Image Processing - Exercise 3

Neriya Ben David, neriyabd, 206698581

מבוא :

ראשית, מטרת התרגיל מחולקת לשני חלקים :

החלק הראשון עוסק במיזוג שתי תמונות באמצעות מסכה, ואילו החלק השני עוסק ביצירת תמונת משולבת המכילה שתי תמונות. באמצעות התמונה המשולבת ניתן לראות את אחת משתי התמונות לפי נקודת המבט (תמונה אחת נראה כאשר נסתכל מקרוב, ותמונה אחרת כאשר נביט מרחוק).

הכלים המרכזיים בהם השתמשתי כדי לבנות אלגוריתמים שייענו על שתי הבעיות הינם :

- שימוש בפירמידת גאוסיין תוך שימוש של טכניקות טשטוש והקטנה של התמונה בכל רמה.
- הגדלה של התמונות ע"י שימוש בלקיחת כל פיקסל שני ושימוש בקרנל הנלמד בכיתה לצורך כך.
- שימוש בפירמידת לפלסיאן.
- סכימה של פירמידת הלפלסיאן על מנת לשחזר את הרמות השונות בפרמידת גאוסיין.
- אלגוריתם הנלמד בכיתה למיזוג תמונות.

אלגוריתמים - כללי :

ראשית, כרקע לאלגוריתמים אותם אפרט בהמשך אציג מספר פונ' שמימשתי :

- **גאוסיין:** $gaussian\_pyramid$ - פונ' זו מקבלת כקלט את התמונה, כמות הרמות הרצויות וקרנל, ומחזירה את פירמידת הגאוסיין עם כמות הרמות הנדרשת במערך.
מעבר בלולאה על כל אחת מהרמות, כך שבכל רמה אנו מפעילים קרנל טשטוש על התמונה באמצעות הפונ' $kernel1D$ לאחר מכן אנו מקטינים את התמונה תוך שימוש ב$np.array[::2,::2]$ המורידה כל העמודות השורות הזוגיות (משאירה כרבע מהפיקסלים).
- $gaussian\_rgb\_pyramid$ - פונ' זו משתמשת ב- $gaussian\_pyramid$ עבור כל צבע ומחזירה את שלושת פירמידות הגאוסיין שהתקבלו.

- **הגדלה:** $pyramid\_up$ - פונ' זו מקבלת תמונה וממדים, ומגדילה את התמונה החדשה לממדים שהיא קיבלה באמצעות הוספת פיקסלים בכל הקואו' בהן הפיקסל נמצא בקואו' בה השורה אי זוגית והעמודה אי זוגית (כרבע מהפיקסלים). לאחר מכן מופעל קרנל הנועד לשחזר את התמונה המקורית עם $[1,2,1]$ ו-$transpose([1,2,1])$.
- $expand\_pyramids$ - פונ' זו מקבלת כקלט פירמידת גאוסיין ומחשבת באמצעות $pyramid\_up$ את כל התמונות המוגדלות של הפירמידה שהתקבלה.

- **לפלסיאן:** $laplacian\_pyramid$ - מקבלת כקלט את פירמידת גאוסיין והגדלה שלה, ומחשבת את פירמידת הלפלסיאן ומחזירה אותו.
- $laplacian\_rgb\_pyramid$ - מקבלת כקלט פירמידות גאוסיין עבור כל צבע ובונה בעזרת $expand\_pyramids$ ו-$laplacian\_pyramid$ את פירמידות הלפלסיאן (עבור כל צבע) ומחזירה.
- $sum\_laplacian$ - סוכם את כל רמות הלפלסיאן תוך התאמה לגודל המתאים.

- **קרנל:** $kernel1D$ - מקבלת כקלט תמונה וסוג קרנל k חד ממדי ומפעילה אותו על התמונה פעמיים, פעם אחת את הקרנל המתקבל ופעם שנייה את השיחלוף שלו (כפי שראינו שיותר יעיל בכיתה).

