# פענוח המידע הגולמי המקודד ב-JPEG

## רקע

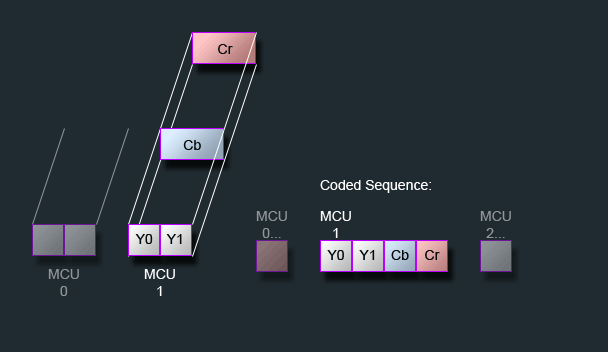
מסמך זה יתאר את אלגוריתם פענוח המידע הגולמי המקודד ב-JPG, בהינתן שכבר פורסרו כל המרקרים. בפרט, קיימות טבלאות ההאפמן, הקוונטיזציה לכל component, וכן ידועים ה-sampling factors לכל קומפוננט. המידע הגולמי ב-JPEG נמצא מייד אחרי המרקר SOS.

דגש חשוב: רציף מהצורה FF 00 עשויים להופיע. ה-00 הוא PADDING שיש להתעלם ממנו, ואם זה לא 00 צריך לפתוח מרקר חדש.

## מבנה התמונה – רצף MCUs

בקידוד תמונה בפורמט JPEG, מחלוקת התמונה לתתי-תמונות, אשר על כל אחת מתתי-התמונות מתבצעת הדחיסה בנפרד. כל תת-תמונה בתיאור מכונה MCU: Minimum Coded Unit. המבנה של כל תת-תמונה תלוי בראש ובראשונה במספר רכיבי הצבע הנדרשים לתיאור התמונות. לדוגמה, תמונה בשחור-לבן ניתנת לתיאור על-ידי רכיב צבע יחיד. תמונה בצבעים ניתנת לתיאור בשיטות שונות, כאשר הנפוצה מביניהן מכונה YCbCr (ראו מסמך אחר). בכל MCU, מופיע המידע על כל רכיב צבע בנפרד. כך למשל, עבור MCU בגודל 8\*8, מבנה אפשרי של המידע הוא תיאור רכיב ה-Y, לאחריו רכיב ה-Cb ולאחריו רכיב ה-Cr.

על-מנת לצמצם את גודל הקובץ, נעשה לעתים תהליך בשם Chroma Subsampling. בייצוג תמונה בשיטת YCbCr, רכיב ה-Y נקרא Luma (בהירות) והרכיבים Cb ו- Cr מכונה Chroma (צבע). העין האנושית רגישה משמעותית יותר לשינויים בבהירות מאשר לשינויים בצבע. לכן, לעתים מתבצע צמצום של רכיבי הצבע בקובץ – זהו תהליך של Chroma Subsampling. דוגמה אחת היא דחיסה המכונה 4:2:2, בה דוחסים רכיבי צבע אופקיים צמודים:



בשיטה זו, כל אחד מהריבועים הקטנים מייצג תת-תמונה בגודל 8\*8. על-מנת לקודד MCU, שגודלו הפעם 16\*8, מייצגים כל אחד מ-2 הריבועים של 8\*8 ע"י רכיב Y נפרד, אך באמצעות אותם רכיבי Cb ו-Cr, ש"מורחים" על פני שני פיקסלים אופקיים צמודים. צמצום 2 הרכיבים הצמודים נעשה במקום ע"י מיצוע שלהם. השינוי הנ"ל הינו קטן, ולכן לרוב אינו מורגש לעין האנושית. יחסי הצמצום מוגדרים ע"י ה-Sampling Factors שמוגדרים ב-metadata של הקובץ.

לפי אותם Sampling Factors ניתן לדעת מהו גודל של כל MCU (לרוב מדובר ב8\*8, 16\*8, 8\*16 או 16\*16). לכל גודל של MCU קיימת הגדרה אחידה לסידור רכיבי הצבע שמתארים את ה-MCU. בדוגמה שבתמונה שלמעלה, בתחילה מקודדים 2 רכיבי ה-Y, לאחר מכן ה-Cb ולבסוף ה-Cr.

