דחיסת תמונה וקול

מרצה: מר נמרוד פלג, סמסטר ב' 2017
נכתב ע"י קובי בר-חנין

<u>הערות:</u>

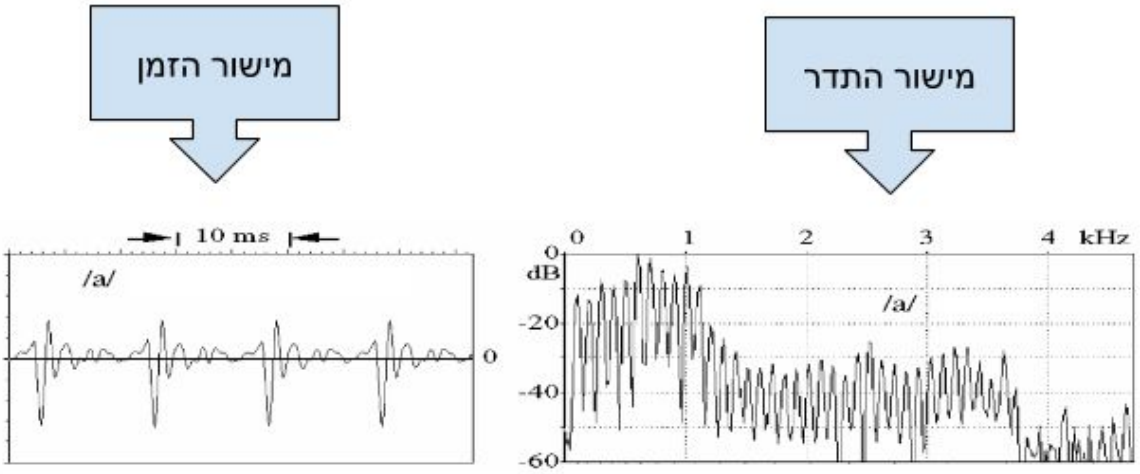
- **סיכום זה נכתב ע"מ לעזור לי להבין את החומר ולא במטרה להיות בשימוש נרחב, בהחלט יכולות להיות טעויות.**
- **חלק מהסיכום מבוסס על סיכומים אחרים שנכתבו ע"י סטודנטים שלקחו את הקורס איתי.**

בהצלחה!

דחיסת קול

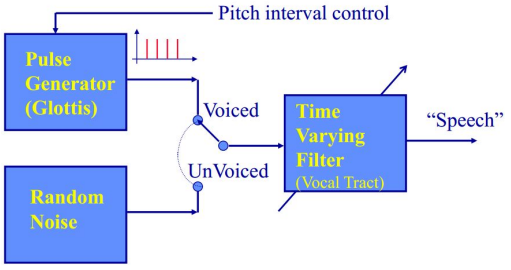
1. הברות בדיבור מתחלקות לשני סוגים:
   a. Voiced – מחזורי. נוצר על ידי מיתרי הקול.
   b. UnVoiced – לא מחזורי. נוצר על ידי שינוי צורת חלל הפה, בעזרת הלשון, השפתיים וכו'.
2. בקורס אנו לומדים שתי גישות עיקריות לקידוד (דחיסת) דיבור: קידוד של **גל הדיבור**, וקידוד של **מאפייני הדיבור** (VOCODERS). בתעשייה משתמשים בעיקר בשילוב בין שתי השיטות.
3. ה-PITCH = תדירות רעידת מיתרי הקול (לכל צורך מעשי בקורס זה):
   a. נמדד ביחידות של הרץ (X הרץ = X מחזורים בשניה).
   b. אם ה-PITCH גבוה אנחנו תופסים את הצליל כגבוה ולהיפך.
   c. **<u>הרחבה:</u>** ה-Pitch זו תכונה של גל הקול הנובעת ממחזוריות פתיחה וסגירה של מיתרי הקול, זו התכונה המאפשרת לקבוע האם קול נתון הוא "גבוה" או "נמוך". עבור גברים ה-Pitch בתחום 50-200 הרץ, ועבור נשים 120-500 הרץ.
   היינו רוצים לאמור שה-Pitch הוא תדר הדיבור שלנו בכל נקודת זמן, אך דיי במבט על התמונה הרגעית של הקול כשם שמיוצג במישור התדר כדיי להבין שבכל צליל שאנו מפיקים מעורבים תדרים רבים במידת השפעה שונה. אז נשאלת השאלה- מהו התדר לו ניתן לשייך את תכונת "גובה" הצליל? נרצה לדעת לשערך את התדר האופייני ל-Pitch (בהמשך).
4. ייצוג של אות קולי כגל (מישור הזמן - מישור התדר):

- כך נראה גל קול במישור התדר ובמישור הזמן- בין השניים מקשרת התמרת פורייה:



- קל לראות כי מרבית האנרגיה בקול נמצאת בתדרים הנמוכים, מעט אנרגיה בתדרים הגבוהים (נשתמש בעובדה זו בהמשך).

5. מסלול גל הקול:
   a. אוויר יוצא מהריאות לעבר מיתרי הקול.
   b. במיתרי הקול מועמסת על האויר תדירות ה-PITCH.
   c. האויר ממשיך לחלל התהודה בו מודגשים (מתעצמים) תדרים מסוימים, אלו נקראים **פורמנטים** (בגל קול סטנדרטי אנושי יש כ-4 פורמנטים= "תדרים מוגברים").
   d. בשלב האחרון מגיע האויר (בשלב זה כבר גל קול גולמי) לחלל הפה המהווה חלל תהודה נוסף (דינמי) ובו מתעצב הגל לצורתו הסופית.
6. **מודל הדיבור:**



a. Pulse Generator - יוצר את התדירות הבסיסית. <u>ההקבלה</u>: מיתרי הקול ה"מלבישים" את התדירות על האויר ביציאתו מהריאות.

b. Random Noise - במודל הזה, מתייחסים ל-Unvoiced בתור רעש לבן – רעש ללא תדר או הרמוניה, ואין שום דבר במיתרי הקול שמעצב את צורתו. <u>ההקבלה</u>: במערכת הדיבור שלנו גם כאשר ישנו עיצור עדיין ישנה יציאת אויר מהריאות, זו עוברת עיצוב "גס" יותר בחלל הדיבור (ובפרט הפה). האויר הזה מתייחס לרעש הלבן שמזניק את התהליך העיצורי.

c. Time Varying Filter – הוא מעצב את אות הדיבור, כלומר- מגביר תדרים מסוימים לעומת תדרים אחרים. <u>ההקבלה</u>: מורכב מתא התהודה שסמוך למיתרי הקול ומחלל הפה המהווה פילטר תלוי זמן היות שמושפע מתנועת הלשון, השפתיים וכו' בחלל הפה.

7. הנחות במודל:

a. תחת פרקי זמן קצרים, ניתן לתאר את הטרנספורמציה של גל הקול כקונבולוציה לינארית של הקול שמופק על ידי מיתרי הקול ביחד עם צורת חלל הפה (ה-GLOTTAL).

b. הנחה שנייה עליה אנו מתבססים היא כי השמיעה שלנו היא לוגריתמית ברמת התדרים. (**יש לנו הפרדה יחסית גבוהה בתדרים נמוכים, אבל לא בתדרים הגבוהים**).

8. שיערוך PITCH (לפי אוטוקורלציה):

a. האוטוקורלציה מחזירה את עוצמת הדמיון בינה לבין גרסה מוזזת של עצמה.

b. כאשר מתקבל מקסימום ראשון בגרף האוטוקורלציה ניתן לקבוע כי בקירוב טוב עברנו מחזור אחד של תדירות הדיבור- המיקום הזה אם כן מייצג את זמן המחזור.

c. נציב את זמן המחזור בנוסחה הסטנדרטית ונקבל את תדירות ה-PITCH.

d. ניתן להעריך את ה-PITCH בצורה גסה ללא אוטוקורלציה:

i. נספור את מספר השיאים בגרף זמני על פני פרק זמן קצר (מאיות השניה): נניח שספרנו 7 שיאים ב-5 מאיות שניה.

ii. נחלק את משך הזמן במספר השיאים ונקבל זמן המחזור הממוצע למקטע זה: $T = 0.05/7 = 0.007\ sec$

iii. נמיר לתדירות לפי הנוסחה הסטנדרטית: $f = 1/T = 1/0.007 = 142\ Hz$

e. **<u>עוד על אוטוקורלציה:</u>** - פונקציית האוטוקורלציה מוגדרת כ: $f_{auto-correlation} : K \to I$ כאשר:

◄ K מייצג הזזה נתונה מפונקציית המקור

◄ I מייצג את "מידת התאימות" בין ההזזה לפונקציית המקור (מעשית- את ערך הסכימה של מכפלות פונקציית האות בפונקציה המוזזת)

פונקציית האוטוקורלציה היא בעצם מכפלה של פונקציית הקול במישור הזמן על גבי קטע נתון בגרסה מוזזת שלה... ערכי הפונקציה הם למעשה סריקה משמאל לימין ע"ג אות הדיבור והכפלה שלו בעצמו ("הקצנה") כך שכאשר תהיה חפיפה חלקית של הפונקציה המוזזת בפונקציית המקור נקבל מקסימום מקומית, ועבור המקרה של חפיפה מלאה המתרחש מעשית לפני ההזזה הראשונה נקבל מקסימום גלובאלי.

האוטוקורלציה מחשבת את התאימות לאורך כל החלון שהגדרנו N עבור הסטה מסוימת של הפונקציה מעצמה (k=1,2,3...N-1)... כלומר עבור הסטה k=1 יחושב סכום עבור **כל** המכפלות לאורך החלון N בין הפונקציה המקורית לפונקציה המוזזת- הפלט יהיה מספר בודד שהוא "ערך התאימות" בין הפונקציות עבור הזזה זו... כמובן שבנקודת ההתחלה נקבל תאימות מקסימלית (כי הרי יש זהות) וערך גבוה מאוד (מקסימלי) של אוטוקורלציה.. כעת נמשיך להזיז את הפונקציה (כלומר נקדם k ונחשב עבור כל ערך שלו את התאימות) וכך נוכל לצייר את פונקציית התאימות או אוטוקורלציה... הפסגה הבאה בגרף שנקבל תהיה המקום בו ישנה תאימות שיא נוספת.

- <u>אז איך זה עוזר לחשב את תדר ה-Pitch?</u>

כאשר האוטוקורלציה מקבלת מקסימום (פרט למקסימום הראשון), היא תקבל אותו עבור k מסוים שמציין באיזה הסטה של הפונקציה הכפלנו בפונקציית המקור, היות ש-k מייצג דגימות (כמו n) אזי נדע בעצם באיזה דגימה (k) התקבלה תאימות גבוהה ואותו נתון יהיה למעשה זמן המחזור T ממנו נוכל למשוך את תדר ה-pitch ע"י חישוב סטנדרטי $f = \frac{1}{T}$. לא יכולנו פשוט לחשב זאת מתמונה רגילה של גל הקול כיוון שהיא אינה מייצגת מדד מדויק לתדירות (רק אולי אינטואיציה גסה), <u>האוטוקורלציה עוזרת לנו לחשב את המחזוריות שבגל הקול, שהיא מורכבת בהרבה ממחזוריות סינוסיאדלית רגיל.</u>

לסיכום עבור $\hat{k}$ שמהווה אינדקס של המקסימום השני של האוטוקורלציה נקבל $f_{pitch} = \frac{1}{\hat{k}}$

9. דגימה וכימוי- כללי:
   a. תהליך הדחיסה מכניס רעש להקלטה, וגורם לאובדן מידע. <u>רעש בפורמנטים גבוהים לא יהיה משמעותי כמו בנמוכים. נרצה למקד את את הרעש בפורמנטים הגבוהים.</u>
   b. **Sampling (דגימה)** – כמה דגימות יש לבצע בשנייה, כך שנוכל לשחזר את האות בצורה מלאה, בלי הפסדים. מרווח הדגימה יקרא T_s.
   c. **Quantization (כימוי)** – מיפוי של טווח ערכים גדול לטווח ערכים קטן יותר. קובעים סט ידוע מראש של עוצמות, כך שהאות שלנו יהיה מיוצג רק על ידי העוצמות האלה. מעבר מסט ערכים מסוים לסט ערכים קטן יותר, תמיד יגרום לאיבוד מידע. לכן נרצה למזער אותו.
10. **דגימה** - מאות אנלוגי לאות דיגיטלי. תהליך:
   a. מסנן מעביר נמוכים (LPF)
      i. נועד ע"מ למנוע ALIASING.
      ii. בד"כ מאפס כל מה שמעל תדר נייקוייסט.
   b. המרה מאנלוגי לדיגיטלי ע"י דגימה במרווחי זמן קבועים
      i. גורם לקוונטיזציה לא מכוונת
      ii. במרחב התדר נקבל שכפולים בשיקופים עד אינסוף לשני הכיוונים

אות קול גולמי > מסנן מעביר נמוכים > A2D > אות קול דיסקרטי + שכפולים במישור התדר (תופעת לוואי)

11. **שחזור** - מאות דיגיטלי לאות אנלוגי. תהליך (הפוך בדיוק):
   a. המרה מאות דיגיטלי לאות אנלוגי
   b. מסנן מעביר נמוכים (LPF) - מבטל את השכפולים במישור התדר

אות קול דיסקרטי > D2A > מסנן מעביר נמוכים (ע"מ לבטל שכפולים במישור התדר) > אות קול רציף

12. **משפט הדגימה של שנון** -

$$f_s \geq 2 * f_{max}$$

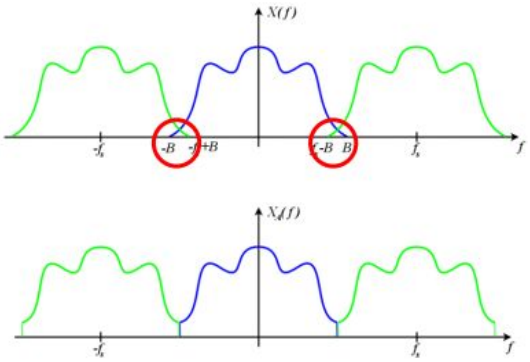
f_s = תדר הדגימה שנבחר

f_{max} = הוא התדר המקסימלי שקיים ברצועה אותה אנו דוגמים

a. המשפט אומר, שקצב הדגימה שלנו קובע את התדר המקסימלי אותו נוכל למדוד. ניתן לשחזר את האות המקורי רק אם דוגמים בקצב שהוא לפחות פי 2 מהתדר הכי גבוה אותו דגמנו, ואם לא דוגמים צפוף מספיק, אי אפשר לשחזר את האות המקורי.
b. למשל: כדי לדגום דיבור עד לתדר של 4KHz, נדגום בקצב שקצת יותר גבוה מ-8KHz.
c. המשפט מגדיר את הדגימה היעילה ביותר. כלומר, דגימה האוספת את כמות האינפורמציה המינימלית הדרושה להפקה מחודשת ומלאה של האות לאחר קידודו.

13. ALIASING

a. לאחר הדגימה, במרחב התדר יש אינסוף עותקים של האות, בתשקיף מראה אחד לשני.
b. במידה ודגמנו בקצב הנמוך מקצב נייקויסט נקבל כי ישנה חפיפה בין האות המקורי לעותקים הסמוכים אליו (משני הצדדים).
c. אם אכן מתקבלת חפיפה אז לאחר ה-LPF ב**תהליך השחזור** נקבל עיוות של האות כיוון שנחתוך יחד עם האות שלנו "זנב" מהאות ההעתק.
d. אם בכל זאת נרצה לדגום בתדר נמוך מהאידאלי- נוכל להעביר את האות ב LPF עם רף נמוך יותר שיגרום לזנבות של ההעתקים להיות חלקים יותר ולהידגם ללא רעשים. מסיבה זו מסנן זה מכונה גם ANTI ALIASING.



14. כימוי: מיפוי חד-ערכי מקבוצת ערכים גדולה לקבוצת ערכים קטנה.

a. קבוצת ערכים גדולה = ערכי אות אנלוגי
b. קבוצת ערכים קטנה = כמות תאי זיכרון שמוקצים לשמירת האות הדיגיטלי.
c. לרוב הקוונטיזציה תהיה הלכה למעשה פעולת "עיגול".

15. סוגים נפוצים:

a. **קוונטיזציה יוניפורמית (אחידה) -** בהינתן שגודל קבוצת המטרה הוא K, ושהקלטים הם בקטע ערכים ידוע מראש X, נחלק את הקטע X ל-K חלקים בגודל שווה, כאשר כל חלק ממופה לערך יחיד בקבוצת המטרה. <u>טובה כאשר לא ידוע כלום על הקלט או כאשר מודדים רעש לבן.</u>
b. **קוונטיזציה אופטימלית -** בהינתן שגודל קבוצת המטרה הוא K, נחלק את ערכי המטרה שלנו כך שקבוצות בעלי התפלגות ערכים גבוהה יותר יקבלו יותר ערכים מקבוצות אחרות בתוך קלט מסוים. את המיפוי ניתן לחשב בעזרת אלגוריתם לויד-מקס.
c. **קוונטיזציה אדפטיבית -** היא קוונטיזציה בה גודל הצעד משתנה בהתאם לשינוי באות הקלט (אם יש שינוי חד - נקודד אותו עם יותר ביטים. אם השינוי חלק יותר - נקודד אותו עם פחות ביטים).

16. דחיסת צורת הגל (Waveform Coding) -

a. שומרים את הצורה הכללית של האות (כמעט שלא כוללים מידע על מאפייני הדיבור).
b. הדחיסה נעשית בין דגימה לדגימה.
c. קל לממש אותה בשני הצדדים.

d. צורת דחיסה זו מאוד פופולרית.