אלגוריתם 1 - $image\ blending$:

שלבי האלגוריתם:

- קריאת קבצי שתי התמונות בצבע, וקריאת המסכה בגווני אפור והפיכת המסכה לבינארית.
- עבור כל אחת משתי התמונות: בניית פירמידת לפליסאן בעלת 5 רמות לכל ערוץ צבע.
- בניית פירמידה גאוסיינית - 5 רמות עבור המסכה ופירמידות לפלסיאן לתמונות עבור כל צבע בנפרד.
- סכימת כל אחת משלושת פירמידות הלפלסיאן (אחת שהתקבלה עבור כל צבע).
- מיזוג שלושת הפירמידות לפירמידה אחת המכילה את שלושת ערוצי הצבע והחזרתה.

פירוט שלבי האלגוריתם:

- קריאת התמונות והמסכה באמצעות פונ' $cv2.imread$ והמרתן ערכיהן ל-$float$ (כדי לא בהכרח לעגל אחרי כל פעולה למספרים שלמים) ושינוי המסכה לתמונה בינארית באמצעות $cv2.IMREAD\_GREYSCALE$, נרמול ב-255 ועיגול הערכים ע"י $np.round$.
- בניית פירמידת גאוסיין עבור כל ערוץ צבע בנפרד עם מספר הרמות הרצויות עבור כל אחת משתי התמונות באמצעות הפונ' $gaussian\_rgb\_pyramid$.
- בניית פירמידות לפלסיאן עבור כל אחד מערוצי הצבע של שתי התמונות באמצעות $laplacian\_rgb\_pyramid$
- אתחול רשימה ריקה שאליה יוכנסו הלפלסיאן המשולבים של כל ערוץ צבע לפי האינדקסים הבאים. $B: 0, G: 1, R: 2$.
- עבור כל צבע, נעבור על פירמידות הלפלסיאן משתי התמונות, נחבר אותן ע"י שימוש בנוסחה הבאה:
$$Mask[i] \cdot L^1_{color}[i] + (1 - Mask[i]) \cdot L^2_{color}[i]$$
כך ש- L^i_{color} מייצג את פירמידת הלפלסיאן בצבע $color \in \{R, G, B\}$ והתמונה ה- $i \in \{0\ ,1\}$ ונכניס לרשימה.
- נסכום עבור כל צבע את פירמידת הלפלסיאן באמצעות הפונ' $sum\_laplacian$.
- איחוד שלושת פירמידות לפלסיאן אלו ע"י פונ' $numpy.stack$ עבור הממד השלישי (ממד ערוצי הצבע), והפיכת התמונה למספרים שלמים והחזרתה.

אלגוריתם 2 - $image\ hybrid$:

שלבי האלגוריתם:

- קריאת קבצי שתי התמונות בגווני אפור.
- עבור כל אחת משתי התמונות: בניית פירמידת לפלסיאן בעלת n רמות.
- בניית לפלסיאן המכיל את k הרמות הראשונות של הלפלסיאן של התמונה הראשונה ו-$n - k$ הרמות האחרונות של הלפלסיאן של התמונה השנייה. (בחירת ה-k תלוי בתמונה)
- סכימת הלפלסיאן והחזרת התמונה שהתקבלה.

פירוט שלבי האלגוריתם:

- קריאת התמונות בדרגת אפור באמצעות פונ' $cv2.imread$ ושימוש ב-$IMREAD\_GREYSCALE$.
- בניית פירמידת גאוסיין באמצעות פונ' $gaussian\_pyramid$ המקבלת כקלט את התמונה, כמות רמות לפירמידה, וקרנל טשטוש (לשם ביצוע ההקטנה) ומחזירה את פירמידת הגאוסיין.
- בניית פירמידות המכילות את הגדלת התמונות באמצעות פונ' $expand\_pyramids$, כך שתמונה תבצע $expand$ לפי הממדים המתאימים ברמה שקדמה לה. פונ' זו מקבלת את פירמידת הגאוסיין ומחזירה את פירמידת שחזור (בלי הרמה הראשונה, לה לא מוגדר לה $expand$).
- בניית פירמידות לפלסיאן לשתי התמונות באמצעות פונ' $laplacian\_pyramid$ המקבלת את פירמידת הגאוסיין ופירמידת השחזור ומחשבת באמצעותן את הלפלסיאן.
- יצירת לפלסיאן מאוחד ע"י לקיחת 2 הרמות הראשונות של הלפלסיאן שהתקבל מהתמונה הראשונה ו- 4 הרמות התחתונות של התמונה השנייה והחזרתו.