מכאן, היות שנתונים מימדי התמונה ב-metadata של הקובץ, ניתן לחשב מהו מספר ה-MCU-ים שמהם מורכב הקובץ. במקרה של "שארית" בחלוקת מימדי התמונה במימדי MCU, נוצר מצב בשם Partial MCU, שהוא MCU שלם נוסף, אשר בו ערכי הקצוות שחורגים מגודל התמונה (ואשר אינם מוצגים בתמונה בסופו של דבר עקב מימדיה) לרוב זהים לערך האחרון באותה שורה/עמודה.

כעת, לאחר פרסור הקידוד של כל MCU (בו נדון בחלק הבא), חיבור התמונה המלאה מתוך ה-MCU-ים המפורסרים נעשה ע"י הצבת כל MCU במקומו לפי מימדי התמונה ומימדי ה-MCU.

## פענוח MCU

בפרק זה נדון בפענוח MCU. למעשה, נדון בפענוח רכיב צבע כלשהו בתוך MCU, כאשר הטיפול בכלל רכיבי הצבע זהה. נניח לצורך הדוגמה כי מדובר ברכיב Y ושמימדי ה-MCU הם 8\*8. כפי שראינו, משמעות רכיב הצבע ב-MCU היא תיאור ערכי הבהירות של התמונה ב-64 הפיקסלים האלו.

עם זאת, לצורך שיפור הדחיסה ב-JPEG, ערכי הבהירות לא שמורים כפי שהם. נניח כי ערכי ה-MCU נתונים ע"י המטריצה שנסמן ב-, שמימדיה הם כאמור 8\*8. אזי, ביצירת קובץ ה-JPG, התבצע התהליך הבא:

1. ביצוע התמרת DCT על המטריצה A, לקבלת מטריצה שנסמן ב-B. לערך השמאלי עליון ב-B נקרא מקדם ה-DC, וליתר הערכים נתייחס כמקדמי AC.
2. ביצוע תהליך קוונטיזציה למטריצה B תוך שימוש במטריצת קוונטיזציה כפי שמוגדר ב-metadata של הקובץ. למטריצה לאחר הקוונטיזציה נקרא C.
3. קידוד המטריצה C באמצעות קידוד האפמן המוגדר ע"י טבלת האפמן עבור רכיב זה, כפי שמוגדר ב-metadata של הקובץ.

במסמך זה, לא נדון בהתמרת ה-DCT ובקוונטיזציה, עקב פשטות התהליך באופן יחסי. במקום זאת, בחלק הבא נדון בפענוח המידע הגולמי המצוי בקובץ ה-JPG (מייד אחרי סוף ה-marker בשם SOS), עד לקבלת המטריצה C – כלומר, מטריצת מקדמי ב-DCT, לאחר תהליך קוונטיזציה. לאחר שמתקבלת המטריצה C, יש לבצע עליה את תהליך הקוונטיזציה ההפוך באמצעות מטריצת הקוונטיזציה, ולאחר מכן ביצוע התמרת DCT ההפוכה לקבלת ערכי רכיב הצבע המקוריים, בשיטת YCbCr.

