מבוא לראייה ממוחשבת: מטלת מנחה (ממ"ן) 11 - נדב כחלון; דו"ח שאלה 4

<u>סעיף a</u>
הקוד לסעיף זה מופיע בקובץ Question4PartA.py, והוא מכיל את המימוש של הפירמידה הלפלסיאנית (והגאוסיאנית) בפונקציות השונות, יחד עם הרצה שלהן בתכנית הראשית על תמונה בספריית התמונות.
*** במהלך הדו"ח אאזכר שוב ושוב את המאמר "The Laplacian Pyramid as a Compact Image Code" של Peter J. Burt ו-Edward H. Adelson מ1983 (אתייחס אליו בתור "המאמר הנ"ל").

ראשית, בתחילת הקובץ מוגדר הפילטר שנשתמש בו לצורך החלקה בכל השאלה. זהו kernel בעל התכונות שהפירמידה דורשת (נירמול, סימטריה, equal-contribution). ההגדרה שלו (ושל תכונת equal-contributionה) מובאת במאמר הנ"ל, ולא אחזור עליה כאן. אנחנו יוצרים גרסה חד-מימדית של הkernel, ובעזרת הכפלה בשחלוף של עצמו אנחנו מקבלים את הגרסה הדו-מימדית שלו. אנחנו מגדירים אותו בצורה גלובלית כדי שנוכל להשתמש בו גם בקוד של הסעיף הבא.

הפונקציה reduce:
פונקציה זו מממשת את שלב הreduce בו משתמשים במעבר בין הרמות בפירמידה הגאוסיאנית. המימוש שלו עוקב בדיוק אחר האלגוריתם המתואר במאמר הנ"ל - ראשית אנחנו מטשטשים את התמונה בעזרת הkernel (בהנחה שחלות עליו הדרישות הנ"ל - ובפרט equal-contribution), ואז אנחנו פשוט דוגמים כל פיקסל שני.

הפונקציה expand:
פונקציה זו מממשת את שלב expand בו משתמשים בשחזור של רמה נמוכה יותר בפירמידה הגאוסיאנית, כחלק מתהליך יצירת הפירמידה הלפלאסיאנית. המימוש שלו עוקב בדיוק אחר האלגוריתם המתואר במאמר הנ"ל - אנחנו מכפילים את הרזולוציה בעזרת הכנסת פיקסלים שחורים בין הפיקסלים הנתונים, מריצים פילטר עליו חלות הדרישות הנ"ל (ובפרט equal-contribution), ואז מכפילים את ערכי הפיקסלים פי 4 (מכיוון שלאחר הכפלת הרזולוציה כל פיקסל התרחב ל-4 פיקסלים, ששלושה מהם היו שחורים).

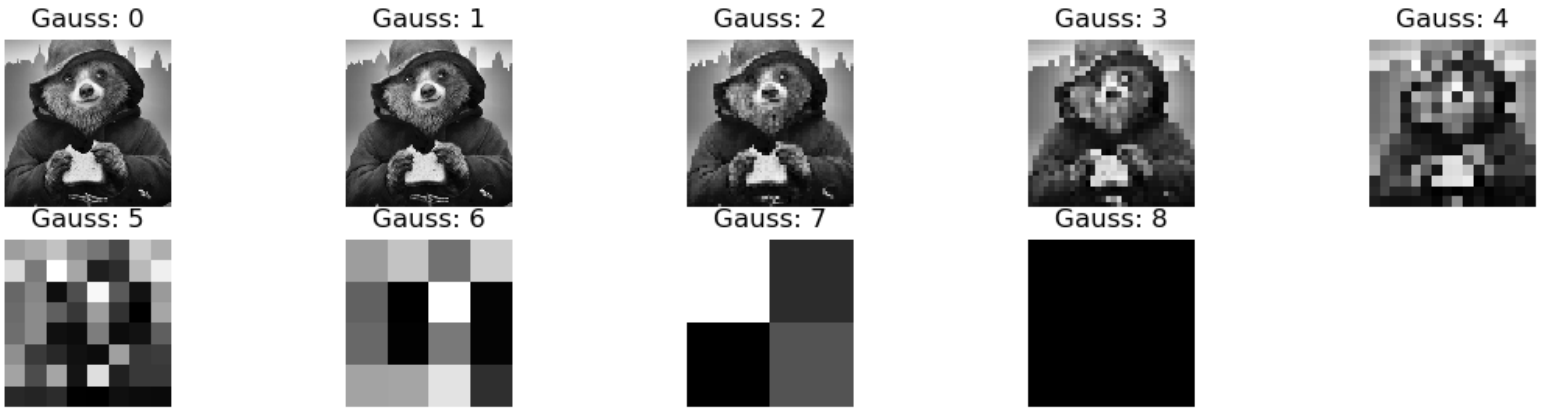
הפונקציה inagePyramids:
פונקציה זו מייצרת את הפירמידות והגאוסיאנית והלפלסיאנית לתמונה נתונה. נשים ב שאם לא נתון גובה לפירמידה, אנחנו מייצרים פירמידה עם גובה מקסימלי (שורה 39): גובה זה הוא כמספר הפעמים שנוכל לחצות את הרזולוציה של התמונה ועדיין להישאר עם תמונה ברזולוציה שלמה.
ראשית אנחנו מייצרים את הפירמידה הגאוסיאנית רמה אחר רמה, בכל פעם מבצעים reduce לרמה הקודמת (בדיוק כמו שמתואר במאמר הנ"ל). אנחנו עוצרים כשהגענו לגובה הדרוש.
גם את הפירמידה הלפלסיאנית מייצרים בהתאם אחר כך - עוברים על כל הרמות של הפרמידה הגאוסיאנית, ובמעבר בין כל זוג רמות מבצעים expand לרמה הגבוהה ומחסרים בין התמונות שקיבלנו (לקבלת המידע שאבד במעבר לרמה הגבוהה יותר). לבסוף אנחנו מכניסים את "קודקוד הפירמידה" הגאוסיאנית לראש הפירמידה הלפלסיאנית, כמבוקש.

הפונקציה plotPyramids היא פונקציה שנועדה להקל על הסרטוט של זוג הפירמידות זו לצד זו, והיא לא רלוונטית לתהליך יצירת הפירמידות (שבאמת חשוב לנו), לכן לא אפרט עליה.

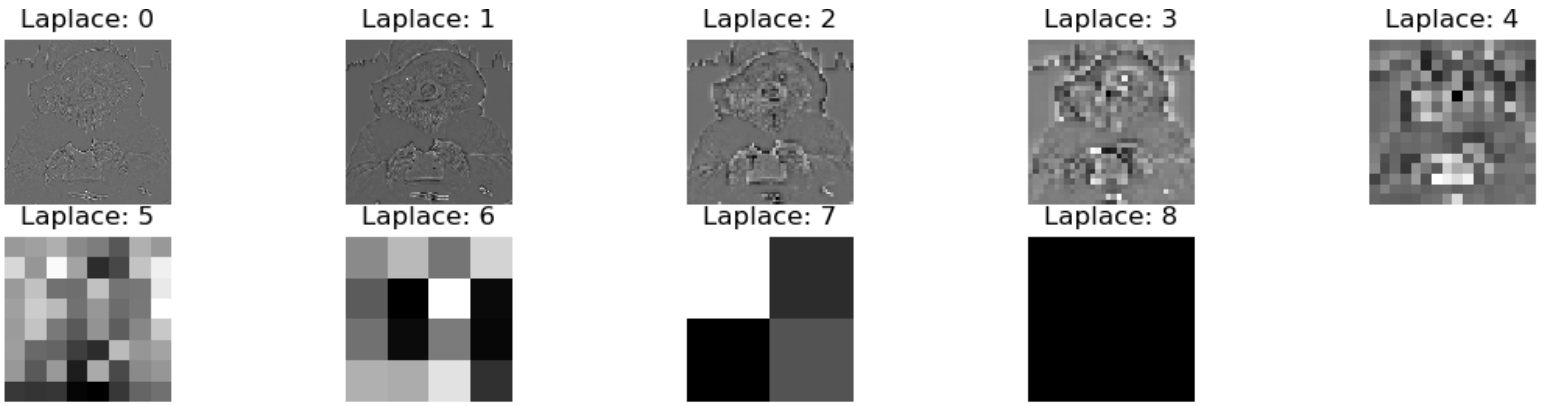
התוכנית הראשית:
התכנית הראשית די ברורה - אנחנו קוראים תמונה מהזיכרון, מייצרים לה פירמידות מתאימות, ומשרטטים אותם - הכל בעזרת פונקציות שכבר הגדרנו.

התמונה המקורית:

הפירמידה הגאוסיאנית:



הפירמידה הלפלסיאנית:



סעיף b

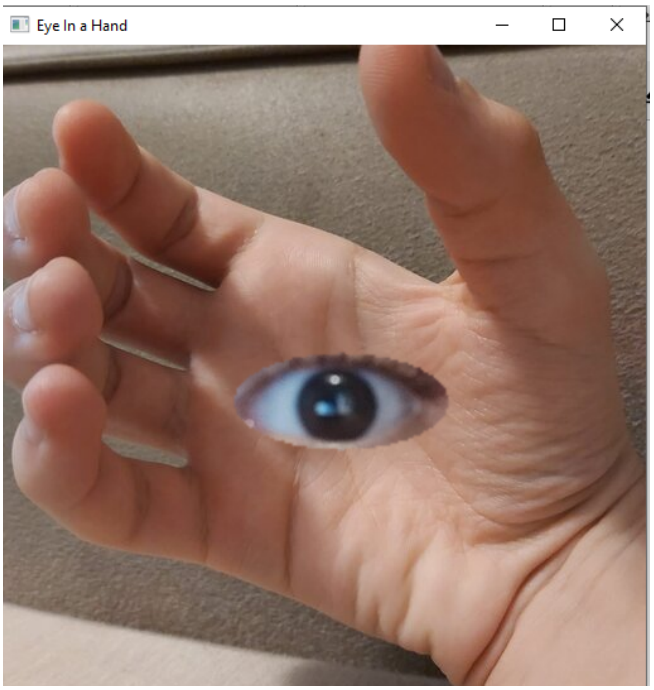
הקוד לסעיף זה מופיע בקובץ Question4PartB.py, והוא משתמש בפונקציות שהוגדרו בקוד של הסעיף הקודם (ראה import בשורה 7), ומגדיר כמה פונקציות נוספות. התוכנית הראשית יוצרת תמונה ובה אשליה שעין נמצאת בתוך כף-יד.

הפונקציה collapse:
פונקציה זו "מקריסה" פירמידה לפלסיאנית ומפיקה את התמונה המקורית ממנה הופקה הפירמידה. הרעיון פשוט, וגם הוא מתואר במאמר הנ"ל. אנחנו מתחילים בקודקוד הפירמידה, ובכל שלב עורכים expand, ומחברים את התוצאה לרמה שמתחת. אנחנו ממשיכים כך עד הרמה האחרונה. היות והפירמידה הלפלסיאנית הופקה מההפרשים בין רמות הפירמידה הגאוסיאנית, והודות לעובדה שexpand ו-reduce אלו פעולות חילופיות, תוצאת הפעולה היא הרמה התחתונה ביותר בפירמידה ה**גאוסיאנית** - שזו בדיוק התמונה המקורית.

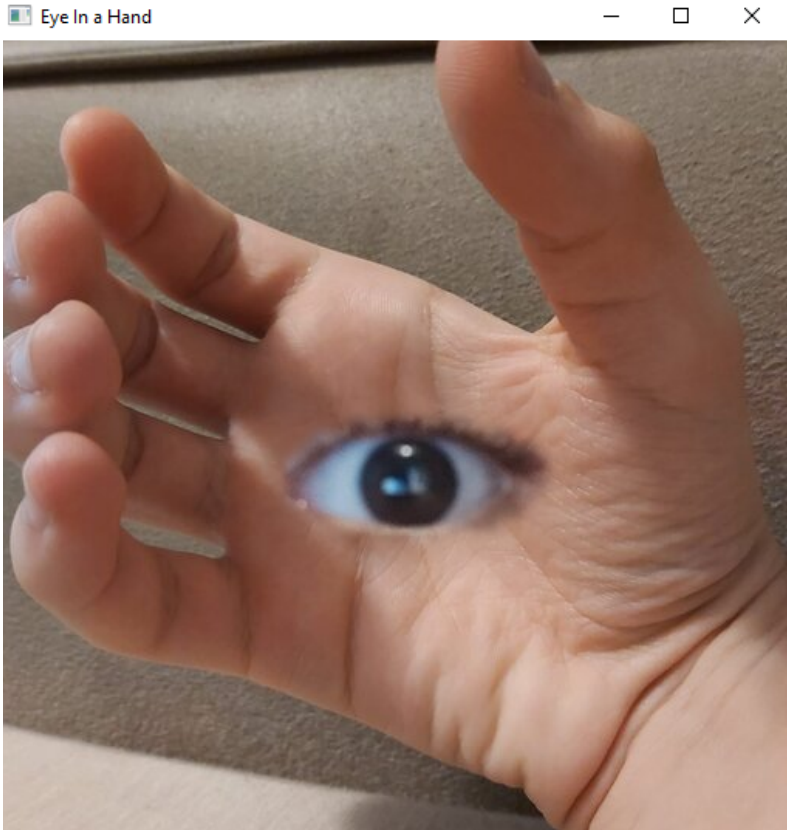
הפונקציה joinLPyramids:
פונקציה זו נועדה למזג פירמידות גאוסיאניות (של תמונות מאותה רזולוציה) בהתאם לפילטר מיזוג נתון. פילטר המיזוג הוא מטריצה שמימדיה כמימדי התמונות מהן הופקו הפירמידות, שמכילה את הערכים 1 ו-0. במקומות בהם הפילטר הוא חיובי (למעשה- לא 0) נבחרת הפירמידה הראשונה, ובמקומות בהם הפילטר הוא 0 נבחרת הפירמידה השניה.
האלגוריתם מאוד פשוט: אנחנו עוברים על כל רמה של הפירמידות (מלמטה למעלה) - בכל פעם ממזגים בין הרמות המתאימות בהתאם לפילטר (שורה 27), מבצעים reduce לפילטר (כדי שיתאים לרמה הבאה - שהרזולוצייה שלה קטנה פי 2), ועושים עדכון קל כדי שהפילטר יקבל רק את הערכים 1 ו-0 (שורה 29), אך בפועל יכולנו לוותר על זה (השארתי את זה כדי למנוע שגיאות כלשהן).

התכנית הראשית:
המטרה של התכנית הראשית היא למזג בין תמונה של עין וכף-יד ברזולוציה 512x512, כדי ליצור אשליה שהעין יוצאת מתוך כף-היד. המיזוג נעשה בעזרת מיזוג הפירמידות הלפלסיאניות שלהן (גובה הפירמידות נקבע בשורה 34). אנחנו טוענים את התמונות המתאימות ויוצרים לכל אחת פירמידה גאוסיאנית ופירמידה לפלסיאנית (שורות 35-42). את הפירמידות אנחנו יוצרים בעזרת הפונקציה imagePyramids שמימשנו בסעיף הקודם. בשורה 47 אנחנו יוצרים את פילטר המיזוג: בעזרת ניסוי ותהייה, הגעתי לפילטר אליפטי במבנה מדויק לצרכינו - חתיכת העין מהתמונה הראשונה והשתלתה בתוך היד מהתמונה השניה. בשורה 51 אנחנו מבצעים את הקריאה למיזוג הפירמידות, ובשורה 52 אנחנו "מקריסים" את הפירמידות לקבלת התמונה הממוזגת. שאר השורות נועדו להצגה של התהליך + התוצר הסופי בעזרת matplotlib.pyplot ו-OpenCV, ולא אפרט עליהן מכיוון שהן לא רלוונטיות לתהליך המבוקש.

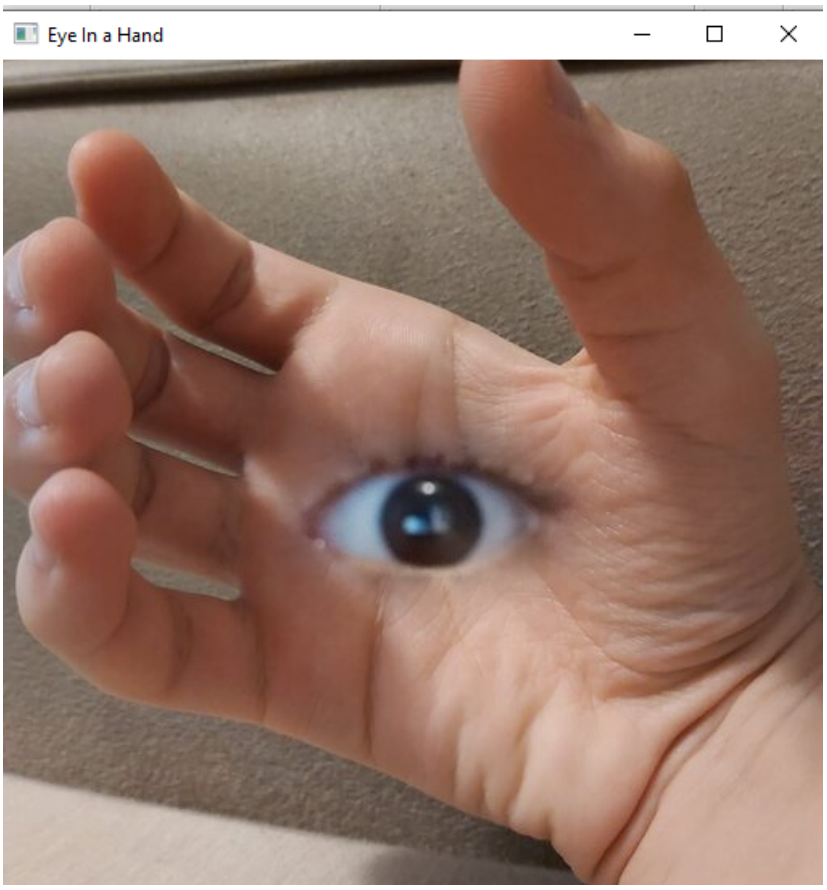
התוצאה שקיבלתי עבור 2 רמות (מעבר מאוד חד, כמצופה):



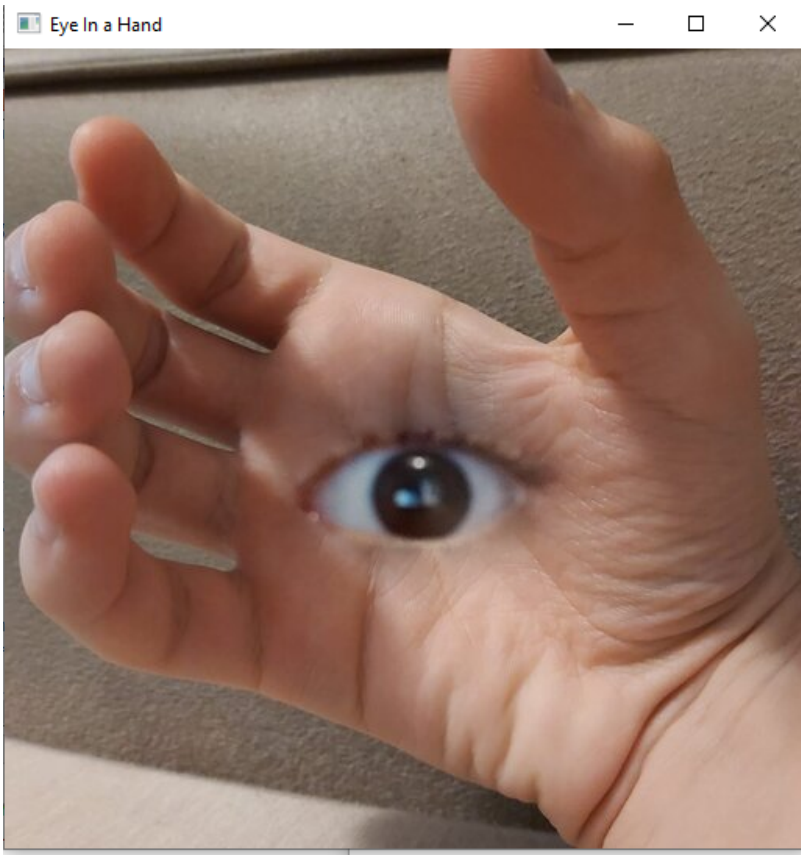
גם עבור 4 רמות קיבלתי תוצאה שנראת מעט "מלאכותית":



תוצאה דומה קיבלתי גם עבור 5 רמות. עבור 6 רמות קיבלתי תוצאה לא רעה:



אבל הרגשתי הכי מסופק עם 7 רמות:



מעבר ל7 רמות הפילטר נהיה מאוד גס ו"משובץ" (במקור הוא אמור להיות אליפטי, והדבר נהיה רע כשננסה לייצג אליפסה ברזולוציה מאוד נמוכה) כך שהתמונה נהייתה מעט כהה ומוזרה ליד העין. החלטתי להסתפק ב-7 רמות.

התהליך כולו מתואר בתרשים הבא, שהפיקה התוכנית הראשית:

בהצלחה!