(עבור הפונק' אותם מימשתי בעצמי והשתמשתי בהן בפירוט שלבי האלגוריתמים, פירטתי לעיל את המימוש שלי עבורן ואופן פעולתן (עמוד 1)).

היפר - פרמטרים, $thresholds$, ובחירות מימוש באלגוריתם:

1. באלגוריתם 1 השתמשתי בפירמידה בעלת 5 רמות כיוון שהתוצאות הכי טובות התקבלו מכך.
באלגוריתם 2 בחרתי להשתמש בפירמידת לפלסיאן בעלת 6 רמות (2 רמות נלקחו מהתמונה הראשונה בתחילתה (התדרים הגבוהים) ו-4 הרמות הנמוכות נלקחו מהתמונה השנייה (התדרים הנמוכים)).
2. בחרתי להשתמש בקרנל [[1, 2, 1]] בטשטוש לפני הקטנה כיוון שקרנל זה מתחשב בפיקסלים השכנים הקרובים ביותר, כיוון שלא הייתי מעוניין שהתפירה תהיה בלי רעש של השכנים שרחוקים יותר (ספציפית לתמונות שהשתמשתי בהן).
3. כדי להתמודד עם תמונות צבעוניות, בחרתי לפצל את ערוצי הצבע של כל תמונה ולבצע את האלגוריתם באופן בלתי תלוי על כל צבע בנפרד.

הבדלים: ההבדל הראשון בין שני האלגוריתמים נובע מהשוני בשימוש שעשיתי בפירמידת הלפלסיאן בכל אחד מהם. במקרה הראשון רצינו לבצע $blinding$ בין שתי תמונות, כך שהמרחב המשותף שלהם יהיה רק התפר ביניהן ולכן לקחנו מכל אחת את כל פירמידת הלפלסיאן שלה (בכל מקום שאינו התפר). במקום התפירה חיברנו בין כל הרמות של פירמידת הלפלסיאן של שתי התמונות בהתאם למסכה שאיתה ביצענו החלקה ומעבר נקי ביניהן. לעומת זאת, באלגוריתם השני, שתי התמונות הופיעו בכל מרחב התמונה המשולבת ולכן בחרתי לקחת תדרים גבוהים מהתמונה הראשונה ותדרים נמוכים מהתמונה השנייה (דבר היוצר את האפקט המתבקש) השתמשתי ברמות שונות של הלפלסיאן עבור כל אחת מהתמונות. (בהתאם למה שרצינו עבור התמונה שנראה מרחוק ומה מקרוב).
בנוסף, באלגוריתם הראשון השתמשתי בצבע כיוון ששילוב התמונות בצבע משתלב באופן חלק ויוצר תוצאה מרשימה יותר לעומת האלגוריתם השני בו השימוש שלי בצבע איבד את האפקט אותו ניסיתי להשיג בגלל השוני בדרגות הצבע בכל אחת משתי התמונות.

תוצאות:

אלגוריתם 1:

תמונות מקור + מסכה

שילוב של תמונת יער עם תמונה של אסטרונאוט על הירח

אלגוריתם 2:

תמונות מקור

מרחוק רואים את אנג ואילו מקרוב רואים את קורה

תוצאות לא מוצלחות:

באיחוד זה השתמשתי בכל רמות הפירמידה עד לרמה של רמה 1 x 1.
ניתן לראות כי החליפה של האסטרונאוט ירוקה.
הסבר אחד לכך הוא ששימוש בכל רמות הפירמידות עלול להגביר רעש, וליצור דגש על התדרים הנמוכים בין התמונות (הרקע הירוק), כפי שניתן לראות על התמונה של החליפה של האסטרונאוט בה אינם רואים טוב את החדות והצבעים של חליפת האסטרונאוט.

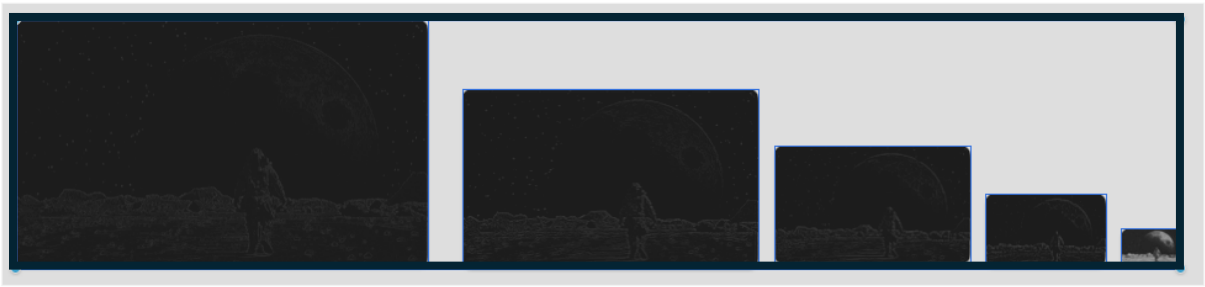
באיחוד זה נלקחו 3 הרמות הראשונות של הפירמידה מהתמונה של קורה ואילו 3 הרמות האחרונות מהפירמידה מהתמונה של אנג.
ניתן שלא רואים את התמונה של אנג (התמונה עם התדרים הנמוכים).
הסבר לכך הוא שלקחנו יותר מדי תדרים נמוכים גם מהתמונה הראשונה (קורה) כך שהיא ממלאת גם את הרקע של התמונה המשולבת.

הצגת פירמידות הגאוסיין והלפלסיאן :

Gaussian Pyramid

בפירמידת הגאוסיין כל רמה קטנה פי 2 (בשורות ובעמודות), וכי ניתן להבחין בתמונות כי שככל שהרמה יורדת כך התמונה נהיית פחות חדה ומטושטשת. דבר זה נובע מכך שככל שיורדים ברמות, התדרים הגבוהים של התמונה האחראים על קווי המתאר "נאבדים" ולכן רואים בעיקר את הרקע הכללי של התמונה.

Laplacian Pyramid



פירמידת הלפלסיאן מייצגת את איבוד המידע עבור כל רמה של פירמידת הגאוסיין. לכן, ניתן לראות את איבוד המידע מכל שלב בפירמידה בלפלסיאן. (מעצם הגדרתו של הלפלסיאן - $G_n - expand(G_n)$).
כפי שניתן לראות בתמונות המרכיבות את הפירמידה אנו רואים בעיקר את קווי המתאר של התמונה, זאת כיוון שבכל ירידה ברמה נעלמים בעיקר התדרים הגבוהים האחראים על החדות וקווי המתאר של הרמה שמעליה.
בנוסף, עבור הרמה האחרונה של הפירמידה ניתן לראות שהיא זהה לרמה הנמוכה של פירמידת הגאוסיין שהצגנו, זאת מכיוון שרמה זו אינה מופחתת מתמונה אחרת, שכן $G_{-1} = L_{-1}$ (כאשר -1 היא הרמה האחרונה).

מסקנות :

בתרגיל זה התנסיתי לעומק בשימוש בפירמידות ובפרט בשימוש בפירמידת לפלסיאן. בשני חלקי התרגיל השתמשתי בלפלסיאן לשימושים שונים. אחד כדי למזג בין שני תמונות על ידי שימוש בפירמידת לפלסיאן של שתי התמונות ומשקול ביניהם באמצעות מסכה, ובשני, שימוש ב"פירוק" של התדרים הגבוהים והנמוכים המתקבלים בפירמידת הלפלסיאן על מנת לאחד בין שתי תמונות.
הבנה נוספת מהתרגיל היא שלא בהכרח נרצה להשתמש בפרמידת לפלסיאן המכילה את כל הרמות אלא נרצה להשתמש כל פעם בכמות רמות שונה שתביא לתוצאה טובה ביותר לפי חשיבה על מרכיב התמונות ולפי הבדיקות שנבצע.
בחרתי בתרגיל זה לממש את מירב הפוני בעצמי ולא להשתמש בספרייה חיצונית, הרגשתי שזה העמיק את רמת ההבנה של החומר הנלמד בכיתה וכי אנסה לשלב זאת גם בהמשך.
בנוסף, לראשונה יצא לי לעבוד עם תמונות בצבע ובפירוק התמונה לערוצי צבע, דבר שלא יצא לי לעסוק בו עד כה.