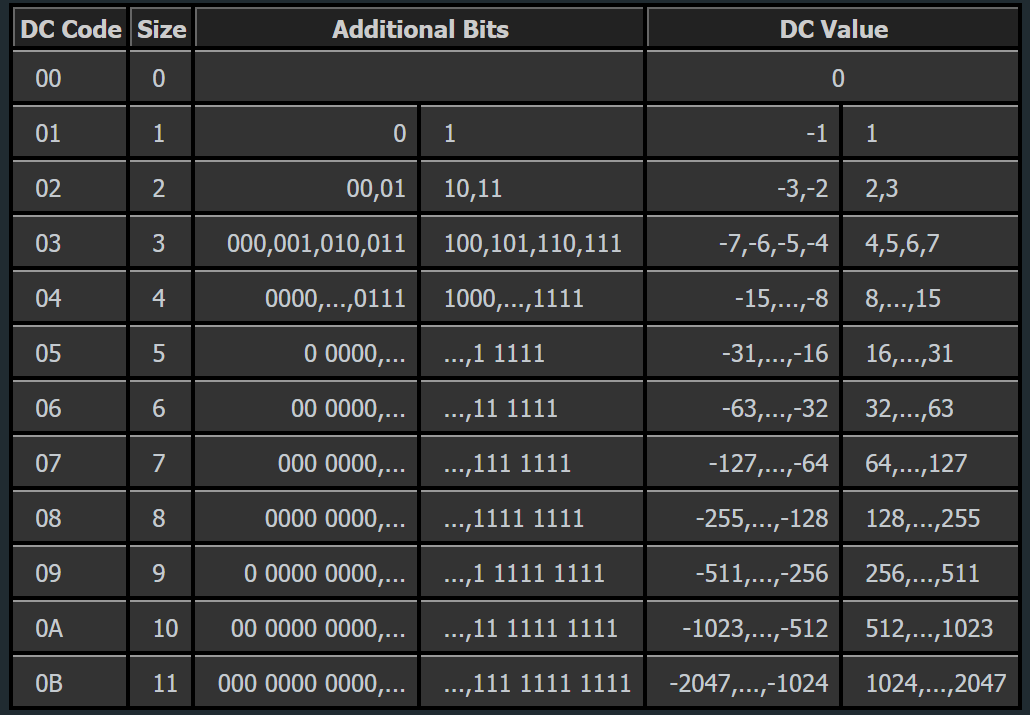
## פענוח המידע הגולמי בקידוד האפמן

תהליך פענוח המידע הגולמי מורכב מ-2 חלקים: פענוח רכיב ה-DC, ולאחר מכן פענוח רכיבי ה-AC. פענוח ה-DC נעשה תחילה, ולכן התיאור יתחיל ממנו.

### פענוח רכיב ה-DC: שיטת DPCM

תחילה נדגיש – בכניסה השמאלית-עליונה של המטריצה לא שמור ערך רכיב ה-DC כפי שנדרש, אלא ההפרש בין רכיב ה-DC הנ"ל לבין רכיב ה-DC מהבלוק הקודם (עבור ערך ה-DC הראשון נשמר הערך עצמו). נתייחס בהמשך ההסבר לערך המפוענח בתור ערך ה-DC, ובסוף תהליך פענוח ה-MCU-ים יש לחשב את ערכי ה-DC האמיתיים (ולא את ההפרשים) באמצעות סריקה קדמית.

פענוח ערך ה-DC מתוך המידע נעשה בצורה הבאה, בהינתן טבלת ההאפמן:

1. סרוק את המידע הגולמי ביט אחר ביט, עד הגעה למחרוזת ביטים אשר יש לה ערך בטבלת ההאפמן. נסמן את ערך זה ב-DC Code.
2. קרא עוד DC Code ביטים מהמידע הגולמי. נסמן ערך זה כ-Additional Bits.
3. ערך ה-DC (DC Value) יחושב כעת באמצעות הטבלה הבאה (שאינה מופיעה ב-JPG):

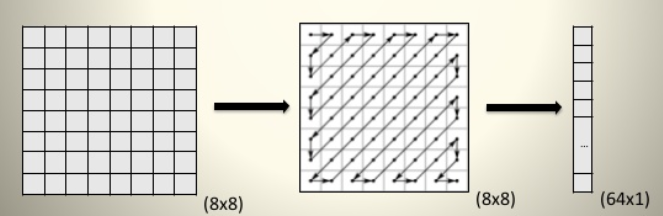
נתבונן בשורה בטבלה שמתאימה ל-DC Code שמצאנו. כעת, נתבונן ב-Additional Bits שקראנו. במידה שה-MSB הוא 0, נחבר את הערך למספר השמאלי בתת-העמודה השמאלית של העמודה DC Value. כך למשל, עבור DC Code=5, וערך Additional Bits ששווה ל- 00011, נחבר את ערך ה-Additional Bits (הוא 3) למספר השמאלי בתת-העמודה השמאלית של העמודה DC בשורה, הוא (-31), ונקבל 28. במידה שהערך Additional Bits הוא בעל MSB שערכו 1, נוסיף את הערך למספר השמאלי באותה השורה, **לאחר איפוס ה-MSB ב-Additional Bits**.