17. PCM (Pulse Code Modulation)

a. ייצוג ישיר של גל האודיו.

b. ללא עיבוד או חיזוי.

c. בלי קונטסט לדגימות קודמות.

d. ניתן לבצע קוונטיזציה אופטימלית או לוגריתמית.

18. DPCM (Differential Pulse Code Modulation)

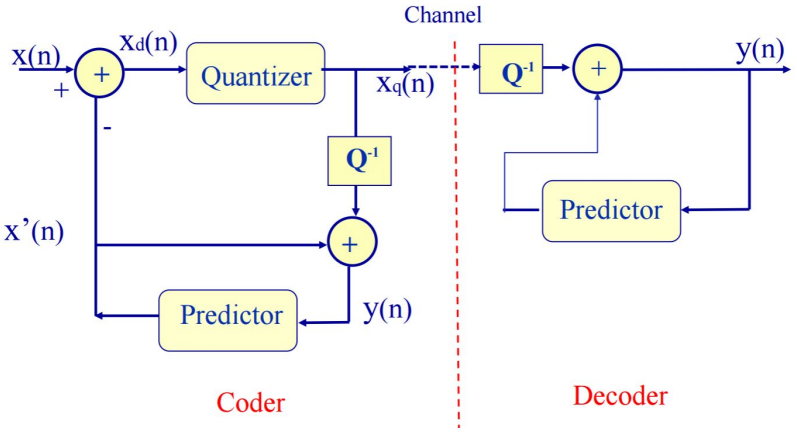
a. היות שהאודיו לא משתנה מאוד בזמנים קצרים נקבל כי דגימות סמוכות יהיו דומות.

b. בנוסף- קצב השינוי בין דגימות הוא כמעט לינארי (אם היה שינוי X למטה בפעם הקודמת סביר שכך יהיה גם בפעם הבאה).

c. המשמעות היא שניתן **לחזות** את הדגימה הבאה, עם טווח שגיאה קטן.

d. בשיטה זו נעביר (או נשמור) את השגיאה, שהיא בטווח מצומצם ביותר וכך נקבל דחיסה.

e. החזאי תמיד מקבל פידבק לגבי ההפרש בין האות המקורי לחיזוי שלו וכך יודע לעדכן את עצמו במידה והוא שוגה.



19. נתאר את התהליך באיור הזה:

<u>בצד שמאל - המשדר (באמצע התהליך):</u>

- נניח שאנו בדגימה ה-N. (באיור: $x(n)$) ניקח את ההפרש בין הדגימה ה-N לבין הערך שחזינו שיופיע בדגימה ה-N. (כלומר $x_d(n) = x(n) - x'(n)$)
- נכמת את הטעות (באיור: $x_q(n)$) ונבצע היפוך קוונטיזציה לטעות כדי לקבל את "הטעות" שמתקבלת בצד המקלט.
- נוסיף את הטעות הזאת לאות שחזינו, וכך ניצור את האות "השלם" שבצד המקלט. (באיור: $y(n)$). **שני החזאים מקבלים את אותו פידבק בסיום מעגל- כך הם נשארים מסונכרנים**.
- נבצע חיזוי של הדגימה ה-N+1. באיור, הפלט מסומן ב-$x'(n)$.
- נחזור על התהליך עבור הדגימה הבאה.

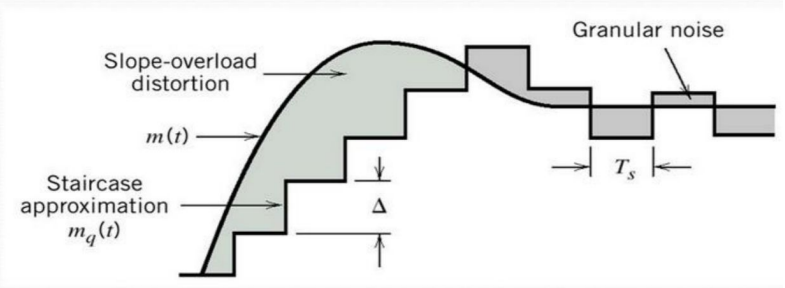
<u>בצד ימין - המקלט:</u>

- אנו מקבלים את הטעות.
- נבצע היפוך קוונטיזציה ונחבר אותה עם הערך שהחזאי נותן לנו עבור הדגימה ה-N.
- האות שקיבלנו הוא האות שנוציא החוצה. (באיור: $y(n)$).
- נעביר אותו לחזאי כדי לחזות את הדגימה הבאה ונחזור על התהליך.

שימו לב: **החזאים זהים לחלוטין בשני הצדדים** ומביאים את אותם ערכים.

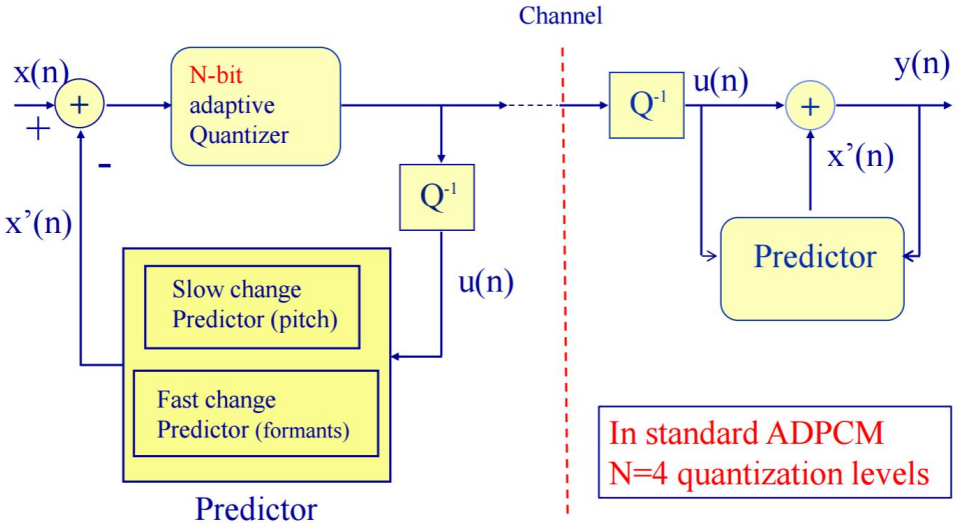
20. DM (Delta Modulation)

a. דחיסת DPCM כאשר לשגיאה מוקצה ביט אחד בלבד (0 עבור שגיאה שלילית ו1 עבור חיובית)
b. ישנו רעש שמצטרף בהכרח לקידוד- "רעש גרגרי" הנובע מאי היכולת לקודד שגיאת 0.
c. דחיסה זו מאופיינת ע"י אפקט מדרגות:



21. ADPCM (Adaptive Differential Pulse Code Modulation)

a. דומה לדחיסת DPCM, ההבדל: מספר הביטים שמוקצה לשגיאה.
b. שגיאות קטנות ניתן לקודד במספר מועט של ביטים, ושגיאות גדולות יקודדו במספר גדול של ביטים.
c. הקוונטייזר מפיק את מספר הביטים הדרוש (אם צריך ביט אחד - נוציא ביט אחד, ואם צריך ארבעה - נוציא ארבעה).
d. החזאי כאן, לעומת ב-DPCM, לוקח בחשבון את השינוי ב-PITCH ואת השינוי בפורמנטים כדי לעזור בהקצאת ביטים נכונה בהמשך.



22. VOCODERS(Voice Coders)

a. בשונה מדחיסת <u>צורת הגל</u> בה העברנו ייצוג דגום ומכומת של <u>גל הדיבור</u>, בשיטה זו נקודד את **הדיבור עצמו** ע"י **העברת פרמטרים** המייצגים את אות הדיבור כך שנוכל לסנתזם במקלט.
b. השיטה הבסיסית ביותר לבצע זאת נקראת - LPC(Linear Prediction Coding):
   i. נחלק את האות לפריימים ("מסגרות") - קבוצות קטנות של דגימות רציפות (ב LPC-10 מדובר ב-180 דגימות לפריים, ו-50-60 פריימים בשנייה).
   ii. בצד המשדר, אנו מחשבים את הפרמטרים של הפילטר לכל פריים ומעבירים אותם למקלט, שמייצר את אות הדיבור בצד המאזין. הפרמטרים שיחושבו עבור כל פריים:
      1. קולי\לא-קולי

2. עוצמה
3. PITCH
4. מקדמי החיזוי הלינאריים (מקדמי פורמנטים? רוב הביטים נצרכים פה)

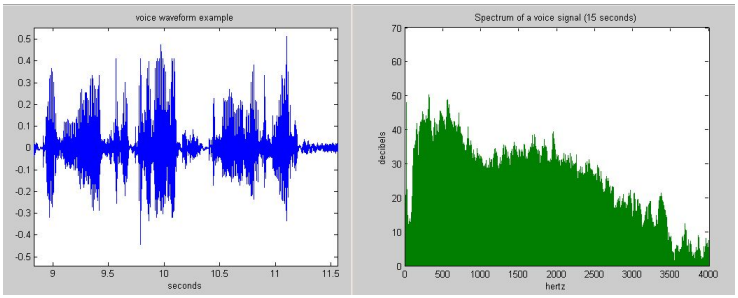
iii. בצד הקולט, אנו מתרגמים את הפרמטרים לאות קולי מסונתז באופן הבא:
1. יצירת אות עירור כלשהו (אחד ל-VOICED, ואחד ל-UNVOICED).
*אות העירור ל-VOICED הוא שמשתמש בפרמטר ה-PITCH- זו מחזוריותו.
2. מתג הבורר בין VOICED ל-UNVOICED.
3. מגבר עוצמה.
4. פילטר מקדמים לינאריים.

c. ניתן לכווץ עד פי 50 או 60.

d. <u>הגל המתקבל הוא אינו הגל המקורי</u> - אנו מסנתזים את הגל בצד המקלט ומייצרים גל אשר תכונותיו קרובות לתכונות שהעבירו לנו (למשל, מעטפת ספקטרלית דומה ופורמנטים שמיקומם דומה לאלו שבאות הדיבור המקורי).

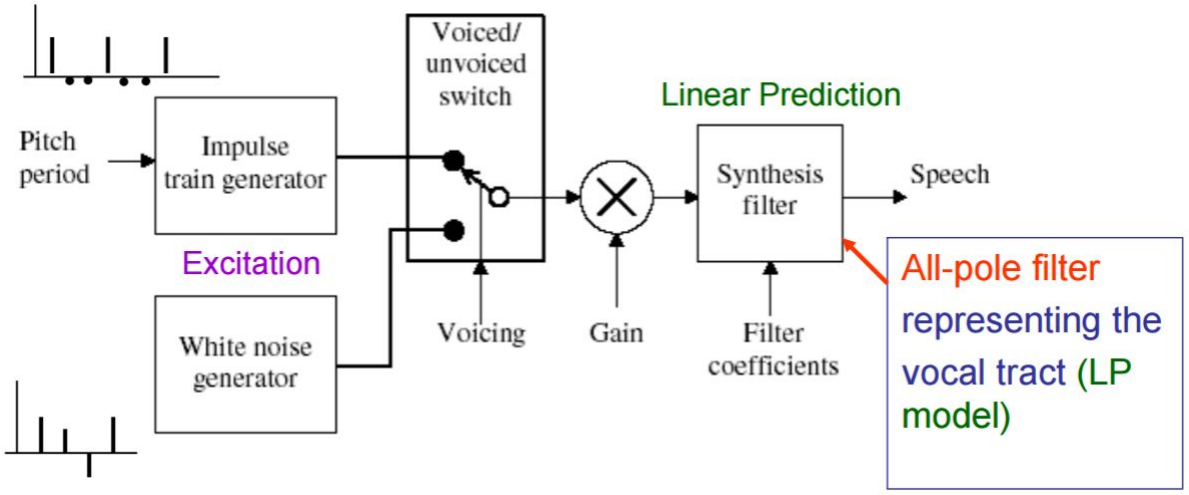
e. PSD (Power Spectral Density):

i. PSD = התפלגות עוצמת האות בתדרים השונים. במקרה שלנו בד"כ רוב האנרגיה נמצאת בתדרים הנמוכים:

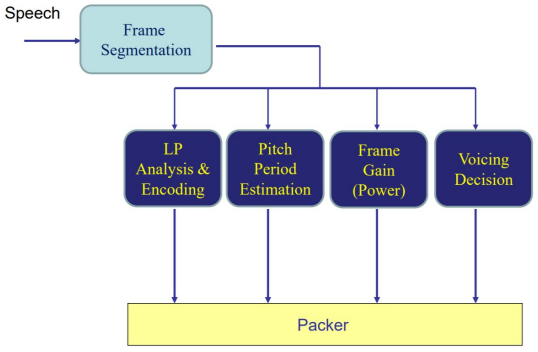


ii. בהקשר לפענוח אות ע"י VOCODER: ה- PSD של גל המקור נשמר בקירוב גם בגל המסונתז. הכוונה בכך היא שבקירוב, אזורים צפופים יותר מבחינה תדרית בגל המקור יהיו צפופים יותר גם בגל המסונתז.

f. השיטה הזאת זורקת את כל "המידע הגולמי" של הגל (צורת הגל) ושומרת את העוצמה התדרית שלו. <u>הגל המסונתז נשמע דומה למאזין האנושי משום שצורת הגל אינה חשובה לאוזן שלנו כמו המידע התדרי עצמו</u>. זאת גם הסיבה ש<u>ה-SNR הוא אינו מדד טוב לאיכות אות דיבור מסונתז</u>.



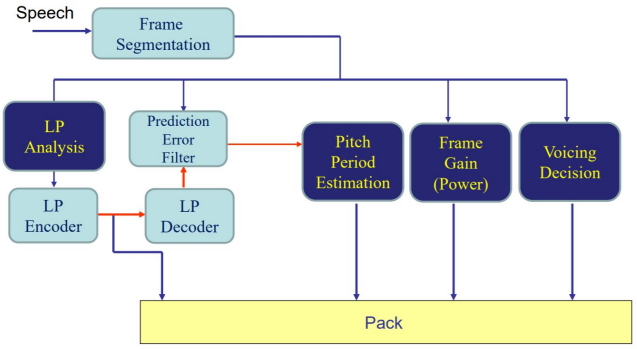
23. באיור למעלה מוצג המודל הבסיסי לסינטיסייזר שבצד המקלט, כאשר:
    a. Voicing - מסגרת דיבור VOICED או UNVOICED.
    b. Gain - רמת האנרגיה של מסגרת הדיבור
    c. Filter Coefficients - מקדמי הפילטר (מודל LP)
    d. Pitch Period - משך הזמן בין פולסי דיבור (VOICED).
24. **בגדול מה שקורה זה שאנחנו לוקחים איזה דף חלק (אות עירור) ומציירים עליו את אות הדיבור כאשר בכל תחנה אנחנו מוסיפים רכיב כלשהו לציור.**
25. פילטר המקדמים:
    a. עבור אות UNVOICED
        i. הפילטר מעצב את הרעש הלבן (שלו יש ספקטרום פחות או יותר מלבני) לצורה שקרובה למעטפת הספקטרלית של פריים ה-UNVOICED ששודר.
        ii. זה אפשרי מטעמי פשטות המעטפת הספקטרלית באות UNVOICED
        iii. **חשוב לזכור שזה נכון רק לגבי UNVOICED.**
        iv. אפשר להגיע לרמת דיוק טובה ב-UNVOICED עם 10 מאפיינים.
    b. עבור אות VOICED
        i. טכנית- על מנת להגיע לרמת דיוק טובה, נצטרך להעביר הרבה פרמטרים, אך עקב השימוש באות מחזורי גם בצד המקלט, מספיק שנעביר 10 פרמטרים כדי להגיע לרמה סבירה לאוזן.
26. מקודד ה-LPC:



    a. Frame Segmentation - מחלק את האות הנכנס לפריימים (אוגר את הדגימות).
    b. Voicing Decision - מחליט האם מדובר ב-Voiced או Unvoiced.

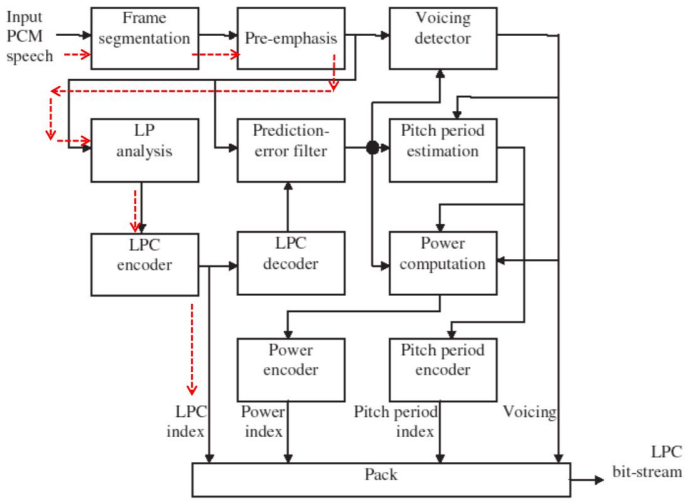
c. Frame Gain - מחשב את העוצמה של הפריים הנוכחי.
d. Pitch Period Estimation - משערך את ה-PITCH.
e. LP Analysis & Encoding - מחשב את הפרמטרים של הפילטר ומקודד אותם.

27. אם נרחיב את המערכת לשימוש ב**שגיאת החיזוי**:



a. מטרת המודל המתקדם - PITCH "נאמן למקור":
   i. נזכור כי אנו בונים את האות המסונתז ממרכיביו- כלומר נרצה מרכיבים כמה שיותר נאמנים למקור. בניגוד לאינטואיציה- הקול שאנו מפיקים לא מייצג PITCH נקי.
   ii. ה-PITCH היוצא מחלל הפה שלנו עובר עיוות ע"י חללי התהודה ממיתרי הקול ואילך.
   iii. אם נוכל לסנן את העיוות הזה החוצה נקבל PITCH "טהור יותר", שמייצג טוב יותר את רעידות מיתרי הקול שלנו.
b. תהליך זיכוך ה-PITCH:
   i. במקום להעביר את ה-PITCH ישירות, אנחנו נשתמש בחישוב של מקדמי האות והמעטפת הספקטרלית כדי לבנות "פילטר הופכי".
   ii. נפעיל את הפילטר הזה על האות הנכנס.
   iii. נקבל ייצוג נקי יותר **של האות** במונחי PITCH.
   iv. נשלח את האות לשיערוך ה-PITCH, שכעת יהיה מדויק יותר.
c. בסכמה עצמה:
   i. המקדמים מחושבים ב-LP Analysis, עוברים קידוד במודל ה-LP Encoder ונשלחים כרגיל אל ה-Packer (בצהוב).
   ii. במסלול מקביל אנחנו בונים את הפילטר שמייצג את הקול ב-LP Decoder.
   iii. ב-Prediction Error Filter מפעילים את ההופכי לפילטר על האות הנכנס כדי לנרמל את האות ולהגיע ל-PITCH המקורי.
   iv. האות המנורמל הולך למודול ה-Pitch Period Estimation. משם החישוב מתבצע כמו קודם (אבל מדויק יותר).

28. מקודד LPC-10



a. מבצעים חלוקה של אות המקור לפריימים במודול ה-**Frame Segmentation**.
b. נבצע שלב עיבוד נוסף, **Pre-Emphasis**, בו נגביר את התדרים הגבוהים.
   i. אלו משמעותיים ע"מ שהקול שלנו יישמע טבעי וייחודי.
   ii. רוב הדיבור מתרכז בתדרים הנמוכים, ויש מעט אנרגיה בתדרים הגבוהים- ללא ההגברה הזאת, בשלב הקוונטיזציה המידע של התדרים הגבוהים עלול להיאבד.
   iii. בשלב מתקדם יותר בתהליך נפעיל פילטר שיחזיר תדרים אלו למקור.
c. במודל ה-**Voicing Detector** נקבע האם מדובר באות שהוא Voiced או Unvoiced.
   i. אנו מעבירים את היציאה של מודול זה אל ה-Packer,
   ii. נעביר את היציאה בנוסף למודול ה-Pitch period estimation (ע"מ לא לאמוד PITCH במקטעי UNVOICED)
   iii. נעביר את היציאה בנוסף למודול ה-Power computation.
d. הדיבור המוגבר (אחרי Pre-Emphasis) נכנס למודל ה- **LP analysis**, לביצוע אנליזה ספקטרלית של מקדמי ה-LPC.
e. האות ממשיך ל-**LPC encoder** ולאחר מכן ל-**LPC dncoder**. וכמו שפירטנו קודם, הוא מנורמל ונשלח למודול ה-Pitch period estimation. המקדמים שחושבו מועברים כמובן אל ה-Packer דרך ה-Encoder.
f. מודול ה-**Pitch period estimation** מחשב את ה-PITCH ומעביר אותו אל ה-Pitch period encoder ומשם אל ה-Packer.
g. מודול ה-**Power computation** מקבל גם את אומדן ה-Pitch, את החלטת ה-Voiced/Unvoiced, ואת "האות המנורמל". ומבצע את החישוב:
   i. עבור UNVOICED:

$$P = \frac{1}{N}\sum_{n=0}^{N-1} e^2{}_{[n]}$$

1. $e(n)$ - עוצמת האות יחסית לדגימה n.
2. N - אורך הפריים - כמות הדגימות בסגמנט.

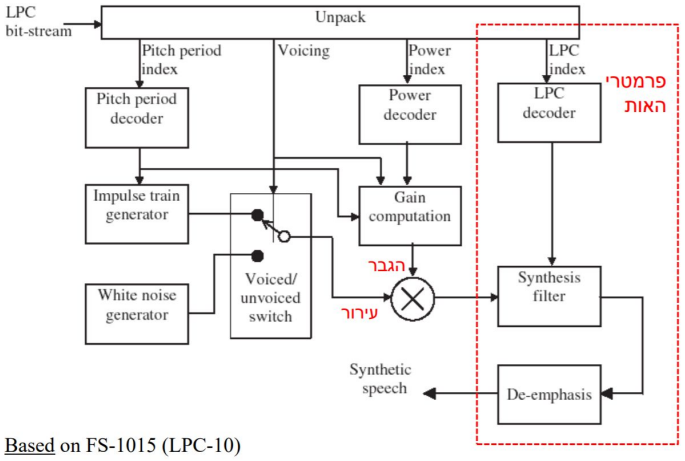
כלומר אנחנו מחשבים את סכום האנרגיה בסגמנט וממצעים.

   ii. עבור VOICED: נרצה להתחשב ב-PITCH.

$$P = \frac{1}{\lfloor N/T \rfloor T} \sum_{n=0}^{\lfloor N/T \rfloor T-1} e^2[n]$$

T - זמן מחזור ה-PITCH.

29. מפענח LPC-10



a. **Unpacking.**
b. ה-Pitch נשלח לפענוח ב- **Pitch period decoder** ועובר למודול ה-Impulse Train Generator, שמייצר פולסים מחזוריים בזמן המחזור של ה-PITCH.
c. **Voiced/Unvoced switch:** ביט ההחלטה בין ה-Voiced ל-Unvoiced הולך ל"מתג" שמחליף בין אותות העירור השונים (האות המחזורי או הרעש הלבן).
d. ה-Power Index הולך אל ה-**Power decoder**,וביחד עם ה-PITCH והחלטת ה-VOICING, ה-GAIN מחושב ב-**Gain computation**.
e. **Synthesis filter** - הפרמטרים של ה-LPC מועברים אל הפילטר, שמופעל על אות העירור.
f. **De-Emphasis** מבטל את הגברת התדרים הגבוהים שביצענו במשדר.

30. **Voicing Detection**:

a. כללי:
   i. בשלב זה אנחנו מזהים האם מדובר בפריים שהוא Voiced או Unvoiced.
   ii. אחת ההגבלות של מודל ה-LPC היא העובדה שצריך לקבוע בפירוש האם פריים הוא Voiced או Unvoiced, על אף שייתכנו מצבי מעבר באמצע סגמנט.
   iii. החלטה לא נכונה בשלב זה עלולה להביא לתוצאות הרסניות בסינתוז הדיבור.
b. זיהוי ע"י אנרגיה:
   i. השיטה הבסיסית ביותר לזיהוי VOICING לפי האנרגיה של הדגימות. לרוב, לדגימות שהן Voiced יש אנרגיה גבוהה, ולדגימות שהן Unvoiced יש אנרגיה נמוכה.
   ii. ניתן לחשוב בשתי דרכים:
      1. כמו ממקודם:

$$E[m] = \sum_{n=m-N+1}^{m} s^2[n]$$

      2. שימוש בטכניקה שנקראת MSF:

$$MSF_{[m]} = \sum_{n=m-N+1}^{m} \left|s_{[n]}\right|$$

3. שימוש בריבוע האנרגיה טוב יותר שכן הוא מדגיש את ההבדל בין אנרגיות גבוהות לנמוכות.

c. זיהוי ע"י ZERO CROSSING:
   i. נספור את מספר החציות של האפס.
   ii. מספר חציות גבוה מצביע על Unvoiced (כי Unvoiced נמצא בתדרים גבוהים יותר, ולכן הוא מתחלף יותר מהר - הוא יותר דומה לרעש).
   iii. מספר חציות נמוך מצביע על Voiced (כי Voiced נמצא בתדרים נמוכים יותר, ולכן הוא מתחלף לאט יותר - יש רכיב של זמן המחזור)

d. זיהוי ע"י PREDICTION GAIN:
   i. מזכיר את שכלול השיטה להערכת ה-PITCH:
   ii. נפעיל על הסיגנל חיזוי לינארי כדי לייצר את מקדמי המסנן. ניקח ונייצר בחזרה את האות במשדר, נשווה את האות המסונתז לאות המקורי, ונבדוק מה השגיאה ביניהם. נעשה כך לכל הדוגמאות בפריים:

$$PG_{[m]} = 10\log_{10}\left(\frac{\sum_{n=m-N+1}^{m} s^2_{[n]}}{\sum_{n=m-N+1}^{m} e^2_{[n]}}\right)$$

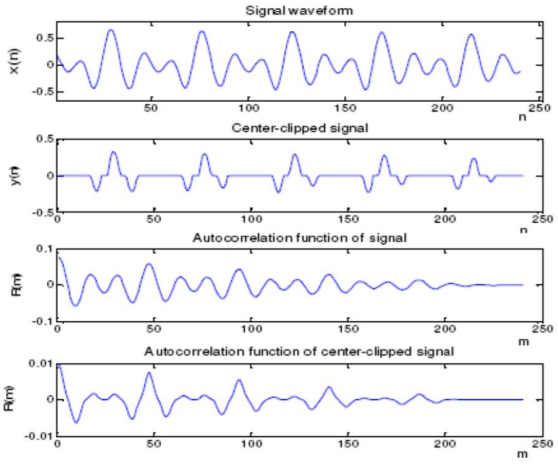
   iii. **שגיאת החיזוי של Voiced תהיה יותר קטנה** (באזור 3 דציבל), שכן חלק זה אינו דומה לרעש, ויש לו <u>התנהגות שניתן לחזות</u>**.**
   iv. שגיאת החיזוי של Unvoiced תהיה הרבה יותר גדולה, שכן Unvoiced מתנהג בצורה שדומה לרעש.

e. סיכום הדרכים לזהות VOICED/UNVOICED:

מדד\ערך	כשזה גבוה	כשזה נמוך
אנרגיה	Voiced	Unvoiced
Zero-Crossings	Unvoiced	Voiced
Prediction (error) Gain	Unvoiced	Voiced

31. **Pitch Detection**:

a. ננסה למצוא PITCH שמתנהג בתחומי ה-PITCH של גברים (50 עד 250 הרץ) או של נשים (120 עד 500 הרץ). נקבע את חלונות זיהוי ה-PITCH לפי התחומים הנ"ל.
b. זיהוי ע"י אוטוקורלציה (ראינו בתחילת הקורס).
c. זיהוי ע"י CENTER-CLIPPED AUTO-CORRELATION:
   i. אוטוקורלציה עם חלון מרכוז (Center-Clipped Autocorrelation) - איפוס של כל מה שנמצא (בערך מוחלט) מתחת לסף מסוים.



ii. כך נקבל את ה-PEAK-ים בגרף בצורה הרבה יותר ברורה.

d. זיהוי ע"י Magnitude Difference Function:

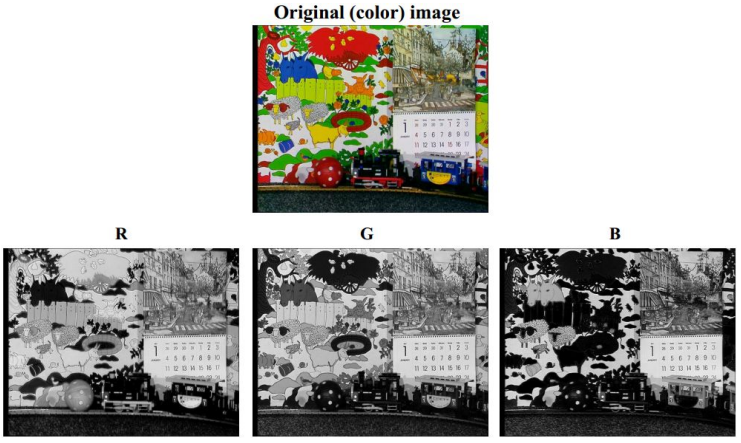
i. במערכות שבהן יש סיבוכיות נמוכה, נשתמש בדבר הזה. נחשב את הסדרה לפי הנוסחה הבאה:

$$MDF[l,m] = \sum_{n=m-N+1}^{m} \left| S[n] - S[n-l] \right|$$

ii. נחפש ב-MDF את נקודת המינימום, כי כשהם יהיו זהים, נקבל 0.

<u>דחיסת תמונה</u>

1. תמונה – האור המוחזר מאובייקט לאחר מעבר במערכת אופטית והיקלטות במערך חיישנים.
   a. תמונה היא פונקציה של שני משתנים: $f(x,y)$.
   b. כל נקודה (x,y) מגדירה פיקסל = אלמנט זעיר של התמונה.
   c. הערך של f מבטא את מידת הבהיקות (עוצמת האור) בנקודה.
   d. בתמונה דיגיטלית מידת הבהיקות מיוצגת ב-8 סיביות: כלומר 256 ערכים (0-255).
   e. בתמונות צבע כל פיקסל הוא מגדיר 3 פונקציות: $r(x,y), g(x,y), b(x,y)$.
   f. הצבעים הבסיסיים הנ"ל לכל פיקסל הם אדום, ירוק וכחול = RGB.
   g. בקורס זה לרוב נעסוק בתמונת רמת אפור.
2. רמות האפור הנקלטות:
   a. המידע שאנחנו קולטים מהסביבה הוא רציף, גם במרחב (אינסוף נקודות שונות), וגם בבהירות ובצבע (באינסוף ערכים שונים).
   b. כמות המשאבים שלנו היא סופית (זיכרון). לכן, כמו בקול נצטרך לדגום ולכמת את הערכים ע"מ לעבד אותם דיגיטלית.
3. **אפיון של תמונות:**
   a. צבעוניות
   b. בהיקות – Brightness ובהירות – Luminance
   c. ניגודיות – Contrast
   d. כושר הפרדה – Resolution
   e. תדרים מרחביים
   f. רעש – SNR, MSE
4. אפיון של תמונות - **צבעוניות:**
   a. תמונת צבע מכילה פי 3 מידע מתמונת רמות אפור.
   b. ייצוג צבע ישיר - RGB:
      i. בייצוג זה נגדיר כי כל אחד מהצבעים מתורגם לתמונת רמות אפור באופן הבא: ככל שהצבע הנתון (אדום, ירוק או כחול) יותר דומיננטי- כך הוא מקבל ערך בהיר יותר בתמונה רמות האפור.
      ii. נקבל את התמונות הבאות:



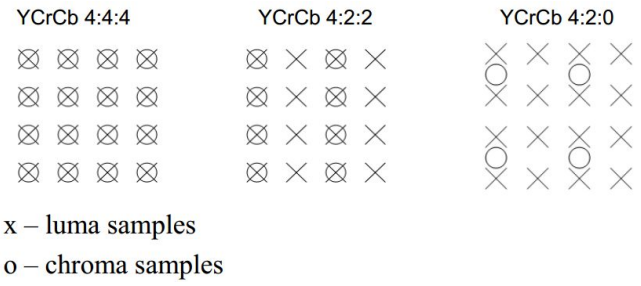
      iii. קל לראות כי יש דמיון בין התמונות.
      iv. **דמיון = יתירות = דחיסה.**
      v. מסקנה אחת שניתן להסיק היא כי העין האנושית <u>רגישה יותר לשינויים בבהירות מאשר לשינויים בצבע</u>.

c. Y-Cb-Cr:

i. נגדיר מרחב צבע חדש:

- Y = מידע על הבהירות בלבד.
- Cb-Cr = מידע על רמת הצבע.

ii. נוריד את רזולוציית הצבעים:



iii. בדגימה של 4:2:0 אנחנו מקטינים את תמונות ה-Chroma פי 2. זאת הקטנה מאבדת מידע (LOSSY), אך מכיוון שהעין האנושית לא רגישה לשינויים כאלו, <u>איבוד המידע לא מוריד מאיכות התמונה כפי שנתפסת ע"י אדם</u>.

5. אפיון של תמונות - **בהיקות – Brightness ובהירות – Luminance**
6. אפיון של תמונות - **ניגודיות – Contrast**
7. אפיון של תמונות - **כושר הפרדה – Resolution**
8. אפיון של תמונות - **תדרים מרחביים**
9. אפיון של תמונות - **רעש – SNR, MSE**

a. ההשוואה תהיה בין התמונה המקורית לתמונה מעובדת.

b. מדד MSE:

$$MSE_{(X,Y)} = \frac{1}{N}\sum_{i=1}^{N}(Xi - Yi)^2$$

i. כאשר X_i הם הפיקסלים בתמונה X, ו-N מספר הפיקסלים בתמונה.

ii. **ככל שהשגיאה קטנה יותר כך התמונות דומות יותר**.

iii. יתרון - בגלל הריבועיות בסכימה השיטה מדגישה שגיאות חזקות ומזניחה שגיאות חלשות.

iv. חיסרון - החישוב גלובלי ולא ניתן לקבל ממנו מידע לגבי חומרת השגיאה באזורים שונים בתמונה.

c. מדד PSNR-Peak Signal to Noise Ratio:

i. בפועל זה המדד בו נשתמש.

$$PSNR = 10\log_{10}\frac{\left(2^n - 1\right)^2}{MSE}$$

ii. כש-n הוא מספר הסיביות של הדגימות. כלומר, במונה נמצא ריבוע הערך הגבוה האפשרי ביותר של האות. במקרה של תמונות, מדובר בדרך כלל ב-n=8, כלומר שהערך המקסימלי הוא 255.

iii. ה-PSNR משקף יותר טוב את מה שהצופה הממוצע יגיד.

iv. בדרך כלל, <u>PSNR של 35 ויותר נחשב לטוב</u>.

v. באופן כללי, **נרצה PSNR גבוה ו-MSE נמוך**.

10. דחיסת תמונות - כללי:

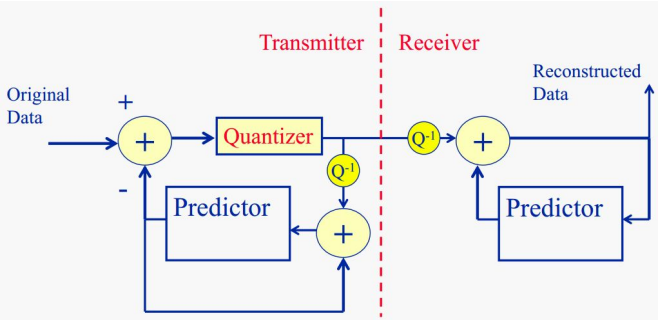
a. Lossless - דחיסה ללא איבוד מידע. בדחיסה כזאת נצליח לשחזר את המידע המקורי. לדוגמא: 7Z, ZIP, RAR.
b. Lossy - דחיסה עם איבוד מידע. כלומר לא נצליח לשחזר את המקור אחד לאחד, אלא נגיע לקירוב שלו.

11. דחיסת LOSSELESS:

a. נקרא גם "קידוד אנטרופיה".
b. סוגים:
   i. קידוד "האפמן"
   ii. Run Length
   iii. קידוד אריתמטי
   iv. Lempel-Ziv (דחיסת ZIP)
   v. Asymmetric Numeral Systems.
c. יתרונות:
   i. אין איבוד מידע בתמונה.
   ii. קל ליישום.
   iii. פשוט לשימוש.
d. חסרונות:
   i. יחס דחיסה נמוך (1:2 או פחות).
   ii. חלק מהדוחסים מוגנים בפטנט (אלגוריתמים).
e. <u>נחזור לנושא זה בהמשך.</u>

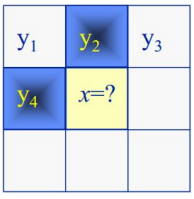
12. דחיסת LOSSY:

a. הנחת היסוד בדוחסים אלו היא כי <u>חלק מהמידע הוא בעל השפעה מועטה על התוצאה הסופית</u> ולכן נוכל לוותר עליו.
b. המדדים לדחיסת LOSSY נוגעים לתפיסת איכות סובייקטיבית ע"י שימוש בסולם MOS, שנותן ערך בין 1 ל-5 המייצג את איכות האות לצופה אנושי
c. נתייחס לתהליך הדחיסה כפי שלמדנו אותו עבור עיבוד קול - סכמת DPCM. כלומר, ישנם משדר ומקלט, והמשדר שולח למקלט את שגיאת החיזוי בערוץ חסר הפסדים:



d. לולא הקוונטייזר זו היתה סכמה חסרת הפסדים.
e. תהליך הכימות הוא מה שגורם לאיבוד המידע. היפוך התהליך לא משחזר אותו במדויק, אלא בקירוב.

13. שיטת החיזוי - LINEAR PREDICTION:



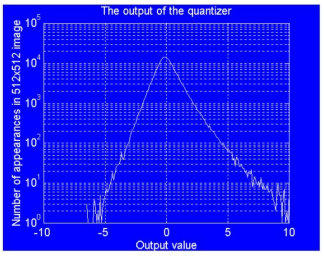
a. המקלט מקבל את שגיאת החיזוי, ומוסיף אותה לתוצאת החזאי שלו כאשר התהליך מתבצע שורה שורה, משמאל לימין ומלמעלה למטה.
b. נניח שאנחנו באמצע התהליך ומקבלים שגיאת חיזוי עבור פיקסל x.
c. החזאי חוזה את הערך ל-x לפי הנוסחה:

$$x = h_1 * y_1 + h_2 * y_2 + h_3 * y_3 + h_4 * y_4$$

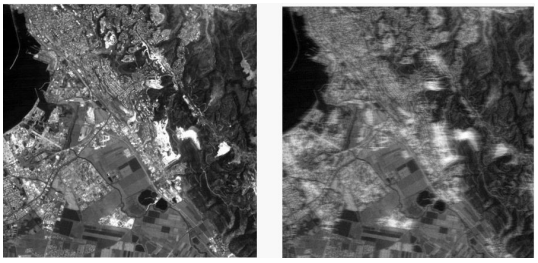
d. לאחר החיזוי מתבצעת סכימה של שגיאת החיזוי והערך שנחזה ומתקבל פיקסל התוצאה.
e. שגיאות החיזוי הערך של x:
   i. באזורים החלקים בתמונה, שגיאת החיזוי עבור x תהיה קטנה, כי באזורים האלה מתקיים כי כל פיקסל הוא אכן קומבינציה לינארית של הסביבה שלו.
   ii. לעומת זאת, במקומות שבהם יש שינוי (Edge), שגיאת החיזוי עבור x תגדל, כי יש שינוי בסדר הגודל של הערכים שלא ניתן לקבלו ע"י הפונקציה של x. במקרה כזה ההיזון החוזר לחזאי אמור לגרום לכך שהחיזויים הבאים יהיו מדויקים יותר.

14. Adaptive Quantizer
a. הקוונטייזר הוא תלוי אפליקציה.
b. יש אפשרות לקוונטייזר לויד-מקס, שהוא אופטימלי לפי MSE, ונותן תוצאת דחיסה טובה מספיק לעין האנושית.
c. ניתן לתכנן קוונטייזר שמשתנה לפי דגימה או סטטיסטיקה של התמונה.

15. Laplacian Quantizer
a. קוונטייזר לפלסיאני מותאם במיוחד לצילומי אוויר, כי **פילוג שגיאת החיזוי** של צילומי אוויר הוא לפלסיאני. התפלגות כזו היא נפוצה עבור תמונות טבעיות:



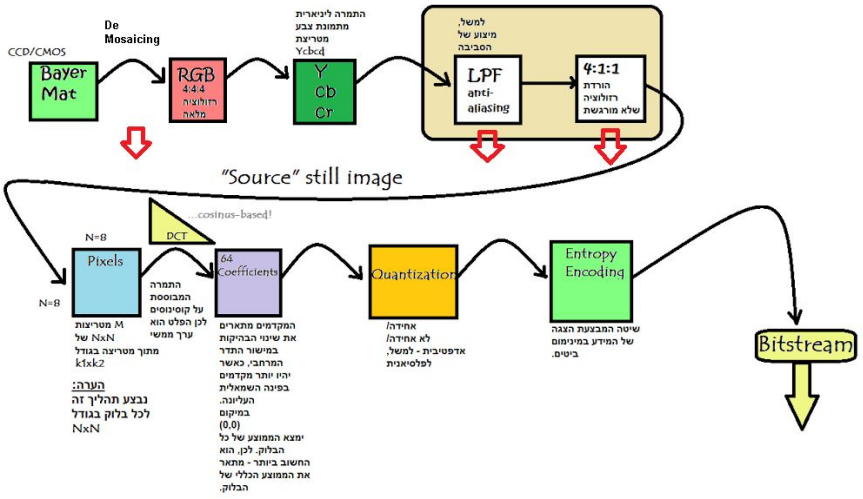
b. ניתן לראות בגרף כי שגיאת החיזוי 0 היא הנפוצה ביותר וככל שמגדילים את השגיאה (לשני הכיוונים) היא הופכת להיות פחות נפוצה.
c. עבור קוונטייזר יוניפורמי נקבל:



d. עבור קוונטייזר לפלסיאני נקבל:

16. JPEG:

a. JPEG ראשי תיבות של Joint Photographic Experts Group.
b. פרוטוקול דחיסת תמונות שמשלב דחיסת LOSSY עם דחיסת LOSSELESS.
c. יעיל במיוחד לדחיסת תמונות טבעיות: אנשים, נוף וכדומה. פחות יעיל עבור טקסטים.
d. דחיסה מבוססת DCT- Discrete Cosine Transform.
e. **תקציר תהליך הדחיסה:**



i. חלק ראשון בתהליך- SENSOR TO IMAGE (קליטה):

1. **יצירת Bayar Matrix**: נשתמש בשבב מסוג CMOS או CCD ל"לכידת" התמונה הגולמית. בתהליך שמבוסס על שבב בו מטריצת קבלים שנטענים במתח בהתאם לכמות האנרגיה שהם צוברים. טרם פגיעת קרני האור בקולטנים על גבי החיישן, הן עוברת בפילטר מסוג מטריצת BAYER שתפקידו לאפשר מעבר של גלי אור בתדירות התואמת לצבעים RGB לסירוגין כך שלכל קולטן מוקצה צבע (ביחס של 2:1 לטובת הירוק על פני האדום והכחול).
2. בשלב הבא האינפורמציה עוברת דרך אלגוריתם **De Mosaicing** שמחשב עבור **כל** תא במטריצה את הערך של **כל** אחד משלושת צבעי היסוד (בעזרת מיצוע של תאים קרובים).
3. לאחר מכן נעביר את המטריצה בעזרת התמרה ליניארית (כנראה כפל במטריצת מעבר) **מ-RGB אל YCbCr**. כאשר Y מייצג את מידת הבהירות של התמונה (גווני אפור) וCbI וCrI מייצגים את ההפרשים המספריים מהערכים של הכחול והאדום בהתאמה.
4. היות שהעין האנושית פחות רגישה לדיוק בצבע נבצע קוונטיזציה על הצבעים. אם כן לפני ביצוע הקוונטיזציה נצטרך להעביר את התמונה דרך מסנן LPF על מנת למנוע\להחליש את תופעת ה-ALIASING המוכרת לנו מעיבוד קול.
5. לאחר הורדת הרזולוציה קיבלנו אינפורמציה המכונה SOURCE DATA, ואותה נשלח לתהליך הדחיסה.

ii. חלק שני בתהליך- SOURCE IMAGE TO BITSTREAM (דחיסה):

1. נחלק את התמונה לבלוקים של 8X8 בשלב זה האינפורמציה בפנים היא של עוצמת הפיקסל וערכיה הם בטווח 0-255 כולל.

2. נבצע התמרת DCT- בשלב זה אין איבוד מידע וניתן מעשית לחזור למטריצה הקודמת ע"י התמרת IDCT.
3. נבצע קוונטיזציה לערכי הDCT.
4. נבצע קידוד אנטרופיה (LOSSELESS)
5. הפלט כעת הוא BITSTREAM של התמונה הדחוסה.

17. Transform Coding - DCT

a. העין רגישה יותר לתדרים נמוכים מאשר לתדרים גבוהים. לכן, נרצה למרכז את רוב ה"נזק" של הדחיסה בתדרים הגבוהים.
b. ה-DCT היא טרנספורמציה משמרת מידע (LOSSLESS) המאפשרת לנו להסתכל על תמונת הפיקסלים "במישור התדר" המרחבי.
c. לכן משתלם להעביר את התמונה למישור התדר, לבצע שם את הדחיסה כך שאובדן המידע יוכוון לתדרים הגבוהים, ואז להחזיר אותה למישור המקורי ולקבל תמונה דחוסה "אופטימלית".
d. האלגוריתם הכללי:
   i. נחלק את התמונה לבלוקים, כל אחד בגודל N*N פיקסלים, **זרים**. (אצלנו N=8, כלומר כל בלוק הוא 64 פיקסלים)
   ii. לכל בלוק נבצע התמרת DCT ונקבל בלוקים של 8*8 עם מקדמי DCT.
   iii. כל בלוק 8*8 עובר התמרה ממרחב דרגות האפור למרחב התדר, כך שכל איבר בבלוק הוא מקדם של תדר.
   iv. את המקדמים נעביר קוונטיזציה וקידוד.
e. התמרת ה-DCT היא **ממשית** (רק COS), **דקורלטיבית** (גרירת שינויים פרופורציונלית, טובה מ-FFT), **הפיכה** (קיימת IDCT), **ספרבילית** (התמרה דו מימדית = 2 התמרות חד מימדיות) ו**קיים לה מימוש מהיר** (n*logn).
f. נוסחת ההתמרה (חד מימדית):

$$F_{(u)} = \left(\frac{2}{N}\right)^{1/2} \sum_{i=0}^{N-1} A_{(i)} \cos\left[\frac{\pi u}{2N}(2i+1)\right] f_{(i)}$$

$$A_{(i)} = \begin{cases} 1/\sqrt{2} & \text{for i=0} \\ 1 & \text{Otherwise} \end{cases}$$

g. נוסחת ההתמרה (דו מימדית):

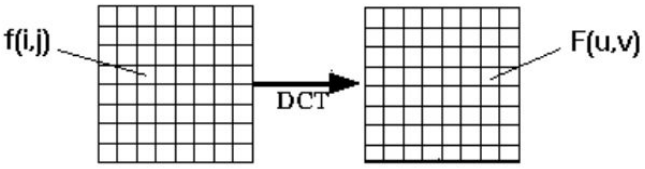
$$F(u,v) = \frac{C(u)C(v)}{4} \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos\left(\frac{(2x+1)u\pi}{16}\right) \cos\left(\frac{(2y+1)v\pi}{16}\right)$$

DCT coefficients

$$C(n) = \begin{cases} \frac{1}{\sqrt{2}} & n = 0 \\ 1 & n \neq 0 \end{cases}$$

Samples

h. דוגמה להתמרה של בלוק פיקסלים לבלוק מקדמים:



139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

Data (pixels)

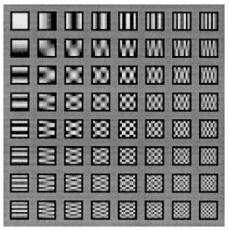
235.6	-1.0	-12.1	-5.2	2.1	-1.7	-2.7	1.3
-22.6	-17.5	-6.2	-3.2	-2.9	-0.1	0.4	-1.2
-10.9	-9.3	-1.6	1.5	0.2	-0.9	-0.6	-0.1
-7.1	-1.9	0.2	1.5	0.9	-0.1	0.0	0.3
-0.6	-0.8	1.5	1.6	-0.1	-0.7	0.6	1.3
1.8	-0.2	1.6	-0.3	-0.8	1.5	1.0	-1.0
-1.3	-0.4	-0.3	-1.5	-0.5	1.7	1.1	-0.8
-2.6	1.6	-3.8	-1.8	1.9	1.2	-0.6	-0.4

Spectral coefficients

i. הערך הכי גדול במטריצת המקדמים: פינה שמאלית עליונה.
הערך הזה הוא הממוצע של דרגות האפור בבלוק (מוכפל בקבוע). נקרא גם מקדם ה-DC.

ii. כל מקדם משקף את העוצמה של התדר המתאים לו באיברי הבסיס. בתמונות, המקדם הכי משמעותי תמיד יהיה ה-DC.

i. משמעות מטריצת (ה-DCT) ההתמרה (2 מימדים):



i. איברי מטריצת ה-DCT מייצגים את התדרים (המרחביים).
1. ככל שנלך יותר ימינה, כך התא מייצג תדר גבוהה יותר בציר X, וככל שנלך יותר למטה, כך התא מייצג תדר גבוהה יותר בציר Y.
2. הערכים בתאים עצמם מעידים על עוצמת התדר המדובר (בציר X או Y)
3. בהקבלה לתמונת הפיקסלים- ככל שנעלה ב-X, האיבר במטריצת ה-DCT מייצג **שינויים מהירים יותר בציר הX**, בתמונת המקור.

ii. בהתמרה ההופכית (Inverse DCT), משחזרים את הבלוק המקורי על ידי סכימת המקדמים, כאשר כל מקדם מוכפל בפקטור המתאים לו הנובע ממיקומו במטריצת המקדמים.

iii. בהצבה בנוסחת ה-DCT נקבל מקדמים עם אינסוף ספרות אחרי הנקודה. במקרה הזה נחתוך את המקדמים אחרי מס' ספרות מסוים. איבוד המידע יהיה זניח כך שבשלב זה האינפורמציה עדיין מוגדרת LOSSELESS.

j. טבלאות קוונטיזציה להתמרת DCT:

i. נבצע את הקוונטיזציה ב-DCT על ידי חלוקת כל בלוק במטריצת הקבועים:

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

ii. נכפול את מטריצת הקבועים בקבוע α, שנקבע על ידי ה-Quality Factor.

iii. האדום מתייחס לתדרים הנמוכים יותר, הכחול לגבוהים.

iv. לאחר החילוק עושים עיגול כלפי מטה, כך שאם המקדם קטן מספיק הוא יאופס.

v. בפועל כמעט כל מקדמי התדרים הגבוהים יתאפסו.

$$\begin{bmatrix} -415 & -33 & -58 & 35 & 58 & -51 & -15 & -12 \\ 5 & -34 & 49 & 18 & 27 & 1 & -5 & 3 \\ -46 & 14 & 80 & -35 & -50 & 19 & 7 & -18 \\ -53 & 21 & 34 & -20 & 2 & 34 & 36 & 12 \\ 9 & -2 & 9 & -5 & -32 & -15 & 45 & 37 \\ -8 & 15 & -16 & 7 & -8 & 11 & 4 & 7 \\ 19 & -28 & -2 & -26 & -2 & 7 & -44 & -21 \\ 18 & 25 & -12 & -44 & 35 & 48 & -37 & -3 \end{bmatrix} ./ \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

$$= \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -3 & 4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

e.g. for the DC coefficient: $\text{round}\left(\frac{-415}{16}\right) = \text{round}(-25.9375) = -26$

vi. טבלת החילוק של עבור מטריצות הצבע:

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

vii. זו טבלה פשוטה יותר היות שעבור CbCr כבר הפעלנו Low Pass Filter, ולכן מראש יש פחות תדרים גבוהים.

viii. הערה: <u>טבלאות הקוונטיזציה ופקטור האיכות לא מוגדרים בתקן, אלא נקבעים על ידי האפליקציה.</u>

k. פקטור האיכות לטבלאות DCT:

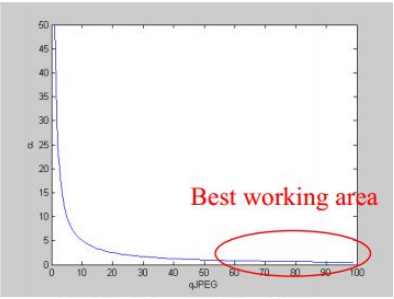
i. בנוסף לקוונטיזציה, ניתן לכפול את מטריצת ה-DCT בקבוע α, אשר מגביר את הדחיסה ככל שנרצה:

$$\alpha = \begin{cases} \dfrac{50}{q\_JPEG} & 1 \le q\_JPEG \le 50 \\ 2 - \dfrac{2q\_JPEG}{100} & 51 \le q\_JPEG \le 99 \end{cases}$$

ii. הקטנה של q_JPEG ⇐ הגדלה של alpha ⇐ חלוקת טבלאות הDCT בטבלאות קוונטיזציה עם ערכים יותר גבוהים ⇐ קוונטיזציה חזקה יותר ⇐ הקטנה באיכות התמונה.

iii. בדרך כלל נבחר פקטור איכות בין 60 ל-80.

iv. גרף המתאר את הקשר בין alpha לפקטור האיכות:



Alpha Vs. quality factor

18. קידוד אנטרופיה:

a. אנטרופיה - מידת אי הסדר במערכת. <u>כאשר יש רעש, האנטרופיה היא מאוד גדולה.</u>

b. קידוד אנטרופיה - נרצה לקודד את **כמות המידע** שיש בסימבול (פיקסל) מסוים.
ככל שהסתברות של פיקסל היא גדולה יותר, ככה הוא נותן לנו פחות מידע.

c. נגדיר את כמות המידע שיש בסימבול:

$$F = log_2 1/Pi$$

d. כאשר Pi = שכיחות ההופעה של הסימבול.

e. משפט שאנון - בהינתן מידע וההתפלגות שלו, את האנטרופיה נחשב על ידי הנוסחה:

$$H = -\sum_{i=1}^{M} Pi * log_2(Pi)$$

f. מעשית האנטרופיה שמחושבת כאן היא מס' הביטים הממוצע לקידוד המינימלי של מידע פר סימבול.

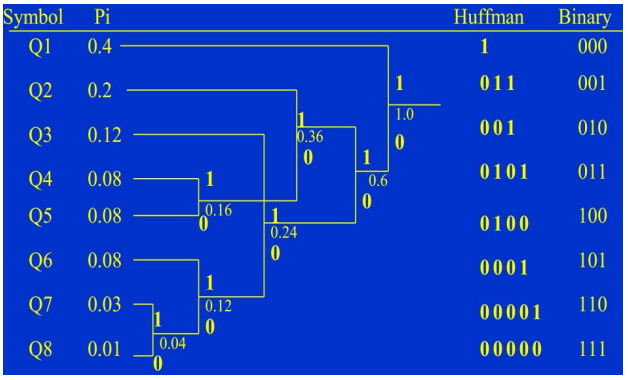
19. קידוד האפמן:

a. קוד האפמן מקצה לכל סימבול קידוד בינארי לפי הכלל הבא:
שככל שסימבול פחות נפוץ, הוא מקבל מילת קוד ארוכה יותר.

b. הנחה במודל: <u>מילת קוד לא יכולה להיות רישא של מילת קוד אחרת.</u>

c. האלגוריתם:

i. (נוח, אך לא חובה, להתחיל במיון הסימבולים לפי שכיחויות)
ii. נבחר את שני הסימבולים עם הסתברויות ההופעה הקטנות ביותר
iii. לראשון נקצה את הסימן 1, ולשני את הסימן 0.
iv. נחבר את ההסתברויות שלהם, ונתייחס לסכום בתור סימבול חדש.
v. נחזור לשלב ii. בסוף סך ההסתברויות שווה ל-1 (בדיקת שפיות).



d. דוגמא הזאת התחלנו מהסימבולים Q7,Q8, כי להם יש את ההסתברויות הכי נמוכות.
חיברנו בניהם, וקיבלנו את הסכום 0.04, וכעת במקום להתייחס ל-Q7,Q8, נתייחס לסכום הזה ומעליו נמשיך לבנות את העץ.

e. באיטרציה הבאה, שני ההסתברויות המינימליות הן 0.04 שיצרנו, ו-0.08 של Q6, אז נחבר בניהם וניתן 1 ל-Q6, ו-0 לענף מתחתיו. נסכום ונקבל 0.12, ונמשיך הלאה.

f. סיימנו לבנות העץ. הולכים 'אחורה' בעץ (מהשורש לעלים), והמסלול מהשורש לסימבול הוא קוד ההאפמן שלו.

g. קידוד רגיל מ-000 ל-111, היה דורש 3 סיביות. בקוד האפמן קיבלנו:

$$R = \sum Pi \bullet Li = 2.52$$

h. כלומר, אורך קוד ממוצע של 2.52 ביטים. יחס הדחיסה הוא:

2.52/3 = 0.84

i. כלומר, הפעלה של קוד האפמן ישירות על התמונה לא תיתן חיסכון גדול.

j. בקרוב נראה על איזו אינפורמציה כן ישתלם לנו להפעיל את קידוד האפמן.

k. חסרונות בקוד האפמן:

i. רגיש לשגיאות בביטים (Bitrate Error): טעות בקידוד האפמן יכולה להרוס את תהליך ה-decoding.

ii. תלוי בסטטיסטיקה - ייתכן מצב שסימבול נדיר הופך פתאום לנפוץ, ואז מתבזבזים ביטים כי מקצים לסימבולים נדירים הרבה ביטים.

iii. לא אחיד.

20. DC Codes

a. כאשר נשלח מטריצת DCT של תמונה, נרצה לשלוח אותה בלוק בלוק. נרצה לשלוח את ההפרשים בין מקדמי ה-DC בין בלוקים סמוכים, במקום לשלוח את המקדמים עצמם. לכן את ההפרש הזה נרצה לקודד.

b. נעשה זאת ע"פ הטבלה הבאה:

DC Coef Difference	Size	Typical Huffman codes for Size	Additional Bits (in binary)
0	0	00	-
-1,1	1	010	0,1
-3,-2,2,3	2	011	00,01,10,11
-7,...,-4,4,...,7	3	100	000,...,011,100,...111
-15,...-8,8,...,15	4	101	0000,...,0111,1000,...,1111
|	|	|	|
-1023,...-512,512,...,1023	10	1111 1110	00 0000 0000,...,11 1111 1111
-2047,...-1024,1024,...2047	11	1 1111 1110	000 0000 0000,...,111 1111 1111

c. דוגמאות:

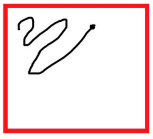
i. עבור הפרש 0 נגדיר קידוד 00.

ii. עבור ההפרשים 1,1- נגדיר את הקידוד 010. נשרשר 0 נוסף אם מדובר בהפרש אחד, אחרת נשרשר 1 אם מדובר במינוס אחד. זה נדרש כי אנחנו רוצים לדעת מהו בדיוק ההפרש והסימן שלו.

iii. עבור הפרשים: 3-,2-, 2, 3 נגדיר את הקידוד 011, ונוסיף לו שני ביטים נוספים על פי ערכו (אחד מתוך הארבעה).

iv. נשים לב בטבלה שככל שההפרש יותר גדול, ככה ניתן לו יותר סיביות. ההסתברות להפרש מאוד גדול בין שני מקדמי DC היא מאוד נמוכה, לכן הקודים עבורו יהיו ארוכים. בנוסף ניתן להסיק את הסתברות ההופעה מאורך הקוד.

21. AC Codes - Run-Size Technique



a. לאחר קוונטיזציה נקבל כי האפסים מתרכזים בתדרים הגבוהים.

b. בסריקת ב"זיג-זאג", לאחר מספר קצר של מקדמים, נגיע למקום שממנו הכל מתאפס, אז נשלח למקלט קוד שנקרא END OF BLOCK.
c. לא נרצה לקודד את מקדמי ה-AC עצמם, כי יש יותר מדי אפשרויות למטריצת DCT.
d. נקודד את מקדמי ה-AC לפי הפורמט הבא:

(Run, Level) Value

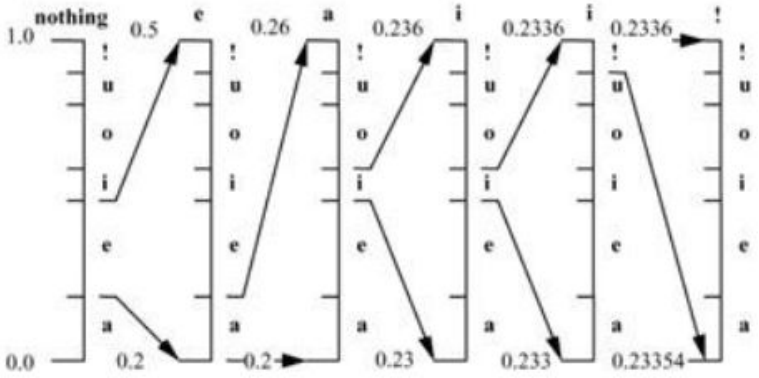
e. כאשר:
   i. Run - כמות אפסים יש לפני בסריקה.
   ii. Level - כמות סיביות שתופס הערך.
   iii. Value - ערך ה-AC עצמו.
f. **את זוגות ה-(Run, Level) נקודד בקידוד האפמן**.
g. הקידוד יעיל יותר, שכן יש קורלציה גבוהה בין מספר האפסים לערכים שלפניו: מקדמים נמוכים מגיעים לרוב לאחר מספר רב של אפסים.
h. אם נצטרך לשלוח הרבה אפסים (מעל 15), נשלח Zero Run Length (נקרא לו ZRL):
   i. כל ZRL שווה 16 אפסים.
   ii. אם נרצה לשלוח 20 אפסים נשלח ZLE וב-Run העוקב נוסיף 4.
i. דוגמה לקידוד המקדמים (המקדמים בעמודה השמאלית ביותר):

(Run,Size)	Code Byte (hex)	Code Word (binary)	(Run,Size)	Code Byte (hex)	Code Word (binary)
(0,1)	01	00	(0,6)	06	1111000
(0,2)	02	01	(1,3)	13	1111001
(0,3)	03	100	(5,1)	51	1111010
(EOB)	00	1010	(6,1)	61	1111011
(0,4)	04	1011	(0,7)	07	11111000
(1,1)	11	1100	(2,2)	22	11111001
(0,5)	05	11010	(7,1)	71	11111010
(1,2)	12	11011	(1,4)	14	111110110
(2,1)	21	11100			
(3,1)	31	111010	(ZRL)	F0	11111111001
(4,1)	41	111011			

j. (0,1) = אפס אפסים לפני המקדם. הקוד תופס סיבית אחת.
k. ככל שהקודים קצרים יותר, סימן שהם נפוצים יותר.

22. קידוד אריתמטי

a. קידוד אריתמטי = קודם "נארוז" סימבולים בווקטור, ולא נייצג סימבול בקוד, אלא ישירות באמצעות ההסתברות שלו.
b. שיטה זו דוחסת יותר טוב מהפמן, אבל עם סיבוכיות גבוה.
c. קוד האפמן אומר משהו ממוצע - נגיע בממוצע למשהו יותר קצר מהקוד הבינרי הרגיל, אבל עדיין צריך לתת לכל סימבול מס' שלם של סיביות.

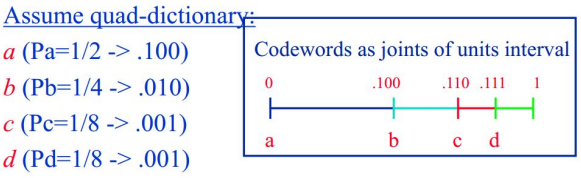


d. נפלג את הסימבולים על ציר המספרים מ-0 עד 1 לפי ההסתברות.
e. אלגוריתם הקידוד:

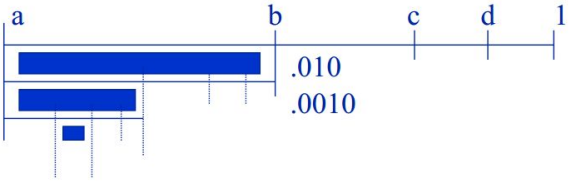
i. לכל סימבול, נלך לטווח שמתאים לו מתוך הטווח הכללי.
ii. כעת נקבע את הטווח שקיבלנו להיות <u>הטווח הכולל</u> ונחלק עליו את הסימבולים באותו יחס.
iii. נבצע שוב עבור הסימבול הבא.
iv. כאשר הגענו לסוף המילה, נבחר ערך מייצג בתוך הטווח. (נניח האמצע- **זהו הקידוד שלנו עבור המילה**.

f. אלגוריתם הפענוח:
i. בהינתן ההסתברות X שמייצגת דחיסה של סימבול.
ii. נתחיל מהטווח 0-1.
iii. נלך לטווח שבו X נופל, ונרשום את הסימבול אותו הוא מייצג.
iv. נקבע את הטווח החדש לפי טווח הסימבול הראשון.
v. נחזור לשלב iii.
vi. נשים לב כי בכל שלב דיי יהיה להסתכל על מספר מסוים של ספרות ב-X (במקרה הפשוט ביותר ספרה אחת) ולא יהיה צורך לבצע השוואות עד לרמת הדיוק המירבית אחרי הנקודה העשרונית.

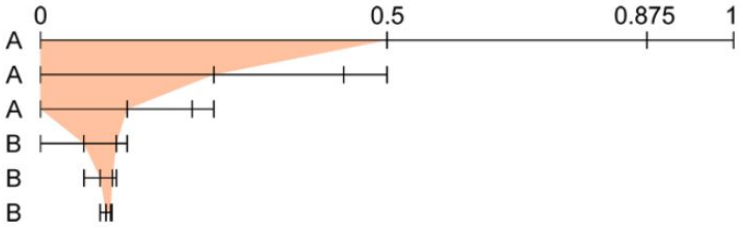
g. דוגמה:



i. בדוגמה הזו, המילון שלנו הוא a,b,c,d. נרצה לקודד את הרצף aab.

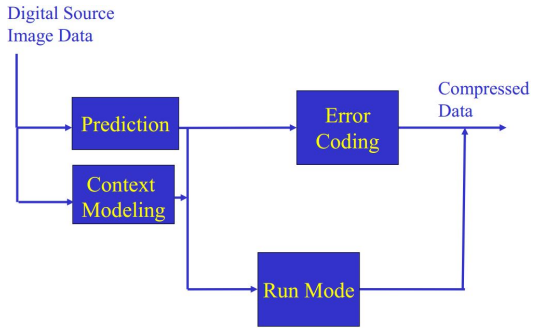


ii. נתחיל מה-a הראשון.
iii. ה-a תופס חצי מהטווח בגלל הסתברות שלו. נסמן את הטווח שלו בכחול. נחיל על הקטע הכחול את החלוקה היחסית. זהו הטווח החדש.
iv. נעבור ל-a השני. גם הוא תופס חצי מהטווח.
נעבור ל-b השלישי. הוא תופס את הרבע שנמצא בין חצי לשלושה רבעים.
v. בסוף הריצה, ניקח מספר רנדומלי בתחום הכי מצומצם שהגענו אליו- למשל 0.16.
vi. נשדר מספר זה למקלט, המכיר את טבלת ההסתברויות
vii. המקלט יבצע בניה מחדש של המילה לפי המספר שקיבל.
viii. המחשה:



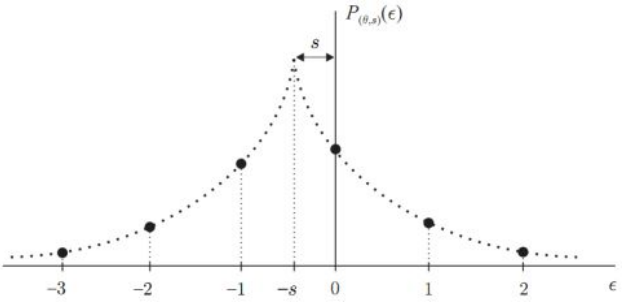
23. LOCO Coder

a. סכימת המקודד:



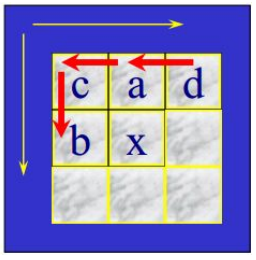
b. מפרט המודל:
   i. Prediction - חזאי רגיל המעביר את שגיאת חיזוי.
   ii. Context Modeling - קובע לכל פיקסל את הקונטקסט שלו בהתאם לסביבתו.
   iii. Error Coding - מקודד שגיאת החיזוי לפי הקונטקסט.

c. יחידת ה-Context- כללי:
   i. החדשנות במודל נובעת מיחידה זו.
   ii. הנחת המוצא היא שאנחנו רוצים להעביר שגיאת חיזוי מינימלית.
   iii. ע"מ להעביר שגיאה מינימלית ניעזר **נקודת ייחוס דינמית לשגיאה, נקודת הייחוס הזו היא** הקונטקסט.
   iv. מבחינה מעשית הקונטקסט הוא גרף פילוג גיאומטרי:



1. הגרף מגדיר עבור פיקסל נתון את הערך ממנו נמדדת השגיאה- בסיס הקונטקסט.
2. בסיס הקונטקסט (במקרה הזה s-) הוא השגיאה הנפוצה ביותר.
3. ככל שסוטים מבסיס הקונטקסט הסבירות לשגיאה פוחתת.
4. עבור כל פיקסל נקבע קונטקסט- את שגיאת החיזוי נעביר יחסית לקונטקסט ולא באופן ישיר. בצורה זו נקטין את המספר שאנו מעבירים. בדוגמה לעיל שגיאה בשיעור s- תעבור בתור שגיאת 0.
5. במודל הבסיסי של ה-LOCO הוגדרו 728 קונטקסטים שונים.

d. חישוב הקונטקסט:



D1= d-a
D2= a-c
D3= c-b

   i. את הקונטקסט בכל תא X נחשב בעזרת שלושת הגרדיינטים המוגדרים לעיל D1,D2,D3.

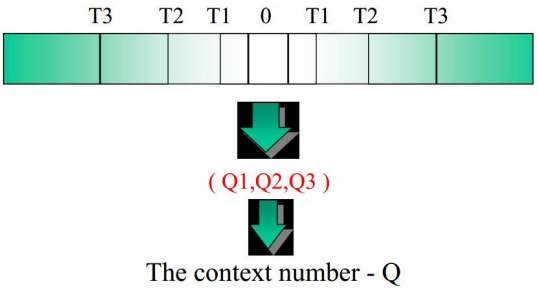
ii. אבל אם כל פיקסל יכול לקבל ערך בתחום 0-255, אזי תחום השגיאה הוא: $[-255, 255]$.

iii. אנו מגדירים 3 גרדיינטים ולכן נקבל כי יש 512 בשלישית אפשרויות לקונטקסטים.

iv. זה מספר עצום ולא נוכל לקבוע לכל פיקסל קונטקסט אם נצטרך לסרוק מרחב אפשרויות כה גדול.

v. ניעזר בקוונטייזר ע"מ לפתור בעיה זו- חשוב לשים לב כי הקוונטייזר **לא פועל על הפיקסלים** בשום שלב בתהליך ולכן הדחיסה נשארת LOSSLESS. הקוונטייזר מופעל רק על הקונטסטים שהם לא יותר מכלי עזר סטטיסטי לשיפור יעילות הקידוד.

e. הקוונטייזר של הקונטקסטים:



i. נקבע 4 תחומי שגיאה סימטריים ל-0, נקבל כי כולל תחום ה-0 יש לנו <u>9 תחומים</u>.

ii. סה"כ במקום 512 בשלישית, יש לנו 9 בשלישית אפשרויות להפרש - 729.

iii. בפועל זה 728 כי מתעלמים מקונטקסט האפס.

iv. ניתן היה לכמת אל כל מספר אי זוגי של תחומים.

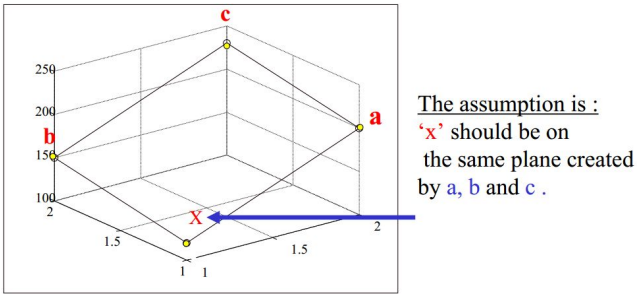
f. חיזוי LOCO:

i. החיזוי עבור פיקסל X: (מתייחס לתמונת הפיקסלים הנמצאת בעמוד הקודם)

$$Px = \begin{cases} \min(a,b) & c \geq \max(a,b) \\ \max(a,b) & c \leq \min(a,b) \\ a+b-c & otherwise \end{cases}$$

1. שני המקרים הראשונים הם עבור מצב ש-X נמצא באזור בו יש Edge.
   a. במקרה הראשון c נמצא בפסגה יחסית ל-X לכן סביר ש-X ממשיך את הירידה ולכן ניתן לו ערך מינימלי.
   b. המקרה השני אנלוגי.
2. אם X ממוקם באזור "חלק" בתמונה נבצע "ממוצע" a+b-c (ע"מ להימנע מחלוקה ב3 שהיא בזבזנית מבחינת סיבוכיות).

ii. המחשה:



g. מקודד Golomb-Rice

i. מקודד אנטרופיה אופטימלי עבור פילוג גיאומטרי.

ii. הפיקסל מגיע למקודד לאחר שקיבל קונטקסט ושגיאת חיזוי.

iii. יתרונות:

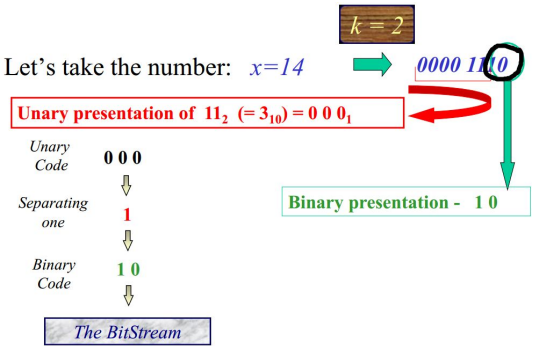
1. לא דרושה טבלה (כמו בהאפמן).
2. בעלי פרמטר אחד : k, אשר יועבר על ידי הקונטקסט.

iv. אופן הקידוד:

1. המקודד מקבל מספר k ומחלק את שגיאת החיזוי לפי המוראה לעיל.
2. ייצוג אונרי = המספר 3 מיוצג '0001' , כל מילה מסתיימת ב-1.

v. נמחיש בדוגמה:

1. רוצים לקודד את המספר x=14, כאשר k=2.
2. הייצוג הבינארי של 14 הוא 00001110
3. נפריד את הייצוג הבינארי של 14: 01 + 000011
4. קיבלנו 3 בצד שמאל, נקודד באונרי : 000.
5. צד ימין נשאר בקידוד בינארי.

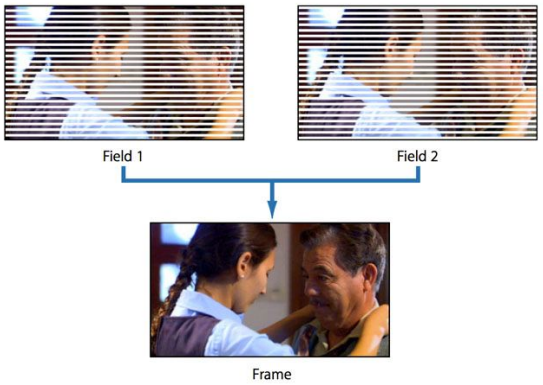


h. LOSSY MODE

i.

דחיסת וידאו

1. מבוא:
   a. שני שדות:
      i. בעבר, הווידאו לא היה בנוי מפריים אחד מלא, אלא משני "שדות" ששודרו בהתחלפות (alternating): השדה הזוגי - שייצג את התוכן בשורות 0,2,4,6 וכו', והשדה האי-זוגי - שייצג את התוכן בשורות 1,3,5,7 וכו'.
      ii. התמונה השלמה נוצרה על ידי סריקה ועדכון של אחד מהשדות: פעם עדכנו את השורות הזוגיות (שדה זוגי), ופעם את השורות האי-זוגיות (שדה אי-זוגי). הסריקה והעדכון הם כל כך מהירים שאיננו מסוגלים לראות אותם.



   b. תקנים:
      i. שני תקנים עיקריים: PAL האירופאי וNTSC האמריקאי.
      ii. ל-NTSC היה קצב "רפרוש" מהיר יותר, אך תמונה קטנה יותר, ול-PAL היה קצב "רפרוש" איטי יותר, אך תמונה גדולה יותר ואיכותית יותר.
   c. צבע:
      i. בתחילת דרכה, הטלויזיה שודרה בשחור לבן. ע"מ להוסיף שידורים בצבע מבלי לפגוע במקלטי השחור לבן הקיימים נמצא תחום ריק בפס השידור בו שודרו תכני הצבע ב- YCbCr מדולל ל4:1:1 .
      ii. ארטיפקטים בצבע:

Original, **single field**. The moving text has some motion blur applied to it.

Original still image.

4:2:0 **progressive** sampling (**single field**) applied to moving interlaced material. the chroma leads and trails the moving text.

4:2:0 **progressive** sampling applied to a Still image. Both fields are shown.

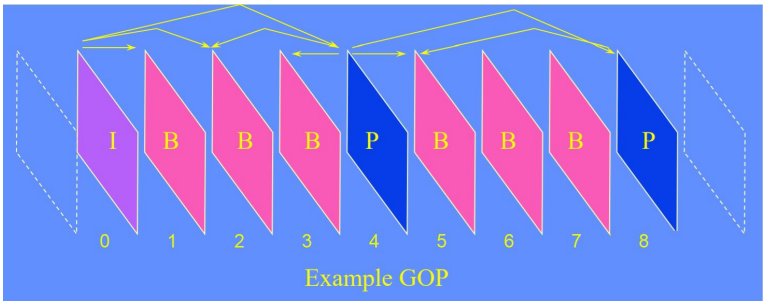
4:2:0 **interlaced** sampling (**single field**) applied to moving interlaced material.

4:2:0 **interlaced** sampling applied to a still image. Both fields are shown.

2. יתירות זמנית:

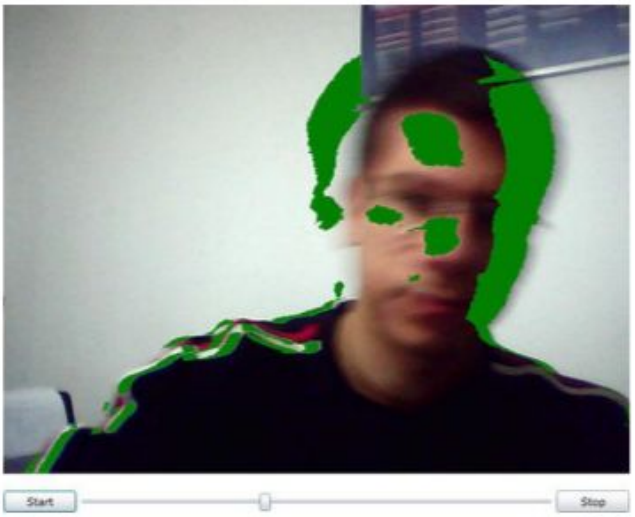
a. בדחיסת תמונה קיימת יתירות מרחבית (spatial redundancy) - פיקסלים סמוכים דומים אחד לשני.
b. בדחיסת ווידאו קיימת בנוסף **יתירות זמנית** (temporal redundancy) - פיקסלים **בפריימים סמוכים** דומים אחד לשני.
c. עדיין ישנה יתירות סטטיסטית, אבל לא נקודד ב-HUFFMAN אלא ב-ARITHMETIC CODING, כי היעילות הרבה יותר חשובה מהמהירות. כמובן שגם נבצע קוונטיזציה לצבעים דומים.

3. **מבוא לדחיסה זמנית:**
a. ניתן לחלק את הפריימים בוידאו דחוס לשני סוגים:
i. **Intra Frames** - תמונה עצמאית, שאינה תלויה בפריימים סמוכים (כמו שאנו שומרים תמונה ב-JPG). מכונה גם I-Frame.
ii. **Inter Frames** - פריים "הפרש" בין תמונות עוקבות. (הפרש עם חיזוי התנועה מהפריימים הסמוכים).
b. תמונות ה-Inter מוגדרות ע"פ חיזוי:
i. **P-Frame (קיצור של Predicted Frame)** - פריים Inter שמחזיק את ההפרש בין הפריים הקודם לנוכחי. עוברים יותר קוונטיזציה מ-I-Frames.
ii. **B-Frame (קיצור של Bidirectional predicted Frame)** - פריים Inter שמחזיק את ההפרש בין הפריים הקודם לנוכחי וגם בין הפריים הבא לנוכחי. עובר הכי הרבה קוונטיזציה.
iii. הערה חשובה: החיזוי של B-FRAMES משתמש ב-I-FRAMES ו-ב-P-FRAMES בלבד.
iv. נבחין בין חיזוי קדימה לחיזוי אחורה:
1. חיזוי אחורה - מוצאים איפה פיקסלים היו בעבר.
2. חיזוי קדימה - מוצאים לאן פיקסלים ילכו בעתיד.
v. בין כל שתי Intra Frames נעביר קבוצה של תמונות הפרש שיהוו חיזוי של הפריים המלא על סמך המלא הקודם, כמו שעשינו ב-DPCM.
vi. קבוצת הפריימים שמתחילה ב-I-Frame ונגמרת ב-I-Frame שני נקראת GOP (קיצור של Group of pictures).את גודל ה-GOP אנו קובעים על ידי שני מספרים, M ו-N, כש-M מציין את המרחק בין שני פריימים "חשובים" (I-Frame או P-frame) וביניהם B-Frames, ו-N מציין את המרחק בין שני I-frames. בתמונה הנ"ל, M הוא 4 ו-N לא ידוע (נניח 12).



4. **קידוד תמונות הפרש (תוך שימוש בהערכת תנועה):**
a. **קידוד הפרש רגיל** (תוך התעלמות מתנועה)
i. לא יועיל בדחיסה.
ii. עלול להוסיף מידע שיכול לגרום לדחיסה פחות יעילה.
iii. למשל:
1. בתמונה למטה ניתן לראות תנועה ימינה של ראשו של איש.

2. ניתן לראות כי בניסיון לקודד את תמונת ההפרש ישירות קיבלנו כי ישנה רצועה באמצע בה אין הפרש בין התמונות ואכן ניתן לראות את פני האיש.
3. אך בצידי הראש ישנם הפרשים הנובעים מצד אחד מ"עליית הראש על הרקע" ומצד שני מ"חשיפת הרקע ע"י הראש".



b. **נעדיף לקזז (לפצות) את התנועה** ע"מ להביאה בחשבון בקידוד:
   i. נרצה לזהות אילו דברים זזו ממקום למקום מפריים לפריים.
   ii. נרצה לזהות אילו דברים נוספו בין פריימים סמוכים.
   iii. כלומר, נרצה לזהות מהי "התנועה" שהייתה בין שני פריימים סמוכים.
   iv. נגדיר את התנועה הזו ב-**וקטור תנועה**.
   v. לשם כך, נשתמש בטכניקה שנקראת התאמת בלוקים.

c. התאמת בלוקים:
   i. נניח ונרצה להתאים בין הפריים הנוכחי F0 לפריים הבא F1.
   ii. נחלק את **F0** לאזורים קטנים - ריבועים בגדלים מוגדרים מראש (מאקרובלוקים) בגודל של 16x16, כלומר 4 בלוקים של 8x8.
   iii. עבוד כל בלוק ב-F0 ננסה למצוא איפה הבלוק הכי דומה לו ב-F1.
   iv. נגדיר **אזור חיפוש** ב-F1 שמסביב למיקום הגיאומטרי של הבלוק ב-F0.
   v. בשיטת FULL SEARCH למשל, נעבור כעת על חלון החיפוש ב-F1 פיקסל פיקסל ונבצע השוואה מלאה ע"י הפרשים לבלוק מ-F0, נסכום את ההפרשים ונקבל מדד.
   vi. כאשר המדד מינימלי ישנה התאמה בין הבלוקים.

d. קריטריונים להשוואה בין בלוקים:
   i. ישנם מספר קריטריונים שלא נעמיק בהם (MSE מינימלי, קרוס-קורלציה מקסימלית וכו'...)
   ii. SAD (Sum of Absolute Differences)
      1. קריטריון מאוד פופולרי, אבל בעייתי בעקבות בעיה של "נפילה לתוך מינימום מקומי" ולא לתוך מינימום גלובלי. מביאה לתוצאות פחות טובות מ-MSE, כמובן (אך טובה ממנה מבחינת סיבוכיות).
      2. ההגדרה המתמטית של SAD:

$$SAD = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|curr\_block(i,j) - ref\_region(i,j)\right|$$

   iii. פעולת התאמת הבלוקים מתבצעת במקודד בלבד ("לא מעניין את המפענח")

e. **שיטות חיפוש (עבור התאמת בלוקים):**

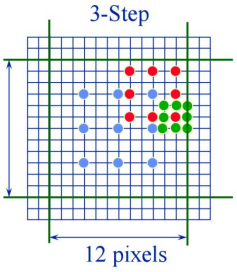
i. כעת צריכים לבחור באסטרטגיה שבה נעבור על חלון החיפוש ונבחר בבלוק המתאים ביותר.

ii. ראינו: ב**חיפוש מלא (Full Search)**, אנו עוברים על כל פיקסל בחלון החיפוש, ובודקים האם הבלוק שהפיקסל הזה הוא פינתו השמאלית העליונה "הכי דומה" לבלוק שאנו מחפשים.

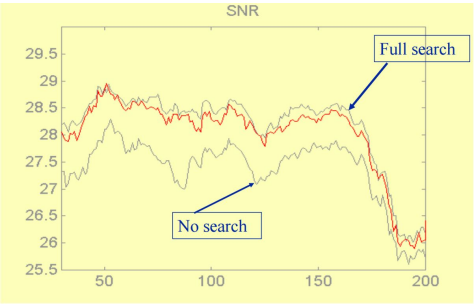
1. הבעיה בחיפוש מלא היא **עלותו היקרה** מבחינת סיבוכיות.
2. ולכן ננסה למצוא חלופות שדרכן נגיע לתוצאה תת-אופטימלית טובה מספיק.

iii. **N-Step Search**

1. נבצע כאשר N=3, ונניח כי חלון החיפוש הוא ריבוע בעל צלע 12 פיקסלים.
2. נתחיל את האלגוריתם במיקום הגיאומטרי של <u>נקודת התחלת הבלוק</u> מ-F0, נקודה זו תהיה אמצע אמצע חלון החיפוש ב-F1.
3. מסביבה נבחן את 8 הנקודות המקיפות אותה במרחק 3.
4. נבחר מה-8 את הנקודה שבה ה-SAD הכי נמוך (<u>נחשב את ה-SAD עבור נקודה כאילו נקודה זו היא נקודת התחלת הבלוק לו מחושב ה-SAD</u>).
5. מסביב לנקודה זו נבחן שוב את 8 נקודות המקיפות אותה הפעם במרחק 2
6. נבחר מה-8 את הנקודה שבה ה-SAD הכי נמוך.
7. כעת ברשותנו 9 נקודות (בהכרח לפחות 3 משיקות לשפת החלון), ביניהן הפרש של פיקסל בודד.
8. נבחן מה-9 את הנקודה שבה ה-SAD הכי נמוך.
9. נקודה זו היא נקודת התחלת הבלוק הכי מתאים.



10. השיטה יעילה יותר מחיפוש רגיל, כי אנחנו לא עוברים על כל הבלוק.
11. לשיטה זו ביצועים טובים יחסית, שכן הסיכוי ש<u>ניפול בה על המינימום הגלובלי הוא גדול יותר</u>.



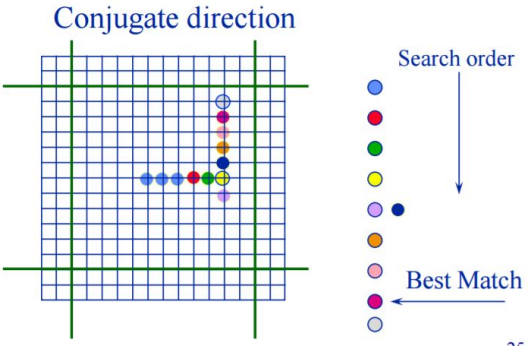
iv. **N-Step Search (עבור N כללי)**

1. דומה לחיפוש עבור N=3, כעת הרדיוס ההתחלתי הוא N.
2. גודל הבלוק שנחפש אינו משתנה, אלא רק הרדיוס שבו מתבצעת בחירת הנקודות.

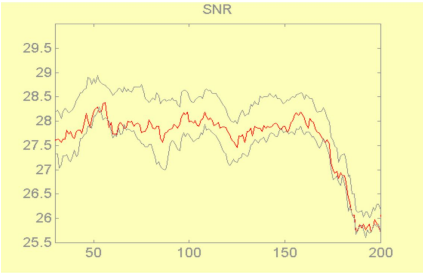
3. שאלה אפשרית במבחן היא "מהו ה-N המינימלי שדרוש על מנת לכסות חלון בגודל של MxM?".
   a. התנאי לכך הוא: $\sum_{i=1}^{N} i \leq \frac{M}{2}$.
   b. זה נובע מכך שבמקרה הגרוע (כך שה-SAD הנמוך ביותר נמצא על שפת חלון החיפוש), נתקדם כל פעם את מרחק הרדיוס (המתחיל ב-N ויורד ב-1 כל פעם), על פני מרחק של מחצית מחלון החיפוש.
4. שאלה נוספת שאפשר לשאול היא "מה מספר הפיקסלים המקסימלי שאנו מכסים עבור חיפוש ב-N שלבים?".
   a. הפתרון הוא כזה: $9 + 8(N-1)$.
   b. כלומר- עבור 3 שלבים נבדוק SAD עבור 25 פיקסלים.
   c. מתחילים את החיפוש ב-9 פיקסלים, ובכל פעם ש"נקטין את הרדיוס" מסתכלים רק על שמונת הפיקסלים.

v. Conjecuate Search

1. כללי: בשיטה זו אנחנו נחפש על ציר X, וכשנגיע לנקודה האופטימלית, נחפש את הנקודה האופטימלית בציר Y.
2. בודקים את ה-SAD באמצע, ובשני הפיקסלים מימין ומשמאל לו.
3. בוחרים בפיקסל שבו ה-SAD הכי נמוך. אם באמצע - מפסיקים. ואם בצדדים - ממשיכים לחפש בצד שבו ה-SAD הכי נמוך.
4. כשנגיע לנקודה שבה ה-SAD נהיה גבוה יותר, נעבור לחיפוש בציר האנכי מהנקודה בה עצרנו.
5. נבצע חיפוש כזה גם בציר האנכי.

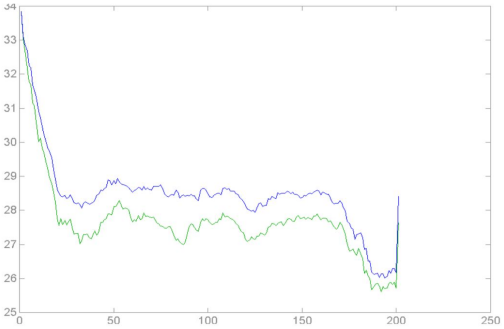


6. חלופה זו נחשבת פחות טובה מחיפוש רגיל, כי אנחנו עוברים על פחות פיקסלים ונוקטים בגישה שלא בהכרח תמיד נכונה (שה-SAD תמיד יורד למינימום בציר X ובציר Y).



vi. Zero MV Search

1. בשיטה זו, אנו פשוט מעבירים ווקטור תנועה 0, בלי קשר למה שקרה קודם לכן.



f. הערה- **Block Matching != Motion Tracking.**

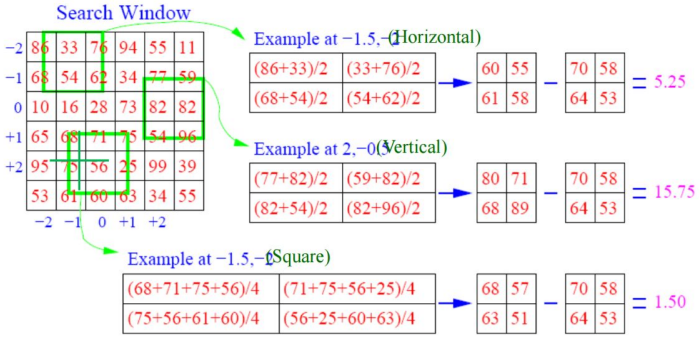
5. Half Pel Refinement

i. בעת תהליך התאמת הבלוקים, בגלל שהפיקסלים נדגמים בצורה בדידה, ייתכן כי הפיקסל שמתחיל את הבלוק שאנו מחפשים (בעל ה-SAD הנמוך ביותר) "נבלע" בין פיקסלים סמוכים. כך שלעולם לא נוכל "לקלוע" לבלוק שמתאים בצורה מיטבית לבלוק המקור, ולכן נצטרך להשקיע יותר ביטים בדחיסה של בלוק הפרש פחות מוצלח.

ii. ב-MPEG-1 ניתן לפתור את הבעיה ע"י הגדלת הרזולוציה פי 2.

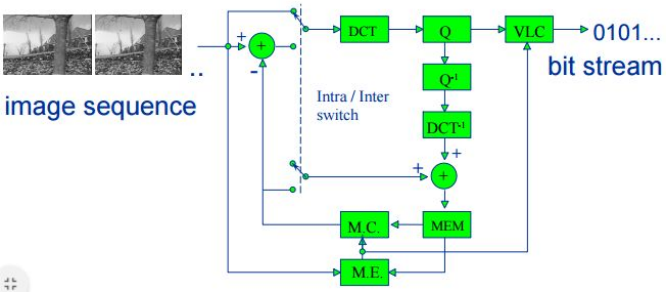
iii. בתהליך ההגדלה מעשית נרווח את הפיקסלים הקיימים ע"י הכנסת פיקסלים ממוצעים בתווך- בצורה זו נחשוף "פיקסלי ביניים" שעשויים להוביל ל-SAD מיטבי. וכפועל יוצא לקידוד בלוק שגיאה יעיל יותר (וחיסכון בביטים).

iv. למשל:



1. בדוגמה 1 - אינטרפולציה (הרחבה) אופקית.
2. בדוגמה 2 - אינטרפולציה אנכית.
3. בדוגמה 3 - אינטרפולציה בשני הכיוונים.
4. האינטרפולציה נעשית על ידי מיצוע פשוט של כל הפיקסלים "שביניהם" יוצב הפיקסל החדש.
5. כמובן שהשגיאה המינימלית ניתנת עבור דוגמה 3, משמע היא תקבל את ציון ה-SAD הנמוך ביותר ותקודד ביעילות מירבית.

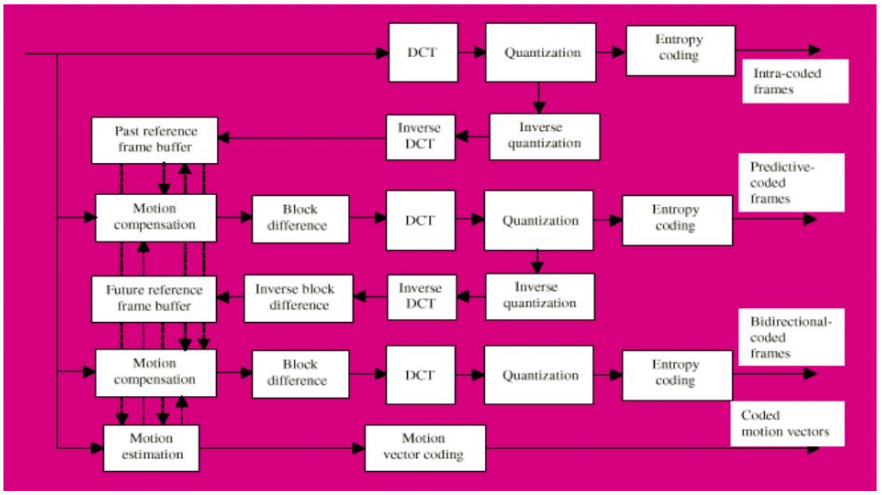
6. H.261 MODEL



a. תיאור הרכיבים בתמונה: (רק מה שחדש)
   i. MEME - מקום בזיכרון ששומר את הפריים בשלמותו.
   ii. M.E. - מודול שמבצע התאמת בלוקים ואומדן תנועה.
   iii. M.C. - פיצוי התנועה.
b. רשימת הנחות:
   i. הקלט של המערכת הזו הוא Macro Blocks והם מגיעים אחד אחרי השני בסדר כלשהו.
   ii. יש לנו מערכת שקובעת לכל MB אם הוא INTER או INTRA (המערכת לא בשרטוט, היא אחראית על המתגים).
c. תהליך:
   i. עבור INTRA:
      1. המתגים במצב עליון.
      2. במסלול העליון מתבצע קידוד שקול ל-JPEG.
      3. מבצעים מסלול הופכי (מהאמצע למטה) ע"מ לשמור על עקביות עם המקלט.
      4. שומרים את ה-MB בזיכרון וממשיכים לבלוק הבא.
   ii. עבור INTER:
      1. המתגים במצב עליון.
      2. המידע נשלח ל-ME ולסוכם.
      3. במודול ה-ME בונים את ווקטור התנועה של הבלוק מהפריים הקודם שמגיע מה-MEM ומהבלוק הנוכחי.
      4. את ווקטור התנועה שהתקבל נשלח ל-MC, ולקידוד אנטרופיה ב-VLC.
      5. ה-MC מקבל את ווקטור התנועה, ואת הפריים הקודם מהזיכרון (MEM). מפעיל את הווקטור על הפריים השמור ושולח את הבלוק המתאים לאחר קיזוז התנועה לסוכם בחלק השמאלי העליון.
      6. הסוכם מחסיר את ה-MB המקורי מה-MB שיצרנו ב-MC, שולח את בלוק ההפרשים לקידוד סטנדרטי במסלול העליון.
      7. בשלב האחרון, נחבר בין אותו בלוק שנשלח לקידוד <u>לאחר שעבר פענוח</u> לבלוק ההפרש, ונשמור את הבלוק המתוקן בזיכרון (כשם שצפוי להתקבל במקלט).
   iii. נקודות נוספות:
      1. ה-H261 הוא דוחס זמן אמת ולכן מאוד מאוד יעיל יחסית (האיכות רק סבירה) ולכן טוב לשיחות וידיאו.
      2. 70% מהחישובים שלו מתבזבזים ב-ME שבו הוא מוצא את ווקטור התנועה.

7. MPEG-1 MODEL

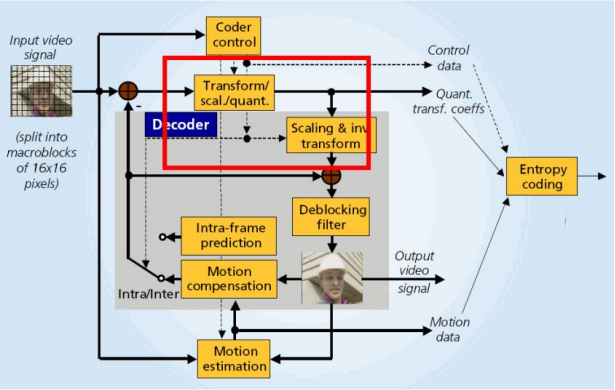
a. MPEG-1 היא דחיסה שנועדה כדי לשמור על וידאו באיכות VHS בצורה דיגיטלית בקצב העברת ביטים של 1.5Mbps.
b. הדחיסה הזאת היא הרחבה של H.261.
c. מודל הדחיסה:



d. הבדלים מ-H.261
   i. ההחלטה ל-Inter/Intra והקידוד עצמו מתבצע ברמת הבלוקים, כלומר שבפריים אחד ייתכן מצב ובו חלק מהבלוקים הם Inter וחלק אחר מהבלוקים הם Intra.
   ii. הבלוקים מחולקים ל-Slices.
   iii. כפי שתיארנו למעלה, אפשר לבצע התאמת בלוקים ב-Half Pel Refinement.
   iv. ה-MEM יכול להכיל יותר מפריים אחד.
   v. ניתן לבצע חיזוי דו-כיווני.

8. H.264 MODEL

a. תקן דחיסת וידאו מודרני שנחשב כ-GENERAL PURPOSE. המהפכות העיקריות שלו:
   i. דחיסת וידאו General purpose, החל מוידאו צ'אט ועד לשימור סרטים.
   ii. שימוש ב-ICT לעומת DCT.
b. שכלול של התאמת הבלוקים:
   i. גדלי בלוק משתנים (רוחב או גובה של 4,8 או 16).
   ii. רזולוציה של רבע פיקסל (בהשוואה לחצי פיקסל של MPEG.2) - נקרא Quarter pel refimement.
   iii. חיזוי ווקטורי תנועה (בשני הצדדים).
c. בנוסף:
   i. שימוש ב-Intra prediction.
   ii. שימוש בקידוד אנטרופיה מתקדם.
d. כל אלו מביאים לשיפור משמעותי בחיזוי, ומכאן גם ביכולת הדחיסה ובביטרייט הסופי.
e. כדי לתמוך ביכולות האלו - אנחנו מוותרים את התאימות אחורנית.
f. המודל:



g. רכיבים:
   i. **Coder Control** - מחליט אם MB הוא Inter או Intra.
   ii. **Transformation/Scaling/Quantization** - רכיב שמבצע קוונטיזציה וטרנספורמציה ל-ICT (קירוב ל-DCT עבור מספרים שלמים).
   iii. **Scaling & Inverse Transform** - פילטרים שהופכים את הבלוק בחזרה לייצוג התמונה שלו, ומשמשים כדי לקבל שמתקבלת במקלט.
   iv. **Deblocking FIlter** - מודול שמחבר בין כל הבלוקים של הפריים לפני שנשלח החוצה.
   v. **Intra frame prediction** - מודול שאחראי לביצוע חיזוי מרחבי על ה-Intra (מעין DPCM על התמונה עצמה).
   vi. **Motion estimation** - מודול שאחראי לחשב ווקטורי תנועה. ב-H.264, מעבר לתת-רזולוציה גבוהה, ושימוש בצורות בלוקים שונות, אנו גם חוזים את ווקטורי התנועה עצמם ומשלימים אותם (בעיקרון דומה לחיזוי דגימות ב-DPCM).
   vii. **Motion compensation** - מודול קיזוז תנועה, פועל בדומה לזה שיש ב-H.261.
   viii. **Entropy Coding** - מקודד אנטרופיה ייחודי, מותאם לוידאו.
   ix. **MEM** (אזור התמונה) - בזיכרון שלנו אנחנו זוכרים את כל הבלוקים של 5 פריימים קדימה ו5 אחורה.
h. תהליך:
   i. עבור INTRA:
      1. ה-Coder Control אומר שזה Intra
      2. המתג של Inter\Intra עולה למעלה
      3. ב-Intra Prediction מבצע חיזוי של בלוק Intra
      4. מחסרים את הבלוק שקיבלנו עם החיזוי.
      5. מבצעים דחיסה ושולחים.
   ii. עבור INTER:
      1. ה-Coder Control אומר שזה Intra
      2. המתג של Inter\Intra יורד למטה
      3. שולחים את הבלוק ל-Motion estimation ומוצאים ווקטור תנועה.
      4. שולחים את ווקטור התנועה ל-Motion compensation.
      5. יוצרים את הפריים החדש עם פריימים קודמים שיש לנו בזיכרון בעזרת ווקטורי התנועה.
      6. מחסירים עם הבלוק המקורי ועושים לבלוק שנוצר מהחיסור את הטרנספורמציה והקוונטיזציה.
      7. מקודדים ושומרים את הבלוק החדש ב-BUFFER בזיכרון (ואת הבלוקים האלו אוספים לכדי הפריים החדש).

ICT & IICT

התמרת ICT היא התמרה דמויית DCT עבור **מספרים שלמים**. ההתמרה מוגדרת עבור מספרים שלמים המיוצגים על ידי עד 16 סיביות.

את ההתמרה אנו מגדירים בעזרת המטריצה הבאה:

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{pmatrix}$$

בהינתן מארקובלוק של 16x16 פיקסלים, נחלק אותו ל-16 בלוקים של 4x4 פיקסלים. כדי להפעיל את ההתמרה על בלוק נתון, נבצע מכפלת מטריצות בין הבלוק לבין מטריצת הטרנספורמציה. נשים לב שכל הפעולות שנעשות כאן הן פעולות שינוי סימן (הכפלה במינוס 1), השארת הסימן (הכפלה ב-1) או פעולות shifting (הכפלה ב-2) - פעולות פשוטות שמורידות את הסיבוכיות החישובית פי כמה וכמה ומספקות תוצאות דומות ל-DCT ממשי.

Hadamard Transform

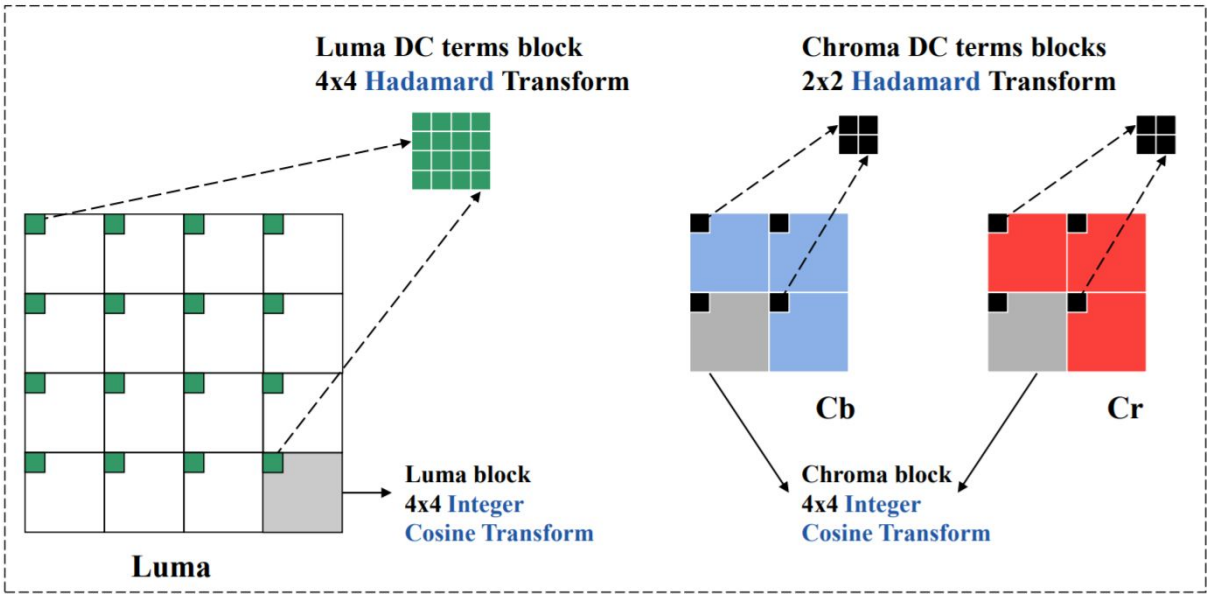
על הנושא הזה הוא דיבר ממש בקצרה.

הצעה נוספת שהייתה היא לקחת את מקדמי ה-DC של **בלוקים** סמוכים ב-Luma, כלומר 4x4 מקדמי DC, ולקודד אותם באמצעות התמרה נוספת שנקרא Hadamard Transform .

לבלוקים של הצבע עושים אותו דבר, רק עם קבוצות בלוקים של 2x2.

גם ההתמרה הזאת מתבצעת במספרים שלמים, והסיבוכיות שלה משנית.

סיכום ההתמרות ב-H.264



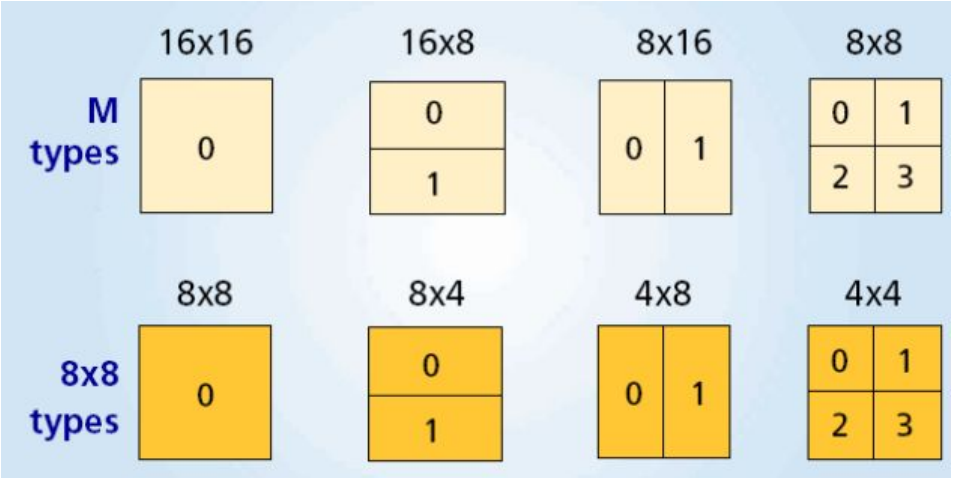
Quantization ב-H.264

הקוונטיזציה ב-H.264 נעשית בצורה של 52 מדרגות לא אחידות, שהקפיצה ביניהן היא תוספת של 12.5% בכל פעם. הקוונטיזציה נעשית באופן שונה ל-Luminance ול-Chrominance. מעבר לזה לא ממש דיברנו (למרות שבאמת אפשר לשאול אותו על זה).

Motion Compensation & Estimation in H.264

הנחות היסוד היא שהחיזוי נעשה רק על צירי X,Y. הסיבוכיות של חישוב העומק עצומה ונראה שאין שיפור באיכות הקידוד אם משתמשים בה. הנחה נוספת היא כי ב-H.264, רזולוציית החיפוש היא ¼ פיקסל (הגדלה פי 4), ולא ½ פיקסל (הגדלה פי 2) כמו ב-MPEG-2. נקרא גם **רבע PEL**.

ב-H.264, אנחנו נשתמש בטכניקות חדשות שיעלו את סיבוכיות החישוב, אבל יביאו ליתרון משמעותי בצמצום המידע בתמונת ההפרש "מקוזז התנועה".

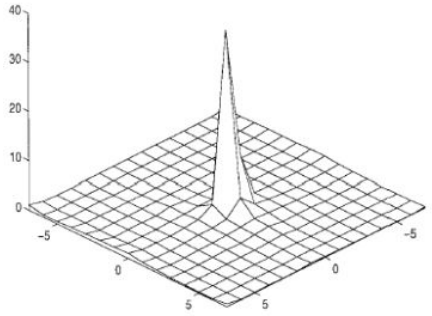


השינוי הראשון (והמשמעותי ביותר) שנעשה הוא הוספת תמיכה ב**גודל בלוק משתנה**. המאקרובלוקים הסטנדרטיים שבגודל 16x16 עדיין קיימים, אך אנחנו תומכים גם בבלוקים בגדלים של 8x16, 16x8, 8x8, 8x4, 4x8 ו-4x4. תכונה זו תורמת לשיפור עצום, כיוון שבהרבה מקרים, גם בתוך בלוק יש תזוזה של אובייקטים שונים לכיוונים שונים.

הבעיה העיקרית בשימוש ברעיון זה הוא גילוי החלוקה הכי טובה למאקרובלוק. יש הרבה חלוקות אפשריות (בערך 259), ואם ננסה להריץ את אלגוריתם זיהוי התנועה על כל אחת מהחלוקות נגיע לסיבוכיות גבוהה במיוחד. לכן, פותחו היוריסטיקות ואלגוריתמים רבים שאומרים איך לחלק את הבלוק, בין אם בהסתמכות על פריימים קודמים, לפי הסביבה ואפילו רשתות עמוקות.

נראה כי ניתן לאזורים שבהם התזוזה יותר חזקה בלוקים יותר קטנים, ולאזורים "חלקים", ללא הרבה תנועה, בלוקים גדולים.

בנוסף, אנחנו מריצים אלגוריתמים לחיזוי ווקטורי התנועה על פי הסביבה, זאת על מנת לשפר את יעילות הקידוד עצמו.אנחנו יכולים להרשות לעצמנו לעשות זאת כי השונות של ווקטורי התמונה באזורים קטנים היא מזערית. בתמונה אפשר לראות את התפלגות ווקטורי התנועה הסמוכים

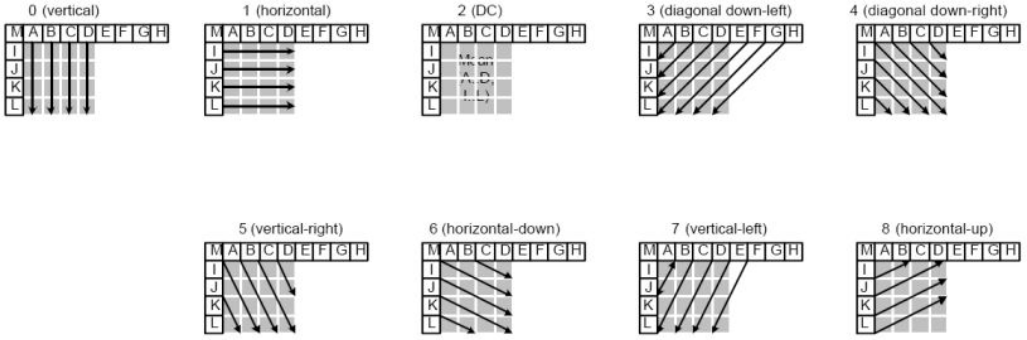


Distribution of adjacent motion vectors

Intra Frame Prediction (ייחודי ל-H.264)

ב-H.264, מעבר לדחיסה דמוית JPEG, אנו מבצעים חיזוי מרחבי על כל בלוק של התמונה. בגלל שאנחנו מניחים מונוטוניות של רוב התמונה , רוב הפיקסלים בבלוק יהיו דומים אחד לשני, והשגיאה תהיה יחסית נמוכה.

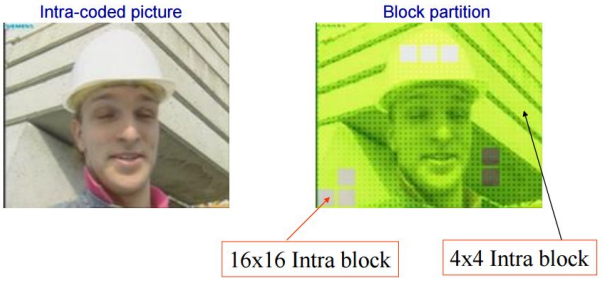
להלן השיטות השונות לחיזוי (או השלמת) הפיקסלים:



ההחלטה על מוד החיזוי היא פחות בעייתית: יש 9 אפשרויות ונוכל לנסות את כולן ולראות מי הכי טובה לנו. אבל בפועל אין צורך שנריץ את כולן - פותחות מספר היוריסטיקות שבוחרות שיטה קרובה לאופטימלית לחיזוי עד אחוד אחד.

ניתן למקבל את התהליך בהרצה על THREAD-ים שונים ובכך לחסוך עוד זמן.

כדי לשפר את היעילות, נחלק את הבלוקים כך שבלוקים מונוטוניים יהיו גדולים יותר מבלוקים עם שינוי רב בצבע.



Deblocking

בדחיסות מבוססות בלוקים, עקב החלוקה של הפריים לבלוקים, ייתכן מצב שבו בחיבור התמונה נוצרים גבולות במקומות שלא אמורים להיות, וגבולות שכן אמורים להיות בתמונה נעלמים. מודול ה-Deblocking ב-H.264 מנסה לאחות את הגבולות שבין

הבלוקים השונים.

גם זו הייתה טכניקה קיימת, אך ב-H.264 היא נחלה הצלחה רבה.

הרעיון הכללי הוא פשוט: נסתכל על 2 בלוקים שכנים שעלינו לחבר. אם ההפרש בין הגבולות הוא קטן, נריץ עליהם פילטר חכם כדי להסיר את ההבדל, ואם ההפרש ביניהם גדול, אז היה שם גבול גם לפני החלוקה לבלוקים ואין צורך להחליק אותו. הפרש זה נקבע לפי הסביבה ופרמטרים אחרים.

נספחים ונושאים שקשורים אבל אין לי מושג איך לשלב אותם

יתרונות וחסרונות של B-FRAMES

יתרונות:

- הם תופסים פחות מקום מסוגי פריימים רגילים
- הדחיסה בהם הכי טובה.

חסרונות:

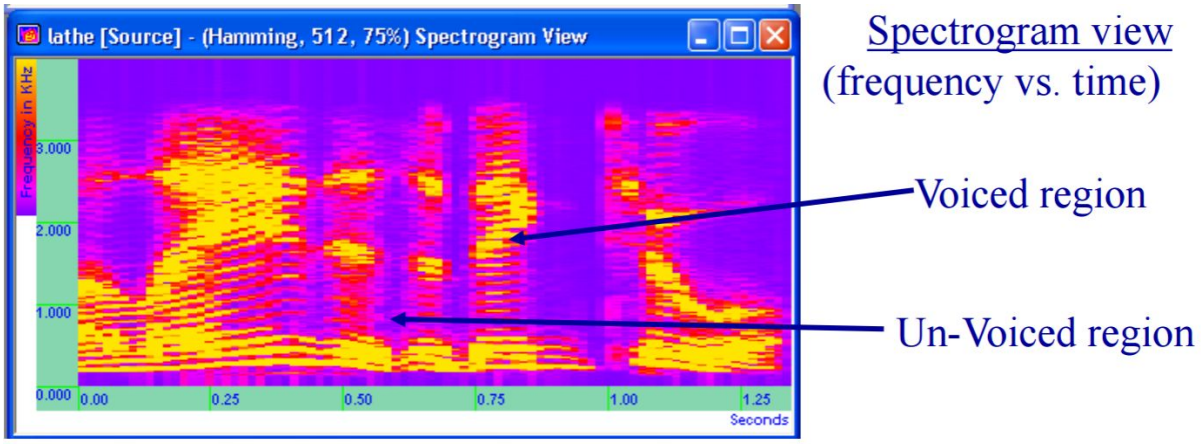
- הם דורשים הכי הרבה כוח חישוב. (גם בצד המקלט וגם בצד המשדר).
- שימוש בהרבה B-FRAMES יוצר אפקט BLOCKINESS בתמונה, עקב הקוונטיזציה החזקה.

- ב-LOCO, הקונטקסט הוא ווקטור של 3 מספרים שעברו קוונטיזציה.
- כאשר עושים קוונטיזציה ב-LOCO, זה נקרא NEAR-LOSSLESS. הקוונטיזציה נעשית בפקטור קטן של 2. (לכן נקרא NEAR LOSSLESS).
- בקידוד AC, מספרים שליליים מקודדים ב-ONE'S COMPLIMENT.
- ניתן לפענח בלוקים של אינטרה בעזרת השכנים, לא משנה כיצד הם מקודדים.
- ב-LPC יש 44 מסגרות לשנייה X לכל מסגרת 54 ביטים (5 ביטים לכל מקדם, ל-PITCH וה-UNVOICED ביחד יש 7 סיביות)

תוספות

דחיסת קול

1. פונמה: יחידת המבע הקטנה ביותר שיש במילה. סדר היחידות הוא זה:
   a. פונמה
   b. מספר פונמות יוצרות הברה
   c. מספר הברות יוצרות מילה
   d. במילה משולש הפונמות הן: מ-שו-ל-ש
      i. נשים לב כי האות שין והאות וו התחברו- זה ייתכן רק עבור אותיות תנועתיות ('ו', 'י' וכו'...). כלומר 'שו' היא פונמה אך 'לש' היא לא פונמה אחת כי אם 2 (הברה).
      ii. במקרה הזה 'שו' היא גם הברה (על אף שהיא פונמה יחידה- היא פונמה המורכבת מאות עיצורית ואות תנועתית)
2. ניתן להעריך PITCH מאנליזת LPC (מישור התדר) ע"י חישוב ההפרש בין 2 שיאים עיקריים עוקבים (פורמנטים)
3. ספקטרוגרמה:



   a. ניתן לראות את ההשתנות הספקטרלית (מישור התדר) בזמן בעזרת ספקטרוגרמה. ניתן לחשוב על זה כעל שילוב של תמונת האות במישור הזמן ובמישור התדר.
   b. ציר ה-X: זמן
   c. ציר ה-Y: תדר
   d. הצבע (אדום גבוה): אמפליטודה.
   e. קל לדמיין כי הפורמנטים פרוסים אנכית לציר ה-X, כלומר בכל נקודה בזמן ניתן לקחת פרוסה מהגרף (בה הזמן לא משתנה) ולקבל תמונה במישור התדר בלבד של הפורמנטים כאשר גובהם ע"פ ציר Y הוא מיקומם בתדר וצבעם הוא אינדיקציה לעוצמה.
   f. תמונת ספקטרוגרמה של רעש היא לא אחידה בעוד שתמונת ספקטרוגרמה של אות דיבור מציגה תבניתיות שמאפיינת את מחזוריות האות. את אותו הדבר אפשר להגיד על אוטוקורלציה.