הנקודות הנמצאות סביבה. כסינון נוסף, נבחרים כמועמדים רק ערכים שהם בעלי קורלציה השווה ל70%- או יותר מערך הקורלציה המקסימלי שהתקבל.

מתוך כל 100 ערכי pitch שנבדקים, מזוהים בדרך כלל כ3-4- מועמדים. כלומר, עיקר הסינון מתבצע בשיטת הקורלציה לטווח קצר. עבור מועמדים אלה, מחושבת הקורלציה לטווח ארוך, וצפוי לקבל מספר מועמדים בעלי מרווחים קצרים וקורלציה נמוכה (הנובעים מבעיית זיהוי ההרמוניות) ומספר מועמדים בעלי מרווחים ארוכים יותר וקורלציה גבוהה (אחד מהם הוא ה-pitch האמיתי, והאחרים נובעים מבעיית זיהוי השברים). כדי לזהות את ה-pitch, בוחרים את המועמד שעבורו אורך ה-pitch הקצר ביותר (כדי להמנע מזיהוי שברים), אך שהקורלציה ארוכת הטווח שלו היא לפחות 70% מערך הקורלציה המקסימלית של המועמדים (כדי להמנע מזיהוי הרמוניות).

3.3.3 דיוק

דיוק אלגוריתם האוטוקורלציה מוגבל על ידי המרווח שבין ערכי p אפשריים, כלומר, מרווח הזמן בין דגימות. לדוגמה, עבור קצב דגימה של 8000 הרץ, המרווח בין אורכי pitch אפשריים הוא כ0.1- מילישניה. עבור תדרים סביב 200 הרץ, מרווח זה מתבטא בהבדלים של כ 5- הרץ. ערך זה כמעט שאינו מספיק כדי להפריד בין שני תווים סמוכים, ואינו משאיר מרווח שגיאה נאות. לעומת זאת, עבור קצב דגימה של 44,100 הרץ, המרווח בין אורכי pitch אפשריים הוא כ0.02- מילישניה, המתאים ליכולת אבחנה של כ1- הרץ סביב תדר של 200 הרץ.

עם זאת, מבחינה מסוימת יש בעייתיות בצורך להגדיל את קצב הדגימה, ובכך גם את זמן החישוב ונפח האחסון של האות, אך ורק כדי לקבל דיוק רב יותר במדידת התדרים הנמוכים. הגדלת תדר הדגימה משמשת להגדלת טווח התדרים שניתן להקליט, ואילו במקרה זה אנו מעוניינים אך ורק במידע על התדרים הנמוכים; לעיתים אף נהוג לבצע, לאחר ההקלטה

בתדר גבוה, סינון מעביר נמוכים! עובדה זו רומזת על כך שניתן להוציא יותר מידע גם מאות הדגום בקצב נמוך, וזו מטרתה של שיטת הסופר-רזולוציה.

4.3 סופר-רזולוציה

שיטת הסופר-רזולוציה הינה הרחבה של שיטת האוטוקורלציה, אשר מאפשרת קבלת דיוק גבוה מאוד תוך חישוב אוטוקורלציה בקצב דגימה נמוך יחסית (Medan et al., 1991). הדיוק הגבוה מתבטא בכך שאורך ה-pitch המחושב הוא מספר לא שלם של דגימות, בניגוד לחישוב אוטוקורלציה כבסעיף 3.3, שבו דיוק המדידה נקבע על ידי הדרישה למספר שלם של דגימות.

שיטת הסופר-רזולוציה מבוססת על ההנחה שהאות, גם אם הוא דגום בקצב נמוך, מכיל את כל המידע על תדר נמוך כמו זה של ה-pitch, ולכן דגימה בקצב גבוה יותר לא תוסיף מידע על ה-pitch עצמו, אלא רק על תדרים גבוהים יותר. לכן, ניתן לקבל קירוב טוב של האות בקצב גבוה, על ידי אינטרפולציה של האות בקצב נמוך. בכך ניתן להגדיל אפקטיבית את דיוק האוטוקורלציה, מבלי להגדיל את נפח האחסון הדרוש עבור האות.

על ידי דוגמה מספרית ניתן להבהיר נקודה זו. נניח שמקליטים pitch של כ200- הרץ בתדר דגימה נמוך יחסית של 8000 הרץ. תדר דגימה זה גבוה פי 20 מתדר נייקויסט של האות. לכן, האות הדגום הוא קירוב טוב מאוד לאות המקורי, ושחזור מעשי על ידי אינטרפולציה כלשהי יהיה שחזור בעל שגיאה נמוכה מאוד.

ניתן למצוא נוסחה סגורה לשינוי באורך ה-pitch כתוצאה מהאינטרפולציה, במקרה של אינטרפולציה לינארית. נסמן ב- p_I את ערך ה-pitch שמתקבל בחישוב אוטוקורלציה סטנדרטית ("pitch שלם"). ניתן למצוא ערך תיקון p_F, כך שאם נבצע חישוב אוטוקורלציה בשיטה הסטנדרטית עבור האות לאחר אינטרפולציה, התוצאה שתתקבל תהיה $p_I + p_F$. הנוסחה לחישוב p_F היא ארוכה אך אנליטית, וניתנת לחישוב תוך שימוש רק בתוצאות חישובי האוטוקורלציה של p_I בתוספת 2 חישובי אוטוקורלציה נוספים, ו- $O(1)$ פעולות כפל וחיבור. לכן, חישוב p_F הינו מהיר מאוד ביחס למהירות חישוב p_I.

האלגוריתם מאפשר חישוב של p_F עבור אינטרפולציה לאות רציף, כך שתיאורטית אין מגבלה על הדיוק של אלגוריתם זה. עבור אותות מסונתזים ללא רעש, התקבלו תוצאות המדויקות פי 100 ויותר מהדיוק של שיטת ה-pitch השלם. עם זאת, עבור אותות רועשים מידת הדיוק יורדת. כדי לבדוק את הדיוק בתנאי אמת, הוקלט פסנתר המנגן תו קבוע (עבורו ה-pitch נשאר קבוע). סטיית התקן של הזיהוי היתה 0.045 דגימה, כלומר, הזיהוי הוא מדויק פי 20 מה-

pitch השלם. זאת, למרות שתכונות אחרות של הגל, כגון התדרים הגבוהים, השתנו תוך כדי הזיהוי, וכן חלה ירידה של כ80%- בעצמה (Volume) לאורך הקטע שנבדק.

בעייה אחת שהתגלתה בעת היישום של האלגוריתם הינה חוסר היציבות של p_F. הנוסחה לחישוב p_F עשויה, במקרים מסוימים, לתת ערכים גבוהים מ1- או קטנים מ-1-. ערכים אלה מצביעים, לכאורה, על אי-דיוק בזיהוי ה-pitch השלם, מכיוון שה-pitch הסופי מרוחק מ- p_I ביותר מדגימה אחת. במאמר שבו מתוארת שיטת הסופר-רזולוציה (Medan et al., 1991) מוזכרת בעייה זו, ומוצע כפתרון לשנות את p_I בדגימה אחת בכיוון הנדרש, ולחזור על חישוב p_F.

ואולם, הצעה זו גורמת לבעיה אחרת: נוצרים מצבים שבהם מתקבלת לולאה אינסופית. לדוגמה, נניח שהתקבל ערך של $p_F = 2.2 \quad ,p_I = 60$. בהתאם לאלגוריתם הנ"ל, p_I מוגדל ל61- ו- p_F מחושב שנית, אך כעת ייתכן שמתקבל $p_F = -1.8$. כעת p_I מוקטן חזרה ל60-, p_F מקבל שוב ערך גבוה, וחוזר חלילה. בניגוד לרושם הנוצר מ-Medan et al., 1991, מצב זה אינו נובע מזיהוי שגוי של ה-pitch השלם, אלא התרחש מספר פעמים, במסגרת הבדיקות של התוכנה, עבור אותות שהוקלטו בתנאי אמת, וגם במצבים שבהם זיהוי ה-pitch השלם היה נכון.

ב-Medan et al., 1991 מוצע גם פתרון אחר לבעיה: להתעלם מהעובדה ש- p_F אינו הגיוני, ולחבר אותו ל- p_I בכל זאת. למרות שהוכח שפתרון זה נותן תוצאות נכונות עבור אותות סינוסואידליים נקיים, הפתרון אינו מספק עבור תנאי אמת, משום שבמקרים נדירים (בעיקר בתנאי רעש) מתקבלים ערכי p_F גבוהים מאוד (מעל 10), הנובעים מחלוקה באיבר קטן. במצב כזה, לא רק שהסופר-רזולוציה אינה תורמת לזיהוי, היא גם הורסת את זיהוי ה-pitch השלם.

לאור כל זאת, הוחלט להשתמש באלגוריתם הבא במקרה של ערכי p_F מחוץ לטווח: p_I מוגדל או מוקטן ב1-, ו- p_F מחושב מחדש, עד לגבול של 3 פעמים עבור כל חישוב pitch. אם p_I השתנה כבר 3 פעמים, ועדיין לא מתקבל ערך הגיוני של p_F, אנו מניחים כי זהו מצב שבו אלגוריתם הסופר-רזולוציה אינו פועל, ומשתמשים בערך ה-pitch השלם המקורי שהתקבל.

5.3 חישוב volume

ה-volume מוגדר ככמות האנרגיה המצויה באות הדיבור, והוא פרופורציונלי לשורש של ערך התוחלת של ריבוע האות, או $\sqrt{E[x^2(t)]}$. כדי לחשב את ה-volume עבור קטע זמן קצר,

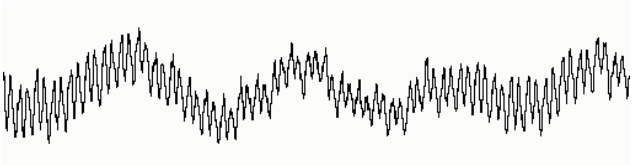
מחושב שורש ממוצע הריבועים של הדגימות באותו קטע זמן. שיטת חישוב זו מכונה Root Mean Square (RMS) Volume.

קיים trade-off בבחירת אורך החלון שעליו יבוצע חישוב ה-volume. מצד אחד, רצוי לבחור חלון קצר ככל האפשר, על מנת לקבל ערכי volume מיידיים, שאינם תלויים בעצמת הקול בפרקי זמן אחרים. מצד שני, חלון זמן קצר מאוד יכלול מספר קטן של דגימות, וחישוב הממצע על דגימות אלה לא יהיה בהכרח קרוב לערך התוחלת (הנחת הארגודיות לא תהיה תקפה).

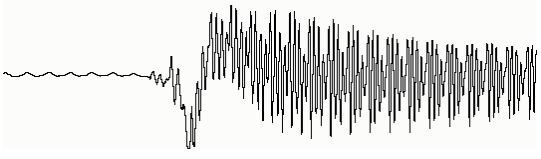
כפשרה בין אילוצים אלה, נבחר בפרויקט זה לחשב את ה-volume על סמך חלון שאורכו שלושה מחזורי pitch. באופן כזה, מובטח כי החלון יכיל מספיק דגימות כדי שהאות יהיה קווזי-מחזורי ולפיכך ארגודי בקירוב. הקטנה נוספת של החלון גורמת לאפקטים לא רצויים, כגון תלות במקום חיתוך החלון ביחס למחזור ה-pitch.

הנחה סמויה בחישוב ה-volume הינה שרמת ה-DC של הערוץ קבועה על אפס. מכיוון שגל קול מועבר ע"י תנודות בלחץ האויר, המידע המעניין בגל הקול הינו השינויים בלחץ, או רכיב ה-AC. ההנחה היא, שהמיקרופון או מערכת ה-A/D מוציאים החוצה את רכיב ה-DC, ומתקבל גל "נקי", בעל תוחלת אפס.

עם זאת, רעשים מסוימים (כגון רוח) עשויים לגרום לרכיב בעל תדר נמוך מאוד (איור 9). כמו כן, התחלה פתאומית של צליל גורמת, במיקרופונים מסוימים, למצב רוויה זמני שבו הוצאת רמת ה-DC אינה מושלמת (איור 10). בשני המקרים האלה, חישוב ישיר של ה- RMS Volume יגרום לקבלת תוצאות גבוהות מהערכים האמיתיים, מכיוון שרכיב ה-DC, אותו איננו רוצים לקחת בחשבון, מופיע ומגדיל את ערכי ה-volume. המצב של איור 9 הוא גרוע במיוחד, מכיוון שה-volume הנמדד יושפע בראש ובראשונה מהרעש, והשפעת עצמת אות הדיבור עצמו תהיה זניחה.



איור 9: רכיב רעש בתדר נמוך



איור 10: מצב רוויה זמני בתחילת הקלטה

כדי להמנע ממצב זה, הוחלט לבצע קיזוז של רמת ה-DC בכל חלון, לפני חישוב ה-volume. כלומר, הערך הממוצע של החלון מופחת מה-RMS Volume. התוצאה היא שערך ה-volume המחושב הוא למעשה סטיית התקן של האות בתוך החלון. כאשר התוחלת של האות היא אפס, סטיית התקן שווה ל-RMS; אך כאשר התוחלת שונה מאפס, סטיית התקן היא נמוכה יותר. באופן כזה מוקטנת ההשפעה של הפרעות בתדר נמוך.

6.3 סיכום

בפרק זה נעשתה סקירה של מגוון שיטות זיהוי pitch, ביניהן שיטות נקודתיות, שיטות מבוססות-תדר ושיטות אוטוקורלציה. שיטת האוטוקורלציה היא השיטה המקובלת ביותר, למרות ששיטות ספקטרליות שימושיות במקרים של זיהוי קולות סימולטניים.

האלגוריתם הספציפי שנבחר מבוסס על אוטוקורלציה קצרת-טווח בשיטת סופר-רזולוציה, עם מספר שינויים טכניים המבוססים על "משחק" עם פרמטרי הזיהוי ונועדו להקטין את השגיאה בזיהוי עבור הקלטות של קולות שירה והמהום, שהם סוגי הקלט הצפויים בתוכנה.

כמו כן הוצג מנגנון לזיהוי volume, המבוסס על חישוב סטיית התקן של האות. שיטה זו דומה לשיטת ה-RMS המקובלת בדרך כלל, אך מקטינה את ההשפעה של רעשים והפרעות בתדר נמוך.

זיהוי ה-pitch וה-volume הינם השלב הראשון לקראת זיהוי היצירה המוקלטת. בפרקים הבאים יוצגו שלבי החישוב הבאים, אשר מסתמכים על תוצאות החישוב של פרק זה. בפרק 6 יוצגו מספר בדיקות של מנגנון זיהוי ה-pitch, על מנת לבדוק את אמינותו וחסינותו לרעש.

4. סגמנטציה

תהליך הסגמנטציה הוא השלב שבו מתורגם מידע גולמי של pitch ו-volume, אל מידע על תווים בודדים: התדר ומשך הזמן של כל תו. שלב זה הינו השלב הבעייתי ביותר מבחינת המחשב, מכיוון שיש צורך להבין למה התכוון המשתמש כאשר שר את השיר: למשל, כאשר חל שינוי בתדר, ייתכן כי שינוי זה מציין תו חדש, אך ייתכן גם כי מדובר בתו יחיד, שבוצע עם זיוף או אפקט ויברטו.

בגלל בעיות אלה, תהליך הסגמנטציה הוא השלב שבו הושקעו רוב המאמצים בעבודה זו. בפרק זה נתאר שתי שיטות סגמנטציה נפרדות. בשיטה הראשונה, הסגמנטציה מבוססת רק על זיהוי שינויים ב-pitch, ואילו בשיטה השניה מתבצע שילוב בין מידע ה-pitch וה-volume. כפי שיוסבר בהמשך, השיטה המשולבת מתאימה יותר לשימוש פרקטי והיא זו שנבחרה לתוכנה הסופית.

בפרק זה נתייחס לערכי ה-pitch וה-volume כאל רשימת ערכים או מערך. כל איבר במערך מכיל נתון אחד של pitch או volume, ומיקומו במערך מציין את הזמן שבו נמדד ערך זה. קיים איבר אחד במערך עבור כל מחזור pitch (כלומר, זיהוי pitch-synchronous), ולכן הפרשי הזמנים בין נקודות במערך אינם קבועים. לתיאור המערכים ייעשה שימוש בביטויים השאולים מפונקציות בדידות. לדוגמה, כאשר מדובר על שני השכנים הקרובים ביותר לערך pitch מסוים, הכוונה היא לשני המקומות במערך המופיעים מיד לפני ומיד אחרי ערך ה-pitch הנ"ל.

1.4 עיבוד ראשוני

ערכי ה-pitch מתקבלים מתהליך ארוך של זיהוי pitch, שתואר בפרק 3. לפני השימוש בהם, מתבצע עיבוד ראשוני שמטרתו לשפר את איכות מידע ה-pitch, בלי להכביד בזמן חישוב על תהליך זיהוי ה-pitch. תהליך זה כולל שני שלבים: הוצאת ערכי pitch שגויים בעליל, ואינטרפולציה על ערכי ה-pitch.

בשלב ראשון נסרק מערך ערכי ה-pitch ומבוצעת החלקה שמטרתה לבטל מקרים נקודתיים של זיהוי pitch שגוי. בתהליך זה, נערך חיפוש לערכי pitch שונים מאוד מסביבתם. אם נמצא ערך pitch שמרחקו משני שכניו הוא יותר מ400- סנט, ערך pitch זה מוכרז כשגוי; זאת מכיוון שאדם אינו מסוגל לשנות את ה-pitch שלו בקצב מהיר כל כך. במקום ערך שגוי זה, מציבים את הממוצע של שני שכניו של ערך ה-pitch השגוי. יש לציין כי תיקון זה הינו נדיר

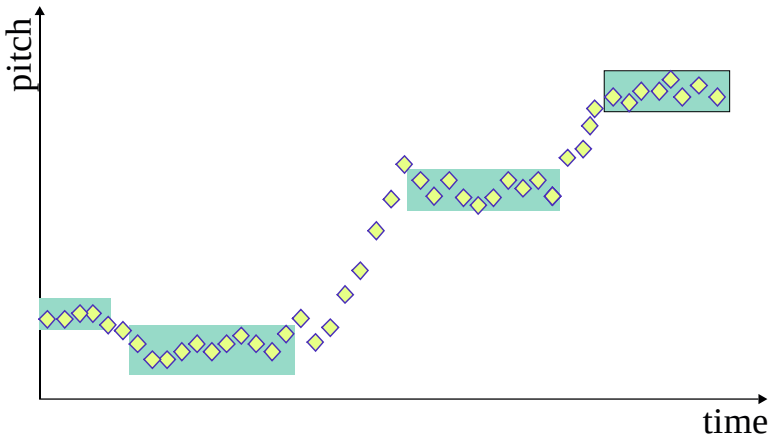
מאוד במצבי הקלטה תקינים. עם זאת, פעולה זו עשויה לתקן מקרים נקודתיים של זיהוי שברים או זיהוי הרמוניות (ראו סעיף 2.3.3).

עיבוד נוסף שמבוצע על מערך ערכי ה-pitch הוא אינטרפולציה. בין כל שני ערכי pitch מוסיפים ערך pitch נוסף, השווה לממוצע בין שניהם. בצורה כזו, מוגדל מספר ערכי ה-pitch פי שניים, כמעט ללא צורך בחישובים נוספים. הגדלה זו חשובה על מנת לאפשר סגמנטציה מדויקת יותר, אך בגלל המחיר הכבד (מבחינת זמן חישוב) הדרוש להגדלת מספר חישובי ה-pitch, נבחר לבצע אינטרפולציה בין ערכי pitch במקום חישוב pitch במרווחים קצרים יותר. כיוון שתיאורטית ה-pitch אמור להיות רציף, אינטרפולציה זו אינה גורעת משמעותית מדיוק הזיהוי.

2.4 סגמנטציה מבוססת Pitch

השיטה המקורית שנבחרה לביצוע הסגמנטציה היא בעלת עקרון פשוט. מכיוון שכל תו אמור להיות בעל pitch קבוע, יש למצוא סגמנטים שבהם ה-pitch הוא קבוע בקירוב[1].

דוגמה למצב כזה מוצגת באיור 11. באיור, ישנה הפרדה ברורה בין תווים שונים, שבהם ה-pitch הוא קבוע בקירוב. בין תווים אלה ישנם איזורי מעבר, אשר יכולים להיות איזורים בעלי pitch המשתנה במהירות, או איזורים שבהם הדיבור הוא לא-קולי (unvoiced) וכתוצאה מכך זיהוי ה-pitch נותן ערכים אקראיים ולא רציפים.

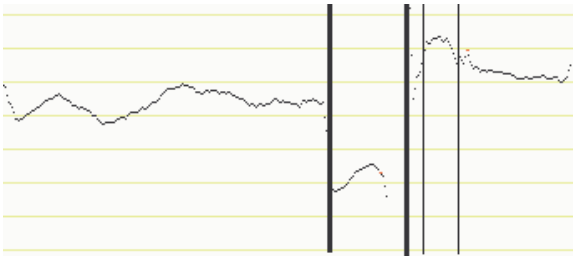


איור 11: סגמנטציה מבוססת pitch

בשלב ראשון, נכתב אלגוריתם פשוט לסגמנטציה ע"פ pitch שפעל באופן הבא. האלגוריתם עבר לפי הסדר על כל נקודות הזמן במערך ערכי ה-pitch. בכל שלב, חושב הממוצע מסוף הסגמנט הקודם ועד לנקודת הזמן הנוכחית. (אם לא נמצא עדיין אף סגמנט, חושב הממוצע מתחילת ההקלטה ועד לנקודה הנוכחית.) ערך ה-pitch הנוכחי הושווה לממוצע. במקרה שנוצר הפרש גבוה מ100- סנט בין שני ערכים אלה, נקבע כי נקודת הזמן הנוכחית היא סוף הסגמנט הנוכחי ותחילתו של סגמנט חדש.

על פי הגדרה זו, גם איזורי שקט בין שני תווים הינם "סגמנטים". איזורים כאלה מזוהים מאוחר יותר, על סמך העובדה שה-pitch שלהם אינו אחיד (כלומר, בעל סטיית תקן גבוהה), או על סמך היותם סגמנטים קצרים. במקרה שמזוהה סגמנט כזה, הוא אינו מתורגם לתו.

הבעיה באלגוריתם זה היא שבמקרים מסוימים עשויים להיות סטיות גדולות מהממוצע, גם בתוך תו יחיד. דוגמה לכך מוצגת באיור 12. באיור זה מוצגים בקווים עבים ערכי הסגמנטציה הנכונים (שנקבעו באופן ידני, ע"י האזנה להקלטה). בקווים דקים מוצגים סגמנטים נוספים שאותם הכניס המחשב, כאשר בוצעה סגמנטציה בשיטה שתוארה למעלה.



איור 12: מצב שבו סגמנטציית pitch אינה פועלת נכון

(קווים עבים – סגמנטציה רצויה ; קווים דקים – סגמנטציית pitch)

באיור זה מתבטאת תופעה כללית שבה ה-pitch אינו נשאר קבוע, אלא משתנה בטווח ערכים גבוה יחסית (ולעיתים אף יותר מסמיטון). תופעה זו נפוצה במיוחד בתחילת תו. במצב כזה מתרחש overshoot: אם התו הנוכחי נמוך יותר מהתו הקודם, המשתמש מתחיל לשיר אותו עוד יותר נמוך מהתדר הנכון שלו. לאחר מכן המשתמש שומע את התוצאה, ומתקן אותה על ידי מעבר לתדר גבוה יותר. לעיתים תהליך זה חוזר על עצמו מספר פעמים: אם התיקון הוא

---

[1] שיטה דומה הוצגה על ידי McNab et al., 1996, אך במאמר מאוחר יותר (McNab et al., 1997) הם תיארו שיטה זו כ"פחות אמינה" מסגמנטציה מבוססת volume.

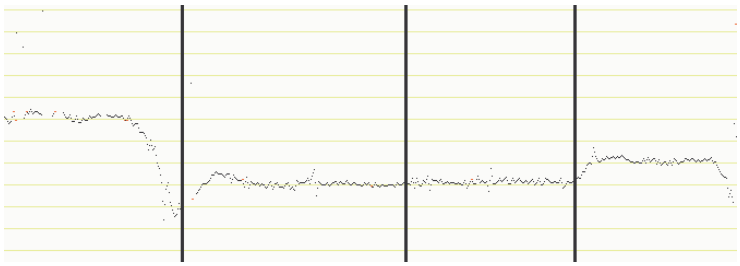
חזק מדי, מתבצע תיקון "מסדר שני" כלפי מטה. תופעה מקבילה מתרחשת גם כאשר התו הנוכחי הוא גבוה יותר מהקודם. המשתמש אינו מודע לתהליך הזה, ומבצע אותו מבלי לשים לב; ובכל זאת, התופעה קיימת כמעט אצל כל משתמש שנבדק (גם בהקלטות של זמרים מקצועיים), ויש צורך להתחשב בה.

ברור, לכן, שהאלגוריתם כפי שתואר למעלה אינו מתאים לסגמנטציה של שירה על ידי בני אדם. נבדקו מספר שינויים של האלגוריתם, על מנת להקטין את השפעת חוסר היציבות של ה-pitch.

שיפור אחד של האלגוריתם מאלץ אורך מינימלי כלשהו של סגמנטים. שינוי זה מיועד למנוע מצב שבו תחילת התו (שהיא באופן טיפוסי בעלת תדר שונה במקצת משאר התו) מזוהה כסגמנט נפרד. תוספת זו אכן הקטינה את ההשפעה של חוסר היציבות בתחילת התו, אך יוצרת אילוץ על מהירות השירה האפשרית, וגורמת לכך שתווים קצרים אינם מזוהים.

שיפור נוסף שהוכנס מיועד למנוע שינויים קצרים ב-pitch ולוודא שהם אינם גורמים ליצירת סגמנט נוסף. תוספת זו היא למעשה שינוי של התנאי למציאת סוף הסגמנט: התנאי החדש דורש לא רק שערך pitch יחיד יהיה במרחק של יותר מ100- סנט מהתדר הממוצע של הסגמנט, אלא שמספר ערכי pitch רצופים יהיה מרוחקים יותר מ100- סנט מהתדר הממוצע. בצורה זו מובטח כי ישנו שינוי אמיתי של התדר, ולא רק קפיצה פתאומית שעשויה להווצר כתוצאה מזיוף או חוסר יציבות ב-pitch.

גם אם הבעיות שהוצגו עד כה הן חמורות, עקב אכילס של שיטת הסגמנטציה הזו הוא סגמנטציה בין שני תווים בעלי תדר זהה. כדוגמה לבעיה, איור 13 מציג הקלטה של ארבעת התווים הראשונים של השיר "יונתן הקטן". התו השני והשלישי הם בעלי תדר זהה. הסגמנטציה הרצויה מסומנת בקווים עבים, אך קשה לראות כיצד אלגוריתם הסגמנטציה יכול להפריד, על סמך מידע pitch בלבד, בין התו השני והשלישי, כיוון שה-pitch שלהם הוא אחיד לחלוטין.



איור 13: סגמנטציה בין שני תווים זהים רצופים

חוסר היכולת להבדיל בין תווים זהים, וכן הבעיות האחרות שתוארו בסעיף זה, הובילו לחיפוש אחר אלגוריתם סגמנטציה בעל ביצועים טובים יותר. אלגוריתם כזה מתואר בסעיף הבא.

3.4 סגמנטציה משולבת Pitch-Volume

כפי שהוסבר בסעיף הקודם, סגמנטציה המבוססת רק על מידע pitch אינה כוללת כמות מספיקה של מידע כדי לאפשר זיהוי מדויק לחלוטין בכל המקרים. זאת מכיוון שיש צורך להבדיל בין שני מצבים דומים מאוד מבחינת pitch: מצב של תו יחיד שבו ה-pitch אינו קבוע בגלל מגבלות יכולת השירה של המשתמש, לעומת מצב שבו ישנם שני תווים נפרדים בעלי תדר דומה או זהה. לכן, נוצר הצורך באלגוריתם סגמנטציה שיסתמך על מידע נוסף, מעבר למידע ה-pitch בלבד. בעקבות Ghias et al., 1997, הוחלט להשתמש, בנוסף למידע על ה-pitch, גם במידע של עצמת הקול (volume).

1.3.4 בחירת אופן השירה

בצורות שירה מסוימות, אשר יתוארו מיד, ישנה ירידה משמעותית ב-volume בין כל שני תווים ביצירה. ניתן לנצל שינויי volume אלה על מנת לזהות בצורה מדויקת יותר את ההתחלה והסיום של כל תו. טיב הזיהוי מושפע ממידת ירידת ה-volume בין תווים, ולכן חשוב למצוא מהי צורת השירה האופטימלית עבור זיהוי.

אין הסכמה כללית על צורת השירה האופטימלית. יש המשתמשים בקלט על ידי המהום (Ghias et al., 1997), ואחרים משתמשים בקלט על ידי שירת ההברות "טָה-טָה-טָה" (McNab et al., 1997). לכן, כדי לבחור בשיטת השירה האופטימלית הוקלטו מספר קלטים באופני שירה שונים, והשוו הגרפים של ה-volume שלהם (איור 14).

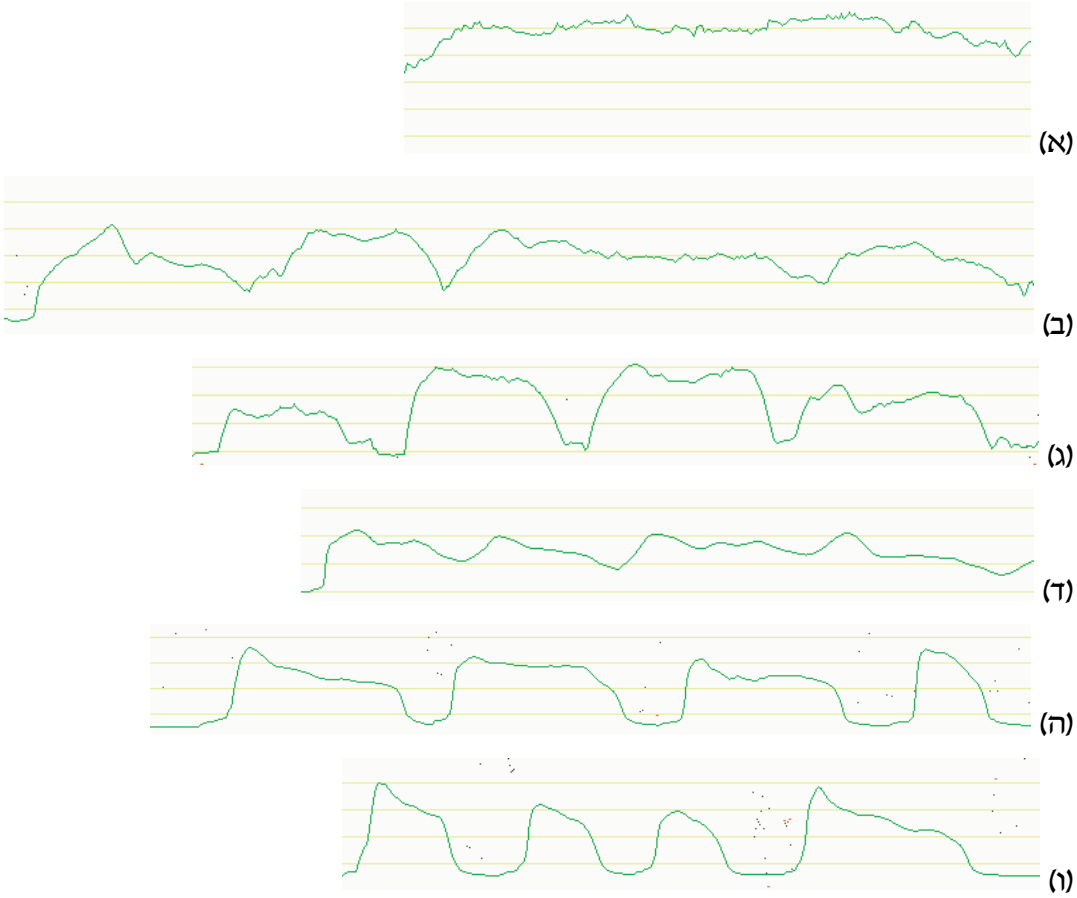
ניתן לראות כי במקרה של שירה באמצעות ההברות "לָה-לָה-לָה" או "הָה-הָה-הָה", אין הפרדה טובה בין התווים; גרף ה-volume הוא פחות או יותר אחיד, או שאין ירידה מלאה של ה-volume בין התווים. הסיבה לכך היא שהעיצורים /ל/, /ה/ אינם גורמים לעצירת הקול היוצא מהפה, ולכן ה-volume אינו יורד גם בעת הגיית העיצור.

לעומת זאת, המצב עבור ההברות המכילות את העיצור /ט/ וכן עבור המהום הוא הרבה יותר טוב. ברוב המקרים ישנה ירידה משמעותית מאוד של ה-volume.

עבור העיצור /ט/ הסיבה לירידה ב-volume היא היותו עיצור פּוֹצֵץ (plosive), וכדי להגות אותו יש צורך לעצור לחלוטין את יציאת האוויר מהפה למשך כ50- מילישניות, דבר

הגורם לירידה של ה-volume כמעט עד לרמת רעש רקע. תוצאות דומות מתקבלות גם עבור עיצורים פוצצים אחרים, כגון /ד/.

לעומת זאת, עבור המהום הסיבה לירידה ב-volume בין תווים היא, כנראה, הרגל של בני אדם: נהוג לעצור בין שני תווים כאשר שרים בעזרת המהום. הסבר אפשרי לכך הוא שהגיה באצמעות הצליל /מ/ היא אחידה מאוד, ולכן ללא עצירה זו היה קשה גם למאזין אנושי להבחין בין התווים. אך מכיוון שאין סיבה פיזיולוגית לירידה ב-volume, היא אינה תמיד מתבצעת באופן מלא (לדוגמה ראה איור 14ד).



איור 14: volume כפונקציה של הזמן עבור צורות שירה שונות

(כל דוגמה מכילה ארבעה תווים)

(א) "לָה-לָה-לָה" ; (ב) "הָה-הָה-הָה" ; (ג)-(ד) המהום ; (ה) "טִי-טִי-טִי" ; (ו) "טָה-טָה-טָה"

המסקנה מכך היא, שכדי להבטיח סגמנטציה אופטימלית יש לבקש מהמשתמש לבצע את ההקלטה באמצעות הגיית ההברה "טָה" או "טִי" (או שילוב שלהם). שיטה זו מבטיחה ירידה משמעותית של ה-volume בין התווים.

2.3.4 אלגוריתם הסגמנטציה

לאחר שנקבע אופיו של הקלט, יש למצוא את אלגוריתם הסגמנטציה שיתאים בצורה הטובה ביותר לקבוצת הקלטים האפשריים. במאמרים קיימים (McNab et al., 1997; Ghias et al., 1997) לא פורט אופן פעולת אלגוריתם הסמנטציה, והאלגוריתם המתואר כאן נבחר לאחר התבוננות במספר רב של קלטים, ובדיקת אפשרויות שונות של האלגוריתם.

האלגוריתם מורכב משני שלבים. סגמנטציה ראשונית מתבצעת על פי ה-volume בלבד. מטרת שלב זה היא לזהות סגמנטים, שכל אחד מהם מכיל תו יחיד. ואולם, שלב זה אינו מזהה את תחילתו וסופו של התו במדויק, מכיוון שבמקרים רבים ה-volume עולה לפני תחילת החלק הקולי של ההברה. לכן, מתבצע שלב סגמנטציה שניונית, שבו בכל סגמנט נבחר האיזור אשר מכיל צליל מוסיקלי. שלב זה מסתמך על מידע ה-pitch.

בכל שלבי האלגוריתם, קיימים פרמטרים מספריים המאפיינים את תהליך הזיהוי. בכדי להמנע מבחירה שרירותית של מקדמים אלה, נערכה אופטימיזציה של הערכים על מדגם של הקלטות בתנאי אמת. תהליך האופטימיזציה מתואר בסעיף 7.2. בפרק זה, יצוינו הערכים המספריים של פרמטרים אלה ללא הצדקה.

סגמנטציה ראשונית

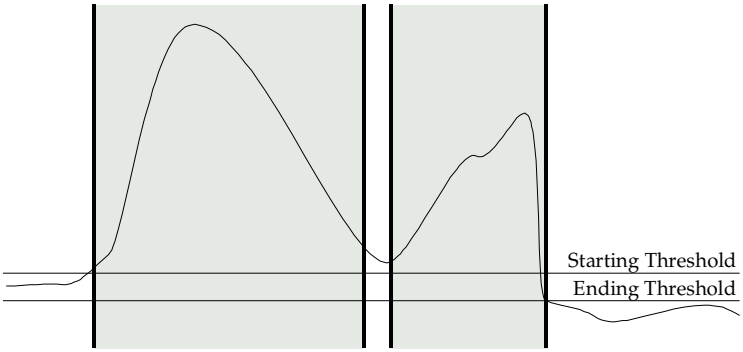
כאמור, מטרת שלב זה היא לזהות באופן גס את נקודות ההתחלה והסיום של כל תו, בעזרת שימוש ב-volume של האות. כדי למנוע מצב שבו קיימת תלות בעצמת ההקלטה הכללית (אשר תלויה בגורמים סביבתיים כגון סוג המיקרופון, קירבת המיקרופון לדובר וכד'), מחושב תחילה ה-volume הממוצע של ההקלטה כולה, וכל הפרמטרים נקבעים ביחס לערך זה.

זיהוי נקודות ההתחלה והסיום של הסגמנטים מתבצע על ידי שתי מערכות: מערכת קבועה ומערכת אדפטיבית. שתי המערכות פועלות במקביל, כך שכל אחת מהן עשויה לסמן את תחילתו או סופו של סגמנט. צורת פעולה זו מאפשרת לנצל את היתרון של המערכת האדפטיבית במקרה של סגמנטים הדומים זה לזה באופיים, ועדיין לזהות מקרים חריגים באמצעות המערכת הקבועה.

המערכת הקבועה מכילה ספים אחידים (המיוצגים כאחוזים מה-volume הממוצע), עבור ההתחלה והסיום של הסגמנט. סף ההתחלה הינו 20% מה-volume הממוצע, ואילו סף הסיום הינו 14% מה-volume הממוצע. ההפרש בין סף ההתחלה לסף הסיום נועד למנוע זיהוי

סגמנטים קצרים מאוד כאשר ה-volume מתנדנד סביב ערך הסף. כאשר מתקבל רצף של 10 ערכי volume, שכולם גבוהים מסף ההתחלה, מוחלט כי מדובר בתחילת סגמנט. לעומת זאת, עבור סיום סגמנט מספיק ערך volume אחד, וזאת על מנת לאפשר זיהוי של סוף סגמנט גם כאשר המרווח בין תווים הוא קצר מאוד.

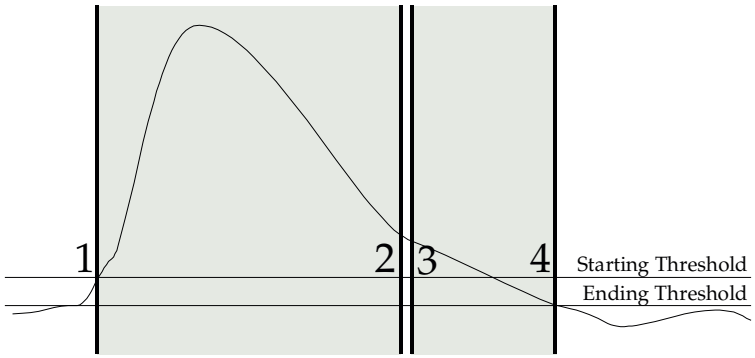
תחת תנאי הקלטה אידאליים, ניתן היה להסתפק בסף הקבוע; ואולם, בתנאי אמת נוצרים לפעמים מצבים שבהם המשתמש אינו עוצר לחלוטין בין תווים, וכתוצאה מכך אין ירידה מלאה של ה-volume בין התווים (איור 15). הפתרון הפשוט, של הגבהת סף הסיום, דורש הגבהה גם של סף ההתחלה, ובכך מקטין את יכולת הזיהוי של תווים שקטים. לכן, נבחר פתרון המאפשר לסיים סגמנט גם אם ה-volume שלו עדיין לא ירד לרמת סף הסיום, בעזרת מערכת אדפטיבית. במערכת זו, סיום הסגמנט מזוהה כירידה מתחת ל30%- מה-volume המקסימלי של הסגמנט הנוכחי.



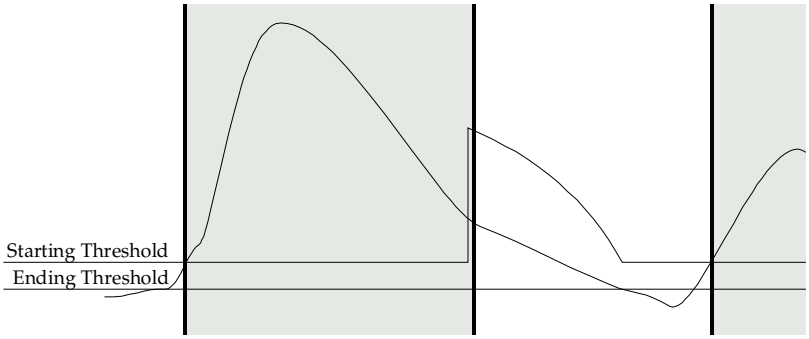
איור 15: מצב הדורש סגמנטציה אדפטיבית

השילוב הלא-סטנדרטי בין מערכת הסגמנטציה הקבועה והאדפטיבית יצר בעיה לא-צפויה, המודגמת באיור 16: בתחילה (שלב 1 באיור), מזוהה תחילתו של הסגמנט על-ידי הסף הקבוע. לאחר מכן (שלב 2 באיור), מזוהה הסוף של אותו סגמנט בצורה אדפטיבית ותקינה. אך כעת רמת ה-volume עשויה עדיין להיות גבוהה לא רק מסף הסיום, אלא גם מסף ההתחלה, וכתוצאה מכך יזוהה מיידית תחילתו של סגמנט נוסף (שלב 3 באיור), למרות שמדובר עדיין באותו תו; ולבסוף מזוהה הסיום של הסגמנט השגוי בעזרת סף הסיום הקבוע (שלב 4 באיור).

כדי לפתור בעיה זו, הוכנס תיקון לאופן בחירת סף ההתחלה, כך שגם סף ההתחלה אינו קבוע. על-פי תיקון זה, סף ההתחלה של כל תו מקבל ערך של 60% מה-volume המקסימלי של התו הקודם. ערך זה הוא, ברוב המקרים, גבוה בהרבה מסף ההתחלה הקבוע. ככל שמתרחקים מסוף הסגמנט הקודם, הולך ודועך סף זה, עד שהוא מגיע לערך הסף הקבוע (בדרך כלל תוך כ30- מחזורי pitch). מצב זה מאפשר ערכי volume גבוהים מיד לאחר סיום הסגמנט הקודם, גם ללא זיהויים כסגמנט נוסף; ערכי volume דומים, אם יימצאו רחוק מהתו הקודם, יגרמו לזיהוי תחילת התו. איור 17 מציג דוגמה לפעולת סף ההתחלה המלאכותי.

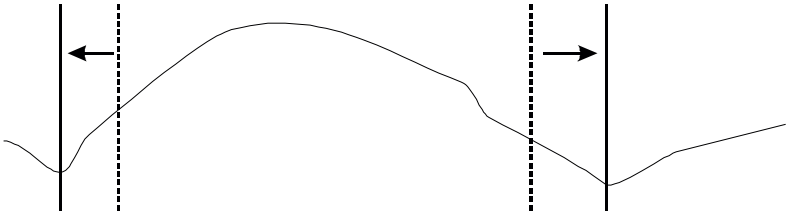


איור 16: דוגמה לסגמנטציה שגויה בעקבות שילוב הסף האדפטיבי



איור 17 : סף התחלה "מלאכותי"

בסיום השלב הראשוני של הסגמנטציה, מתבצע תהליך תיקון שבו קצוות הסגמנט מורחבים עד לקצה איזור דעיכת ה-volume. תיקון זה משפיע במצב שבו ה-volume ממשיך לדעוך מעבר מעבר לסיום הסגמנט, או מתחיל לעלות לפני תחילת הסגמנט (איור 18). במצב כזה, התוכנה מעבירה את קצה הסגמנט אל המינימום המקומי של ה-volume, על מנת להבטיח שכמות מקסימלית של התו נכנסת בתוך הסגמנט.



איור 18: תיקון הסגמנטציה הראשונית

(קו מקוטע – תחילתו וסופו של הסגמנט הראשוני ; קו רציף – הסגמנט המתוקן)

סגמנטציה שניונית

כזכור, תהליך הסגמנטציה הראשונית מפריד בין התווים באופן גס, ותפקידו של תהליך הסגמנטציה השניונית הוא לזהות באופן מדויק את נקודות ההתחלה והסיום של כל תו, בתוך הסגמנט שלו. תפקיד נוסף של הסגמנטציה השניונית הוא לזהות מצבים שבהם הסגמנט אינו מכיל מידע מוסיקלי (למשל, מקרים של רעש רקע), ולבטל את אותם סגמנטים. כדי לבצע את שתי המטלות האלה, נעשה שימוש במידע ה-pitch של הסגמנט.

סגמנט תקין צפוי להיות קטע זמן הכולל בתוכו את התו המבוקש, שהוא איזור בעל pitch קבוע פחות או יותר, ובנוסף עשוי לכלול גם איזורים מהם אנו מעוניינים להתעלם: איזורי שקט, איזורי תחילת וסיום התו שבהם המידע אינו קולי, וכן איזורים בתחילת התו שבהם ה-pitch אינו יציב (לדוגמה, איור 12 והדיון לגביו בסעיף 2.4).

כדי להתאים אלגוריתם לדרישות אלה, הוחלט להגדיר את איזור התו המבוקש כאיזור הרציף הגדול ביותר בתוך הסגמנט, שעבורו סטיית התקן של ערכי ה-pitch נמוכה. התוכנה עוברת על תת-האיזורים הרציפים בתוך הסגמנט, מחשבת את סטיית התקן של ה-pitch עבור כל אחד מהם, ומוצאת את תת-האיזור הארוך ביותר בעל סטיית תקן שאינה עולה על 50 סנט. איזור זה מוכרז כאיזור התו של הסגמנט.

מספר תת-האיזורים הרציפים בתוך סגמנט באורך n הינו $n(n-1)$. במקרים של סגמנטים קצרים, ניתן לבדוק את סטיית התקן עבור כל הסגמנטים בלי לצרוך זמן רב. אך לעיתים קיימים סגמנטים ארוכים של כמה מאות ערכי pitch, עבורם חיפוש ממצה כזה לוקח כמות לא סבירה של זמן. לצורך כך הוכנסו לתוכנה מספר שיטות לחיפוש תת-אופטימלי.

שיטה אחת כזו מיועדת לחסוך טיפול במקרים של איזורים בעלי סטיית תקן גבוהה מאוד. במקרה שנמצא איזור מסוים בעל סטיית תקן מעל 150 סנט, התוכנה מניחה שאיזור זה אינו יכול להכיל תו, והחיפוש מדלג על האיזור הבעייתי וממשיך הלאה.

שיטה נוספת מופעלת במקרה של סגמנטים שאורכם עולה על 100 מחזורי pitch. במקרים כאלה ההנחה היא שתת-האיזור שייבחר בסופו של דבר יהיה ארוך יחסית, ולכן הדיוק במציאת נקודת ההתחלה והסיום שלו הינו פחות קריטי. לכן, החיפוש אינו מתבצע על כל נקודות ההתחלה והסיום האפשריים. במקום זאת, מתבצע חיפוש בדילוגים, כך שנבחנים לא יותר מ10,000- תת-איזורים.

לאחר מציאת תת-האיזור האופטימלי, יש לקבל החלטת קְבִילוּת בקשר לסגמנט הנוכחי: כלומר, האם הוא מכיל תו אמיתי, או שמא זהו סגמנט רעש שיש לבטלו. החלטה זו מתקבלת על סמך אורך תת-האיזור האופטימלי. אם אורכו גדול מאוד, הרי שיש איזור גדול בתוך הסגמנט שבו ה-pitch קבוע ולכן הגיוני לצפות שהסגמנט מכיל תו. לעומת זאת, אם אורכו קצר מאוד, ניתן להניח כי לא היה בסגמנט אף תו מוסיקלי, וכי תת-האיזור התקבל באופן מקרי ממספר ערכי pitch אקראיים, אשר נמצאו באופן מקרי קרובים אחד לשני.

תרגום לתווים

בשלבים קודמים של תהליך הסגמנטציה נקבעו כבר נקודות הזמן שבהן מתחיל ומסתיים כל תו. נותר לקבוע רק את התדר המתאים לאותו תו. על-פי תהליך הסגמנטציה השניונית מובטח כי כל תו יכיל ערכי pitch בעלי סטיית תקן שאינה עולה על 50 סנט, אך עובדה זו עדיין משאירה מקום לספק באשר לערך ה-pitch אליו התכוון המשתמש להגיע.

מצב נפוץ במיוחד הוא שתחילתו או סופו של התו מכילים ערכי pitch שגויים, כפי שתואר בסעיף 2.4. ייתכנו גם שגיאות נוספות כותצאה מזיהוי pitch שגוי. בשני המקרים עלול האיזור השגוי להכלל בתוך תת-האיזור שנחשב כתַו, מכיוון שהמגבלה על סטיית התקן מאפשרת לקבל מספר מסוים של ערכים רחוקים מהממוצע, אם שאר הערכים קרובים מאוד לממוצע.

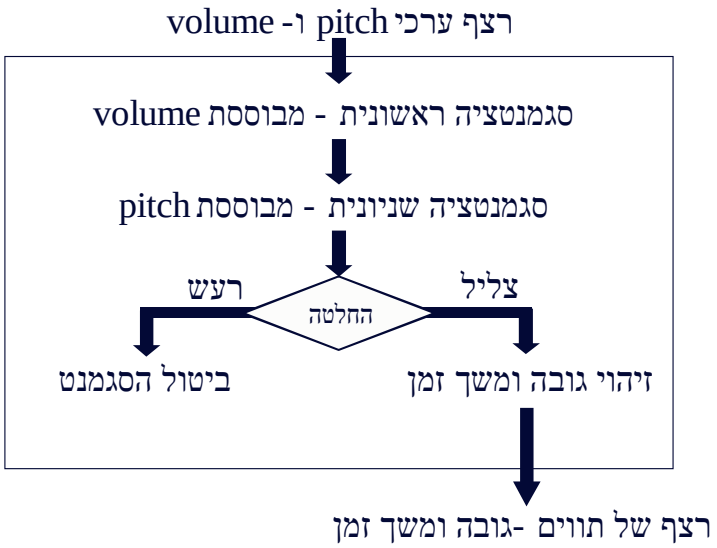
כדי להתעלם עד כמה שאפשר מערכי ה-pitch השגויים, נעשה שימוש באלגוריתם של מיצוע איטרטיבי, על מנת לזהות את התדר של כל תו. האלגוריתם פועל באופן הבא: ראשית, מחושבים הממוצע וסטיית התקן של איזור התו כולו. לאחר מכן, חוזרים על חישוב הממוצע, אך כעת נכללים בתהליך רק ערכי pitch שנמצאים בטווח של 2.5 סטיות תקן מהממוצע. באופן כזה ערכי pitch קיצוניים אינם נכללים בממוצע ואינם משפיעים עליו. מתקבל ממוצע חדש, וכעת מבצעים שוב את אותו תהליך, כאשר הפעם נכללים רק ערכי pitch שנמצאים בטווח של 2.2 סטיות תקן מהממוצע החדש. מיצוע זה חוזר על עצמו שוב ושוב, כאשר בכל פעם קטן טווח סטיות התקן המותר הולך וקטן ב0.3-. לפי משפט מתמטי, תהליך זה יביא בהכרח להתכנסות, והנסיון מראה שברוב המקרים 5 איטרציות מספיקות כדי לקבל ערך קרוב לגבול.

ערך הממוצע המתקבל לאחר האיטרציות הוא התדר שנבחר לייצג את התו הנידון. באופן כזה התקבל, מתוך רשימת ערכי pitch ו-volume, רצף של תווים בעלי תדר ומשך-זמן ידועים.

4.4 דיון ומסקנות

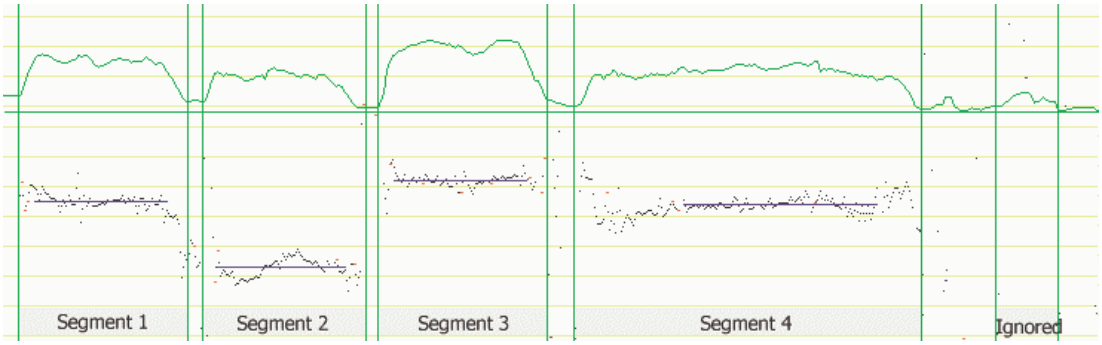
בפרק זה נידון התהליך המורכב של ביצוע סגמנטציה, או הפרדה מערכי pitch ו-volume כמעט רציפים לערכים בודדים של תדרי התווים. התהליך המורכב של הסגמנטציה מתואר באיור 19. רשימת ערכי ה-pitch וה-volume עוברת סגמנטציה ראשונית על ידי הפרדה לא-מדויקת לתווים בודדים, על סמך ערכי ה-volume באותו איזור. לאחר מכן מתבצעת סגמנטציה שניונית שמטרתה לזהות באופן מדויק את מקומות ההתחלה והסיום של כל תו, תוך הסתמכות על ערכי ה-pitch. על פי אורך התו, מוחלט האם מדובר ברעש או בצליל. במקרה

הראשון הסגמנט מבוטל, ואילו במקרה השני העיבוד ממשיך לתהליך של זיהוי גובה הצליל בעזרת מיצוע איטרטיבי.



איור 19: סיכום תהליך הסגמנטציה

דוגמה לתהליך הסגמנטציה כולו נתונה באיור 20. באיור זה, הגרף העליון מייצג את ה-volume, והגרף התחתון מייצג את ה-pitch. כל קו אופקי מייצג מרווח תדר של סמיטון אחד עבור ה-pitch (יחידות ה-volume הן שרירותיות). הקווים האנכיים מציינים את מקומות ההתחלה והסיכום של הסגמנטציה הראשונית. לעומתם הקווים האופקיים הכהים בכל סגמנט מייצגים את הטווח וגובה הצליל של כל אחד מהתווים כפי שנקבעו על ידי הסגמנטציה השניונית.



איור 20: תהליך הסגמנטציה

כפי שצוין בתחילת הפרק, תהליך הסגמנטציה הוא התהליך המורכב ביותר בעבודה, והוא מהווה את מקור השגיאה העיקרי בתהליך הזיהוי, למרות שברוב המקרים של שירה ברורה עם רעש מועט הוא פועל היטב. הסיבה לקושי היא שמבין שלושת השלבים של התוכנה (זיהוי pitch, סגמנטציה וחיפוש), זיהוי ה-pitch הוא תהליך סטנדרטי ומקובל, ותהליך החיפוש מושתת על תיאוריה מבוססת היטב של התאמת מחרוזות חלקית. לעומת זאת תהליך הסגמנטציה המוצג כאן פותח כולו עבור הבעיה הספציפית (והחדשה) של חיפוש מנגינה על סמך קלט קולי, ורובו פותח במהלך פרויקט זה.

עם זאת, יש לציין כי התהליך שתואר בפרק זה פועל בצורה טובה מאוד ברוב המקרים, תחת מגבלות ההגייה שהוגדרו בסעיף 1.3.4. בדיקה מפורטת יותר של יכולת הפעולה של האלגוריתם מתוארת בפרקים 6 ו7-, והצעות לשיפורים עתידיים במנוע הסגמנטציה מתוארים בסעיף 8.2.1.

5. חיפוש במאגר המידע

בפרק זה נתאר את השלב האחרון בביצוע זיהוי של מנגינה מוקלטת. בשלבים קודמים תורגמה ההקלטה לרצף של תווים, כאשר המידע הקיים בקשר לכל תו הוא אורכו והתדר שלו (פרקים 3-4). מטרת שלב החיפוש הינה להשוות בין רצף תווים זה לבין כל אחד מהשירים המצויים במאגר המידע של התוכנה, כדי למצוא את השיר או השירים הדומים ביותר להקלטה המקורית. בסופו של דבר תתקבל רשימה של השירים המתאימים ביותר, בצירוף מדד כשלהו למידת ההתאמה בין כל שיר להקלטה.

1.5 מטרת תהליך החיפוש

שני גורמים עיקריים מונעים מהתוכנה לבצע השוואה ישירה בין התווים של ההקלטה לבין התוים של השירים במאגר המידע: חוסר דיוק של המשתמש המבצע את ההקלטה, וחוסר דיוק של שלבים קודמים בתהליך הזיהוי.

מבחינת זיופים של המשתמש עצמו, טעויות נפוצות מאוד מתרחשות בעקבות התופעות הבאות:

- **טרנספוזיציה**, כלומר, שירה בסולם שונה מהסולם המקורי, המתבטאת בהזזה של כל התווים מספר קבוע של סמיטונים כלפי מעלה או מטה.
- **זיופים**; תחת כותרת רחבה זו נכנסות מספר תופעות, ביניהן חוסר דיוק בשירת התו המבוקש וחוסר יציבות ב-pitch במהלך שירת התו.
- **מגבלות הטווח הדינמי** של המשתמש, כלומר, חוסר יכולת פיסית של המשתמש לשיר תו גבוה או נמוך מאוד.
- **זכרון מוגבל**; לא תמיד זוכר המשתמש במדויק את השיר. ייתכן שהוא ישיר מספר תווים גבוה מדי או נמוך מדי, או ישלב בין שני חלקים דומים של אותו שיר. תופעה זו קיימת במיוחד במקרים שבהם ההיכרות של המשתמש עם השיר אינה טובה.
- **שירה חלקית** של המנגינה, כלומר, המשתמש מבצע חלק קצר יותר או ארוך יותר מהקטע המצוי בזכרון התוכנה.

עיוותים בטֶמְפּוֹ (קצב ניגון היצירה) מתבטאים הן בניגון היצירה בקצב שונה מהקצב המקורי שלה (בדרך כלל בקצב מהיר יותר), וכן בחוסר אחידות בטמפו לאורך היצירה.

בנוסף, יש לקחת בחשבון כי ייתכן ששלבים קודמים בתהליך הזיהוי לא פעלו באופן מושלם, וכתוצאה מכך ייתכנו התופעות הבאות:

- **זיהוי שגוי של תו** עשוי להיות תוצאה של זיהוי pitch לא נכון (לדוגמה, זיהוי הרמוניות או זיהוי שברים, כמתואר בסעיף 3.3.2).
- **החסרת תו** (כלומר, אי זיהוי תו הקיים בהקלטה) עשויה להתרחש במספר מקרים, לדוגמה: אם התו קצר מדי בכדי להחשב לצליל על ידי תהליך הסגמנטציה השניונית, או אם ה-volume של התו נמוך מדי בכדי לאפשר זיהוי על ידי תהליך הסגמנטציה הראשונית (ראו סעיף 4.3.2).
- **הכפלת תו** (כלומר, זיהוי של תו יחיד כשני תווים) עשויה להתרחש במקרים שבהם ה-volume של התו יורד ועולה חזרה, וגורם לכך שהתו מופרד לשני סגמנטים על ידי תהליך הסגמנטציה הראשונית.

ניתן לסכם ולומר שמגוון רחב של תנאים לא-אידאליים עשויים לגרום לעיוותים משמעותיים בין התווים שהתכוון המשתמש לבצע, לבין התווים שמגיעים בסופו של דבר למנוע החיפוש. רוב השגיאות שתוארו עשויות לגרום לאחת משלוש תוצאות אפשריות: הבדל בין התכונות האמיתיות לבין התכונות המזוהות של תו מסוים; דילוג על תו מסוים; או הכנסה של תו שאינו מצוי בשיר המקורי.

כיוון שכך, מטרתו של מנוע החיפוש היא לבצע התאמה "חכמה" בין רצף התווים המולקטים לבין מאגר המידע. הכוונה היא שטעויות בשלבים הקודמים של השירה והזיהוי יגרמו לירידה במדד ההתאמה בין השיר האמיתי להקלטה, אך עדיין יאפשרו זיהוי נכון של היצירה.

יש להזכיר כאן את העקרון המכונה GIGO (Garbage In, Garbage Out): אם המשתמש אינו מסוגל לשיר בצורה מינימלית, או אם אינו זוכר את המנגינה כלל, או אם התכוון לשיר יצירה אחת אך ביצע הקלטה דומה יותר ליצירה אחרת, אין ולא תהיה למחשב אפשרות לזהות באופן נכון את ההקלטה. כל שניתן לשאוף אליו מבחינת מנוע החיפוש הוא תיקון מספר מקסימלי של טעויות; האלגוריתם אינו יכול לעשות ניסים.

2.5 מרחק עריכה

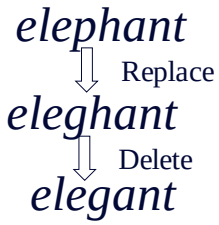
בסעיף הקודם ניתן תיאור של מגוון רחב של בעיות אשר מתרחשות באופן תדיר בעת ביצוע ההקלטה ושלבי העיבוד הראשונים שלה. על מנת לאפשר התאמה בין תווים מהקלטה לתווים ממאגר מידע, גם תחת השפעתם של מספר גדול של הפרעות אפשריות, נבחר להשתמש

במושג "מרחק עריכה" (edit distance), המכונה לפעמים גם "מחיר עריכה" (edit cost). שיטה זו מאפשרת לבצע התאמה בין מחרוזות גם אם חלו תופעות כגון מחיקת תו מתוך אחת המחרוזות, ואף כאשר אורכי המחרוזות אינם שווים (Myers and Miller, 1989; Myers, 1996; Baeza-Yates and Perleberg, 1992; Knight and Myers, 1992). קיימים מספר יישומים והרחבות של עקרון זה לצורך השוואה בין מחרוזות מוסיקליות (Prechelt and Typke, 1999; McNab et al., 1997; Ghias et al., 1997; Mongeau and Sankoff, 1990).

בבסיסה של שיטה זו עומד הרעיון הבא: דמיון בין שתי מחרוזות הוא שקול לכך שניתן להפוך מחרוזת אחת לשניה על ידי מספר קטן של "שינויים", או פעולות עריכה בסיסיות, שיוגדרו מיד. בסעיף זה נגדיר את עקרון מרחק העריכה עבור מחרוזות באופן כללי (מעל אלפבית כלשהו). בסעיף 4.5 תידון הבעיה של בחירת האלפבית והפרמטרים הספציפיים עבור מחרוזות של תווים מוסיקליים.

פעולות העריכה הבסיסיות הן החלפה (replace), מחיקה (delete) והוספה (insert). "החלפה" היא שינוי אות אחת במחרוזת; "מחיקה" היא הוצאת אות אחת מהמחרוזת (ובכך קיצור אורך המחרוזת ב1-); ו"הוספה" משמעותה הכנסת אות חדשה בין שתי אותיות קיימות במחרוזת (ובכך הארכת המחרוזת ב1-).

לדוגמה, כדי לשנות את המילה elephant למילה elegant יש צורך בשתי פעולות עריכה לפחות. דוגמה לתהליך עריכה שמבצע את השינוי הזה הוא התהליך הבא: ראשית מחליפים את האות P באות G, ולאחר מכן מוחקים את האות H (איור 21). קיימים גם תהליכי עריכה שונים, ביניהם תהליכי עריכה ארוכים יותר ואף כאלה שאורכם אינסופי.



איור 21: דוגמה למרחק עריכה

כדי להגדיר באופן מלא את מרחק העריכה מעל אלפבית מסוים, יש צורך לציין את ה"מחיר" של כל פעולת עריכה בסיסית. במודל שבו נשתמש, מחירים אלה יכולים להיות תלויים רק באותיות עליהן מתבצעת הפעולה ובמיקומן בתוך המחרוזת.[2] נהוג להגדיר את

[2] ניתן לפתח מודלים דומים שבהם ישנה תלות, למשל, באותיות הנמצאות בקרבת האות הנוכחית (Mongeau and Sankoff, 1990). מודלים אלה הינם בעלי סיבוכיות חישוב גבוהה יותר ולכן נמנע משימוש בהם בעבודה זו, שבה יש צורך לחשב מרחקי עריכה עבור מספר רב של מחרוזות.

המחיר של "עריכה ריקה" ("החלפה" של תו מסוים באותו תו) כ0-, כך שביצוע או אי-ביצוע פעולה כזו אינו משפיע על מחיר העריכה.

לאחר שהוגדר המחיר של כל פעולות העריכה הבסיסיות, ניתן להגדיר את המחיר של רצף פעולות עריכה בתור סכום מחירי הפעולות הבסיסיות המרכיבות אותו. מרחק העריכה (או מחיר העריכה) ממחרוזת אחת לשניה יוגדר כמחיר המינימלי של רצף פעולות עריכה אשר מתרגמות מחרוזת אחת לשניה[3].

מרחק העריכה הוא מדד לדמיון בין מחרוזות, אשר מאפשר למצוא קשרים בין מחרוזות גם כאשר ישנם הבדלים משמעותיים ביניהם, כגון אותיות החסרות באחד מהם. מרחק עריכה נמוך מציין כי שתי המחרוזות הן כמעט זהות, ואילו מרחק עריכה גבוה מצביע על קשר עקיף או אקראי בלבד.

נחזור כעת לדוגמה של איור 21. נניח לשם פשטות שמחיר כל פעולה בסיסית הוא 1. במצב כזה המחיר המינימלי שווה למספר המינימלי של פעולות עריכה הנדרשות כדי להפוך את המחרוזת הראשונה לשניה. העריכה המתוארת באיור 21 היא העריכה הקצרה ביותר האפשרית, ולכן מרחק העריכה מ-elephant ל-elegant הוא 2. במקרה זה המחירים הוגדרו באופן סימטרי, כך שגם מרחק העריכה בכיוון ההפוך (מ-elegant ל-elephant) הוא 2, אך ניתן להגדיר את המחירים באופן שתכונה זו לא תתקיים.

כדי שהמנגנון יפעל באופן משביע רצון, יש צורך בהגדרה נכונה של מחירי הפעולות הבסיסיות: הבדלים שהם בעלי משמעות רבה צריכים לקבל מחיר גבוה יותר מאשר הבדלים שעשויים להתחולל במקרה. הגדרת המחירים עשויה גם להשתנות בהתאם לאופי המערכת, סוג השימוש בה וסוג התיקונים שהיא צפויה לבצע. לדוגמה, עבור מערכת המחפשת מילים בעברית, יש לתת מחיר נמוך יחסית למחיקה והוספה של האות 'י', בכדי לאפשר התאמה בין מילים בכתיב חסר ובכתיב מלא. לעומת זאת, עבור מערכת תיקון שגיאות כתיב, ייתכן שהגדרת המחירים תהיה שונה, ותעניק מחיר נמוך גם להחלפה בין האותיות 'ק' ו-'כ', למשל.

3.5 התאמת מחרוזות מקורבת

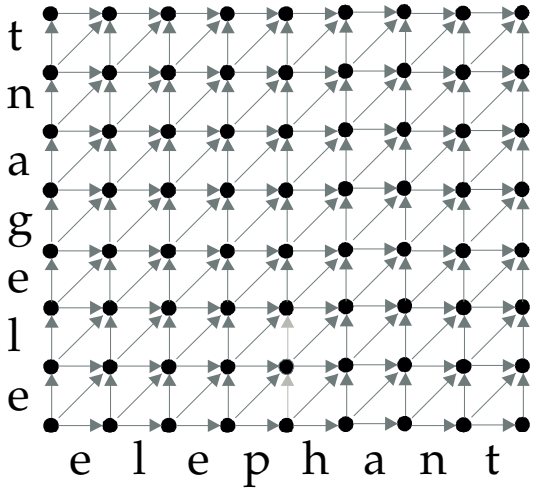
בסעיף הקודם ניתן תיאור תיאורטי של מדד לקירבה בין מחרוזות, הנקרא מרחק עריכה. זהו כלי חזק למציאת דמיון בין מחרוזות, גם כאשר קיימים מספר הבדלים ביניהן. לכן, במבט ראשון נראה שאלגוריתם למציאת ההתאמה בין המחרוזות יהיה בעל סיבוכיות חישוב

[3] רצף כזה תמיד קיים. לדוגמה, ניתן למחוק את כל האותיות של המחרוזת הראשונה ואז להוסיף את כל האותיות של המחרוזת השניה. לכן, מחיר העריכה הוא מושג מוגדר היטב.

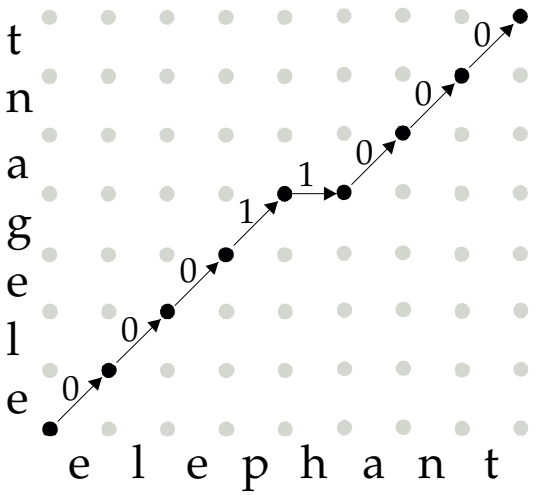
גבוהה. למרבה המזל, קיים אלגוריתם מהיר המבצע את החישוב בזמן $O(nm)$, כאשר n ו-m אורכי שתי המחרוזות ביניהם נערכת ההשוואה. האלגוריתם מכונה "אלגוריתם להתאמת מחרוזות מקורבת" (approximate string matching) (Myers, 1986; Myers and Miller, 1989).

1.3.5 שקילות לבעיה מתורת הגרפים

ניתן לתאר את האלגוריתם באופן מוחשי על ידי אנלוגיה לבעיה בתורת הגרפים. הגרף האנלוגי מכיל $(m+1)\times(n+1)$ צמתים המסודרים ברשת דו-מימדית (איור 22). המרווחים שבין עמודות הצמתים מתאימים לאותיות המחרוזת הראשונה, והמרווחים שבין שורות הצמתים מתאימים לאותיות המחרוזת השניה.



איור 22: גרף לחישוב מרחק עריכה



איור 23: מסלול העריכה בעל המחיר המינימלי

הגרף מכיל קשתות מכוונות מכל צומת לצומת שמעליו, לצומת שמימינו, ולצומת באלכסון לכיוון ימינה-למעלה. כל קשת אנלוגית לפעולת עריכה בסיסית אפשרית מהמחרוזת הראשונה לשניה, באופן הבא: קשת אופקית אנלוגית למחיקה של האות בעמודה המתאימה; קשת אופקית אנלוגית להוספה של האות בשורה המתאימה; וקשת אלכסונית אנלוגית להחלפת האות מהעמודה המתאימה באות מהשורה המתאימה.

על סמך הגדרות אלה של האנלוגיה, ניתן לראות כי כל רצף פעולות אשר הופך את המחרוזת הראשונה לשניה הוא אנלוגי למסלול העובר מהצומת השמאלי-תחתון לצומת הימני-עליון. מסלול כזה ייקרא "מסלול עריכה" (edit path). מסלול כזה יוצר התאמה בין כל אותיות המחרוזת הראשונה לבין כל אותיות המחרוזת השניה, תוך שימוש בחוקי העריכה הבסיסיים[4]. כאשר אות מסוימת נשארת ללא שינוי, אנו מניחים כי נבחרה הקשת האלכסונית עבורה, כיוון שע"פ הגדרת מרחק העריכה, זוהי "עריכה ריקה" שמחירה הוא אפס.

כדי למצוא את מחיר העריכה, לכל קשת בגרף מוקצה המחיר המתאים לפעולת העריכה שהיא מייצגת. כעת, ברור כי הבעיה האנלוגית למציאת מרחק העריכה בין שתי המחרוזות הוא מציאת מסלול העריכה בעל סכום מחירי הקשתות המינימלי. עבור מחירי פעולות בסיסיות של 1 לכל פעולה מלבד עריכה ריקה, מסלול העריכה המינימלי (איור 23) הוא אכן האנלוגי לרצף הפעולות המתאים לעריכה בעלת המחיר המינימלי.

[4] רצפי פעולות אשר מחליפים את אותה אות מספר פעמים אינם ניתנים לייצוג בשיטה זו, אך ההנחה היא כי רצפי פעולות כאלה הינם תמיד בעלי מחיר גבוה יותר מאשר החלפה ישירה של האות לערכה הסופי, ולכן לצורך חישוב מרחק העריכה אין עניין ברצפי פעולות כאלה.

2.3.5 פתרון הבעיה השקולה

עד כה תוארה בעיה בתורת הגרפים והוסברה השקילות שלה לבעייה המקורית של מציאת מרחק עריכה בין שתי מחרוזות. כעת נראה כי קיים פתרון יעיל לבעיה השקולה, ובאנלוגיה, מתאים פתרון זה גם לבעיה המקורית.

כזכור, הגדרת הבעיה השקולה היא כדלקמן: נתון גרף מכוון דומה לאיור 22, שבו לכל קשת ישנו מחיר ידוע. יש למצוא את המסלול מהצומת השמאלי-תחתון לצומת הימני-עליון, שיש לו מחיר מינימלי (איור 23).

פתרון הבעיה מתבצע על ידי הצמדת "מחיר מצטבר" p_{ij} לכל צומת בגרף. המחיר המצטבר של צומת כלשהו הוא סכום המחירים המינימלי הדרוש כדי להגיע לצומת זה מהצומת השמאלי-תחתון. הקואורדינטות מציינות את מיקום הצומת בגרף, כאשר הצומת השמאלי-תחתון יהיה בעל קואורדינטת 0,0.

הצומת השמאלי-תחתון הוא בעל מחיר מצטבר אפס. מחירי הצמתים בשורה התחתונה ובטור השמאלי ניתנים לחישוב ישיר, מכיוון שיש רק מסלול אחד המגיע אליהם. המחיר של שאר הצמתים מחושב באופן איטרטיבי, ע"פ הנוסחה:

$$p_{ij} = \min(p_{i-1,j} + d,\quad p_{i,j-1} + i,\quad p_{i-1,j-1} + c)$$

כאשר d הינו מחיר המחיקה המתאימה למקום הנוכחי בגרף, i הינו מחיר ההוספה המתאימה ו-c הינו מחיר ההחלפה המתאימה. משמעות ביטוי זה היא, שהמחיר המינימלי של צומת מסוים נבחר על פי המחיר המינימלי הכולל שייגבה בעקבות ביצוע כל אחת משלושת הפעולות הבסיסיות, על גבי מחרוזת שבה כבר בוצעו כל שאר השינויים.

תוך שימוש בנוסחה זו ובתנאי ההתחלה (מחירי השורה והעמודה הקיצוניים), ניתן לחשב את המחיר המינימלי עבור העמודה והשורה השניים, ובעזרתם להמשיך הלאה אל השורות הבאות. לאחר מילוי כל יתר הצמתים בגרף יהיה ניתן לחשב גם את המחיר המינימלי הדרוש עבור הצומת הימני-עליון, וזה מחיר העריכה של הגרף כולו.

האלגוריתם שתואר מאפשר למצוא את מסלול העריכה המינימלי ואת מחיר העריכה המתאים לו, תוך ביצוע $O(mn)$ פעולות חיבור. האלגוריתם דורש אחסון מערך בגודל $O(mn)$ אשר מכיל את המחירים המצטברים של הצמתים[5].

---

[5] קיים אלגוריתם דומה אשר מוצא את מחיר העריכה אך לא את מסלול העריכה, וסיבוכיות הזכרון שלו היא $O(n+m)$ בלבד. סיבוכיות הזמן של אלגוריתם זה אינה שונה. עם זאת, במקרה של ההשוואות הנפוצות בעבודה זו הזכרון הנצרך זניח ביחס לזכרון הנתפס, למשל, על ידי ההקלטה עצמה. מכיוון שאלגוריתם משופר זה הינו פחות קריא ואינטואיטיבי, הוא לא הוכנס לשימוש בתוכנה.

מכיוון שהבעיה בתורת הגרפים שקולה לבעיה המקורית, ניתן להפעיל את הפתרון גם על הבעיה המקורית, בלי שינוי בסיבוכיות הזמן והמקום של הפתרון. הוצג, אם כן, פתרון יעיל לבעיית חישוב מרחק העריכה.

4.5 תרגום תווים מוסיקליים למחרוזת

בסעיף 1.5 תואר מגוון רחב של שגיאות העלולות להווצר במהלך שלבי ההקלטה והזיהוי. שגיאות אלה מסתכמות בתופעות של פעולות עריכה בין מחרוזת התווים המקורית לבין מחרוזת התווים שזוהתה.

עיוות יוצא דופן מבחינה זו הוא טרנספוזיציה, התופעה שבה אדם שר מנגינה בסולם שונה מהסולם המקורי של היצירה. כמעט כל בני האדם אינם זוכרים את הסולם שבו מבוצע שיר, וכאשר הם שרים אותו הם בוחרים סולם באופן אקראי. זו הסיבה שבקטעים מוסיקליים שבהם אנשים שונים שרים יחד, ישנה בדרך כלל התחלה אינסטרומנטלית, שתפקידה "להכניס" את כל הזמרים לאותו סולם.

מבחינה אקוסטית ההשפעה של תופעת הטרנספוזיציה היא תרגום כל התווים למעלה או למטה בכמות מסוימת (Taylor, 1992). לכן, התדר המדויק של כל תו ביצירה, בפני עצמו, אינו תורם לזיהוי; המידע החשוב נמצא רק במרווחים שבין התווים. לכן, הגיוני לבצע את תהליך החיפוש רק על רצף המרווחים בין התווים, ולא על התדרים האבסולוטיים שלהם.

לעומת הטרנספוזיציה, שאת השפעתה ניתן לבטל לחלוטין בגלל אופיה הרגולרי, העיוותים האחרים הם הרבה יותר אקראיים. כמו כל תופעת עיוות, ניתן לתאר את העיוותים האלה בעזרת שלוש פעולות העריכה הבסיסיות של סעיף 2.5: שינוי, מחיקה והוספה. היתרון בבחירת פעולות עריכה בסיסיות אלה הוא כפול: מצד אחד, רוב העיוותים שתוארו בסעיף 1.5 גורמים לשינוי אשר מתאים בדיוק לפעולת עריכה בסיסית; ומצד שני, הפעולות הינן רב-תכליתיות וכלליות מספיק על מנת לאפשר תיקון גם במקרים של שגיאות יוצאות דופן. לכן, ניתן לצפות שמסלול העריכה האופטימלי בין המנגינה המוקלטת והמקורית יעקוב בקירוב אחר העיוותים שליוו את תהליך ההקלטה והזיהוי.

מול אפשרות השימוש בשלוש פעולות עריכה כלליות ורב-תכליתיות, קיימת גם אפשרות לנסות ולהתאים פעולות עריכה מורכבות יותר, אשר יתאימו באופן מדויק יותר לסוגי העיוותים הצפויים (Mongeau and Sankoff, 1990; McNab et al., 1997). היתרונות והחסרונות של שתי הגישות יתוארו בקצרה בסעיף 8.2.2; השוואה בין גישות אלה היא נושא למחקר המשך.

בעבודה זו נתמקד במרחק עריכה המבוסס על שלוש הפעולות הבסיסיות של סעיף 2.5. נותר לקבוע כיצד יתורגמו התווים המזוהים לאותיות במחרוזת (כלומר, מהו האלפבית), ומה יהיה המשקל של כל סוגי פעולות העריכה, כך שתהליך החיפוש יתקן באופן אופטימלי את העיוותים של השלבים הקודמים.

הגישה המקובלת ברוב התוכנות הקיימות תכונה כאן "מינימליסטית", וקוראת להשתמש באלפבית קטן ככל האפשר, המכיל מידע מתומצת על כל תו, על מנת להקטין את מידת ההשפעה של זיופים בשירה (McNab et al., 1996; Prechelt and Typke, 1999; Ghias et al., 1997). האלפבית המקובל במקרים אלה הוא קוד Parsons (1975), אשר יתואר בסעיף 1.4.5. כאנתיתזה לגישה זו, ניתן להרחיב את האלפבית ולהכליל בו גם מידע שדיוקו מוטל בספק, אך להגדיר את מחירי השינויים כך שיקחו בחשבון צורות שגיאה נפוצות (גישה "מקסימליסטית"). שני אלפביתים כאלה, שפותחו במהלך עבודה זו, יתוארו בסעיפים 2.4.5 ו-3.4.5. ההשוואה בין גישות מנוגדות אלה תיעשה באמצעות סימולציה, בסעיף 6.3.2, ובאמצעות בדיקה אמפירית, בסעיף 7.3.

1.4.5 חיפוש ע"פ קוד Parsons

קוד Parsons פותח בשנת 1975 והיווה את הנסיון הראשון לאפשר לאדם ללא הכשרה מוסיקלית לחפש קטעי מוסיקה על סמך מנגינתם. השיטה מבוססת באופן קיצוני על הגישה המינימליסטית: Parsons בחר לבנות אלפבית המכיל שלוש אותיות בלבד. אותיות אלה מסמנות את תדר התו הנוכחי ביחס לתו הקודם: עליה בתדר (U), ירידה בתדר (D) או חזרה על אותו תו (R). לדוגמה, תחילת השיר "יונתן הקטן" תיוצג ע"י המחרוזת "DRUDRDUUUURR". רוב האנשים מסוגלים, לאחר אימון מסוים, להבדיל בין עלייה, ירידה וחזרה, ועל ידי כך לרשום קוד Parsons עבור המנגינה אותה הם מחפשים.

היישום המקורי של קוד Parsons היה ספר של כמה עשרות אלפי מנגינות קלאסיות, אשר נרשמו בעזרת קוד זה ומוינו לפי סדר אלפבית. כדי לחפש מנגינה, יש למצוא את קוד Parsons שלה ולחפש אותו לפי סדר האלפבית שלו, בדומה לחיפוש במילון.

נראה, אם כן, כי מקורה של הגישה המינימליסטית לחיפוש מנגינות נובעת לא רק מהרצון למנוע שגיאות, אלא גם מהצורך לפשט את תהליך החיפוש, על מנת לאפשר לאדם חסר הכשרה מוסיקלית לבצע את החיפוש ללא עזרה חיצונית. כיום, כאשר החיפוש מתבצע בעזרת מחשב, יש צורך לבחון מחדש את טיבה של הגישה המינימליסטית. בדיקה מחודשת של הגישה המינימליסטית תיערך בסעיף 8.1.4.

קוד Parsons מומש כאלפבית החיפוש במספר מחקרים קודמים (McNab et al., 1996; Ghias et al., 1997), אך רק במחקר אחד ניתן פירוט של מחירי פעולות העריכה הבסיסיות (

Prechelt and Typke, 1999), ואלה הן: למחיקה או הוספה של R ניתן מחיר 1; למחיקה או הוספה של D או U ניתן מחיר 2; ולכל פעולות ההחלפה ניתן מחיר אינסופי. מחירים אלה נמצאו על ידי אופטימיזציית תוצאות החיפוש על כ100- הקלטות. בפרק 7 יתואר תהליך אופטימיזציה דומה שנערך בעזרת התוכנה הסופית של עבודה זו. בתהליך זה התקבלו תוצאות שונות במידת מה מתוצאותיהם של Prechelt and Typke. הסיבה לכך היא שאופן פעולת תוכנה זו שונה בצורה מהותית מהתוכנה של Prechelt and Typke; כתוצאה מכך, סוגי השגיאות ושכיחויותיהן שונים, וצורת התיקון האופטימלית לשגיאות אלה משתנה גם היא.

2.4.5 חיפוש ע"פ דמיון תדר

מול הגישה המינימליסטית של קוד Parsons, ניתן לגשת לבעיית החיפוש על ידי בניית אלפבית שיכלול כמות גדולה של מידע, מתוך ידיעה מראש שהמידע באלפבית זה לא יהיה מדויק. לדוגמה, במקום לרשום רק את כיוון השינוי בתדר, ניתן להכליל בעבודה גם את המרווח המוסיקלי בין הצלילים. צורת חיפוש זו תכונה חיפוש ע"פ דמיון תדר (frequency similarity search).

כחלק מהגישה המקסימליסטית, הוחלט לציין את המרווחים ביחידות של סֶנט. זאת, למרות שרוב בני האדם אינם מסוגלים להבחין במרווח של סנט אחד, ועל אחת כמה וכמה אינם יכולים לשיר בדיוק שכזה. האלפבית ייוצג כאן כאוסף המספרים השלמים, כאשר 0 מציין חזרה מדויקת על אותו תו, מספר חיובי x מציין עלייה ב-x סנט, ומספר שלילי $-x$ מציין ירידה של x סנט. לשם פשטות נניח כי המספרים האלה שלמים (מימוש האלגוריתם כ-fixed-point).

נותר כעת "לתַמְחֵר" את האלפבית. כלומר, יש לקבוע מחירים לכל הפעולות הבסיסיות.

כאשר אדם מחפש מנגינה, הוא מתבקש לשיר לפחות 10 תווים ממנה, אך ייתכן שיבצע מעט יותר או פחות, ואין כל סיבה שמספר התווים שהוא מבצע שווה למספר התווים של המנגינה במאגר המידע. כדי לאפשר התאמה בין מנגינות שאורכן שונה, הוגדר מחיר אפס למחיקות והוספות כאשר הן חלות בסוף המחרוזת (Ghias et al., 1997).

נניח כעת שהמשתמש מקליט שיר בעל 30 תווים, ואנו משווים אותו למאגר המידע. נניח ששיר מסוים במאגר המידע מכיל עשרה תווים בלבד. אזי, על ידי ביצוע עשר החלפות על עשרת התווים הראשונים של ההקלטה, ניתן לקבל את השיר שבמאגר המידע באופן מדויק; אך ברור, שהתאמה כזו אינה מצביעה על קשר בין השירים. לעומת זאת, אם שיר אחר במאגר המידע מכיל 30 תווים, וע"י שינוי של עשרה תווים נוצרת התאמה לשיר המוקלט, הרי שהתאמה זו אינה מקרית לחלוטין.

המסקנה מדוגמה זו היא, ששיטת תְמְחוּר חייבת להתחשב באורכי המחרוזות על מנת להיות הוגנת. על המחיר להיות יחסי הפוך לאורך המחרוזת. השאלה היא, באורכה של איזו מחרוזת יש להתחשב: השיר המוקלט או השיר במאגר המידע? כדי לענות על השאלה נתייחס לשני מקרים נפרדים: פעולות החלפה ופעולות הוספה/מחיקה.

במקרה של פעולת ההחלפה, אם מחליפים את כל האותיות במחרוזת הקצרה, ניתן להגיע להתאמה "מקרית" עם המחרוזת הארוכה יותר, ע"פ השיטה שתוארה למעלה. לכן, מחירי פעולות החלפה צריכים להקבע ביחס הפוך לאורך המחרוזת הקצרה.

לעומת זאת, כדי להגיע להתאמה מקרית בעזרת פעולות הוספה ומחיקה, יש למחוק את כל המחרוזת הראשונה ולאחר מכן להוסיף את כל המחרוזת השניה. לכן, מחירי פעולות הוספה ומחיקה נקבעים ביחס הפוך לסכום אורכי שתי המחרוזות.

כעת יש לקבוע את המחיר היחסי של סוגי הפעולות השונות. אמנם אנו מצפים לכך שהתאמה מדויקת בין תווים תהיה נדירה מאוד, אך אנו מניחים שזיופים קטנים יהיו נפוצים יותר מזיופים גדולים. הוחלט, לכן, שהמחיר לשינוי מתו a לתו b (המיוצגים לפי הסימון שהוגדר למעלה) יהיה פרופורציונלי ל-$|a-b|$.

לעומת זאת, המחיר של מחיקה או הוספה אינו תלוי כמעט בערך התו אשר מתווסף או נמחק. כיוון שכך, הוחלט שמחיר מחיקה והוספה יהיה קבוע, ותלוי רק באורכי המחרוזות כפי שתואר למעלה.

מחירי שלוש הפעולות הבסיסיות מסוכמים בנוסחאות הבאות:

$$Cost\left[\text{change}(a \to b)\right] = \frac{\alpha|a-b|}{\min(m,n)}$$

$$Cost\left[\text{delete}(a)\right] = \frac{\beta}{m+n}$$

$$Cost\left[\text{insert}(a)\right] = \frac{\gamma}{m+n}$$

כאשר: a,b - ייצוג בסנטים של כל תו.

α,β,γ - קבועים שאת ערכם יש לקבוע.

m,n - אורכי שתי המחרוזות.

קביעת ערכי הקבועים נעשתה על ידי אופטימיזציה על קבוצה של הקלטות בתנאי-אמת, כמתואר בסעיף 7.2.

3.4.5 חיפוש משולב תדר/משך-זמן

בשני האלגוריתמים שהוצגו עד כה, נעשה שימוש רק בתדר של כל תו על מנת לבצע את החיפוש. אך במוסיקה נהוג לאפיין תווים על פי שני פרמטרים: תדר ומשך-זמן (duration).

מחקרים פסיכולוגיים מראים כי משך-זמן הינו גורם חשוב באופן שבו בני אדם זוכרים מנגינות (Dowling, 1978). כיוון שכך, הוחלט לנסות לבצע חיפוש על סמך שילוב המידע של תדר ומשך זמן (frequency/duration search).

עם זאת, התזמון של בני אדם באורכי שירת תווים אינו גבוה. הטמפו שבו אדם מבצע את היצירה אינו בהכרח הטמפו המקורי שלה, וכן ייתכנו שינויים בטמפו תוך כדי היצירה. באופן כללי, הדיוק בתדר גבוה יותר מהדיוק במשך הזמן. לכן, הוחלט לייצג את משך הזמן במונחים כללים בלבד, וליצור עבורו אלפבית של שלוש אותיות, בדומה לקוד Parsons. עבור כל תו מאופיין משך הזמן בהשוואה לתו הקודם: ארוך יותר (U), קצר יותר (D) או שווה בקירוב (עד כדי 30%) (R) למשך הזמן של התו הקודם.

האלפבית המלא הוא כעת מכפלה ישרה של אלפבית התדרים שהוצג בסעיף 2.4.5, עם אלפבית משכי הזמן. כדי שתתקיים התאמה טובה בין שני תווים, נדרשת התאמה גם בין התדרים וגם בין משכי הזמן. תכונה זו רומזת על אופי התאמה כְּפָלִי (multiplicative), כלומר, מחיר ההתאמה בין תווים הוא מכפלה של מחירי התאמה לתדרים ולמשכי זמן (Estes, 1994; pp. 19-21).

ברוח זו, הוחלט להשתמש במחירים הקיימים עבור התדרים, ולהכפילם בפקטור המייצג את מידת הקרבה בין משכי הזמן: אם משכי הזמן שווים (עד כדי אופן הייצוג באלפבית בעל שלוש האותיות), הפקטור הוא 1; אם משכי הזמן קרובים (החלפה בין U ל-R או בין D ל-R), הפקטור יהיה גדול מ1-; ואם משכי הזמן מנוגדים (מ-U ל-D ולהיפך), הפקטור יהיה גדול עוד יותר. מחירי הוספה ומחיקה אינם תלויים לא בתדר ולא במשך הזמן, ונשארים זהים לערכיהם בשיטת התדר. כמקודם, ערכי הפקטורים נקבעו בתהליך אופטימיזציה שיתואר בסעיף 7.2.

5.5 סיכום

תהליך החיפוש הוא השלב האחרון בפעולת התוכנה הסופית. זהו שלב שבו נעשה מאמץ לתקן שגיאות בתהליכים קודמים בתוכנה. מרחק העריכה מאפשר למצוא דמיון בין מחרוזות תווים, גם כאשר חלו עיוותים מסוגים שונים על המחרוזות: מחיקת תווים, הוספת תווים ושינויי תווים. כיוון שכך, התאמת מחרוזות על ידי חישוב מרחק עריכה הינה שיטה המתאימה לסוגי הטעויות הנפוצים בשירה ובשלבי העיבוד הקודמים של התוכנה.

בפרק זה הוצגו מספר אלגוריתמים אפשריים לביצוע תהליך החיפוש. שיטת קוד Parsons מוציאה כמות מינימלית של מידע מתוך התווים, מתוך מטרה למנוע ככל האפשר את השפעתם של טעויות וזיופים. שיטת דמיון תדר, לעומת זאת, משתמשת בתדר המדויק של כל

תו, ובכך מכילה כמות רבה יותר של על כל תו, אם כי מידע זה הינו פחות מדויק. שיטת החיפוש המשולבת תדר/משך-זמן משתמשת בנוסף במידע של משך זמן הצלילים.

בפרקים הבאים תיערך השוואה בין אלגוריתמים אלה. תתואר סימולציה שבודקת את הדיוק של האלגוריתמים תחת תנאים מבוקרים (סעיף 6.3), וכן תתואר בדיקה אמפירית בתנאי אמת, מהקלטות של שירת בני אדם (סעיף 7.3).

6. סימולציה

בכדי לבדוק את פעולת המערכת, נערכו בדיקות נפרדות על כל אחד משלושת החלקים העיקריים של המערכת: מנגנון זיהוי ה-pitch, אלגוריתם הסגמנטציה ומנוע החיפוש. מטרת בדיקות אלה היתה לוודא את תקינות המערכות, לבדוק את פעולתן בצורה מבוקרת ולקבל תיאור של מידת הדיוק בפעולתן. השימוש בכלי הסימולציה לצרכים אלה מאפשר שליטה מדויקת על הפרמטרים של הניסוי, שאינה קיימת כאשר נערך ניסוי בתנאי אמת. בנוסף, העובדה שהסימולציה אינה דורשת קלט אנושי אפשרה לבצע את הבדיקות על מדגם רחב של קלטים, ובכך לקבל מדגם סטטיסטי רחב יותר של התוצאות.

מובן, שסימולציות אינן יכולות להחליף לחלוטין בדיקה של המערכת בתנאי אמת. לכן, נערך ניסוי מלא של המערכת כיחידה אחת, בעזרת הקלטות של בני אדם. ניסוי זה יתואר בפרק הבא.

1.6 זיהוי Pitch

בפרק 3 נערכה סקירה של שיטות זיהוי pitch מקובלות, ונבחרה שיטת זיהוי המתאימה ביותר לתנאים הספציפיים של עבודה זו. בסעיף זה תיבדק איכות זיהוי ה-pitch תחת תנאים מבוקרים: בהקלטות של אותות מסונתזים, בהקלטות של כלי נגינה ובהקלטות של שירה והמהום.

1.1.6 בדיקת קולות מסונתזים

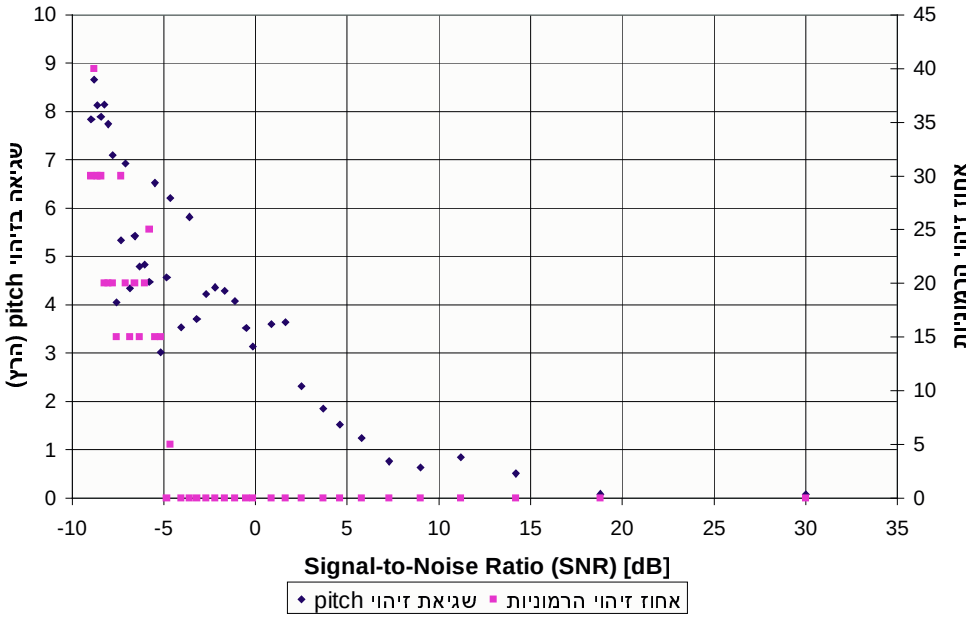
בשלב ראשון נערכה בדיקה על אות מסונתז במחשב, על מנת לוודא שאלגוריתם הזיהוי פועל באופן תקין, וכן על מנת לבדוק את רגישותו לרעשים מבוקרים.

איור 24 מציג תוצאות זיהוי של גל סינוס שהוסף לו רעש גאוסי לבן באמפליטודות שונות. בגרף זה, המעויינים מציגים את סטיית התקן של ההפרש בין התדר המזוהה לתדר האמיתי, ואילו הריבועים מציגים את אחוז המקרים שבהם חל "זיהוי הרמוניות" (ראה סעיף 3.2.3). ערכי זיהוי הרמוניות לא נכללו בחישובי סטיית התקן, אלא נחשבו כ"ערכים חסרים" לצורך חישוב זה.

מגרף זה ניתן לראות כי עבור ערכי SNR סבירים (מעל 5 dB), לא מתרחשות שגיאות זיהוי הרמוניות כלל, וסטיית התקן של הזיהוי קטנה מ1- הרץ. עבור ערכי SNR גבוהים יותר,

סטיית התקן של הזיהוי יורדת מתחת ל0.01- הרץ, דיוק גבוה מאוד שמתאפשר הודות לאלגוריתם הסופר-רזולוציה. חוסר דיוק שעלול לגרום לזיהוי שגוי, וכן זיהוי הרמוניות, מופיעים רק במקרים של אותות בעלי SNR שלילי, כלומר רעש שהאמפליטודה שלו גדולה מהאמפליטודה של גל הסינוס.

השפעת רעש לבן על דיוק זיהוי pitch



איור 24: השפעת רעש לבן על איכות הזיהוי

תוצאות אלה מצביעות על חסינות גבוהה מאוד לרעש לבן. עם זאת, יש להזהר מפרשנות מרחיקת לכת של תוצאות אלה, מכיוון שברוב המקרים, הרעשים המופיעים יחד עם האות המקורי הם בעלי מחזוריות אחת או יותר (כגון הרמוניות או פורמנטים). רעשים כאלה עשויים לגרום לבעיות זיהוי קשות יותר, בעקבות זיהוי מחזוריות של הרעש במקום ה-pitch.

כדי לבדוק זאת, נערכו מספר הרצות של קבצים מסונתזים של גל סינוס בסיסי ("pitch") בתוספת רעשים מחזוריים מסוגים שונים. לכל אות התבצעו כ100- זיהויי pitch. הממוצע וסטיית התקן של חישובים אלה מוצגים בטבלה הבאה, כאשר תדר ה-pitch הנכון הינו 132 הרץ.

יכולת זיהוי pitch עבור אותות בעלי רעש מחזורי

תיאור האות	תדר מזוהה (הרץ)	
	ממוצע	סטיית תקן
$\sin(\omega_0 t)$	132.00	0.00
$\sin(\omega_0 t)+2\sin(2\omega_0 t)$	132.00	0.00
$\sin(\omega_0 t)+\sin(2\omega_0 t)+\sin(3\omega_0 t)$	132.00	0.00
$\sin(\omega_0 t)+0.2\sin(1.7\omega_0 t)$	130.95	1.87
$\sin(\omega_0 t)+0.4\sin(1.7\omega_0 t)$	128.30	4.73
$\sin(\omega_0 t)+0.3\sin(1.7\omega_0 t)+0.2\sin(2.4\omega_0 t)$	132.34	7.24
$\sin(\omega_0 t)+0.3\sin(1.7\omega_0 t)+0.2\sin(2.4\omega_0 t)+0.2\sin(15.5\omega_0 t)$	131.93	7.08

מטבלה זו ניתן לראות כי הרמוניות של תדר ה-pitch אינן גורמות לזיהוי שגוי, גם כאשר האמפליטודה שלהן גבוהה מהאמפליטודה של האות המקורי. תוצאה זו צפויה אם שמים לב לעובדה שתוספת הרמוניות אינה משנה את המחזור של האות, אלא רק את הצורה שחוזרת על עצמה, כפי שניתן לראות מהשוואה בין איור 6 לבין איור 7 בפרק 3.

לעומת זאת, תוספת של תדרים שאינם כפולות שלמות של התדר הבסיסי גורמת לירידה בדיוק הזיהוי. דיוק הזיהוי יורד ככל שגדלים מספר התדרים הנוספים ועוצמתם. בהקלטה של קול אחד צפויים רעשים כאלה להופיע בעיקר ממקורות חיצוניים, ולכן עוצמתם לא צפויה להיות גבוהה. אך במקרים של הקלטה של מספר קולות סימולטניים (אשר אינם חלק מדרישות פרויקט זה), עלול אלגוריתם הזיהוי לפעול באופן שגוי. במקרים כאלה, לכן, יהיה צורך להשתמש בשיטות זיהוי אחרות, כגון זיהוי ספקטרלי.

2.1.6 זיהוי אותות מפסנתר

עד עתה נבדקו תכונות הזיהוי של המערכת על אותות מסונתזים. נעבור כעת לבדיקה של אותות שהוקלטו מפסנתר. אותות אלה דומים יותר לאותות שישמשו את התוכנה הסופית בכך שהם מוקלטים בעזרת מיקרופון וכוללים תנאי רעש מציאותיים. מצד שני, בהקלטה מפסנתר ידוע ערך ה-pitch האמיתי (עד כדי דיוק הכיוון של הפסנתר), וכן ידוע שה-pitch נשאר קבוע לכל אורך התו המוקלט. זהו מידע שאינו קיים בהקלטות אנושיות, שבהן ה-pitch אינו ידוע ובדרך כלל משתנה באופן רציף. לכן, הקלטות מפסנתר מאפשרות לבדוק את תקינות הזיהוי של התוכנה בתנאי אמת.

כדי לבדוק את טיב הזיהוי, הוקלטו 5 תווים מפסנתר. בוצע זיהוי pitch על כל אחד מהתווים לאורך זמן, וחושבו הממוצע וסטיית התקן של ה-pitch. תוצאות אלה מוצגות בטבלה הבאה.

זיהוי pitch בהקלטה מפסנתר

תו	תדר מזוהה (הרץ)	סטיית תקן (סֶנט)	תדר תיאורטי (הרץ)	הפרש (סֶנט)
דו (C-4)	254.9	5.51	264-261	45.2-
רה (D-4)	285.3	9.85	297-293	49.5-
מי (E-4)	321.3	11.09	330-329	44.2-
פה (F-4)	338.4	5.12	352-349	54.3-
סול (G-4)	380.6	16.39	396-392	51.1-

בטבלה זו, התדר התיאורטי מציין את טווח התדרים המקובל עבור אותו תו, בעזרת tempers שונים (ראו פרק "מונחים ויחידות", עמ' 12-13). טווח התדרים נועד לתת סדר גודל לטווח השינויים המקובלים באותו תו, שאינם נחשבים לזיופים חמורים.

מהטבלה ניתן לראות שישנו הפרש קונסיסטנטי של כחצי סמיטון בין התדרים המזוהים לבין התדרים התיאורטיים. ניתן לזקוף הבדל זה לעובדה שהפסנתר עליו בוצעה ההקלטה אינו מכוון. סטיית התקן של הזיהוי, אשר אינה מושפעת מחוסר הכיוון, הינה נמוכה מאוד (5-10 סנט בקירוב), עובדה המעידה על דיוק גבוה בזיהוי ה-pitch גם בהקלטות בתנאי אמת.

3.1.6 זיהוי קולות אנושיים

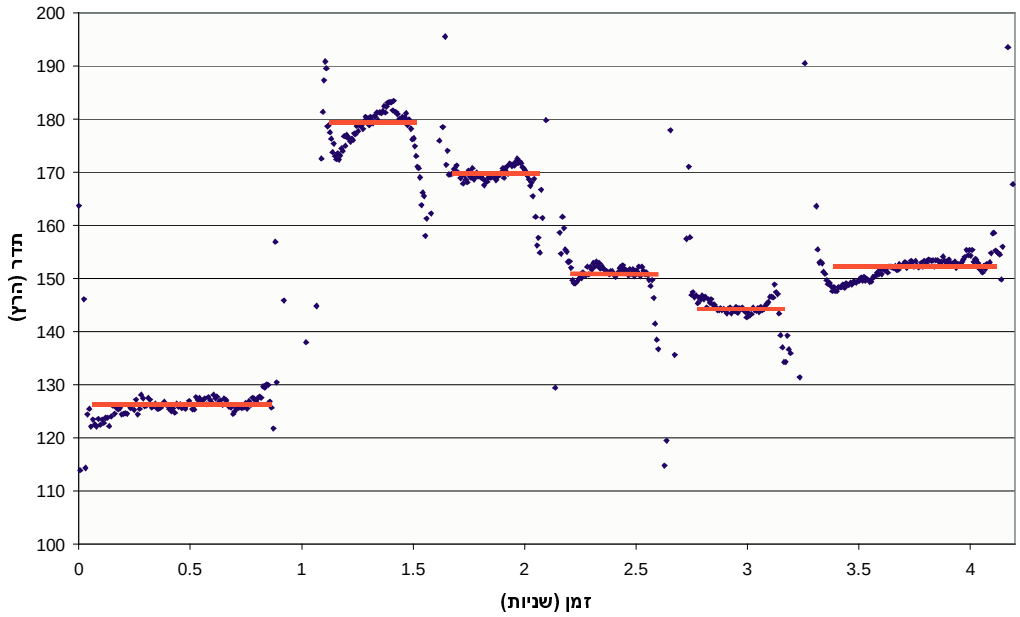
בסעיפים הקודמים נבדק טיב הזיהוי של אלגוריתם זיהוי ה-pitch, והוצגו תוצאות המעידות על כך שבתנאי הקלטה סבירים, זיהוי ה-pitch מתבצע באופן מדויק. בסעיף זה נשתמש באלגוריתם זיהוי ה-pitch כדי לבדוק את טיב הזיהוי של קולות אנושיים.

כפי שניתן לצפות, הקול האנושי הוא הרבה יותר רב-גוני מקולות מסונתזים ואף מכלי נגינה כמו הפסנתר. התוצאות של זיהוי pitch אנושי הינן הרבה פחות אחידות, ותלויות במידת המיומנות של המבצע ובאופי השירה. כמובן שלא ניתן לכסות את מגוון האפשרויות הקיימות במספר סביר של הקלטות, אך ננסה להציג כאן מספר דוגמאות שונות ולהסיק מסקנות בנוגע לאופי הכללי של ה-pitch האנושי.

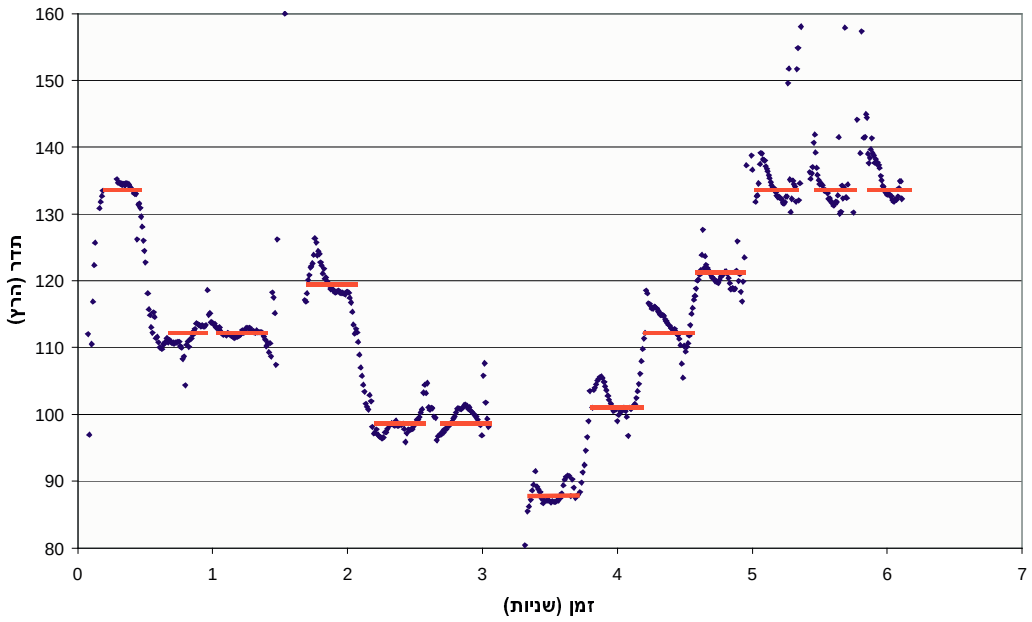
בעמודים הבאים מוצגות מספר דוגמאות של זיהויי pitch של קולות אנושיים. ההקלטות בוצעו בתנאי הקלטה רגילים (כפי שתוארו בסעיף 1.1.1), על ידי שלושה אנשים. ההקלטות בוצעו במספר שיטות הגייה שונות, כדוגמת המהום, שריקה, והגיית הברות שונות. בנוסף, מוצגים בגרפים קווים אנכיים המציינים חלוקה ידנית סובייקטיבית לצלילים בדידים, בהתבסס על אופי הגרף ועל התווים האמיתיים של השיר.

מתוך הגרפים האלה ניתן ללמוד על מספר תופעות חשובות הקיימות בקולות אנושיים, שלא הופיעו בסעיפים הקודמים. התכונה הבולטת ביותר היא העובדה שקיימים שינויים משמעותיים ב-pitch לאורך תו יחיד, אשר אמור תיאורטית להכיל תדר קבוע. שתי תופעות גורמות לתוצאה זו: שינויים אקראיים ושינויים רציפים.

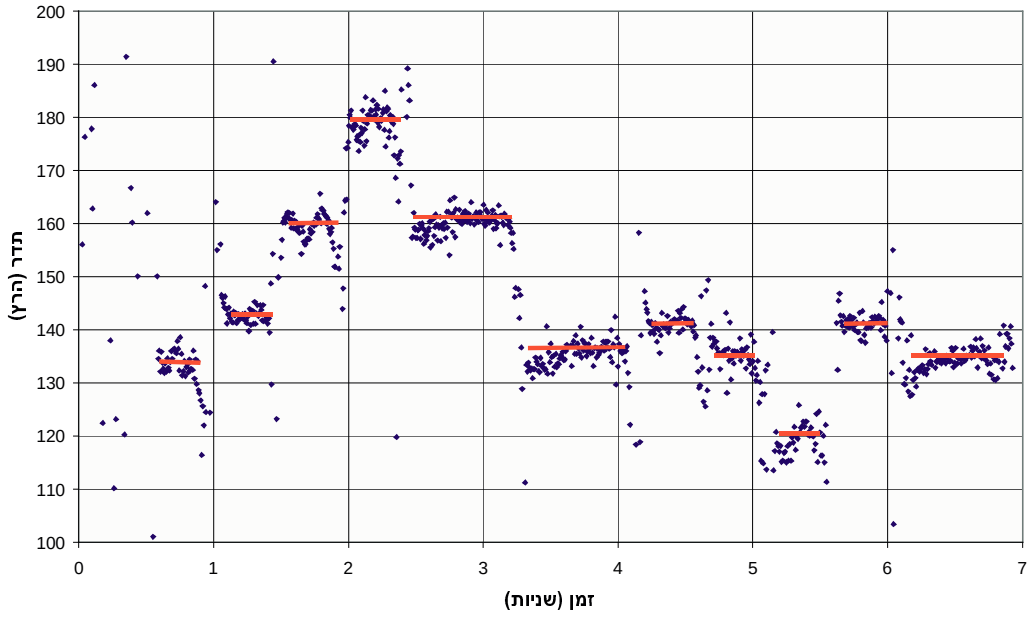
שינויים אקראיים הם שינויים מהירים ב-pitch, אשר בולטים במיוחד באיור 27. שינויים אלה הם קפיצות מהירות סביב ערך ממוצע, ונמשכים לכל אורך התו. סטיית התקן של שינויים אלה יכולה להגיע עד כ2- הרץ, אך נעה סביב 1 הרץ ברוב המקרים. למרות שתופעה זו אינה מופיעה בכל סוגי השירה, ברור כי יש צורך להשתמש בסוג כלשהו של מיצוע בעת תרגום מידע ה-pitch הגולמי למידע על תווים, כפי שאכן מבוצע במנגנון הסגמנטציה (פרק 4).



איור 25: מנואט של באך; שירה ע"י הגיית ההברה "הָא"



איור 26: "יונתן הקטן", שירה באמצעות הגיית ההברה "לָה"



איור 27: "נר לי נר לי", הגייה באמצעות המהום

במקרים אחרים ניתן לראות שינויים "רציפים" ב-pitch. שינויים כאלה בולטים, למשל, באיור 26. השינויים מופיעים באיזור שבין שני תווים, והם תוצאה של העובדה שה-pitch האנושי אינו יכול להשתנות באופן פתאומי לחלוטין. כתוצאה מכך, ישנו תחום מעבר בין

תווים סמוכים, שבו ה-pitch עובר במהירות בין שני התדרים. זמרים מקצועיים מודעים למגבלה זו של הקול האנושי, ומונעים אותה עד כמה שאפשר על ידי הורדת ה-volume בין תווים (Levarie & Levy, 1980). אך הבעיה מופיעה בצורה בולטת בשירה לא-מקצועית, במיוחד באופני שירה שבהם יש מעבר חלק מתו לתו, כגון שירה באמצעות הגיית ההברה "לָה". בהמהום ובצורות שירה מסוימות אחרות, שבהן ישנה עצירה בין כל שני תווים, הבעיה כמעט שאינה מופיעה. עובדה זו הינה הבסיס לסגמנטציית ה-volume שתוארה בפרק 4.

מעבר לעובדה שנקלטים קטעי מעבר בין שני תווים, תופעה הנובעת ממגבלות פיסיולוגיות של מערכת הדיבור, ישנן תופעות נוספות של זיופים של ממש. לדוגמה, תופעה החוזרת על עצמה פעמים רבות היא "overshoot" במעבר בין תווים. בנסיון לעבור מתו גבוה לנמוך, המשתמש יורד לתדר נמוך מדי, ואז "מתקן" על ידי עלייה חזרה לערך הסופי (לדוגמה, המעבר מהתו השלישי לרביעי באיור 26). בעיה דומה מתרחשת במעברים בכיוון ההפוך. רוב המשתמשים אינם מודעים לכך שהם מבצעים פעולה זו. ייתכן שתופעה זו היא תוצאה של משוב בלתי מודע בין מערכות הדיבור והשמיעה, אשר פועלות יחד לתיקון זיופים של צלילים תוך כדי השמעתם.

מעבר לכך, קיימים מקרים רבים של חוסר יציבות בתדר ללא כל סיבה נראית לעין, לעיתים עד כדי שינויים של סמיטון ויותר במהלך שירת תו מסוים. כך לדוגמה מתרחש בתווים האחרונים של איור 26, וכן (במידה פחותה יותר) באיור 25. תוצאות אלה אינן ניתנות להסבר מלבד העובדה שהזמר אינו מוסיקלי במיוחד. קיומם של בעיות אלה מעידה על כך שגם בתנאי סגמנטציה אידאליים, לא יהיה ניתן לקבל תווים מדויקים לחלוטין מהמשתמש. לכן, יש צורך במערכת התאמה גמישה מאוד בין התווים המוקלטים לתווים במאגר המידע, על מנת לאפשר התאמה גם במקרים של זיופים. מערכת חיפוש כזו תוארה בפרק 5.

2.6 סגמנטציה

מטרת תהליך הסגמנטציה, שתואר בפירוט בפרק 4, הינה להפריד את נתוני ה-pitch וה-volume הרציפים לתווים בדידים, ולזהות את התדר שלהם. תהליך זה מתבצע, בתנאי אמת, על גבי אותות מוקלטים בעלי מגוון רחב של תכונות. על מנת לבדוק את התהליך, הוחלט להשתמש בהקלטות בתנאי אמת, ולמצוא את סוגי השגיאות על ידי השוואה ידנית לנקודות ההתחלה והסיום האמיתיות של כל תו.

לצורך כך, הוקלטו 36 שירים שביצעו תשעה משתמשים שונים. ההקלטות הכילו בין 10 ל33- תווים עם ממוצע של כ17- תווים. ה-pitch של הקלטות אלה זוהה בעזרת התהליך המשמש את התוכנה הסופית (סעיף 3.3.1). לאחר מכן בוצעה סגמנטציה. הסגמנטים שזוהו

נבדקו באופן ידני בהשוואה להקלטה המקורית ובהשוואה לתווים הידועים של השיר, מתוך מטרה למצוא שגיאות בתהליך הסגמנטציה האוטומטי. השגיאות חולקו לארבעה סוגים:

- חוסר הפרדה בין שני תווים שונים
- פיצול שגוי של תו יחיד לשני סגמנטים
- התיחסות שגויה לסגמנט מוסיקלי כאל רעש, או להיפך
- משך-זמן לא נכון באופן מובהק (שינוי של 50% או יותר)

הטבלה הבאה מציגה את עיקרי הממצאים של בדיקה זו.

אפיון שגיאות בתהליך הסגמנטציה

סוג השגיאה	מספר אירועים	הסתברות הופעה	
		בשיר	בתו
חוסר הפרדה	12 (ב5- שירים)	14%	2%
פיצול שגוי	5 (ב5- שירים)	14%	1%
זיהוי רעש	2 (ב2- שירים)	6%	< 0.5%
משך-זמן שגוי	17 (ב11- שירים)	31%	3%
סה"כ	36 (ב18- שירים)	50%	6%

עמודת "מספר אירועים" מציגה את מספר הפעמים שבהם שגיאה מסוימת התרחשה, וכן את מספר השירים שבהם הופיעה שגיאה כזו לפחות פעם אחת. עמודת "הסתברות הופעה בשיר" מתארת את אחוז המקרים שבהם שיר מסוים הכיל לפחות שגיאה אחת מהסוג הנתון. עמודת "הסתברות הופעה בתו" מתארת את אחוז התווים שהכילו שגיאה מהסוג הנתון.

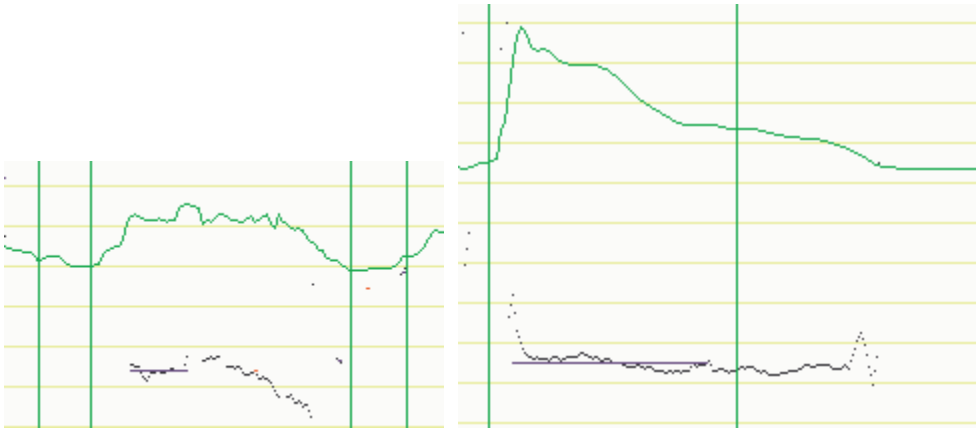
בניגוד לשגיאות בזיהוי pitch, שהינן נדירות מאוד ומופיעות בעיקר במצבים "פתולוגיים" כגון רעש רקע בעל עוצמה דומה לעוצמת ההקלטה, ניתן לראות כי שגיאות בסגמנטציה קיימות בתנאי אמת: מחצית השירים הכילו שגיאה אחת או יותר. כ20%- מהשירים הכילו יותר משגיאה אחת, ואחוזים בודדים הכילו יותר משלוש שגיאות.

העובדה שתהליך הסגמנטציה אינו פועל בצורה מושלמת אינה מפתיעה. היא נובעת מאופיו של התהליך: בניגוד לשלבי זיהוי ה-pitch והחיפוש, שלב זה דורש מידה רבה של הבנה ואף ניחוש של כוונת המשתמש, וכן תלויה בכך שהמשתמש יעקוב אחר צורת ההקלטה הנדרשת ממנו (מה שאינו תמיד מתרחש באופן מושלם).

חלק עיקרי משגיאות הסגמנטציה הינן שגיאות במציאת משך-הזמן של תו מסוים. שגיאות אלה הינן, כמעט תמיד, בחירת קטע קצר מדי מהתו כסגמנט. שני מקרים אפייניים לכך מוצגים באיור 28 (כמו באיורים הקודמים מסוג זה, גם כאן הקו העליון מציין volume, הקו

התחתון מציין pitch והקו הישר בתוך גרף ה-pitch מציין את האיזור שזוהה בסגמנטציה השניונית). במקרה (א) הסגמנטציה הראשונית מסמנת את סיום הסגמנט מוקדם מדי עקב ירידה ב-volume. במקרה (ב) הסגמנטציה השניונית קוטעת את התו מוקדם מדי בשל איזור קצר של חוסר זיהוי pitch המופיע באמצע.

שכיחות תופעת קיצור התו גורמת לירידה באמינות המידע של משכי-הזמן של התווים. ייתכן כי זו אחת הסיבות לכך שחיפוש משולב תדר/משך-זמן אינו משפר בהרבה את ביצועי המערכת, לעומת חיפוש דמיון-תדר (ראו סעיף 7.3).

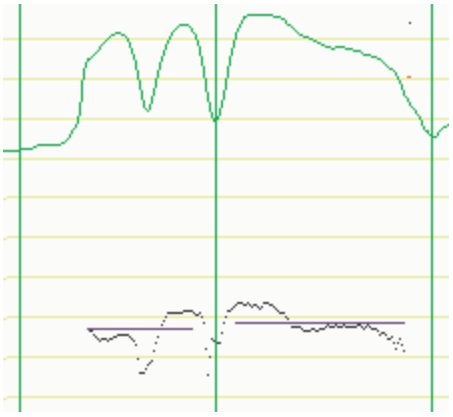


(ב) (א)

איור 28: שני מקרים של זיהוי תו קצר מדי

שגיאה נפוצה נוספת היא חוסר הפרדה בין שני תווים. ככלל, תופעה זו מתרחשת רק במצבים שבהם אין עצירה מלאה בין התווים (לדוגמה, ראו איור 29, שבו אמורה להיות הפרדה לשלושה סגמנטים). תופעה זו מתרחשת בעיקר כאשר השירה אינה מתבצעת בעזרת הגיית ההברות "טָה" או "טִי" (כנדרש בהגדרת הקלט של התוכנה), או כאשר השירה היא מהירה מאוד. התוכנה הסופית מציגה אזהרות המתארות את אופי הבעייה בשני המקרים האלה (ראו נספח א').

שני סוגי השגיאות הנוספים שנמצאו בבדיקה הינם זיהוי שגוי של סגמנט בתור רעש, ופיצול של תו יחיד לשני סגמנטים. שגיאות אלה התחוללו במקרים נדירים מאוד ולכן לא ניתן להסיק מידע אמין על המצבים הגורמים לשגיאות אלה. עם זאת, יש לציין כי שגיאת הפיצול היא, במובן מסוים, הפוכה לשגיאת חוסר ההפרדה בין תווים: אם תוגדל הרגישות להפרדה בין תווים גם כאשר ה-volume ביניהם אינו יורד לחלוטין, יירבו המקרים שבהם התוכנה תפצל באופן שגוי תו יחיד לשני תווים. לכן, קיים tradeoff בין הרצון למזער את השגיאות האלה, ויש צורך בפשרה מסוימת ביניהן.



איור 29: דוגמה לחוסר הפרדה בין תווים

פירוט זה של סוגי השגיאות מעיד על כך שמנוע הסגמנטציה רחוק עדיין מלהיות מושלם. אך למרות מגבלותיו, יש לציין כי באופן כללי התהליך פועל באופן משביע רצון. בתנאי האמת שבהם נבדקה המערכת, ניתן היה ברוב המקרים לבצע זיהוי נכון כאשר חלו עד שתי שגיאות סגמנטציה. כ80%- מהשירים הכילו עד שתי שגיאות סגמנטציה, ואכן אחוזי הזיהוי בתנאי אמת היו 80%, שהם גבוהים מיכולת הזיהוי של אדם ממוצע (סעיף 7.3).

3.6 חיפוש במאגר המידע

השלב האחרון בתוכנה הינו חיפוש של התווים מההקלטה במאגר מידע של שירים ידועים. שלב זה מנסה לבצע התאמה שתתגבר על זיופים של המשתמש ועל שגיאות בתהליכים קודמים במאגר המידע. כדי לבדוק את תקינות תהליך החיפוש, נערכו שני ניסויים. הראשון השווה בין שלושת האלגוריתמים שתוארו בסעיף 5.4, וכן בדק את תלות דיוק החיפוש במספר התווים שבשאילתה. הניסוי השני בדק את השפעת מספר השירים במאגר המידע על דיוק החיפוש.

1.3.6 תיאור מערכת הניסוי

כדי לבצע סימולציה של שלב זה בוצע התהליך הבא: נבחר שיר באקראי מתוך מאגר המידע, ומתוכו נלקחו התווים הראשונים בלבד. לכל אחד מהתווים האלה הוסף רעש אקראי, באופן הבא: לתדר הוסף רעש מפולג אחידות בטווח של ± 100 סנט; משך הזמן הוגדל או הוקטן במשתנה אקראי מפולג אחידות בין 0% לבין 50%. בדיקות לא רשמיות שנערכו על הקלטות בתנאי אמת הראו כי רוב ההקלטות היו בעלות שגיאות וזיופים קטנים מערכים אלה. (לדוגמה,

שינוי של 100 סנט משמעותו מעבר לתו הבא; כאשר שינוי כזה מופיע בביצוע של כלי נגינה, הוא צורם גם לאנשים שאינם מכירים את המנגינה כלל.)

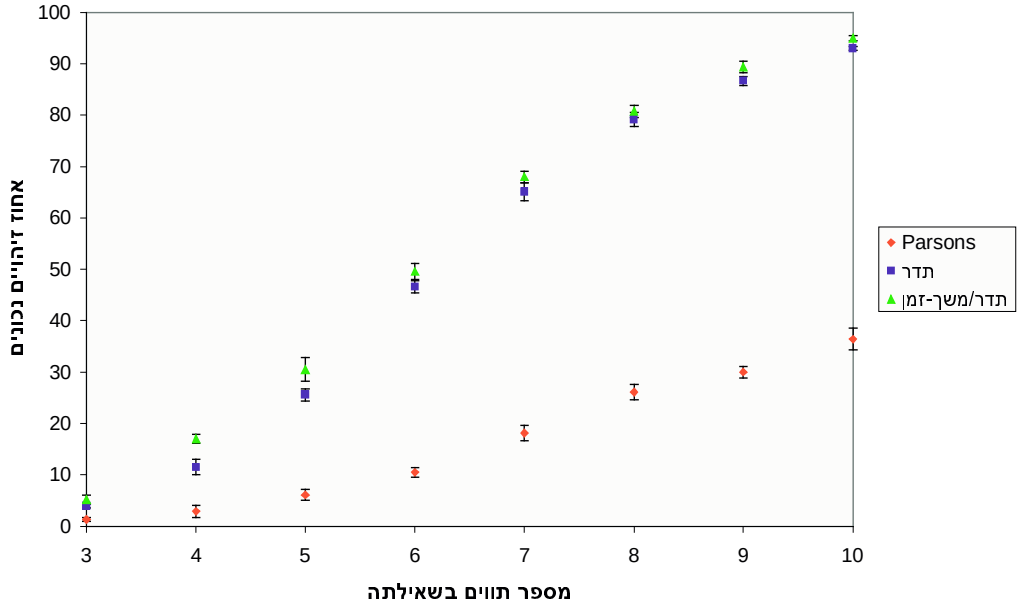
לאחר הוספת הרעש, הועברה רשימת התווים דרך תהליך החיפוש, ונבדק האם עדיין חל זיהוי נכון של השיר. השאלה שנבדקה היתה: האם, על פי אופן ההשוואה של אלגוריתם החיפוש, רשימת התווים בעלת ה"זיופים" קרובה יותר לשיר המקורי מאשר לכל שיר אחר?

עבור כל ערך של המשתנים הבלתי-תלויים, נערכה בדיקה זו 2000 פעם. בכל פעם נבחר קטע שיר חדש והוסף לו רעש אקראי חדש. בצורה כזו התקבל ערך מדויק של אחוז הזיהוי, תחת תנאי הרעש והפרמטרים הנתונים.

2.3.6 השוואה בין האלגוריתמים והתלות באורך השאילתה

כדי להשוות בין שלושת האלגוריתמים שתוארו בסעיף 5.4, נערכה סימולציה של מנוע החיפוש מעל מאגר מידע של 200 מנגינות אמיתיות. הסימולציה בדקה גם את תלות הסתברות הזיהוי באורך השאילתה. תוצאות הסימולציה מוצגות באיור 30.

הקווים האנכיים הינם מדד לסטייה הסטטיסטית האפשרית של כל נקודה. כדי לחשב את ערכי הקווים האלה, חולקו 2000 החזרות ששימשו לקביעת כל נקודה ל5- ניסויים (בעלי 400 חזרות כל אחד), וחושב הממוצע וסטיית התקן של תוצאות ניסויים אלה. הממוצע מוצג כנקודה האמצעית, והקווים האנכיים מציינים את סטיית התקן.



איור 30: תלות הסתברות הזיהוי באלגוריתם החיפוש ובאורך השאילתה

תוצאות הגרף מראות בבירור כי ישנה עלייה באחוז הזיהוי ככל שאורך השאילתה גדל. ההסבר לתופעה זו הוא ששאילתה קצרה עשויה להיות דומה לתחילתם של מספר גדול של שירים. לכן, כאשר מתווסף רעש, גדל הסיכוי שהתוצאה תהיה קרובה יותר לשיר שגוי. לעומת זאת, בשאילתה ארוכה, הסיכוי שרעש אקראי יקרב את השאילתה לשיר שגוי הינו נמוך מאוד. זאת מכיוון שכל תו המתווסף לשאילתה מוסיף דרגת חופש להבחנה בין שירים, כך שכושר ההפרדה בין שירים גדל.

תוצאה נוספת הבולטת מאיור 30 הינה שקיים הבדל משמעותי בין האלגוריתם המינימליסטי המבוסס על קוד Parsons, לבין שני האלגוריתמים המקסימליסטיים המבוססים על מידע מדויק יותר: קוד Parsons הינו בעל ביצועים נמוכים בהרבה מהאלגוריתמים האחרים. תוצאה זו היא עדות לכך שלמרות הזיופים הקיימים בשירה אנושית, מנגינות המבוצעות על ידי שירה עדיין מכילות כמות רבה של מידע מעבר לתיאור הבסיסי של קוד Parsons, וניתן להשתמש במידע זה כדי לקבל זיהוי מדויק יותר.

ההבדל בין שיטת דמיון התדר לבין השיטה המשולבת תדר/משך-זמן הוא פחות בולט. עם זאת, ישנו יתרון עקבי לשיטה המשולבת, אשר מובילה על שיטת התדר בכל המקרים שנבדקו בהפרש משמעותי סטטיסטית. ההבדל בין השיטות משמעותי במיוחד במקרים של שאילתות קצרות מאוד, שבהן כמות המידע קטנה מאוד, וכל תוספת מידע (במקרה זה – משך הזמן של התו) תורמת לטיב הזיהוי. ניתן לשער, כי במקרים של מאגרי מידע גדולים יותר, שבהם יש צורך בשימוש בכל כמות המידע הקיימת, ישוב ויופיע הבדל משמעותי בין שיטת התדר והשיטה המשולבת.

באשר לסיכויי הזיהוי עצמם, ניתן לראות כי בשני האלגוריתמים הטובים יותר מתקבל אחוז זיהוי של כ94%- עבור שאילתה באורך 10 תווים. אחוזי זיהוי גבוהים אלה מעידים על כך שניתן לבצע חיפוש יעיל גם כאשר קיים זיוף במנגינה המוקלטת. אחוז הזיהוי המעשי שיתקבל במערכת תלוי גם בגורמים אחרים, כגון דיוק מנגנון הסגמנטציה. כדי למצוא את הסתברות הזיהוי בתנאי אמת, יש לבצע ניסוי של המערכת כולה על הקלטות אנושיות. ניסוי כזה יתואר בפרק הבא.

3.3.6 תלות הזיהוי בגודל מאגר המידע

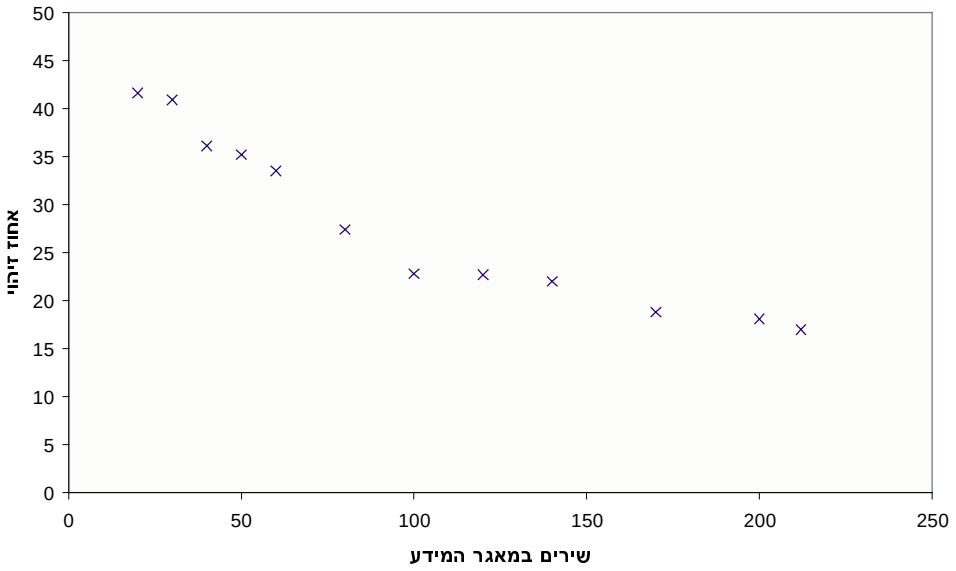
כל הבדיקות שנערכו על דיוק ההתאמה של הקלטות למאגר המידע הסתמכו על מאגר מידע של כ200- שירים, שנכתב במהלך העבודה. בגלל מגבלות טכניות, וביניהן הקושי בהשגת חוברות תווים שמהן נבנה מאגר המידע, לא היה ניתן להרחיב את מספר השירים במאגר מעבר לגודל זה. מאגר מידע כזה מיועד להדגמת פעולת התוכנה, אך ודאי שאינו שימושי ביישום מעשי. לשם השוואה, בספריית הקונגרס של ארצות הברית קיימים מעל שישה מליון גליונות

של תווים מוסיקליים (McNab et al., 1995). ברור כי מחשוב תווים אלה הינו משימה בפני עצמה.

כיוון שהשאיפה היא שתוכנת הזיהוי תוכל לפעול גם על מאגרי מידע גדולים בהרבה, יש צורך בבירור התלות שבין גודל מאגר המידע לבין אחוזי הזיהוי של התוכנה. למרבה הצער, הגדלה של מאגר המידע מובילה כמעט באופן בלתי נמנע לירידה ביכולת הזיהוי של התוכנה. בסעיף זה נדגים ונסביר עובדה זו, ונבדוק את יכולת ההתמודדות של התוכנה עם מגבלה זו.

כפי שהוסבר בפרק 5, תהליך החיפוש במאגר המידע הוא נסיון להתאים בין שתי מחרוזות אשר אינן זהות לחלוטין. יכולת הזיהוי של התוכנה מתבססת על כך שלמרות הבדלים אלה, תהיה ההקלטה דומה יותר למנגינה שהיא מייצגת מאשר לכל מנגינה אחרת. ואולם, ככל שקיימות יותר מנגינות, כך גדל הסיכוי שמנגינה כלשהי תתאים באופן מקרי למנגינה המוקלטת, יותר מאשר המנגינה המקורית.

מכיוון שאין אפשרות להרחיב את מאגר המידע על מנת לבדוק תיאוריה זו, הוקטן מאגר המידע באופן מלאכותי על ידי הוצאה אקראית של שירים מתוכו. כתוצאה מכך התקבלו מגוון גדלים של מאגר המידע, ונבדקה יכולת הזיהוי של התוכנה בתהליך סימולציה, עבור אורך קלט קבוע של 4 תווים. התוצאות מתוארות באיור 31, שבו נתונים אחוז המקרים שבהם התוכנה זיהתה נכון את השיר כעדיפות ראשונה. אחוז הזיהוי הנמוך יחסית הוא תוצאה של העובדה שהשאילתה קצרה מאוד.



איור 31: אחוז הזיהוי כפונקציה של מספר השירים במאגר המידע

ניתן לראות כי אכן ישנה ירידה מונוטונית ביכולת הזיהוי של מנוע החיפוש, כאשר מאגר המידע הולך וגדל. אחוז הזיהוי יורד מכ40%- עבור מאגרי מידע בגדלים של 20-30 שירים, עד לכ18%- במאגר מידע של 200 שירים.

בטרם נעבור להצעת שיטה להתמודדות עם בעיה זו, יש מקום לומר מילה טובה על הגדלת מאגר המידע, שכן התוצאות המוצגות כאן מעוותות במידה מסוימת את המציאות. בניסוי הסימולציה שתואר, נבחר שיר מתוך מאגר המידע הקיים, הוסף לו רעש, ואז התבצע חיפוש בעזרת אותו מאגר מידע. במקרים של מאגרי מידע קטנים, השיר נבחר מתוך מאגר המידע הקטן. הדבר שקול לכך שהמשתמש יתבונן ברשימת השירים הידועה לתוכנה, יבחר מתוכה שיר וישיר אותו. זהו שימוש המתאים לניסוי לבדיקת טיב הזיהוי, אך הוא אינו מתאים ליישום פרקטי, שבו המשתמש מבצע שיר מזכרונו ואינו יודע אם השיר נמצא במאגר המידע. אילו הגדרת הניסוי היתה בחירת שיר מתוך מאגר המידע הרחב ביותר, ונסיון לזהותו בעזרת מאגרי מידע בגדלים שונים, התוצאות היו מראות יתרון עצום למאגרי מידע גדולים. זאת מכיוון שככל שהמאגר גדול יותר, כך גדל הסיכוי שהוא מכיל את השיר המבוקש. לכן, תיאור המערכת כאילו היא פועלת טוב יותר עם מאגרי מידע קטנים הוא נכון רק מנקודת מבט אחת. אין בתיאור זה כדי לומר שיש צורך בשימוש במאגר מידע קטן ככל האפשר ביישום פרקטי.

למרות האמור לעיל, איור 31 מראה בבירור כי ביצועי המערכת, בהנתן כי ההקלטה היא של שיר ידוע, קטנים כאשר מאגר המידע גדל. כיצד ניתן להתמודד עם בעיה זו? הפתרון הוא פשוט: יש לספק למחשב שאילתה בעלת מספר גבוה יותר של תווים.

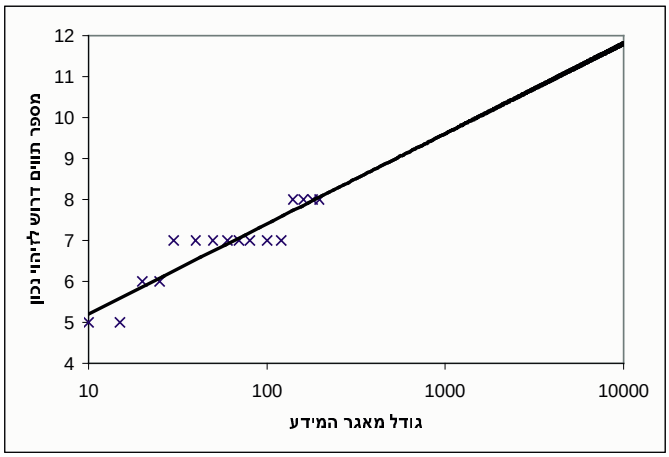
המצב הוא אנלוגי לחיפוש במילון. אין צורך לדעת את אופן הכתיבה של מילה עד לאות האחרונה כדי למצוא אותה במילון. למשל, כדי למצוא את המילה "קריפטון" מספיק לדעת את ארבע האותיות הראשונות שלה, כיוון שאף מילה אחרת בעברית אינה מתחילה באותיות "קריפ". רוב המילים בעברית ובאנגלית ניתנות להבחנה (עד כדי הטיות) בעזרת 4-5 אותיות בלבד. ואולם, אילו היו בשפה העברית 20 מליון מילים, אחוז המקרים שבהם חמש אותיות היו מספיקות לתאר את כל המילים היה מגיע ל25%- לכל היותר. זאת מכיוון שלא קיימים מספיק צירופים של אותיות לתיאור יותר מכ5- מליון מילים. אך על ידי צירופים של שש מילים ניתן היה להפריד בין כל המילים בשפה.

האנלוגיה של חיפוש במילון מדגימה את העובדה שהוספת אות אחת (או תו אחד) למחרוזת גורמת להוספת דרגת חופש מבחינת מספר המחרוזות הקיימות. כתוצאה מכך, מספר המחרוזות (או המנגינות) הניתנות להפרדה גדל אקספוננציאלית כאשר מגדילים את אורך המחרוזת.

האנלוגיה לתווים מוסיקליים אינה מושלמת, כיוון שלא מוגדר בצורה מדויקת גודל אלפבית של תווים, ולכן אין דרך לדעת מראש מה מספר המנגינות הניתנות להפרדה. אך אפילו אם נתייחס לתווים רק כאל קודי Parsons (שהם אלפבית בעל שלוש אותיות בלבד), ניתן

בעזרת קוד סביר של 15 תווים להפריד בין כ14- מליון מנגינות. בעזרת טווח כה רחב של מנגינות אפשריות, ישנם סיכויים מצויינים ששתי מנגינות הנבחרות באקראי מתוך מאגר מידע בגודל סביר, יהיו בעלי תיאור שונה ב"מילון" כזה. ואכן, קיים מדריך המפריד בין כ60,000- מנגינות קלאסיות בעזרת קוד Parsons באורך 15 (Parsons, 1975).

תיאור הבעיה מבחינה תיאורטית מסתבך עוד יותר כאשר מתחשבים באפקטים של התאמה לא-מדויקת. לכן, במקום חישוב תיאורטי, נערכה סימולציה של חיפוש במאגרי מידע בגדלים שונים. עבור כל גודל, נמצא אורך השאילתה המינימלי הדרוש על מנת לקבל 80% זיהוי. תוצאות ניסוי זה מוצגות כ-X-ים באיור 32.



איור 32: אורך השאילתה הדרוש כפונקציה של גודל מאגר המידע

תוצאות אלה רומזות על כך שישנו קשר אקספוננציאלי בין אורך השאילתה לבין גודל מאגר המידע שניתן לזהות, גם במצב של תווים מוסיקליים בתוספת רעש. עם זאת, בשל כמות המדידות הקטנה, אשר מוגבלת על ידי גודל מאגר המידע המקסימלי שהיה ניתן לבדיקה, יש לראות תוצאות אלה כתמיכה בתיאוריית התלות האקספוננציאלית, ולא כהוכחה חד-משמעית. הוכחה חותכת תינתן אם תיערך בדיקה בתנאי אמת על מאגרי מידע בטווח גדלים רחב יותר.

על ידי רגרסיה של נתונים אלה ניתן לקבל את הנוסחה המקורבת הבאה, אשר מוצגת באופן גרפי באיור 32:

$$N = C \cdot b^m$$

$$C = 4.40 \cdot 10^{-2}$$

$$b = 2.84$$

כאשר N הינו גודל מאגר המידע, ו-m הינו מספר התווים הדרוש על מנת לקבל 80% זיהוי מעל מאגר מידע כזה. מידת ההתאמה של נוסחה זו לנתונים המדודים הינה $R^2 = 0.86$.

על ידי אקסטרפולציה של נוסחה זו, ניתן לראות כי גם עבור מאגרי מידע גדולים ניתן יהיה לבצע זיהוי בעזרת מספר סביר של תווים. יש להניח כי מקדמי הרגרסיה נותנים ערכי אקסטרפולציה לא מדויקים במיוחד, ולכן ערכם המספרי עבור גודל מאגר מידע מסוים אינו קריטי; החשיבות של ניסוי זה היא שהוא מדגים את העובדה שהגדלת מאגר המידע אינה פוגעת בטיב הזיהוי, אם רק מגדילים במעט את אורך השאילתה. כדי לבדוק אם אורך השאילתה הדרוש הוא 12 תווים (כפי שמנבאת האקסטרפולציה) או 15 תווים, יש צורך לבנות את מאגר המידע ולבצע בדיקה ישירות עליו. בין כך ובין כך, מספר התווים שיידרש אינו אסטרונומי, והתוכנה תוכל לפעול גם במקרים של מאגרי מידע גדולים בהרבה.

7. בדיקה בתנאי אמת

בפרקים קודמים של עבודה זו תוארו חלקיה של התוכנה הסופית כיחידות נפרדות, ונערכו בדיקות בתנאי מעבדה של כל חלק בנפרד. בפרק זה תתואר בדיקה מקיפה של המערכת כולה כיחידה אחת, ב"תנאי שטח".

כדי לבצע בדיקה זו, הוקלטו מספר רב של אנשים בכמה ניסויים נפרדים. ההקלטות בוצעו על מחשבים שונים, בעזרת מיקרופונים שונים ובתנאי רעש וסביבה שונים, על מנת לבדוק את פעולת התוכנה במגוון מצבים רחב ככל האפשר.

הנבדקים נבחרו באקראי וללא קשר ליכולת השירה שלהם. במקרה אחד הנבדק הודיע שהוא אינו מסוגל לשיר על פי דרישות הקלט של התוכנה, והוא לא הוכלל במדגם. בכל שאר המקרים נכללו כל הקלטותיהם של הנבדקים, מלבד מקרים שבהם הנבדק עצמו חש שההקלטה לא היתה איכותית וביקש לבצע אותה שוב.

כל נבדק ביצע בין שלושה לשמונה שירים בהתאם למידת הזמן שעמדה לרשותו. לפני ההקלטה, הנבדק הקשיב לדוגמת ביצוע שאילתה (של שיר אחר) על מנת לראות באיזה אופן יש לשיר. לאחר מכן הוצגה בפניו רשימת השירים, והנבדק בחר בשיר כלשהו שאותו הכיר. לבסוף, הנבדק הקליט את השיר מזכרונו, מבלי לשמוע את השיר קודם לכן.

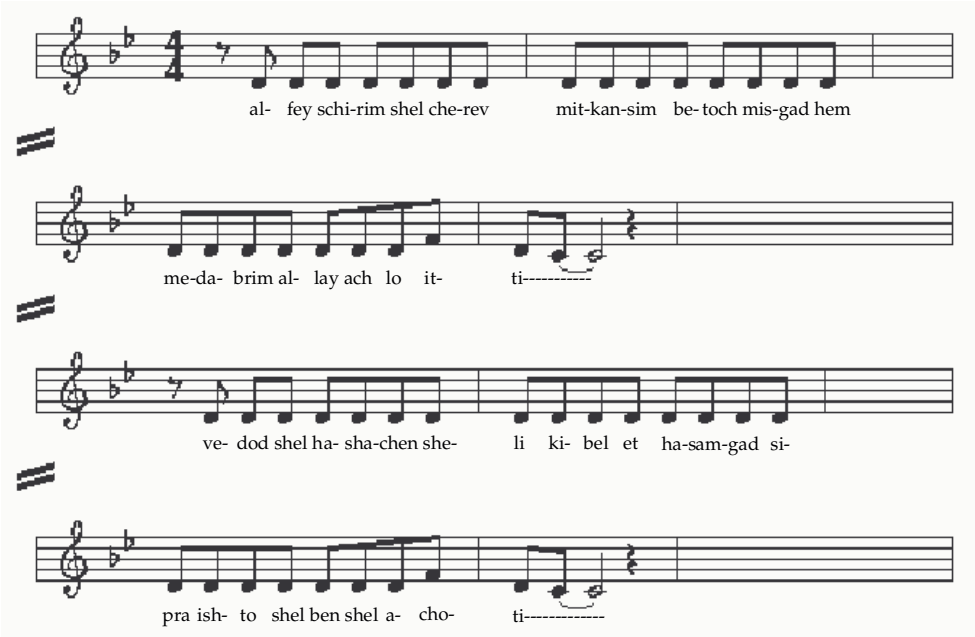
ההקלטות חולקו באופן אקראי לשתי קבוצות, ששימשו לשתי מטרות שונות. הקבוצה הראשונה שימשה לאופטימיזציה של הפרמטרים של תהליכי הסגמנטציה והחיפוש, ואילו הקבוצה השניה שימשה לבדיקת אחוזי הזיהוי, על מנת לוודא שלא חלה התאמת-יתר (overfitting) להקלטות של קבוצת האופטימיזציה.

1.7 בניית מאגר המידע

לצורך ביצוע חיפוש בתנאי אמת יש צורך בבניית מאגר מידע של מנגינות. מאגר המידע צריך להכיל כמות מנגינות שתאפשר למשתמש לבחור בנוחות מנגינה אותה הוא מכיר, וכן על מנת לוודא שתוכנת החיפוש אינה מותאמת לסוגי מנגינות מסוימות.

בעבודה זו נלקחו כל המנגינות מחוברות תווים, ונעשה מאמץ לכסות מגוון רחב של סוגי מוסיקה. מאגר המידע המלא מצורף לעבודה בקובץ Database.mus. אופן ניהול מאגר המידע מתואר בהוראות השימוש בתוכנה הסופית, בנספח א.2. בסעיף זה נעבור בקצרה על מספר נושאים המתארים את אופן הבחירה של שירים וחלקי שירים שהוכנסו למאגר.

שירים מסוימים אינם מתאימים להכלל במאגר מידע של מנגינות. אלה שירים שאינם מלודיים, או שהמנגינה בהם היא מונוטונית מכדי שיהיה ניתן לזהות אותם על סמך מנגינתם בלבד. דוגמה אחת לכך היא ה"מנגינה" של הבתים בשיר "אז למה לי פוליטיקה עכשיו" של להקת "משינה". התווים של שיר זה מוצגים באיור 33.



איור 33: תווים של שיר לא-מלודי

ניתן לראות כי 20 התווים הראשונים של השיר הינם בעלי אותו תדר ואותו אורך. גם בהמשך השיר ישנם רצפים ארוכים של תווים זהים. כיוון שקיימים מספר שירים כאלה, לא ניתן לזהות אותם על פי התחלתם בלבד. במקרים כאלה, ואף במקרים מעט יותר מלודיים, הוחלט שלא להכניס את השירים למאגר המידע, מכיוון שהם אינם מתאימים לזיהוי על ידי זמזום מנגינה.

מכיוון שהתוכנה מיועדת לזיהוי קטע מנגינה, לא הוכנסו למאגר מלוא התווים של כל שיר. במקום זאת, נבחרו באופן סובייקטיבי קטע אחד או יותר, אשר היוו את הקטעים הזכורים ביותר משיר כלשהו. ברוב המקרים, קטעים אלה היו מנגינת הפזמון ומנגינת הבית הראשון. עבור חלק מהשירים, הוכנסה רק מנגינה אחת למאגר; לשירים אחרים הוכנסו שלוש או ארבע מנגינות. במקרה השני, התייחסה התוכנה לכל קטע כאל "שיר" נפרד, ובספירת המנגינות נחשבו קטעים אלה כמנגינות נפרדות.

2.7 אופטימיזציה של הפטרמטרים

אופטימיזציית הפרמטרים של פעולת התוכנה בוצעה על ידי שינוי הפרמטרים, ובדיקת אחוז הזיהוי של התוכנה על מספר הקלטות בתנאי אמת. לצורך האופטימיזציה הוקלטו 64 מנגינות שבוצעו על ידי 14 נבדקים. עבור כל שינוי בפרמטרים, נערכה בדיקה של אחוז המקרים (מתוך 64 ההקלטות) שבהם הזיהוי היה נכון.

בשל המספר הרב של פרמטרים המעורבים בבדיקה, לא ניתן היה לבצע בדיקה מלאה של כל הצירופים האפשריים. במקום זאת, נמצאו בעזרת ניסוי וטעייה ערכי פרמטרים אשר יצרו אחוזי זיהוי סבירים (50%), ולאחר מכן נערכה אופטימיזציה של הפרמטרים בקבוצות קטנות של שניים-שלושה פרמטרים הקשורים לאותו נושא. לאחר סיום האופטימיזציה, אחוז הזיהוי על קבוצת הבדיקה היה 80%. אחוז המקרים שבהם השיר הנכון זוהה באחד מחמשת המקומות הראשונים בהקלטה היה 86%.

הפרמטרים שעליהם נערכה אופטימיזציה מחולקים לשתי קבוצות עיקריות: פרמטרים עבור תהליך הסגמנטציה ופרמטרים של אלגוריתם החיפוש. בכדי שתתאפשר השוואה הוגנת בין אלגוריתמי החיפוש השונים בתנאי אמת, בוצעה אופטימיזציה על הפרטרמים של כל אחד מהאלגוריתמים. זאת למרות שעבור האלגוריתם של Parsons קיים כבר מחקר המתאר את ערכי הפרמטרים האופטימליים (Prechelt and Typke, 1999). השוואה בין תוצאות האופטימיזציה של עבודה זו ושל המחקר הקודם תוצג בסוף סעיף זה.

הפרמטרים שעליהם בוצעה האופטימיזציה, וכן הערכים שנבחרו עבורם, מתוארים בטבלאות הבאות. בתוכנה הסופית, ניתנים כל הפרמטרים האלה לשינוי על ידי המשתמש, למקרה שבדיקה על מדגם גדול יותר תראה שיש צורך בעדכון הפרמטרים.

מכיוון שפרמטרים אלה תלויים באופן הדוק באלגוריתמים עליהם הם משפיעים, תיאורי הפרמטרים מכילים מונחים שהוגדרו בעת הדיון באלגוריתמים אלה. הטבלאות מכילות הפניות לסעיפים שבהם הוגדרו הפרמטרים בהם מדובר.

מטבלת ערכי הפרמטרים של חיפוש Parsons, ניתן לראות כי ישנם הבדלים בין ערכי הפרמטרים של מנוע החיפוש של Prechelt and Typke (1999), לבין הפרמטרים במנוע החיפוש של עבודה זו. ואולם, הבדלים אלה הינם צפויים, מכיוון שתהליכי השירה, זיהוי ה-pitch והסגמנטציה של Prechelt and Typke הוא שונה מאוד מהתהליך המוצע כאן (לדוגמה, זיהוי ה-pitch של העבודה הקודמת מבוסס על אנליזה ספקטרלית). לכן, הגיוני ששני התהליכים ייצרו סוגי שגיאות שונים ויזהו באופן שגוי סוגים שונים של שירה, וכתוצאה מכך התיקונים שתהליך החיפוש צריך לבצע יהיו שונים.

פרמטרים עבור תהליך הסגמנטציה (סעיף 4.3.2)

תיאור הפרמטר	ערך
סגמנטציה ראשונית	
סף אחיד לתחילת סגמנט, באחוזים מה-volume הממוצע	20%
סף אחיד לסיום סגמנט, באחוזים מה-volume הממוצע	14%
אורך מינימלי לזיהוי התחלת סגמנט, במספר מחזורי pitch	10
אורך מינימלי לזיהוי סיום סגמנט, במספר מחזורי pitch	1
סף סיום אדפטיבי, באחוזים מה-volume המקסימלי בתו	30%
תיקון לסף התחלה, באחוזים (איור 16)	60%
דעיכת תיקון לסף התחלה, באחוזים לכל מחזור pitch (איור 16)	2%
סגמנטציה שניונית	
סטיית תקן מקסימלית לתו	50¢
ערך ההפרדה בין תו מוסיקלי לסגמנט רועש, במחזורי pitch	10

פרמטרים עבור אלגוריתם חיפוש לפי קוד Parsons (סעיף 5.4.1)

תיאור הפרמטר	ערך קודם[6]	ערך
ערך מינימלי להתייחסות לתווים כבעלי תדר שונה	-	50¢
מחיר החלפה בין U לבין R	∞	2
מחיר החלפה בין D לבין R	∞	2
מחיר החלפה בין U לבין D	∞	∞
הוספה או מחיקה של R	1	1
הוספה או מחיקה של U	2	3
הוספה או מחיקה של D	2	3

פרמטרים עבור אלגוריתם חיפוש לפי דמיון תדר (סעיף 5.4.2)

תיאור הפרמטר	ערך
מקדם מחיר החלפה (α)	1
מקדם מחיר מחיקה (β)	1000
מקדם מחיר הוספה (γ)	1000

[6] לפי מחקר של Prechelt and Typke (1999).

פרמטרים עבור אלגוריתם חיפוש משולב תדר/משך-זמן (סעיף 5.4.3)

תיאור הפרמטר	ערך
מקדם מחיר החלפה עבור תדרים (α)	1
מקדם מחיר החלפה עבור משכי-זמן זהים	1
מקדם מחיר החלפה עבור משכי-זמן דומים	1.5
מקדם מחיר החלפה עבור משכי-זמן מנוגדים	3
מקדם מחיר מחיקה (β)	1000
מקדם מחיר הוספה (γ)	1000

3.7 בדיקת אחוזי הזיהוי

בסעיף הקודם נעשה שימוש בקבוצה של הקלטות בזמן אמת על מנת לבצע אופטימיזציה של תהליך הזיהוי. הסכנה במצבים כאלה הינה התאמת-יתר (overfitting), כלומר, קיימת אפשרות שערכי האופטימיזציה יותאמו למקרים ספציפיים של ההקלטות בהן נעשה שימוש. כדי לוודא שהתאמת-יתר לא התרחשה במקרה הנידון, נערכו הקלטות נוספות לאחר סיום האופטימיזציה. ההקלטות בוצעו במצבים דומים להקלטות של הסעיף הקודם, וכללו 38 שירים של 9 נבדקים.

אחוזי הזיהוי של קבוצת האופטימיזציה ושל קבוצת המדגם נבדקו. חושבו שני מדדים: זיהוי מלא (כלומר, השיר הנכון זוהה כמתאים ביותר), וזיהוי בחמישיה ראשונה (כלומר, השיר הנכון היה בין חמשת השירים המתאימים ביותר). הבדיקה נערכה עבור שלושת אלגוריתמי החיפוש שתוארו בסעיף 5.4. תוצאות הניסוי מסוכמות בטבלה הבאה.

אלגוריתם	**קבוצת אופטימיזציה**		**קבוצת ביקורת**	
	זיהוי מלא	חמישיה ראשונה	זיהוי מלא	חמישיה ראשונה
תדר/משך-זמן	80%	86%	76%	95%
דמיון תדר	77%	88%	76%	95%
Parsons	52%	73%	53%	66%

מטבלה זו ניתן לראות, כי אין ירידה משמעותית ביכולת הזיהוי במעבר מקבוצת האופטימיזציה לקבוצת הביקורת. להיפך, ישנה עלייה באחוזי הזיהוי עבור החמישיה

הראשונה באלגוריתם התדר ובאלגוריתם תדר/משך-זמן. מספר גורמים עשויים לתרום לעלייה זו. בין גורמים אלה נכללים: שיפור מקרי ביכולת השירה של הנבדקים בקבוצה השניה במקום הקבוצה הראשונה; וכן העובדה שאורך המנגינה הממוצע בקבוצה השניה היה גבוה במקצת מאורך המגינה הממוצע בקבוצה הראשונה, ותרם לזיהוי גם במקרים של זיופים מסוימים. בכל מקרה, ברור שאין מדובר כאן במצב של התאמת-יתר; להיפך, ביצועי קבוצת הביקורת היו טובים במקצת מביצועי קבוצת האופטימיזציה.

העובדה שאלגוריתם התדר ואלגוריתם תדר/משך-זמן קיבלו בדיוק אותם אחוזי התאמה בקבוצת הביקורת היא, ככל הנראה, מקרית. רוב השירים בקבוצה זו דורגו באופן זהה בשני האלגוריתמים, אך היו מספר הבדלים. ככל הנראה, קבוצת הביקורת לא הכילה מספיק נבדקים על מנת להבדיל בין שני האלגוריתמים.

ניתן לראות מטבלה זו כי קיים יתרון ברור לאלגוריתמים המקסימליסטיים של תדר ושל תדר/משך-זמן, על פני אלגוריתם Parsons המינימליסטי. ההבדל בין אלגוריתם התדר לאלגוריתם תדר/משך-זמן הינו פחות בולט, וייתכן שהוא נובע משגיאה סטטיסטית. תוצאות דומות התקבלו גם בסימולציה שנערכה בסעיף 6.3.2. המסקנה ברורה: שימוש באלגוריתם Parsons מוותר על חלק ניכר מהמידע המצוי בשירה, ופוגם ללא צורך ביכולת הזיהוי של התוכנה. נושא זה יידון בהרחבה בסעיף 8.1.4.

כדי לבדוק את יכולתם של בני אדם לזהות שירים, נערך ניסוי נוסף, שבו התבקשו שישה נבדקים להקשיב לקבוצת שירים שהוקלטו על ידי אנשים אחרים. שירים אלה נבחרו באקראי מתוך ההקלטות של קבוצות האופטימיזציה והביקורת. הנבדק הקשיב להקלטה וניסה לזהות את השיר. לא היתה מגבלת זמן, והנבדק היה יכול להקשיב להקלטה מספר פעמים, אם ביקש זאת.

במקרים שבהם הנבדק הכיר את השיר והיה מסוגל לשיר חלק ממילותיו, גם אם לא זכר את שם השיר, נחשב הזיהוי כמלא. במקרים שבהם הנבדק לא היה מסוגל לזהות מידע מילולי לגבי השיר, הוא נחשב כזיהוי שגוי.

במקרה שהנבדק לא היה מסוגל לזהות את השיר, שם השיר נאמר לו, על מנת לוודא שהוא מכיר את השיר. אם הנבדק לא הכיר את השיר, המבחן לא נכלל בתוצאות (הדבר שקול לכך שהשיר "אינו במאגר המידע" של הנבדק). כלומר, בכל המקרים הנכללים בסטטיסטיקה הנבדק הכיר את השיר המדובר, אך בחלקם הוא לא היה מסוגל לזהות אותו. כתוצאה מכך, כל נבדק הקשיב למספר שונה של שירים; סך כל מספרי השירים שניסו הנבדקים לזהות היה 117.

מעניין לציין כי מספר תופעות שונות סווגו בניסוי זה כ"חוסר זיהוי". בחלק מהמקרים, הנבדקים הקשיבו לתחילת השיר, חשבו שמדובר בשיר מסוים, וכאשר המשכו לא התאים ל"תיאוריה" שלהם, החליטו שמדובר בזיוף; במקרים אלה חשבו הנבדקים שהם יודעים על איזה שיר מדובר, אך טעו. כאשר נאמר להם מהו השיר הנכון והם הקשיבו להקלטה שוב, יכלו

לומר כי טעו וזו אכן זו המנגינה שלו. במקרים אחרים, סיפרו הנבדקים כי שם השיר "עומד על קצה הלשון", אך לא יכלו להזכר בו גם לאחר מספר דקות. במצבים נוספים, הנבדקים היו בטוחים שהם אינם מכירים את המנגינה. לפעמים, גם לאחר שנאמר להם שם השיר, סיפרו הנבדקים שהם אינם רואים כל קשר בין מנגינת השיר לבין הביצוע של ההקלטה. תוכנת המחשב הצליחה לזהות גם חלק מהשירים האלה.

תוצאות ניסוי זה השתנו בטווח רחב בין הנבדקים, והיו תלויות לא רק בנבדק עצמו, אלא גם באיכות ההקלטות אליהם הקשיבו. (כזכור, כל נבדק הקשיב לקבוצת שירים שונה.) ממוצע הזיהוי היה 57%, אך חלק מהנבדקים זיהו כ45%- מהשירים ואחרים זיהו כ75%-.

תוצאות אלה מראות כי קבוצת האנשים שנבדקה זיהתה שירים פחות טוב מתוכנת המחשב. עובדה זו מעידה על כך שהתוכנה מסוגלת להתחרות בהצלחה עם בני אדם ביכולת זיהוי, ובתור שכך עשויה להוות כלי עזר שימושי לצורך חיפושי מנגינה ביישומי אמת.

4.7 זמן ה-CPU הנדרש להרצת התוכנה

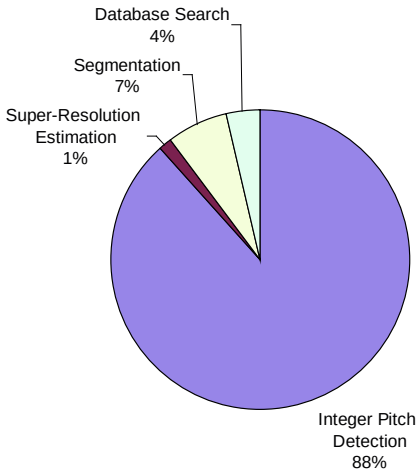
בסעיף זה נתאר בדיקות שנערכו על מנת לבדוק את משך הזמן הדרוש לביצוע תהליך הזיהוי של מנגינה.

נערכה בדיקה על קבוצה של 64 הקלטות בתנאי אמת, בעלות אורך ממוצע של 20.6 שניות. בוצע זיהוי מלא של כל אחת מהיצירות, על גבי מחשב Pentium II/233 MHz המריץ Windows 95, ללא תוכנות נוספות ברקע. החיפוש נערך מעל מאגר מידע של 200 מנגינות. נמצא כי זמן הריצה הממוצע לשיר היה כ4.4- שניות. במילים אחרות, פעולת החישוב לוקחת, בממוצע, פחות מחמישית מהזמן הדרוש לביצוע ההקלטה. עובדה זו מצביעה על כך שניתן בקלות לבצע מימוש של התוכנה בזמן אמת, גם על מחשבים שאינם החדישים ביותר כיום.

נשאלת השאלה, האם ניתן להשליך נתונים אלה גם לתנאים מציאותיים, שבהם החיפוש יתבצע מעל מאגר מידע גדול בהרבה? אם ניווכח כי ישנה תלות חזקה בין גודל מאגר המידע לבין מהירות ביצוע החישוב, אזי הגדלת מאגר המידע לגודל סביר (למשל, 10,000 תווים) עשויה לגרום לכך שהתוכנה תפעל בצורה איטית למדי.

כדי לבדוק (ולהפריך) אפשרות זו, נערכה הרצת profiling על התוכנה, ונמצא זמן החישוב הכולל שהושקע בכל אחד מחלקי התוכנה. ההרצה בוצעה על הקלטה אפיינית בתנאי אמת, שנמשכה כ19- שניות והכילה 17 תווים. החיפוש בוצע על מאגר מידע של 200 מנגינות.

תוצאת ה-profiling מוצגות באיור 34. בתרשים נכללו רק זמני החישוב, ולא זמנים התלויים במשתמש (כגון ביצוע ההקלטה).



איור 34: חלוקת זמן CPU בין חלקי התוכנה

מאיור זה ניתן לראות כי רובו המכריע של זמן ה-CPU מושקע בחישוב ה-Integer Pitch , שהוא השלב הראשון בתהליך זיהוי ה-pitch (סעיפים 3.3 ו3.4-). שאר שלבי זיהוי ה-pitch (Super-Resolution Estimation) ארכו זמן זניח. תהליך החיפוש לוקח כ4%- מזמן ה-CPU. מצב זה מובן לאור העובדה שכמות המידע המעובד הולכת ופוחתת ככל שמתקדם תהליך הזיהוי: ממידע מוקלט ב8000- דגימות לשניה, דרך רשימה של כמה עשרות או מאות ערכי pitch בשניה, ועד לרשימה של שניים או שלושה תווים בשניה.

עבור מערכות בעלות מאגר מידע גדול, תשתנה חלוקת הזמן, ויגדל משך הזמן המושקע בחיפוש. זאת מכיוון שתהליך החיפוש מתבצע על ידי השוואה של מחרוזת התווים המוקלטת לכל אחת מהרשומות במאגר המידע, לעומת שאר חלקי החישוב, שאינם מסתמכים כלל על מאגר המידע. אם נניח כי זמן תהליך החיפוש הוא פרופורציונלי בקירוב למספר השירים במאגר המידע, הרי שעבור מאגר מידע בעל 10,000 שירים, תהליך החיפוש ייקח כ60%- מזמן ה-CPU הכולל – יותר מאשר תהליך זיהוי ה-pitch.

תוך התחשבות בערכי הזמן הממוצעים לשיר, ניתן לחשב ולמצוא כי גם לאחר הגדלת מאגר המידע ל10,000- שירים, זמן החישוב הכולל יהיה כ13- שניות להקלטה בממוצע, כלומר: תהליך הזיהוי עדיין מתבצע במהירות רבה יותר ממשך זמן ההקלטה.

8. דיון מסכם

בעבודה זו הוצגה תוכנה המבצעת חיפוש במאגר מידע מוסיקלי באמצעות קלט קולי. התוכנה מורכבת ממספר חלקים נפרדים, שהוצגו בפרקים הראשונים של העבודה: זיהוי pitch (פרק 3), סגמנטציה (פרק 4) וחיפוש (פרק 5).

נערכו מספר בדיקות כדי לוודא את תקינות התוכנה ולבדוק את ביצועיה. בפרק 6 הוצגו מערכות סימולציה לבדיקת החלקים השונים של התוכנה, ואילו בפרק 7 תוארו מספר בדיקות בתנאי שטח של התוכנה, למציאת הביצועים הכלליים שלה.

בפרק זה נחזור בקצרה על התוצאות העיקריות שנמצאו בעבודה זו, נדון במסקנות העולות מהן ובהשלכותיהן, ונציין מספר נושאים הדורשים בדיקה מעמיקה יותר, ועשויים לשמש בסיס למחקר המשך.

1.8 סיכום ומסקנות

1.1.8 סינון מקדים

בדרך כלל מקובל לבצע סוג כלשהו של סינון מקדים על אותות דיבור, לפני כל תהליך זיהוי pitch. ההנחה היא כי תדרים גבוהים, כגון הרמוניות, פורמנטים ורעשים, מזיקים לתהליך זיהוי ה-pitch.

במקרים שנבדקו בעבודה זו נמצא, למרבה ההפתעה, כי לסינון מקדים ישנה מעט מאוד השפעה על זיהוי pitch באלגוריתם super-resolution autocorrelation שנבחר לשימוש בתוכנה הסופית (פרק 2).

ברוב המקרים שנבדקו נמצא, כי הסינון אינו גורם לשינוי של ממש בערכי ה-pitch המזוהים. במקרים כאלה הסינון דורש זמן חישוב מיותר, אך ניתן לטעון כי הוא מנחית סוגי רעשים מסוימים.

הערכתי היא, כי הסינון אינו משפר את תהליך זיהוי ה-pitch מכיוון שתהליך האוטוקורלציה כולל בתוכו חלק מהמאפיינים של מסנן מעביר-נמוכים. זאת מכיוון שבחישוב האוטוקורלציה, מתבצע סיכום על גבי פרק זמן של מחזור pitch אחד. סיכום כזה גורם להשפעה דומה לזו של מסנן פשוט בעל תדר קטעון נמוך מזה של ה-prefilters המקובלים, ולכן הוצאת ה-prefilter אינה משפיעה במיוחד על מידת הרעש בעת הזיהוי. הסבר מתמטי של תופעה זו ניתן בסעיף 2.3.1.

בנוסף לכך, להרמוניות אין השפעה מזיקה על יכולת זיהוי ה-pitch בשיטת אוטוקורלציה, מכיוון ששיטה זו ישימה על צורות גל לא סינוסואידליות (בעלות הרמוניות) בדיוק כשם שהיא פועלת על גלי סינוס.

הבעיה העיקרית בתהליך הסינון נוצרת במצבים של שינויים מהירים ב-pitch, כגון אלה המופיעים בהקלטות של פסנתר. במצבים כאלה, לא רק שאין שיפור בעקבות הסינון המקדים, אלא שהסינון גורם לירידה של ממש בדיוק הזיהוי, באיזורי השינוי של ה-pitch.

הסבר אפשרי לתופעה זו הוא כדלקמן: השינוי הפתאומי ב-pitch מלווה בתדרים גבוהים, אשר מוחלשים על ידי המסנן. כתוצאה מכך, המסנן יוצר איזור ביניים בין שני התווים, שבו ה-pitch אינו אחיד והזיהוי אינו מדויק.

בין אם יש אמת בהשערות אלה, ובין אם הסיבה האמיתית היא אחרת, העובדות מצביעות על כך שבמקרים של אותות מוסיקליים חד-קוליים, סינון מקדים אינו משפר את יכולת זיהוי ה-pitch, ובמקרים מסוימים אף מחליש אותו.

באשר ליכולתו של המסנן להחליש רעשים חיצוניים, אין ספק כי רעשים בתדר גבוה אכן מונחתים על ידי מסנן כזה; אך בכל ההקלטות שבוצעו בעבודה זו, לא נראה היה שזיהוי ה-pitch בעייתי, גם במצבים של רעש רקע חזק.

המסקנה, אם כן, היא שסינון מקדים אינו תורם לזיהוי pitch לצורך זיהוי תווים מוסיקליים. בתוכנה הסופית לא נעשה שימוש בסינון מקדים. התאמתו של סינון מעביר נמוכים לצרכי זיהוי pitch אחרים היא נושא למחקר מקיף בפני עצמו.

2.1.8 זיהוי Pitch

קיים מגוון רחב של אלגוריתמים לזיהוי pitch, אשר מבוססים על מספר גישות שונות (פרק 3). בעבודה זו נבחרה גישת האוטוקורלציה, בעיקר משום שהיא מבוססת על זיהוי צלילים במישור הזמן. ככל הנראה, זיהוי ה-pitch של אותות שמע במוח האדם מתבצע בצורה כלשהי על פי מאפייני הזמן של האות (Moorer, 1975). עובדה זו באה לידי ביטוי, למשל, ביכולתו של אדם לזהות צליל בתדר נמוך על סמך ההרמוניות שלו, זיהוי שאינו טריוויאלי במישור התדר אך נוצר באופן מיידי במישור הזמן.

בין האלגוריתמים מבוססי-אוטוקורלציה, נבחר אלגוריתם super-resolution (Medan et al., 1991). אלגוריתם זה מאפשר לבצע שיערוך לינארי של ערך ה-pitch, תוך שימוש בקצב דגימה נמוך. בכך הוא מאפשר להשיג דיוק רב בזיהוי ה-pitch, תוך חסכון ניכר בזמן חישוב ובזכרון. כיוון שתהליך זיהוי ה-pitch הוא האיטי ביותר מבין שלבי פעולת התוכנה (סעיף 7.4), חסכון זה במהירות הוא יתרון חשוב. הדיוק הנוסף המוקנה בעקבות השימוש באלגוריתם זה מאפשר לזהות תווים באופן מדויק יותר מאשר בעבודות קודמות.

בבדיקות שנערכו על מנגנון זיהוי ה-pitch (סעיף 6.1) נמצא כי הוא פועל באופן תקין גם תחת מבחנים מחמירים, כגון: רעש רקע חזק; הרמוניות בעלות עוצמה חזקה יותר מהתדר הבסיסי; ובדרך כלל, גם בהוספת תדרים שאינם הרמוניות. האלגוריתם נבדק גם בתנאי אמת, על הקלטות של פסנתר ושל קולות אנושיים, ונמצא כי הוא פועל באופן משביע רצון על כל סוגי הקלט הצפויים לתוכנה.

3.1.8 סגמנטציה

תהליך הסגמנטציה הוא, ללא ספק, הנושא הבעייתי ביותר במשימת חיפוש מידע מוסיקלי על סמך קלט קולי. בשלב הסגמנטציה, יש להפריד את ערכי ה-pitch הרציפים לתווים בודדים. עבודות קודמות בנושא השתמשו באופני סגמנטציה שונים, וגם בעבודה זו נבחנו מספר גישות שונות.

התהליך שנבחר לבסוף לעבודה זו התבסס על שילוב בין שני ערוצי מידע: pitch ו-volume. הסגמנטציה התבססה על ההנחה שה-volume יורד בצורה משמעותית בין כל שני תווים. כדי להבטיח זאת, מתבקשים המשתמשים לשיר באמצעות הגיית הברות "טָה" או "טִי". בעת הגיית העיצור /ט/ ישנה עצירה מוחלטת של האוויר היוצא מהריאות, וכתוצאה מכך יורד ה-volume באופן מספיק כדי לאפשר את הסגמנטציה. הכוונה היא, כי הקלטת שאילתה תבוצע במיוחד עבור תוכנת החיפוש, וההקלטה לא תשמש למטרות אחרות; לכן, ניתן לבקש מהמשתמש להקליט באופן מסוים, בכדי לשפר את יכולת הזיהוי של התוכנה.

בבדיקות שנערכו על תהליך הסגמנטציה, נמצא כי הוא פועל היטב באופן כללי, אך מספר שגיאות זיהוי חוזרות על עצמן (סעיף 6.2). הטעויות העיקריות הינן חוסר הפרדה בין תווים שונים, וזיהוי משך זמן קצר מדי לתווים. מכיוון שמחקרים קודמים לא תארו את אופי ושכיחות השגיאות בתהליכי סגמנטציה שונים, לא ניתן להשוות את יכולת הסגמנטציה של עבודה זו לעבודות אחרות; ניתן לומר רק, כי ביחס לאלגוריתמים אחרים שנבחנו ישנו יתרון לאלגוריתם שנבחר. יתכן, כי אלגוריתם עתידי ישפר עוד יותר את יכולת הסגמנטציה של התוכנה (ראו סעיף 8.2.1).

4.1.8 מנוע החיפוש

תפקידו של מנוע החיפוש הוא למצוא את ההתאמה הטובה ביותר בין רשימת התווים שהתקבלה מההקלטה, לבין התווים של שירי מאגר המידע. התאמה זו צריכה להיות גמישה מספיק על מנת לתקן מגוון רחב של זיופים וטעויות בתהליכי הזיהוי. לצורך כך התבצע חישוב על פי שיטת מרחק העריכה, אשר מהווה מדד לדמיון בין מחרוזות שונות.

כדי לחשב מרחק עריכה בין שתי מחרוזות תווים, יש להגדיר את צורת ההשוואה בין התווים: אלו שינויים יחשבו קרובים או זהים, ואלו שינויים יהיו בעלי מחיר עריכה גבוה. בעבודה זו נעשתה הבחנה בין שתי גישות שונות לבעיה זו. הגישה המינימליסטית גורסת כי כדי למזער את השפעת חוסר הדיוק במידע, יש להוציא מכל תו כמות מזערית של מידע, ולהשתמש רק במידע האמין הזה. לעומתה, הגישה המקסימליסטית טוענת כי יש להוציא מכל תו כמות גדולה של מידע, אך להשתמש במידע זה בערבון מוגבל, ולאפשר שינויים קטנים במידע מבלי לדרוש עבורם מחיר עריכה גבוה.

בכל העבודות הקודמות בנושא, נעשה שימוש בחיפוש על פי קוד Parsons, שהוא חיפוש מינימליסטי מאוד. לכל תו, המידע היחיד שבו נעשה שימוש הוא כיוון שינוי התדר ביחס לתו הקודם: גבוה יותר, נמוך יותר או זהה.

לעומת גישה זו, נבחנה כאן גישה מקסימליסטית, שבה החיפוש מתבצע על סמך השוואת תדרים בדיוק גבוה (חיפוש ע"פ דמיון תדר). בגישה זו, הוענק מחיר עריכה נמוך מאוד לשינויים קטנים בתדר, אך מחיר העריכה הלך וגדל ככל ששינויים אלה נעשו משמעותיים יותר.

בהשוואה בין שתי הגישות ישנו יתרון ברור לחיפוש ע"פ דמיון תדר, כלומר, לגישה המקסימליסטית (סעיפים 6.3.2 ו-7.3). תוצאה זו בולטת הן בסימולציות שנערכו על מנוע החיפוש, והן בתוצאות האמת של זיהוי הקלטות בתנאי שטח.

ניתן להסיק מכך כי למרות מגוון העיוותים והטעויות בזיהוי, יכולת השירה של בני אדם היא טובה מספיק, בכדי שיהיה ניתן להוציא ממנה כמות מידע גדולה יחסית: לא רק את כיוון שינוי התדר, אלא גם את מידת השינוי המקורבת.

ככל שניתן להוציא מידע רב יותר מההקלטה, כך משתפר תהליך הזיהוי. לכן, נעשה נסיון להשתמש גם במידע משך הזמן של כל תו, על מנת לקבל מדד נוסף להתאמה בין מחרוזת התווים המוקלטת לבין שירים במאגר המידע. התוצאה הינה מנגנון חיפוש משולב תדר/משך-זמן.

מידע משך הזמן מדויק פחות מאשר התדרים של התווים, הן בגלל מגבלות השירה של המשתמש והן בגלל שינויים באופן הפרדת התווים בין ביצועים מוסיקליים שונים. כדי להדגיש את חוסר האמינות של מידע זה ביחס למידע התדר, הוחלט להשתמש עבורו בתיאור מינימליסטי וכללי, אשר ייתן תיקון למחיר העריכה שחושב על סמך התדר.

בסימולציות שנערכו הופיע יתרון קטן אך בעל משמעות סטטיסטית לשיטה המשולבת על פני שיטת התדר. לעומת זאת, בבדיקות בתנאי אמת לא נמצאו הבדלים משמעותיים בין השיטות.

להערכתי, במידע של משך-זמן גלום פוטנציאל לשיפור ניכר של איכות הזיהוי. הערכה זו נתמכת גם על ידי מחקרים פסיכולוגיים של זכרון מוסיקלי (Dowling, 1978). ייתכן כי אפשרות זו לא באה לידי ביטוי כתוצאה מכך שרוב ההקלטות היו באיכות מספיקה לביצוע זיהוי גם ללא תוספת המידע על הסגמנטציה, וההקלטות הלא מוצלחות היו במצב גרוע מכדי לזהות אותן בכל שיטה. אם זהו המצב, ייתכן כי האלגוריתם המשולב יראה יתרונות ברורים יותר במקרים שבהם החיפוש מתבצע על מאגר מידע גדול יותר, מכיוון שבמצב זה כל תוספת מידע חשובה על מנת להבחין בין הכמות הרבה של המנגינות.

גודל מאגר המידע הוא, למעשה, הגורם היחיד המפריד בין התוכנה הקיימת לבין יישום שימושי. נשאלת השאלה, לכן, מה תהיה השפעת הגדלת מאגר המידע על ביצועי התוכנה. הגדלה כזו אינה יכולה לשפר את טיב הזיהוי, מכיוון שהיא מגדילה את הסיכויים לקיום מנגינה קרובה מאוד למנגינה המבוקשת, ובכך גורמת לקושי בהפרדה בין המנגינות.

מכיוון שלא ניתן, במסגרת עבודה זו, ליצור מאגר מידע של אלפי מנגינות, יש צורך להתמודד עם שאלה זו באופן תיאורטי, ולנסח מסקנות ראשוניות בלבד. בסעיף 6.3.3 אומץ מודל המילון, לפיו הגדלת מאגר המידע אינה פוגעת ביכולת הזיהוי, אם יחד איתו גדל גם מספר התווים בשאילתה. מודל זה נבדק על מאגרי מידע עד גודל של כ220- שירים, ונמצא מתאים בקירוב. על פי המודל, מאגרי מידע של כ10,000- שירים יהיו ניתנים לזיהוי בדיוק גבוה, אם אורך השאילתה יהיה כ12- תווים לפחות. בהקלטות שבוצעו לבדיקת התוכנה, רוב המשתמשים ביצעו מספר רב יותר של תווים (למרות שלא התבקשו לעשות כן), כך שדרישה זו אינה מהווה בעייה למשתמש הממוצע.

5.1.8 ביצועים כלליים

יכולת הזיהוי של התוכנה נבדקה על קבוצת נבדקים אשר בחרו ושרו מנגינות מתוך מאגר מידע של כ200- מנגינות. ב79%- מהמקרים, התוכנה זיהתה את השיר באופן נכון (כלומר, השיר הופיע בעדיפות עליונה ברשימת ההתאמות). ב89%- מהמקרים הזיהוי היה בין חמש ההתאמות הטובות ביותר.

ביצועים אלה הינם מרשימים במיוחד כאשר לוקחים בחשבון שיכולת הזיהוי של בני אדם את אותם שירים היא נמוכה בהרבה. מספר נבדקים הקשיבו להקלטות שיריהם של נבדקים אחרים, ויכלו לזהות רק 45% עד 75% מהשירים (ממוצע הזיהוי היה 57%). הנבדקים הכירו את כל השירים ששמעו, אך בשל הזיופים בשירה או בשל חמקמקות הזכרון, לא יכלו לזהות את השיר.

נושא נוסף שנבדק הוא זמן הריצה. על מחשב טיפוסי בן-ימינו, נמצא כי זמן העיבוד הנדרש הוא כ13- שניות לכל דקה של הקלטה. לפיכך, יישום בזמן אמת של התוכנה הוא

אפשרי. בדיקת profiling של התוכנה מראה כי גם עבור מאגרי מידע גדולים, ניתן לבצע את התהליך בזמן קצר ממשך זמן ההקלטה.

2.8 נושאים למחקר המשך

בסעיף זה יתוארו מספר הצעות לשימושים והרחבות של עבודה זו, אשר לא נכללו בעבודה מפאת חוסר זמן או משאבים. ישנם תחומים רבים שבהם עדיין לא התבצעה בדיקה מסודרת של אפשרויות הזיהוי הקיימות. מחקר בתחומים אלה עשוי לשפר את ביצועי התוכנה עוד יותר. עצם העובדה שעבודות שונות משתמשות בשיטות סגמנטציה שונות, מעידה על כך שהאלגוריתם האופטימלי עדיין לא נמצא.

1.2.8 השוואה בין שיטות סגמנטציה

היכולת המוצלחת של בני אדם לבצע סגמנטציה כמעט ללא שגיאות ובכל צורת שירה, מוכיחה כי ישנה אפשרות לקבל תהליך סגמנטציה הפועל באופן טוב, גם מבלי לדרוש מהמשתמש לשיר באופן מסוים. הערכתי היא, שתהליך הסגמנטציה במוח האנושי מסובך יותר משאר התהליכים בזיהוי מוסיקה חד-קולית (כגון זיהוי ה-pitch), וכי תהליך זה מבוסס, בין היתר, על זיהוי העיצורים המפרידים בין התווים.

האלגוריתמים הקיימים בעבודות קודמות, וכן האלגוריתם המוצע בעבודה זו, משתמשים בתכונות פשוטות יותר, וכתוצאה מכך מוגבלים בטווח סוגי הקלט שהם מסוגלים לקלוט. השוואה בין יכולת הסגמנטציה של אלגוריתמים אלה לא נערכה עד היום. (חלק מהמאמרים הקיימים אינם מפרטים באופן מלא את האלגוריתם שנבחר.)

ייתכן גם כי נסיונות ההפשטה של הבעיה הן הגורמות לבעייתיות בסגמנטציה. ישנה אפשרות לזהות עיצורים ותנועות בעזרת אלגוריתמים של speech recognition. ייתכן שזיהוי כזה יפעל טוב יותר מהאלגוריתמים הקיימים, ואף יסיר את המגבלה על אופי השירה. אלגוריתם כזה לא נוסה, למיטב ידיעתי, באף יישום של סגמנטציית תווים.

2.2.8 מנוע החיפוש

נראה שהרעיון הבסיסי של מרחק עריכה מתאים מאוד לחיפוש במאגר מידע מוסיקלי. כל העבודות הקודמות התייחסו לחיפוש בשיטה זו כאופטימלית. אך ניתן להעלות על הדעת ואריאציות רבות לשיטת החיפוש המדויקת. בחלקן נעשה שימוש במספר עבודות, אך לא נערכה השוואה ביצועית בין גישות שונות.

לדוגמה, Mongeau and Sankoff (1990) פיתחו אלגוריתם השוואה בין יצירות מוסיקליות, המבוסס על מרחק עריכה. אלגוריתם זה מיועד לזיהוי ואריאציות על נושא מוסיקלי, ובתור שכך הוא מאפשר מגוון פעולות עריכה הקיימות בעת ביצוע ואריאציות. פעולות איחוי ופיצול תווים, למשל, מקבלות מחיר עריכה נמוך, ושינויים המוגדרים "אנהרמוניים" מקבלים מחיר עריכה גבוה. מקצת מהפעולות האלה אומצו על ידי McNab et al. (1997), אך לא נערכה השוואה בין שיטה זו לשיטת מרחק העריכה הסטנדרטי שתוארה בסעיף 5.2.

מצד אחד, ייתכן כי שיטה זו תייעל את זיהוי השירים, על ידי יחס "סלחני" לעיבודים מוסיקליים שונים של המנגינה הבסיסית. בין ההקלטות שבוצעו לצורך עבודה זו היו מספר מקרים שבהם המשתמש יצר (באופן לא מודע) וריאציות על המנגינה המקורית. מצד שני, ייתכן גם כי ה"התמחות" של אלגוריתם מרחק עריכה בסוגי שינויים מסוימים מגבילה את גמישות ההתמודדות שלו עם מגוון רחב של זיופים והגבלות אחרות, כך ששיכלול האלגוריתם אינו ערובה לשיפור ביצועיו.

נושא נוסף שלא נבדק עד תום הוא חשיבות משכי הזמן של התווים בזיהוי יצירה. הסימולציות בעבודה זו הראו כי חיפוש המשלב משכי זמן מהווה יתרון על חיפוש בעזרת תדרים בלבד, ועשוי להיות גורם מכריע בחיפוש במאגרי מידע גדולים. עם זאת, תוצאות בדיקות השטח לא היו חד-משמעיות בנושא זה. יש צורך במחקר נוסף על מנת לבדוק גישות שונות להתחשבות במשכי זמן.

בין היתר, יש לבדוק את נכונות השימוש בגישה המינימליסטית לחיפוש על-פי משכי זמן. בעבודה זו, כל המידע הקיים לגבי משכי זמן תומצת, לפני החיפוש, להשוואה של משך הזמן של התו הנוכחי ביחס לקודם: ארוך יותר, קצר יותר או זהה בקירוב. אך כשם שהשיטה המקסימליסטית הוכיחה את עצמה בחיפוש תדר, ייתכן כי גם כאן יהיה יתרון לחיפוש על פי מידע מדויק יותר, שיכלול לא רק את כיוון השינוי במשך הזמן אלא גם את מידת השינוי; זאת למרות שמידע משכי הזמן הינו מדויק פחות ממידע התדרים.

לבסוף, לפני שימוש מעשי בתוכנה, יש צורך לבדוק את יכולת ההתמודדות שלה עם מאגרי מידע גדולים בהרבה מאלה שנבדקו בתנאי עבודה זו. שני הגורמים שעשויים להיות מושפעים מהגדלת מאגר המידע הינם מהירות הריצה ודיוק הזיהוי. על ידי profiling נמצא כי זמן הריצה יהיה סביר גם עבור מאגרי מידע גדולים. לעומת זאת, לגבי ביצועי המערכת בעת שימוש במאגר מידע גדול ניתן להעלות ספקולציות, אך המבחן האמיתי יהיה בדיקת יכולת הזיהוי תחת תנאי אמת. לצורך בדיקה כזו יש להרכיב מאגר מידע נרחב של אלפי מנגינות. בצורה כזו יהיה ניתן לאשר או להפריך את ההשערה שהועלתה בסעיף 6.3.3, לפיה זיהוי נכון יתאפשר אם אורך השאילתה יגדל באופן לוגריתמי בגודל מאגר המידע.

3.2.8 יישום להתוויה אוטומטית

התוויה אוטומטית היא תהליך של מציאת התווים של הקלטה של קטע מוסיקלי. דוגמת יישום התוויה היא הקלטה משירה אנושית לקובץ MIDI. במקרה כזה, שלבי העיבוד הראשוניים דומים לשלבים הראשוניים של תוכנת החיפוש של עבודה זו: זיהוי pitch וסגמנטציה.

ניתן, לפיכך, להסב את התוכנה הקיימת לתוכנת התוויה אוטומטית של קולות אנושיים, על ידי שינוי שלב העיבוד האחרון. תוכנה כזו היא פתרון נוח וזול עבור מלחינים אשר אינם מעוניינים או אינם מסוגלים להשתמש באורגן להקלטת מוסיקה.

נספחים

א. הוראות שימוש בתוכנה

התוכנה MUSE (MUsic Search Engine) מצורפת לעבודה זו. התוכנית נכתבה ונבדקה תחת Windows 95, אך צריכה לרוץ ללא תקלות על כל מערכת Win32 (Windows NT 4, Windows 98 וכד'). התוכנית משתמשת במערכת MFC 4.2, ולכן כדי שיהיה ניתן להשתמש בה באופן תקין יש צורך לספק את הקובץ MFC42.DLL בספריית Windows\System. גם קובץ זה מצורף לעבודה.

התוכנה מכילה שני מצבי פעולה שונים. הראשון, מצב ה-Wizard, מיועד למשתמש הקצה ומאפשר להקליט מנגינה ולחפש אותה במאגר המידע בצורה פשוטה וידידותית. בנוסף, מכילה התוכנה מערכת מורכבת יותר (מצב Details), המיועדת למשתמש המומחה ולמנהל מאגר המידע. מערכת זו מאפשרת מגוון רחב של פעולות שאינו נגיש דרך ה-Wizard, כגון תצוגה גרפית של שלבי זיהוי המנגינה, הוספת שירים למאגר המידע, ביצוע הרצות batch ועוד.

ברירת המחדל בהפעלה ראשונה של התוכנה היא כניסה למצב Wizard. בהרצות הבאות התוכנה תכנס אוטומטית למצב האחרון בו נעשה שימוש.

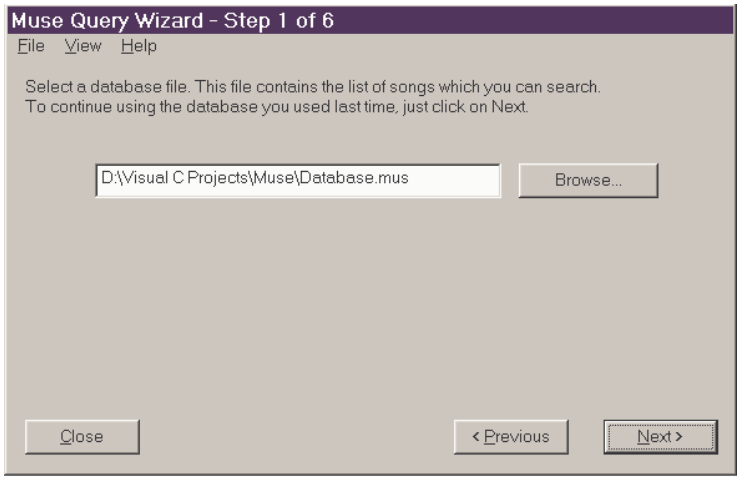
ניתן, בכל שלב, לעבור מה-Wizard אל המצב המפורט על ידי בחירת אפשרות Details בתפריט View. כדי לעבור חזרה ל-Wizard מתוך המצב המפורט, יש לבחור ב-Wizard Mode מתוך תפריט Settings.

א.1 שימוש ב-Wizard

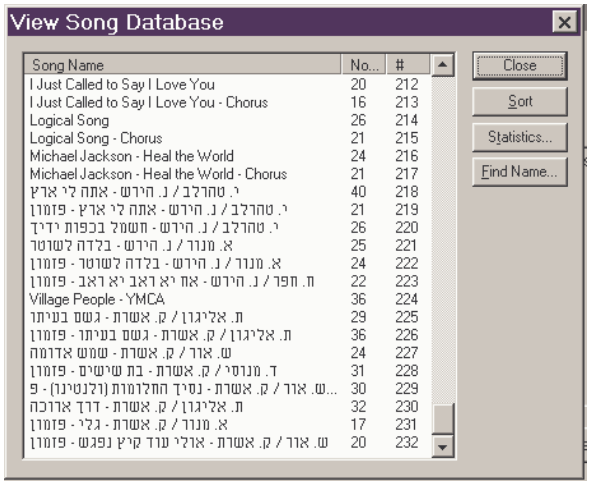
השימוש ב-Muse Wizard הינו פשוט ומהיר. התהליך מורכב משישה שלבים, שבכל אחד מהם המשתמש מקבל הסבר על הפעולות שעליו לבצע. ניתן לעבור בין השלבים על ידי לחיצה על כפתורי Next ו-Previous, ולצאת מהתוכנה על ידי לחיצה על Close. בשלב הראשון (איור 35), בוחר המשתמש את קובץ מאגר המידע שבו התוכנה תשתמש. קובץ זה מכיל את רשימת השירים שעליהם יתבצע החיפוש. ברירת המחדל היא הקובץ האחרון שנעשה בו שימוש. אפשרות זו קיימת על מנת לאפשר עבודה עם מספר מאגרי מידע שונים, למשל: בחירה בין חיפוש במאגר מידע של שירים עבריים ומאגר מידע של שירים לועזיים.

בכל שלב בהמשך, ניתן להתבונן ברשימת השירים שבמאגר המידע על ידי בחירת אפשרות Song List מתוך תפריט View. אפשרות זו מציגה רשימה של כל השירים במאגר

המידע, כמתואר באיור 36. מידע נוסף על שיר כלשהו מתקבל על ידי לחיצה כפולה על השיר המבוקש. פעולה זו פותחת את חלון ה-Note Writer עבור השיר המבוקש (איור 42). הפעולות בחלון זה יתוארו בסעיף א.2.



איור 35: דוגמה ל-Muse Wizard



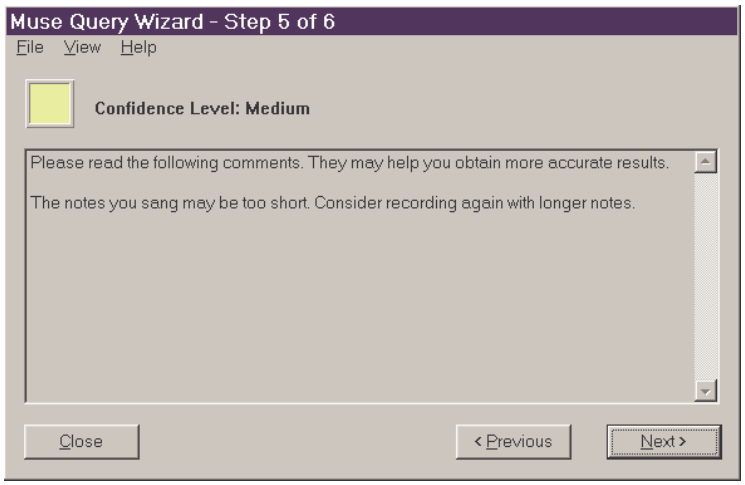
איור 36: רשימת השירים במאגר המידע

בשלב השני בוחר המשתמש את מקור ההקלטה. ניתן לבצע את ההקלטה בזמן ריצת התוכנית (מכרטיס הקול המותקן במחשב), או לחילופין לטעון קובץ הקלטה קיים.

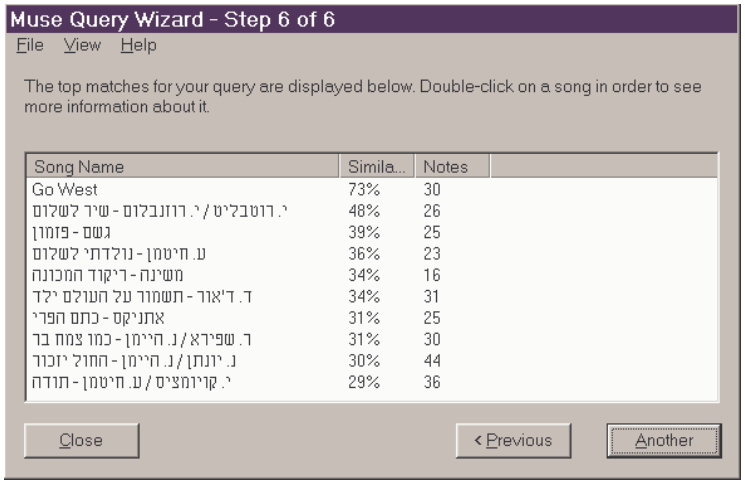
בשלב הבא מתבצעת ההקלטה או נבחר שמו של הקובץ המתאים. ההקלטה צריכה להתאים לדרישות הקלט שהוגדרו בסעיף 1.1.1. במקרה של קובץ מוקלט, יש להשתמש בקובץ מסוג Raw Audio File, בקצב דגימה של 8000 הרץ, רוחב דגימה של 16 ביט, והקלטה מונופונית. סידור הבתים הוא little-endian.

בשלב הרביעי מתבצע תהליך החיפוש. שלב זה עשוי לקחת מספר רגעים, במיוחד במחשבים איטיים, אך ככלל הוא מתבצע מהר יותר מאשר תהליך ההקלטה עצמו (ראו סעיף 7.4).

בשלב החמישי מוצג משוב למשתמש, אשר מאפשר לו לקבל מידע על טיב ההקלטה ועל אפשרויות לשיפור יכולת הזיהוי. התוכנית מחשבת מדד לרמת הבטחון בתוצאות, אשר מוצג באופן גרפי בתור אור ירוק (בטחון גבוה), אור צהוב (בעיות לא חמורות) או אור אדום (בעיות חמורות בזיהוי). במקרה שרמת הבטחון אינה גבוהה מוצגות הערות אשר מודיעות למשתמש כיצד ניתן לשפר את איכות ההקלטה. הערות אלה עשויות לכלול, למשל, הודעה כי קצב השירה מהיר מדי, ה-volume נמוך מדי, או שאין מספיק תווים בכדי לזהות את השיר. דוגמה להערה כזו מוצגת באיור 37.



איור 37: הערות המחשב בנוגע לטיב ההקלטה



איור 38: הצגת תוצאות החיפוש

לבסוף, בשלב השישי מוצגות תוצאות החיפוש. ההתאמה הטובה ביותר מוצגת ראשונה, ואחריה מספר התאמות נוספות. לכל שיר מצורף גם מדד התאמה, באחוזים, המייצג את מידת הקרבה בין ההקלטה לשיר במאגר המידע.

דוגמה לתוצאות כאלה מוצגת באיור 38. גם כאן, ניתן ללחוץ לחיצה כפולה על כל שיר ולקבל את חלון ה-Note Writer עבורו, כדי לקבל מידע נוסף על השיר. השימוש בחלון זה מתואר בסעיף א.2.

א.2 שימוש בתוכנה במצב Details

ניתן לבצע מגוון רחב מאוד של פעולות במצב Details, אך פעולות אלה דורשות כמות מסוימת של הבנה בנוגע לאופן פעולת התוכנית ולשימוש ב-Windows בכלל. בתיאור חלק זה נצא מתוך הנחה שנושאים אלה מובנים למשתמש.

ביצוע שאילתה במצב Details

התוכנית מכילה שני אובייקטים נפרדים שיש לטעון לפני ביצוע החיפוש. האובייקט הראשון הוא "קובץ" או "מאגר מידע", אשר מכיל את רשימת השירים עליהם יתבצע החיפוש. ניתן לפתוח ולשמור קבצים אלה בעזרת הפקודות בתפריט File, בדומה לאפליקציות Windows אחרות.

האובייקט השני שבו נעשה שימוש הוא הקלטה. ניהול ההקלטות מתבצע באמצעות תפריט Record. בתפריט זה, ניתן להקליט הקלטה חדשה (Record) או לטעון אותה מקובץ קיים (Load Raw), בעל פורמט Raw Audio כמתואר בסעיף א.1. בנוסף, ניתן לשמור הקלטה קיימת לקובץ Raw (באמצעות הפקודה Save Raw), וכן להשמיע את ההקלטה הקיימת (Playback).

לאחר פתיחת מאגר מידע והקלטה, ניתן לבצע את שלבי העיבוד, בזה אחר זה, באמצעות פקודות תפריט Analyze: Pitch Detection ולאחר מכן Segmentation. כמו כן, ניתן להעביר את ההקלטה דרך מסנן מעביר נמוכים בעל תדר קטעון של 1000 הרץ (פקודת Low-Pass Filter בתפריט Analyze); אך כפי שהוסבר בפרק 2, פעולה זו אינה נחוצה לביצוע זיהוי נכון, ואינה מתבצעת במצב Wizard.

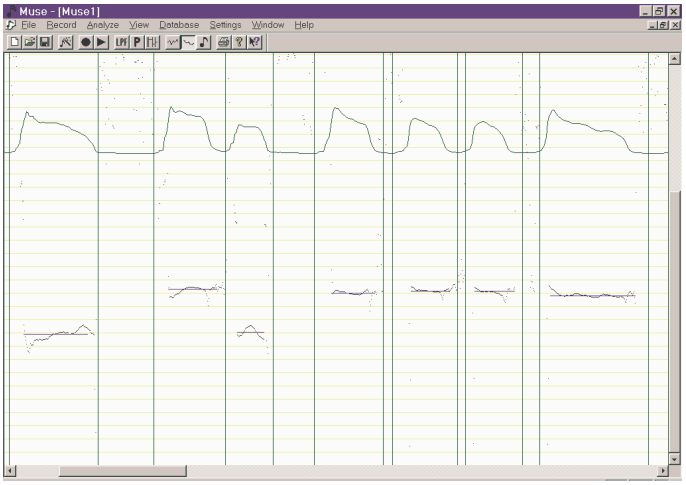
ישנם מספר מצבי תצוגה גרפיים של המידע, הניתנים לבחירה מתוך תפריט View. במצב Waveform, המומלץ בעת ביצוע הקלטות, ניתן לראות את צורת הגל של ההקלטה על המסך תוך כדי ההקלטה.

במצב התצוגה Pitch (אשר פעיל רק לאחר חישוב ערכי ה-pitch), ניתן לראות גרף של ה-pitch וה-volume כפונקציה של הזמן (איור 39). ה-pitch מוצג בצבע שחור, וה-volume מוצג

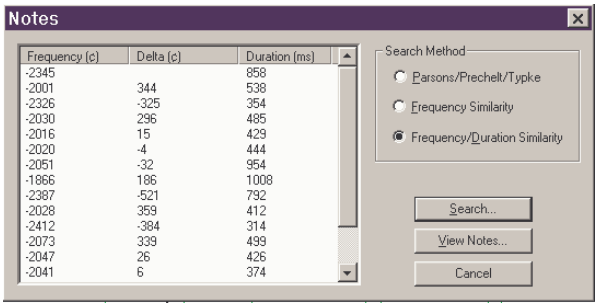
בירוק. כל פיקסל מציין מחזור pitch אחד. הקווים הצהובים האופקיים מייצגים יחידות; המרווח בין שני קווים כאלה הוא סמיטון אחד.

לאחר ביצוע הסגמנטציה, ניתן לראות גם את תוצאות הסגמנטציה במסך זה. תחילתו וסופו של כל סגמנט (ע"פ הסגמנטציה הראשונית) מוצגים על ידי קווים אנכיים ירוקים. ערכי הסגמנטציה השניונית מוצגים כקווים כחולים אופקיים, כאשר אורכם מציג את האיזור שנבחר לסגמנטציה השניונית, וגובהם מציין את התדר שנבחר עבור הסגמנט.

לבסוף, ניתן לראות במצב תצוגת Notes את רשימת התווים המזוהים באופן נומרי (איור 40). מתוך חלון זה ניתן גם לבצע חיפוש על ידי בחירה באופציית Search. כמו כן, ניתן להתבונן בתווים באופן מוסיקלי, בעזרת חלון Note Writer, שהשימוש בו יוסבר בהמשך (איור 42).



איור 39: תצוגה במצב Pitch



איור 40: תצוגת רשימת התווים

ניהול מאגר המידע

ניתן להתבונן ברשימת השירים במאגר המידע על ידי בחירת אפשרות View Songs בתפריט Database. אפשרות זו פותחת חלון (איור 36) אשר מכיל את רשימת השירים. צורת השימוש בחלון זה תוארה בסעיף א.1. נציין בנוסף, כי ניתן למחוק שירים מתוך מאגר המידע על ידי סימון שיר בחלון זה, ולחיצה על מקש Delete. כמו כן, ניתן למיין את השירים לפי סדר אלפבית, על ידי לחיצה על Sort, וכן ניתן למצוא שיר על פי כותרתו ע"י לחיצה על מקש Find Name. אפשרות Statistics מציגה נתונים כגון מספר השירים הכולל, מספר התווים בשיר ועוד.

כדי להוסיף שיר למאגר המידע, יש לבחור באפשרות Add Song מתפריט Database. אפשרות זו פותחת את חלון ה-Note Writer, אשר מאפשר עריכה ויצירה תווים בעזרת ממשק גרפי. הוראות השימוש בחלון זה יוצגו בהמשך.

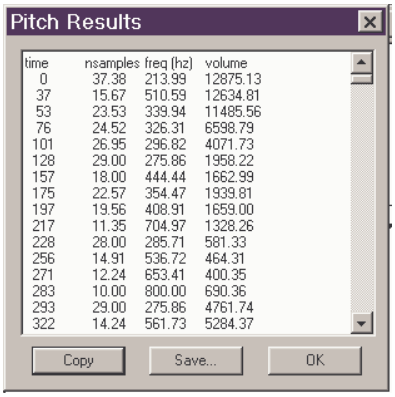
שינוי הפרמטרים של האלגוריתם

התוכנה מאפשרת שליטה רחבה מאוד על פרמטרים של האלגוריתם המבצע את הזיהוי. הגישה לפרמטרים אלה היא באמצעות הפקודות Pitch Detection, Segmentation, ו-Search מתוך תפריט Settings.

כל אחת מהאפשרויות האלה פותחת חלון עם פרמטרים המתייחסים לאותו חלק של האלגוריתם. ככלל, לא מומלץ לשנות את הפרמטרים האלה, מכיוון שהם נבחרו לאחר אופטימיזציה של המערכת. בכל מקרה, ניתן להחזיר את הפרמטרים לערכם המקורי על ידי לחיצה על כפתור Reset בחלון המופיע. כפתור זה מחזיר את ערכי הפרמטרים לערכים המתוארים בעבודה זו (סעיף 7.2), גם אם נשמרו ערכים שונים.

מכיוון שערכי הפרמטרים נשמרים ב-Windows Registry, שינוי של ערך כזה לא יועבר למחשב אחר כאשר התוכנה מועברת ממחשב למחשב. כאשר התוכנה מופעלת בפעם הראשונה על מחשב, היא יוצרת ב-Registry את ערכי הפרמטרים האלה לפי הערכים המתוארים בעבודה זו.

הפרמטרים של מנגנון ה-pitch כוללים מספר ערכים אשר עשויים להיות שימושיים לחלק מהמשתמשים. לדוגמה, ניתן להציג את תוצאות זיהוי ה-pitch באופן נומרי, כך שיהיה ניתן להעביר אותם לתוכנה שונה, כגון Excel או Matlab. כדי להפעיל אפשרות זו יש להדליק את האפשרות Display Numeric Results לאחר בחירת Settings|Pitch Detection. כאשר האפשרות מופעלת, ביצוע חישוב pitch גורם להצגת חלון כדוגמת זה המוצג באיור 41.



איור 41: הצגה נומרית של תוצאות חישוב ה-pitch

כמו כן, קיימת אפשרות ליצור קובץ Raw Audio אשר מכיל שחזור של הקובץ המקורי בעקבות זיהוי ה-pitch. קובץ זה מכיל מחזור אחד של גל סינוס עבור כל מחזור pitch, בעל תדר השווה לתדר ה-pitch המזוהה. התוצאה היא מנגינה אשר נשמעת דומה מאוד להקלטה המקורית, אך מכילה רק גלי סינוס, והיא עשויה לשמש להדגמת תקינות פעולת זיהוי ה-pitch.

בתפריט Settings ניתן גם להפעיל שלושה מצבי פעולה שונים של התוכנה: מצב Wizard, אשר תואר בסעיף א.1, וכן שני מצבי בדיקה של התוכנה (Batch ו-Noise Analysis), אשר מתוארים בהמשך.

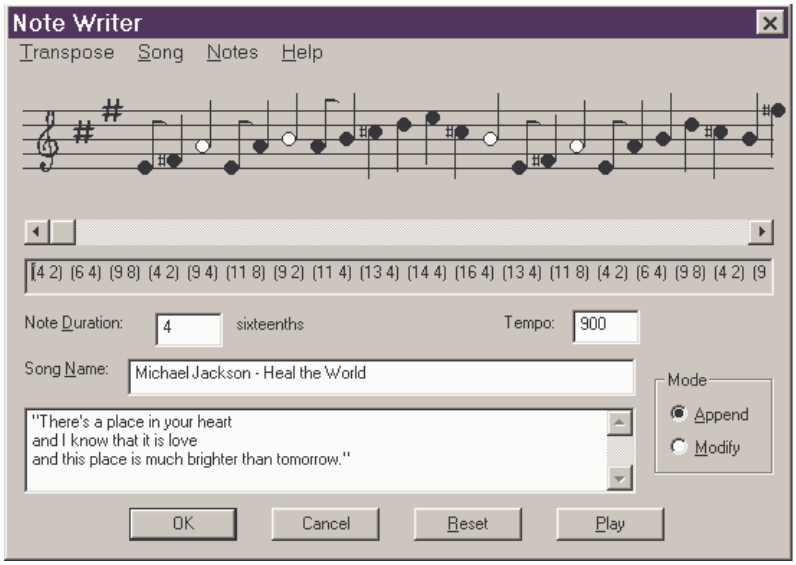
שימוש בחלון ה-Note Writer

חלון ה-Note Writer מוצג באיור 42. חלון זה מאפשר עריכה של שירים במאגר המידע, והוספת שירים חדשים. בנוסף, ניתן להגדיר את שם השיר ולצרף הערות, כגון מילות השיר.

ישנם שני מצבי פעולה בחלון זה, אשר נבחרים על ידי כפתורי ה-Mode בתחתית החלון. המצב הראשון, Append, מאפשר הוספת תווים לסוף השיר על ידי לחיצה על המקום המתאים על גבי החמשה המוצגת על המסך, בעזרת הכפתור השמאלי של העכבר. לחיצה על כפתור זה בתוספת Shift גורמת למחיקת התו האחרון המצוי על המסך, בדומה לפעולת מקש Backspace. משך הזמן של התו נקבע על ידי תיבת הטקסט Note Duration, בה ניתן לכתוב את משך הזמן הרצוי, ביחידות של 1/16 תו. לדוגמה, כדי לייצר תו-שמינית (כגון התו הראשון באיור 42), יש להקיש 2 בתיבת הטקסט.

במצב Append ניתן גם להגדיר סולמות, כלומר, רשימה של תווים שיושמעו כדיאז או במול בעת ניגון היצירה. כדי להגדיר סולם יש לבחור את התווים שיוחלפו בדיאז או בבמול. כדי להוסיף דיאז יש ללחוץ על התו הרצוי על גבי החמשה, בעזרת הכפתור הימני של העכבר. כדי להגדיר תו מסוים כבמול, יש ללחוץ על הכפתור הימני בתוספת Shift. כדי לבטל דיאז או במול

בתו מסוים, יש ללחוץ על הכפתור הימני בתוספת Ctrl. כל הפעולות האלה משפיעות רק על תווים שייכתבו בהמשך, כך שניתן להחליף סולם באמצע היצירה מבלי לשנות תווים קיימים.



איור 42: חלון ה-Note Writer

במצב הפעולה השני, Modify, משתנה השפעתם של כפתורי העכבר. הכפתור השמאלי משמש לשינוי תו קיים. על ידי לחיצה על החמשה, באיזור שבו קיים תו, ניתן לשנות את התדר ו/או משך הזמן של תו זה, כאילו התווסף במקומו תו חדש.

במצב Modify, הכפתור הימני משמש להוספת תווים באמצע היצירה. הוא פועל בדיוק כמו הוספת תווים במצב Append, אך מוסיף אותם בין שני תווים קיימים, בהתאם למקום שבו נמצא העכבר בזמן הלחיצה. כמו כן, ניתן למחוק תו מאמצע היצירה, על ידי לחיצה על Shift יחד עם הכפתור הימני, על גבי התו שיש למחוק.

ניתן לשמוע את היצירה על ידי לחיצה על מקש Play, או בחירת Play מתפריט Song. כדי להפסיק את השמעת היצירה ניתן לבחור ב-Stop מתפריט Play. היצירה מושמעת על ידי סינתוז התווים שלה כגלי סינוס בתדרים המתאימים. לצורך השמעת היצירה, ניתן לבחור Tempo מתאים בתיבת הטקסט המתאימה. ערך ה-Tempo אינו משפיע על תהליך הזיהוי, ומיועד רק לצורך השמעה נוחה יותר של היצירה. ערך זה הוא בעל יחידות שרירותיות ומתאר את משך הזמן של כל תו, כך שערך Tempo גבוה מציין נגינה איטית יותר.

קיימות מספר אפשרויות נוספות המיועדות לעריכה נוחה יותר של היצירה. בעזרת הפקודות בתפריט Transpose ניתן לבצע טרנספוזיציה של היצירה, למעלה או למטה, בסמיטונים בודדים או באוקטבות שלמות. ניתן גם למחוק כמות גדולה של תווים על ידי

בחירת אפשרות Delete Notes מתפריט Notes, והקשת טווח התווים שיש למחוק. כמו כן, ניתן לקבל תקציר של פקודות העריכה על ידי בחירת Instructions מתפריט Help.

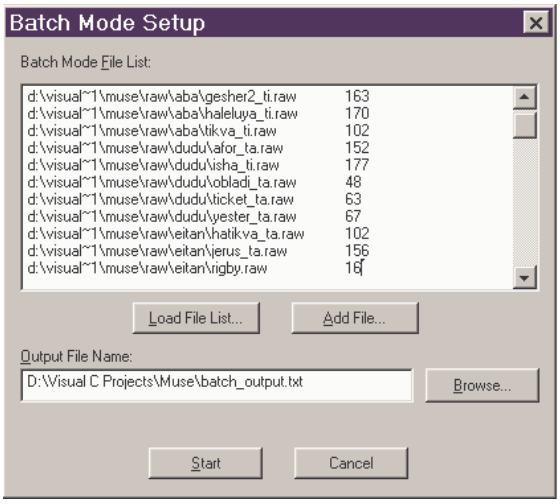
לבסוף, יש ללחוץ על OK על מנת לשמור את כל השינויים שבוצעו בשיר (או להוסיף אותו למאגר המידע, אם מדובר בשיר חדש), או ללחוץ על Cancel כדי לבטל את השינויים. בכל מקרה, השינויים יהיו קבועים רק אם קובץ מאגר המידע נשמר על ידי בחירה באפשרות Save מתפריט File בחלון הראשי.

שימוש במצב Batch

מצב Batch מיועד להפעלת התוכנה במצבים מיוחדים, שבהם יש צורך לבדוק מספר גדול של שאילתות ללא התערבות אדם. במהלך העבודה נעשה שימוש נרחב באפשרות זו על מנת לבדוק את ביצועי התוכנה תוך שינוי פרמטרי האלגוריתמים.

כדי להפעיל את מצב Batch, יש לבחור באופציית Batch Mode מתוך תפריט Settings. אפשרות זו פותחת את חלון Batch Mode Setup שבו מוגדרת מראש רשימת ההקלטות שיש לנתח ושם קובץ שאליו יירשמו התוצאות (איור 43).

בשימוש פשוט של מצב batch, רשימת הקבצים מכילה את אוסף הקבצים שיש לנתח, קובץ אחד בכל שורה, עם full path לכל קובץ. במצב זה יתבצע חיפוש רגיל על כל הקבצים, ותוצאות החיפוש (10 ההתאמות הטובות ביותר), עבור כל קובץ, יירשמו לקובץ התוצאות. יש להקיש את רשימת הקבצים במקום המתאים, או לטעון אותה מתוך קובץ טקסט המכיל את הרשימה, או לבחור את הקבצים אחד אחד באמצעות אופציית Add File.



איור 43: מצב Batch

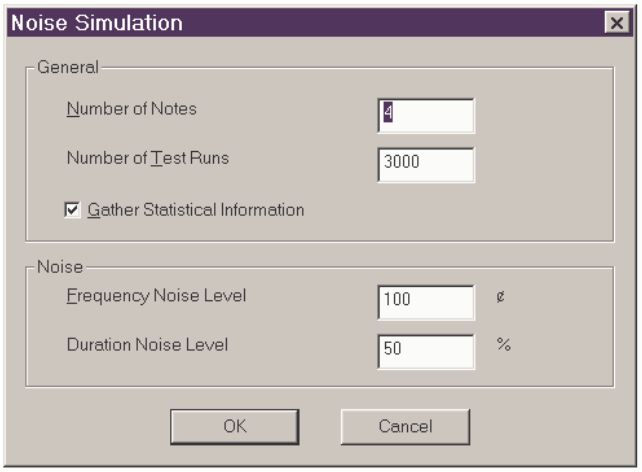
ניתן גם להשתמש במצב batch כדי למצוא אוטומטית את אחוז המקרים שבהם חל זיהוי נכון של השירים. כדי לעשות זאת, יש להוסיף לאחר כל שם קובץ רווח אחד או יותר, ואז להקיש את המספר הסידורי של השיר במאגר המידע. המספר הסידורי של השיר מופיע לצד השיר כאשר מסתכלים על רשימת השירים מתוך Database | View Songs.

במצב פעולה מיוחד זה, קובץ התוצאות אינו מכיל את רשימת עשר ההתאמות הטובות ביותר של כל שיר. במקום זאת, מוצג עבור כל שיר רק אחוז ההתאמה עם השיר הנכון, וכן המיקום של שיר זה בטבלת ההתאמות. לדוגמה, אם היו שני שירים אחרים בעלי התאמות יותר גבוהות, יירשם מיקום השיר כ-3, כיוון שהוא נמצא במקום השלישי בטבלת ההתאמות. בנוסף, יירשמו בסוף הקובץ מספר ההקלטות הכולל, מספר ההקלטות שזוהו במקום הראשון, ומספר ההקלטות שזוהו בין חמשת המקומות הראשונים.

שימוש במצב Noise Simulation

מצב Noise Simulation משמש לביצוע סימולציות על אלגוריתם החיפוש, מהסוג שתוארו בסעיף 6.3.1. כדי לבצע סימולציה כזו, יש לבחור באפשרות Noise Simulation מתוך תפריט Settings. החלון שנפתח מתואר באיור 44.

בחלון זה, ניתן לבחור את מספר התווים שיכללו בשאילתה, מספר ההרצות, ורמת הרעש המתווספת לתווים (ראו סעיף 6.3.1). לביצוע הסימולציה, יש ללחוץ על OK.



איור 44: מצב Noise Simulation

ב. הוראות קומפילציה

התוכנה נכתבה תחת Microsoft Developer Studio 98 ומיועדת לקומפילציה על Microsoft Visual C++ 6.0. התוכנה נכתבה תחת Windows 95, ונבדקה תחת מערכות ההפעלה Windows 95, Windows 98, Windows NT 4.0.

התוכנה משתמשת ב-Microsoft Foundation Classes (MFC), גירסה 4.2, ויש צורך לצרף אליה את הקובץ MFC42.DLL המעודכן לגירסה המתאימה של Visual C++.

בדיסק המצורף לעבודה זו מצויים, בנוסף לקבצי ה-source code, גם קבצי ה-desktop שבעזרתם ניתן לפתוח את התוכנית ב-Visual C++, יחד עם כל אופציות הקומפילציה. בנוסף, כדי לאפשר תאימות מירבית עם גרסאות עתידיות, מצורף לדיסק קובץ MUSE.MAK שמכיל תיאור טקסטואלי של תהליך הקומפילציה והשרשור, בשפת NMAKE של Visual C++.

ג. מבנה התוכנה

נספח זה מכיל מידע טכני הדרוש לשם הרחבה עתידית של התוכנה, וכן לשימוש בחלקים שלה לתוכנות אחרות. בסעיף ג.1 תינתן סקירה כללית של מבנה פעולת התוכנה. סעיף ג.2 יתאר שלושה חלקים של התוכנה אשר ניתנים להעברה לתוכנות אחרות באופן פשוט, ומוסבר אופן השימוש בחלקים אלה בלי תלות בשאר חלקי התוכנה. סעיף ג.3 מתאר את פורמט קבצי מאגר המידע של התוכנה, כדי לאפשר לתוכנות אחרות לקרוא ולכתוב קבצים כאלה.

ג.1 תיאור כללי

התוכנה המלאה מכילה מעל 10,000 שורות קוד ומספר רב של מודולים ומחלקות. כדי לייעל את תהליך הקריאה של קוד התוכנה, מתואר כאן בקיצור המבנה והארגון של התוכנה. בסעיף זה נניח ידע מסוים בשיטת כתיבת קוד תחת MFC.

התוכנה בנויה סביב ארכיטקטורת ה-document/view (מסמך/תצוגה) של MFC, בשיטת multiple document interface (MDI). גם במצב ה-Wizard, קיימים אובייקטים של מסמך ותצוגה, אלא שהם חבויים (אופציית visible של החלונות אינה פעילה). כנהוג ב-MFC, עיקר המידע מצוי כאיברים באובייקט המסמך, ואובייקט התצוגה מכיל רק מידע הקשור ישירות לאופן תצוגת המסמך. בין היתר, מכיל המסמך את המידע הבא:

- מערך של שירים אשר מהווה את מאגר המידע (m_database)

- ההקלטה, בפורמט Raw Audio (m_data)
- מערך ערכי ה-pitch (m_pitch)
- מערך ערכי ה-volume (m_volume)
- מערך ערכי הסגמנטציה הראשונית, המוצג בתור אינדקסים לתוך m_pitch שבהם נמצא ההתחלה או הסיום של הסגמנט (m_break)
- מערך של סגמנטים, המכיל את תדר התו של הסגמנט, זמן ההתחלה והסיום שלו, ועוד (m_segment)

במהלך שלבי העיבוד השונים, מתמלאים איברים אלה של המסמך במידע. עם זאת, יש לציין כי רק מאגר המידע עצמו (m_database) הוא הנשמר לקובץ מאגר המידע.

רוב שלבי העיבוד ממומשים כפונקציות-חבר (member functions) של המסמך. כך, למשל, CMuseDoc::OnAnalyzePitch() מבצעת את תהליך זיהוי ה-pitch. באופן כזה, ניתן לקרוא לפונקציה ממספר מקומות שונים, כפי שנדרש בגלל מגוון צורות ההפעלה האפשריות של התוכנה (מצב Wizard, מצב Batch וכד'). יוצא מהכלל הוא תהליך החיפוש במאגר המידע, אשר מבוצע מתוך הפוקציה CDlgNotes::OnSearch() אשר שייכת לחלון רשימת התווים. כאשר החיפוש מתבצע בלא פתיחת חלון זה (לדוגמה, במצב Wizard), החלון נטען ומושאר במצב לא נראה, על מנת לקרוא לפונקציה זו.

כל חלקי התוכנה נכתבו על ידי המחבר במהלך ביצוע הפרויקט, מלבד המחלקה CDib אשר מיועדת להצגת bitmaps על המסך. המחלקה נלקחה מספרו של Kruglinski (1997), וצורפה לתוכנה באישור Microsoft Corp.

ג.2 מודולים נפרדים

שלושה חלקים של התוכנה עשויים להיות שימושיים גם בתוכנות אחרות, ולכן הם תוכננו כך שניתן יהיה לעשות בהם שימוש מחוץ ל-Muse. מודולים אלה הינם: המחלקה לקבלת קלט מכרטיס הקול, אלגוריתם זיהוי ה-pitch, ואלגוריתם חישוב מרחק העריכה. צורת השימוש בהם תתואר בסעיף זה.

קבלת קלט מכרטיס קול

המחלקה CWaveIn מיועדת לביצוע הקלטות מכרטיס קול, ומאפשרת לשמור את ההקלטות לשימוש מאוחר יותר או להשתמש בהן בזמן אמת. המחלקה נבדקה על Visual C++ בגרסאות 5.0 ו-6.0, בעבודה במצב MFC. המחלקה מצויה בקבצים WaveIn.h ו-WaveIn.cpp המצויים יחד עם שאר קבצי הקוד של התוכנה.

ההקלטה מבוצעת תוך שימוש ב-double-buffering, והזכרון שעליו נשמרת ההקלטה מוקצה באופן דינמי על ידי המחלקה. ההקלטה מתבצעת ל-buffers בגודל אשר נקבע ע"י המשתמש. ניתן לשלוף את ה-buffers בזמן אמת או בכל שלב לאחר סיום ההקלטה. ההקלטה מתבצעת על thread נפרד, אך ניהול ה-thread מתבצע באופן פנימי על ידי המחלקה, והפונקציות החיצוניות של המחלקה הינן thread-safe.

יצירת האובייקט צריכה להתבצע לאחר שהחלון העיקרי של התוכנה נוצר (למשל, על ידי הקצאה דינמית של האובייקט מתוך CDocument::OnInitDocument). ה-constructor של המחלקה הוא בעל הפורמט הנ"ל:

```
CWaveIn::CWaveIn(UINT uDeviceID, LPWAVEFORMATEX pwfx,
        UINT uBufSize, HWND hwndNotify=NULL);
```

uDeviceID: זיהוי של המכשיר המבצע את ההקלטה, או לחילופין הערך WAVE_MAPPER, המורה ל-Windows לקבוע את מכשיר ההקלטה המתאים ביותר.

pwfx: מצביע למבנה המכיל מידע אודות סוג ההקלטה, כגון קצב הדגימה. למידע על מבנה זה, יש לעיין בעזרה של Windows API על מבנה WAVEFORMATEX.

uBufSize: גודל כל buffer מוקלט, בבתים.

hwndNotify: חלון אשר יקבל הודעה כאשר מתמלא buffer (אופציונלי).

לאחר יצירת האובייקט, ניתן להתחיל את ההקלטה על ידי הפונקציה CWaveIn::Start(), ולהפסיק אותה על ידי הפונקציה CWaveIn::Stop(). שתי הפונקציות אינן מקבלות פרמטרים ומחזירות void.

במהלך ההקלטה או לאחריה, ניתן לקבל את ה-buffers המוקלטים באמצעות הפקודות הבאות:

```
BOOL CWaveIn::StackIsEmpty()
```

פקודה זו מחזירה TRUE אם אין אף buffer מוקלט המחכה להשלף, ו-FALSE אחרת.

```
WAVEHDR* CWaveIn::StackPop()
```

פקודה זו שולפת מרשימת ה-buffers המוקלטים את ה-buffer המוקדם ביותר, ומחזירה מצביע אליו ; אם אין buffer מוכן, מחזירה NULL.

הערה חשובה: לאחר סיום השימוש ב-buffer, יש לפנות את הזכרון שהוקצה עבורו. לצורך כך ניתן להשתמש ב-macro הבא, אשר מוגדר ב-WaveIn.h:

```
#define DeleteWaveHdr( pwh )   { delete[] pwh->lpData;  delete pwh; }
```

לצורך שימוש במחלקה ביישומים בזמן אמת, ייתכן שיהיה נוח למשתמש לקבל הודעה לחלון כלשהו בכל פעם שמתמלא buffer, כדי למנוע מצב של קריאות חוזרות ונשנות (polling) לפונקציית StackIsEmpty(). לצורך כך יש להגדיר, בעת יצירת המחלקה, חלון שיקבל הודעות אלה. חלון זה יקבל הודעות WM_WAVEIN_NOTIFY_PUSH בכל פעם שמתמלא buffer. הודעה זו הינה user-defined message אשר מוגדרת ב-WaveIn.h.

במקרה של שגיאה כלשהי בתהליך ההקלטה, התוכנה מציגה הודעת שגיאה מתאימה על המסך, ולאחר מכן יוצרת user exception.

זיהוי pitch

פונקציות זיהוי ה-pitch מצויות בקבצים Pitch.cpp ו-Pitch.h. בעזרת פונקציות אלה ניתן לבצע הן חישוב pitch שלם והן חישוב super-resolution pitch. הפונקציות נבדקו תחת Visual C++ בגרסאות 5.0 ו-6.0, אך מכילות רק קוד C++ סטנדרטי וצריכות לפעול על כל קומפיילר שהוא.

הפונקציות מיועדות להרצה על דגימות ברוחב 16 ביט, והקובץ Pitch.h מגדיר את סוג המשתנה sample כך:

```
typedef short sample;
```

משתנה זה אמור להיות באורך 16 ביט. העברת מידע קולי לפונקציות צריכה להתבצע באמצעות מערכים של משתנים מסוג sample.

לחישוב integer pitch (כלומר, חישוב pitch בשיטת אוטוקורלציה סטנדרטית, כמתואר בסעיף 3.3) יש להשתמש בפונקציה int_pitch הבאה:

```
int int_pitch( sample[] a, int minpitch, int maxpitch, int step, float* correl );
```

a: מערך דגימות שעליהם יבוצע חישוב ה-pitch

minpitch: הערך המינימלי המותר של pitch, בדגימות

maxpitch: הערך המקסימלי המותר של pitch, בדגימות

step: הרזולוציה שבה נבדקים ערכי pitch, בדגימות. ניתן להשתמש באפשרות זו על מנת לבצע חישוב pitch מהיר יותר, על חשבון הקטנת הרזולוציה של התוצאה.

correl: מצביע למשתנה שבו יוכנס ערך האוטוקורלציה של ה-pitch שנבחר.

ערך מוחזר: ערך ה-pitch של האות, בדגימות.

לחישוב super-resolution pitch יש להשתמש בפונקציה sr_pitch הבאה:

```
float sr_pitch( sample[] a, int minpitch, int maxpitch, int step, float* correl,
        int* nrep = NULL );
```

השימוש של כל הפרמטרים זהה לשימוש בפונקציה int_pitch, מלבד הפרמטר step אשר קובע את ערך ה-decimation (ראה Medan et al., 1991), ומקטין את דיוק השערוך (ולא את הרזולוציה). בעבודה זו לא נעשה שימוש באופציית ה-decimation.

הפרמטר nrep מצביע למשתנה, אשר מקבל ערך של מספר הפעמים שבהם בוצע תהליך ה-super-resolution (ראו דיון על חוסר היציבות של התהליך, סעיף 3.4). פרמטר זה הינו אופציונלי.

הערך המוחזר הוא מחזור ה-pitch ביחידות של דגימות, אך באופן כללי לא חייב להיות שלם.

חישוב מרחק עריכה

המחלקה CEditDist הינה abstract template class אשר מאפשרת לבצע חישובי מרחק עריכה (סעיף 5.2) על כל אלפבית שהוא. המחלקה נבדקה על Visual C++, גרסאות 5.0 ו-6.0, וצריכה לפעול על כל קומפיילר ל-C++ אשר תומך ב-templates. הפונקציה מוגדרת בקבצים EditDist.cpp ו-EditDist.h, המצורפים לעבודה זו יחד עם שאר הקוד של התוכנה.

השימוש במחלקה CEditDist מתבצע בכמה שלבים. ראשית, יש לייצר instance של המחלקה עבור סוג האלפבית שעליו ייערך החישוב. לאחר מכן, יש ליצור derivation מתוך מחלקה זו, שבה מוגדרות פונקציות אשר מחזירות את מחיר פעולות העריכה הבסיסיות. לבסוף, השימוש במחלקה ייעשה על ידי יצירת אובייקט של ה-derived class, וקריאה לפונקצייה שלו.

כדי לתאר את פרטי התהליך, יוסברו השלבים ליצירת מחלקה המחשבת את מרחק העריכה מעל אלפבית של כל המספרים השלמים, כאשר מחיר הוספה או מחיקה של איבר יהיה 5 ומחיר השינוי של איבר יהיה 3.

השלב הראשון הוא יצירת instance של המחלקה CEditDist, עבור סוג האלפבית שבו נשתמש. במקרה שלנו האלפבית מתואר על ידי סוג המידע int, ולכן נשתמש בפקודה

```
template class CEditDist<int>;
```

ניתן להשתמש בפקודה זו גם עם סוגי מידע מורכבים (כגון מחלקות), במקרה שיש צורך בתמיכה באלפבית מורכב יותר.

בשלב שני, יש ליצור derivation מתוך המחלקה CEditDist<int>. בתוך derivation זה יוגדרו מחירי פעולות העריכה הבסיסיות, על ידי שינוי הפונקציות הווירטואליות DeleteCost, InsertCost, ChangeCost. הקוד לביצוע ה-derivation הוא:

```
class CIntEditDist : public CEditDist<int>
{
public:
        int DeleteCost(const int& deleted, int x, int y) { return 5; };
        int InsertCost(const int& inserted, int x, int y) { return 5; };
        int ChangeCost(const int& from, const int& to, int x, int y)
                { return (from==to ? 0 : 3); };
};
```

ביישומים מורכבים יותר, ניתן להשתמש בפרמטרים של הפונקציות על מנת להגדיר מחירי עריכה התלויים באיברים הנמחקים. הפרמטרים deleted, inserted, from, to, הינם האיברים שעליהם מתבצעת פעולת העריכה; סוגם של פרמטרים אלה ישתנה עם שינוי סוג המידע שעליו מוגדרת המחלקה. הפרמטרים x,y מתארים את מיקום פעולת העריכה בתוך שתי המחרוזות (או הקואורדינטות בתוך גרף מרחק העריכה המתואר בסעיף 5.3).

ניתן, בנוסף, להשתמש מתוך פונקציות אלה באיברים m_xmax, m_ymax אשר מוגדרים במחלקת הבסיס, על מנת למצוא את האורך הכולל של שתי המחרוזות. איברים אלה הינם בעלי גישת protected ומכילים מידע תקין רק בזמן קריאה ל-EditDistance. השימוש באיברים אלה מאפשר להגדיר מחירים התלויים באורכי המחרוזות, כפי שהוסבר בסעיף 5.4.2.

לבסוף, השימוש במחלקה ייעשה על ידי יצירת אובייקט מסוג CIntEditDist, ושימוש בפונקציה EditDistance (שהתקבלה בירושה מ-CEditDist):

```
int CEditDist<int>::EditDistance(int* ar1, int len1, int* ar2, int len2);
```

הפרמטרים ar1,ar2 הינם מצביעים לשני מערכים המכילים את המחרוזות שעליהן תתבצע ההשוואה. הפרמטרים len1,len2 מציינים את אורכי שתי המחרוזות, בהתאמה.

הפונקציה EditDistance קוראת לפונקציות המחיר על מנת למצוא את מחיר העריכה המינימלי, אותו היא מחזירה. החישוב מתבצע על פי האלגוריתם שתואר בסעיף 5.3.2.

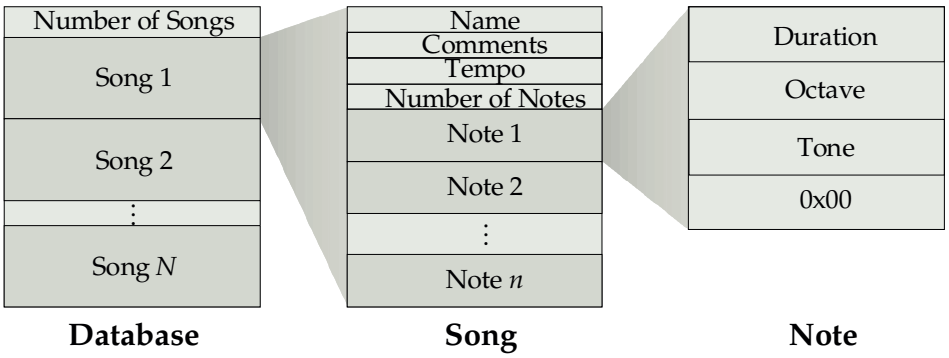
ג.3 פורמט קובץ מאגר המידע

בסעיף זה יתואר הפורמט של קובץ מאגר המידע של Muse, על מנת לאפשר לתוכנות חיצוניות לקרוא ולכתוב קבצים כאלה. הקובץ מיוצר על ידי מנגנון ה-Serialization של MFC, אך יתואר כאן באופן מדויק המאפשר קריאה גם על ידי תוכנות אחרות. הסיומת המוצעת עבור קבצים כאלה היא MUS.*, והם נקראים Muse Database Files.

בסעיף זה, כאשר מדובר על "רישום int לקובץ", הכוונה היא לרישום משתנה באורך 32 ביט, בצורת רישום little-endian.

כאשר מדובר על "רישום מחרוזת לקובץ", הכוונה היא לרישום CString על פי הסטנדרט של MFC. על פי סטנדרט זה, נרשם תחילה אורך המחרוזת ולאחריו המחרוזת כולה. אורך המחרוזת נרשם באופן הבא: אורכים של עד 254 בתים נרשמים על בית אחד. אורכים של עד 65533 בתים נרשמים על ידי רישום 0xFF בבית הראשון, ולאחריו שני בתים של אורך המחרוזת בפורמט little-endian. מחרוזות ארוכות יותר נרשמות על ידי רישום 0xFF בשלושה בתים, ולאחר מכן רישום אורך המחרוזת על ארבעה בתים בפורמט little-endian.

המבנה הכללי של קובץ מאגר המידע מתואר באיור 45. באיור זה, סדר הרישום לקובץ הוא מלמעלה למטה.



איור 45: מבנה קובץ מאגר המידע

הקובץ מתחיל במספר (int) אשר מציין את מספר השירים במאגר המידע. לאחר מספר זה מופיעה רשימה של שירים ברצף, כאשר הפורמט של כל שיר הוא כדלקמן: השיר מתחיל בשתי מחרוזות המתארות את שמו ואת ההערות המצורפות לשיר. לאחר מכן מופיע int המתאר את קצב הניגון הרצוי של השיר. קצב זה אינו משפיע על תהליך זיהוי השיר, אך מיועד להשמעה נוחה יותר של השיר מתוך חלון ה-Note Writer (ראו סעיף א.2). לבסוף, מופיע int המציין את מספר התווים בשיר, ולאחר רשימת תווים בזה אחר זה.

כל תו מיוצג באופן הבא: ראשית מופיע int המציין את משך הזמן של התו, ביחידות של 1/16 תו. הייצוג זהה לייצוג תווים בחלון ה-Note Writer (סעיף א.2). לאחר מכן, מופיע תיאור האוקטבה של התו, בתור בית יחיד (ולא בייצוג int הרגיל, על מנת לחסוך במקום). כל אוקטבה מכילה את התווים מדו ועד סי, כאשר אוקטבה מספר 4 מכילה את התו סול שתדרו

440 הרץ. טווח האוקטבות האפשרי הוא מ0- ועד 8; טווח זה מכסה את כל האוקטבות שבפסנתר ועוד אוקטבה אחת לכל כיוון.

לאחר האוקטבה מופיע בית נוסף המתאר את הצליל. בית זה יכול לקבל את הערכים 0 עד 11, כאשר 0 מייצג דו, 1 מייצג דו דיאז, וכך הלאה עד 11 המייצג את סי. כדי לשמור במולים בפורמט זה, יש לבצע חילוף אנהרמוני (לדוגמה, להחליף את רה במול בדו דיאז). חילוף כזה מתבצע באופן אוטומטי על ידי ה-Note Writer.

בסוף התו, מופיע בית נוסף שערכו צריך להיות 0 תמיד. בית זה מיועד לאפשר הרחבה עתידית של הפורמט.

מקורות

Baeza-Yates, R., and Perleberg, C. H. (1992), "Fast and Practical Approximate String Matching," in Apostolico, A., Crochemore, M., Galil, Z., and Manber, U. (eds.), *Combinatorial Pattern Matching*, vol. 644 in *Lecture Notes in Computer Science*, Springer-Verlag.

Beauchamp, J. W. (1969), "A Computer System for Time-Variant Harmonic Analysis and Synthesis of Musical Tones," *Music by Computers*, John Wiley and Sons.

Borchers, J. (1997), "Worldbeat: Designing a baton-based interface for an interactive music exhibit," *Proc. ACM Intl. Conf. on Comp.-Human Interaction*, pp. 131-138.

Dowling, W. J. (1978), "Scale and Contour: Two Components of a Theory of Memory for Melodies," *Psychological Review* **85**: 4.

Estes, W. K. (1994), *Classification and Cognition*, Oxford Psychology Series Volume 22, Oxford University Press.

Feldstein, S. (1982), *Practical Dictionary of Music Theory*, Alfred Publishing.

Foster, S., Schloss, W. A., and Rockware, A. J. (1982), "Toward an Intelligent Editor of Digital Audio: Signal Processing Methods," *Comp. Music J.* **6**: 1.

Ghias, A., Logan, J., Chamberlin, D., and Smith, B. C. (1997), "Query by Humming: Musical Information Retrieval in an Audio Database," Dept. of Computer Science, Cornell University. Available online at `http://www.cs.cornell.edu/zeno/Papers`

Gold, B. (1962), "Computer Program for Pitch Extraction," *J. Acoust. Soc. Amer.*, **34**: 7.

Helmholtz, H. L. F. (1885), *On the Sensations of Tone as a Physiological Basis for the Theory of Music*, 2nd ed., Dover.

Hess, W. (1983), *Pitch Determination of Speech Signals*, Springer-Verlag.

Knight, J. R., and Myers, E. W. (1992), "Approximate Regular Expression Pattern Matching with Concave Gap Penalties," in Apostolico, A., Crochemore,

M., Galil, Z., and Manber, U. (eds.), *Combinatorial Pattern Matching*, vol. 644 in *Lecture Notes in Computer Science*, Springer-Verlag.

Kruglinski, D. (1997), *Inside Visual C++*, 4th ed., Microsoft Press.

Levarie, S., and Levy, E. (1980), *Tone: A Study in Musical Acoustics*, 2nd ed., Greenwood Press.

McNab, R. J., Smith, L. A., Witten, I. H., Henderson, C. L., and Cunningham, S. J. (1996), "Towards the Digital Music Library: Tune Retrieval from Acoustic Input," *Proc. ACM Digital Libraries '96*, pp. 11-18.

McNab, R. J., Smith, L. A., Bainbridge, D., and Witten, I. H. (1997), "The New Zealand Digital Library Melody Index," *D-Lib Magazine*, May 1997. Available online at `http://mirrored.ukoln.ac.uk/ lis-journals/dlib/dlib/may97/meldex/05witten.html`

Medan, Y., Yair, E., and Chazan, D. (1991), "Super Resolution Pitch Determination of Speech Signals," *IEEE Trans. on Signal Processing*, **39**: 1.

Mongeau, M., and Sankoff, D. (1990), "Comparion of Musical Sequences," *Computers and the Humanities*, **24**: pp. 161-175.

Moore, B. J. C. (1997), *An Introduction to the Psychology of Hearing*, Academic Press.

Moorer, J. A. (1975), *On the Segmentation and Analysis of Continuous Musical Sound by Digital Computer*, Stanford University Ph.D. Thesis, Stanford University, Report No. STAN-M-3.

Myers, E. W. (1986), "An $O(ND)$ Difference Algorithm and Its Variations," *Algorithmica*, **1**: pp. 251-266.

Myers, E. W., and Miller, W. (1989), "Approximate Matching of Regular Expressions," *Bulletin of Math. Biol.*, **51**: pp. 5-37. Available online at `http://www.cs.arizona.edu/people/gene/PAPERS/reg.aprox.ps`

Parsons, D. (1975), *Directory of Tunes and Musical Themes*, Spencer Brown.

Porat, B. (1997), *A Course in Digital Signal Processing*, John Wiley and Sons.

Prechelt, L., and Typke, R. (1999), "An Interface for Melody Input," *Trans. Comp.-Human Interact.*, to be published.

Seiger, N. J., and Tewfik, A. H. (1998), "Audio Coding for Representation in MIDI via Pitch Detection Using Harmonic Dictionaries," *J. VLSI Signal Processing*, **20**, 45-59.

Sondhi, M. M. (1968), "New Methods of Pitch Extraction," *IEEE Trans. Audio Electroacoust.*, **16**, 262-266.

Tanguiane, A. S. (1993), *Artificial Perception and Music Recognition*, vol. 746 in *Lecture Notes in Artificial Intelligence* (Subseries of the *Lecture Notes in Computer Science*), Springer-Verlag.

Taylor, C. (1992), *Exploring Music: The Science and Technology of Tones and Tunes*, Institute of Physics Publishing.

Terhardt, E., Stoll, G., and Seewann (1982), "Algorithm for Extraction of Pitch and Pitch Salience from Complex Tonal Signals," *J. Accoust. Soc. Am.* **71**: 3.