ניתן לבצע חישוב של הטבלה הנ"ל באופן אוטומטי באמצעות מציאת נוסחאות מתאימות לכל כניסה, אולם תהליך זה טכני ולא יפורט כאן.

כזכור, לאחר חישוב ערך ה-DC (DC Value), נזכור כי לא מדובר בערך ה-DC האמיתי, אלא בהפרש מערך ה-DC הקודם.

### פענוח רכיב ה-AC: שיטת RLE

לאחר קריאת המידע הגולמי לצורך חישוב רכיב ה-DC, יופיע במידע הגולמי הנדרש לחישוב 63 רכיבי ה-AC. הקידוד הוא וריאנט של קידוד המכונה RLE (Run LEngth), בו מקבצים רצפים של אותו תו לזוגות, כך שרצף של n פעמים התו a ייוצג ע"י (a,n).

עם זאת, הערכים המקודדים שייקראו לא מהווים ערכי המטריצה שורה-אחר-שורה, אלא קריאת המטריצה בצורת זיגזג, כך:

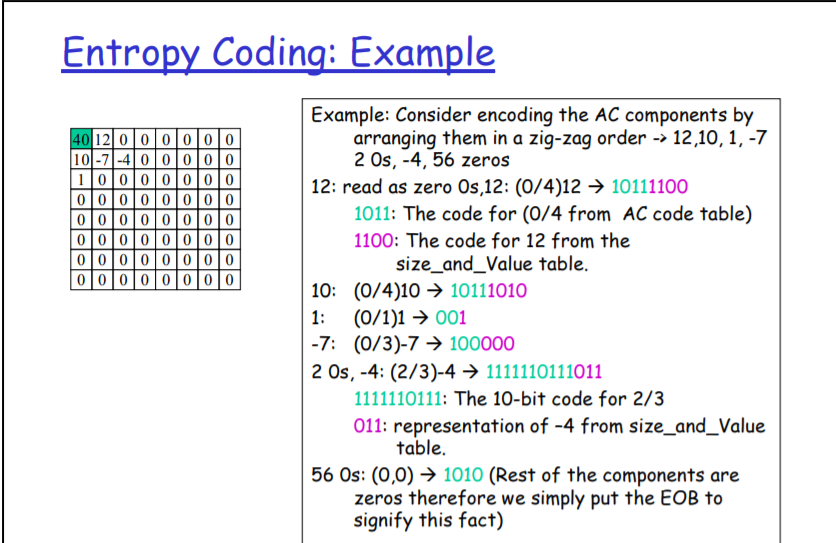
בקידוד ערכי ה-AC, עקב ריבוי האפסים, מקודד המידע במקטעים, כאשר כל מקטע הינו מהצורה S1 S2.   
S1 הוא זוג מהצורה (RunLength, Size), כאשר:

* RunLength הוא מספר בין 0 ל-15 המתאר את ערכי ה-AC הרצופים הבאים שערכם אפס.
* Size הוא מספר בין 0 ל-A, המייצג את מספר הביטים לקידוד ערך ה-AC הבא אחרי רצף האפסים, אשר שונה מאפס.

במידה ש-, מדובר בסימן EOB – End Of Buffer, שמשמעו שהחל מנקודה זו, כלל ערכי ה-AC שווים לאפס. הזוגות הנ"ל מקודדים בקידוד האפמן ע"י טבלת ההאפמן AC. כלומר, קוראים ביט-ביט עד שמגיעים לערך של בית מסוים. שוברים אותו לשניים – חצי חצי, וזה בדיוק S1. S2 הוא באורך Size ביטים.

S2 מכונה Value. חישוב ערך ה-AC נעשה באופן דומה לחישוב ערך ה-DC בחלק הקודם: משתמשים ב-Size לבחירת השורה (בדומה ל-DC Code ב-DPCM) וב-S2 לחישוב הערך (בדומה ל-additional bits ב-DPCM).

דוגמה:



תימוכין:

