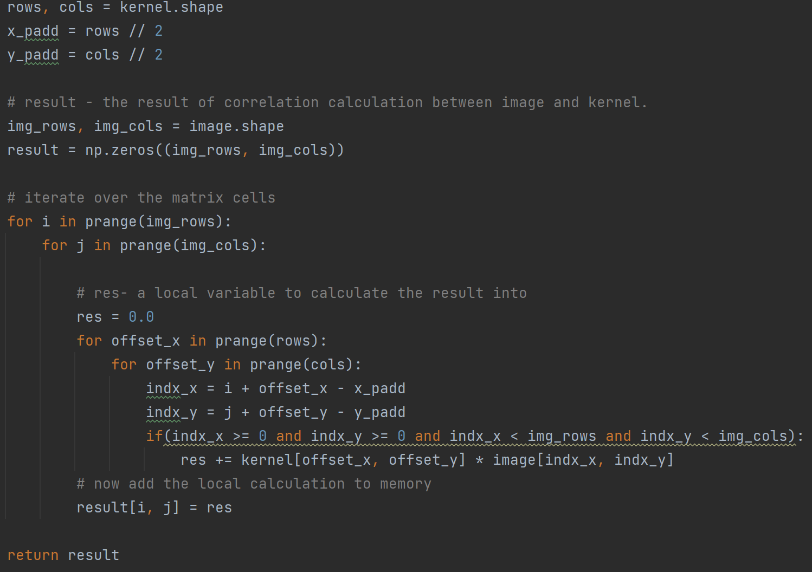
**236370 תכנות מקבילי ומבוזר לעיבוד נתונים ולמידת מכונה**

**דוח תרגיל בית 2**

מרינה ינובסקי 324515659

קורן מועברי 207987314

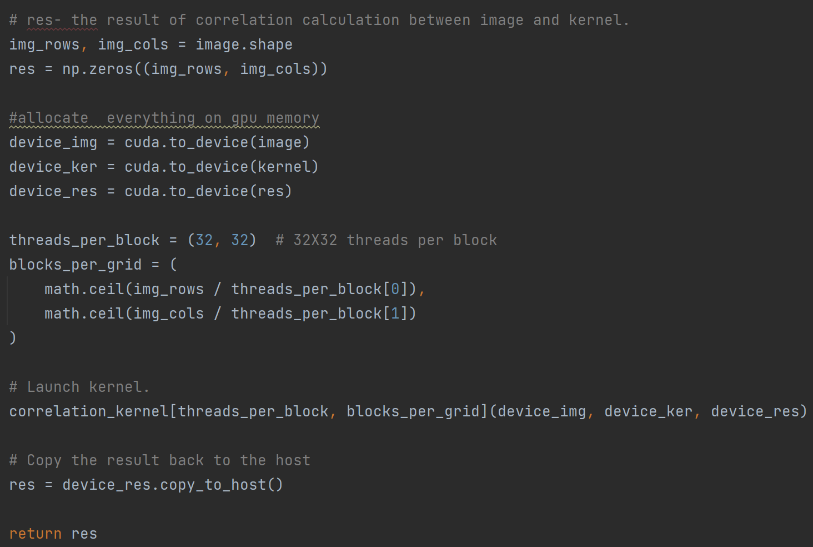
**שאלה 7**

מימוש עם NUMBA:

1. נשמור את מימדי התמונה והקרנל, וכן את הPADDING שמייצג את הגבולות של הקרנל יחסית למרכזו.
2. כעת, באיטרציות על מימדי התמונה (שהם מימדי הפלט), כך שi,j הינו התא שמייצג את הפיקסל עליו עובדים (פלט):
   1. נגדיר משתנה לוקלי אליו נשמור את החישוב כדי להפחית כתיבות לזיכרון
   2. באיטרציות על מימדי הקרנל: נוודא שהאינדקס איתו מחשבים מכפלה בתמונה חוקי. אם כן- נוסיף את המכפלה שתתקבל לסכימה.

* האינדקס בתמונה איתו מחשבים מכפלה = אינדקס התא של התוצאה + padding + האינדקס המתאים של התא בקרנל איתו נכפיל.
* דילוג על הכפלה של ערכים לא חוקיים, שקול להכפלה ב0 והוספה לתוצאה, ולכן אין טעם לבצע חישוב זה.

בנוסף, בחרנו לבצע איטרציות עם prange כדי שניתן יהיה לבצע חישוב מקבילי בהוספת דגל parallel=true.

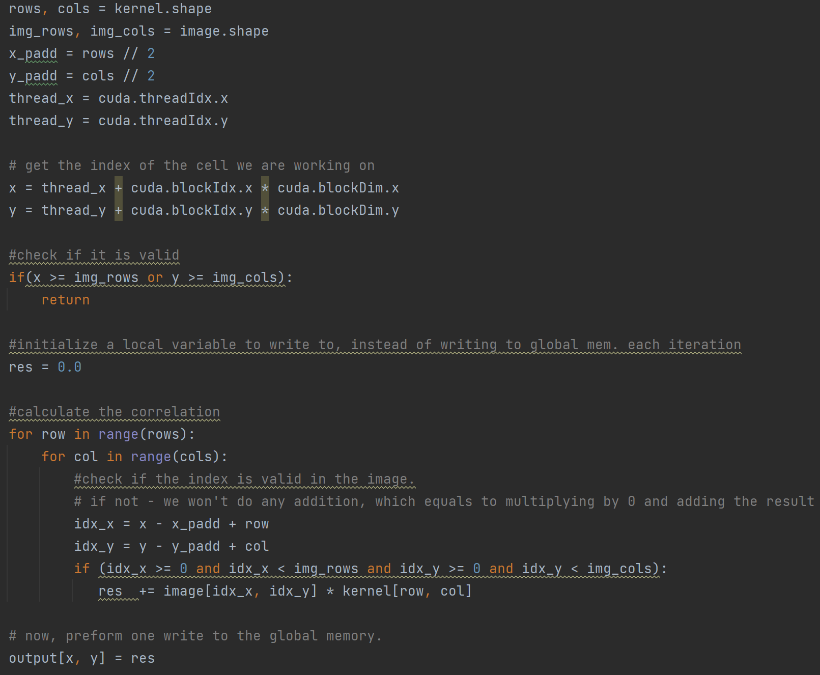
מימוש עם CUDA:

נעתיק לdevice את הפרמטרים המתאימים, כולל את המטריצה של הפלט.

מכיוון שאין הנחה על גודל התמונה, נרצה להקצות את הthreads ואת הבלוקים בהתאם. נגדיר גודל דו מימדי מגודל לכל היותר של 32\*32 כמימד החוטים בבלוק (סה״כ לכל היותר 1024 חוטים, כפולה של 32 שהוא גודל של warp וכן מביא לנצילות של חומרה כי עדיף להריץ כמה שיותר חוטים בבלוק, במקרה שלנו- כמות מקסימלית, כדי לנצל אותו באופן המיטבי). הגדרנו באופן דו מימדי מטעמי נוחות בחישוב (ניתן לקבל ערכי x ו-y בחוטים עליהם נעשה חישוב לאינדקסים).

כעת נגדיר גם כן כמות בלוקים בגודל דו מימדי. נרצה שהכמות הזו תתאים בדיוק לחלוקה של התמונה לתתי בלוקים בגודל 32\*32, כדי לחשב את התוצאה עבור כל התאים בתמונה.

נקרא לקרנל עם המימדים שהגדרנו, ונחזיר את התוצאה שהוא חישב לאחר שהעתקנו אותה חזרה לhost.

מימוש הקרנל:

נשמור את מימדי הקלטים, וכן את הpadding ששימושו הוסבר במימוש הקודם.

נחשב את המיקום המוחלט של תא הפלט לפי מיקום החוט בבלוק ולפי מיקום הבלוק בgrid.

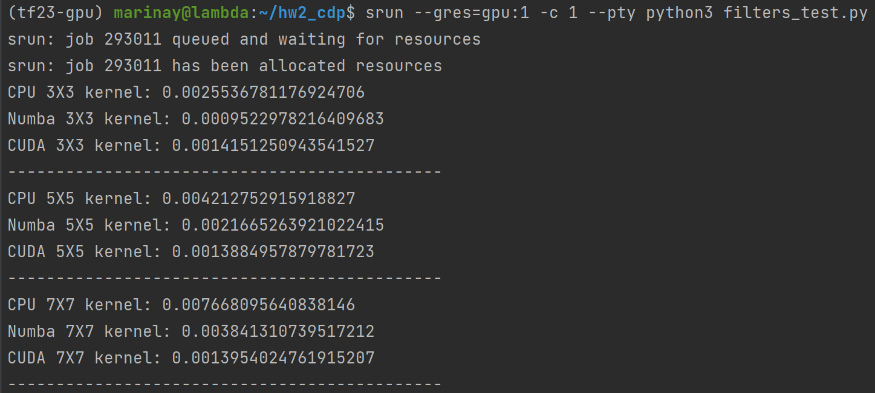
נוודא שהמיקום הנ"ל לא חורג מגודל המימדים המותר.

נאתחל משתנה מקומי אליו נכתוב תוצאות ביניים, וכך נחסוך גישות לזיכרון.

כעת, באיטרציות על מימדי kernel: נוודא שהאינדקס איתו מחשבים מכפלה בתמונה חוקי. אם כן- נוסיף את המכפלה שתתקבל לסכימה

הערות באופן דומה למימוש הקודם:

* האינדקס בתמונה איתו מחשבים מכפלה = אינדקס התא של התוצאה + padding + האינדקס המתאים של התא בקרנל איתו נכפיל.
* דילוג על הכפלה של ערכים לא חוקיים, שקול להכפלה ב0 והוספה לתוצאה, ולכן אין טעם לבצע חישוב זה.

**שאלה 8**

התמונה עליה מתבצע החישוב בגודל 215\*215.

תחילה, הCPU מבצע הרצה רגילה של קוד פייתון: כל שורה מתורגמת ע"י הinterpreter, ומורצת. יהיו 215\*215\* (size\_of\_kernel) איטרציות סדרתיות. לכן ניתן לראות בבירור שזמן הביצוע על הCPU בכל ההרצות הוא הכי איטי, מכיוון שהוא מבצע חישובים רבים באופן סדרתי . לכן נתמקד בהשוואה בין הGPU לNUMBA שמבצעים הרצה מקבילית, דבר שישפר את זמן ההרצה ביחס לCPU.

עבור הקרנל בגודל 3\*3:

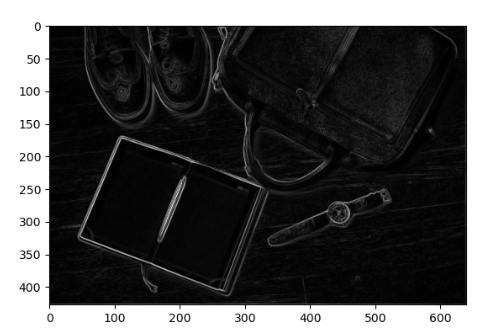
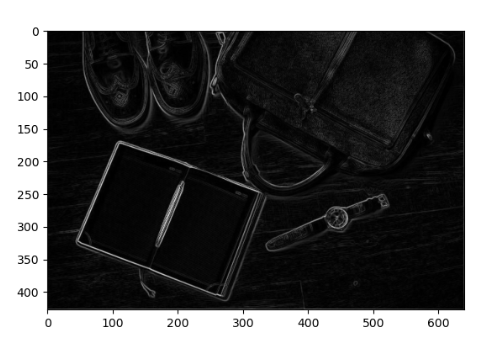
ניתן לראות כי NUMBA, שמבצע הרצה מקבילית על הCPU מהיר יותר מאשר הרצה על הGPU. זאת מכיוון שיש תקורה על העתקת כל המידע לGPU וממנו, וההעתקה זו לא משתלמת ביחס לכמות החישובים הנדרשים לביצוע. בנוסף באיטרציה הראשונה הקוד עובר קמפול (JIT), דבר שמאיץ את החישוב באיטרציות הבאות ולכן נקבל ביצוע מהיר יותר ביחס לCPU הסדרתי האיטי.

עבור שאר הקרנלים: הGPU מהיר יותר מכיוון שכל החישובים בו מתבצעים בצורה מקבילית על כל הCORES שלו, לעומת הרצה מקבילית על הCPU בה ניתן להשתמש בכמות מוגבלת של CORES עליהם מריצים במקביל (הרצה שתתאפשר עם דגל parallel=true עם NUMBA). לפיכך, למרות שאנחנו משלמים מחיר על העתקת המידע לGPU וממנו, אנחנו מקבלים שיפור בביצועים כי יש חישובים רבים שנדרש לבצע, והחישובים מבוצעים במקביליות רבה (החישוב הינו compute bound ולכן הGPU נותן שיפור בביצועים).

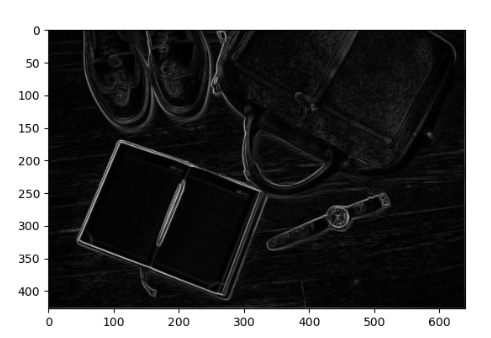
**שאלה 9**

נעדיף להשתמש בnjit כאשר יש מעט חישובים ביחס להעתקת המידע שיש לבצע לGPU ובחזרה (memory bound), ונשתמש בcuda כאשר יש חישובים רבים שיש לבצע, ואנחנו נרוויח מהמקבול על הGPU ולא נפסיד על ההעתקה (compute bound).

**שאלה 10**

הרצה עם הקרנל הנתון: הרצה עם קרנל 1:

A notebook and a watch on a wooden floor

Description automatically generatedהרצה עם קרנל 2: הרצה עם קרנל 3:

**קרנל 1:**

מבנה הפילטר: הפילטר מוגבר על ידי משקלים, כאשר השורה המרכזית מכילה מקדמים גדולים יותר (10 ו 10-) כדי להדגיש את הציר האופקי.

השפעה על התוצאות: קצוות אופקיים מזוהים בצורה בולטת יותר, במיוחד במקומות שבהם יש שינויים חזקים בעוצמה, וקצוות עם הבדלים עדינים מועצמים בשל המשקלים החזקים.

**קרנל 2:**

מבנה הפילטר: הפילטר הזה מאריך את הגובה (המימד האנכי) וכך הוא מכסה יותר שורות.

השפעה על התוצאות: קצוות אנכיים מזוהים יותר, והוא רגיש למעברים על פני מספר שורות במקום שינויים חדים ומקומיים.

**קרנל 3:**

מבנה הפילטר: הפילטר ממצע אחיד עם דגש קל על הפיקסל המרכזי.

השפעה על התוצאות: הפילטר הזה לא מזהה קצוות אלא מבצע סוג של טשטוש או החלקה, מה שגורם לתמונה להיראות רכה יותר. זה קורה כי הוא מחשב ממוצע של עוצמות הפיקסלים, מה שמפחית רעשים ומטשטש את הקצוות.

**הפילטר המקורי:**

מבנה הפילטר: מזהה קצוות על ידי הדגשת גרדיאנטים בכיוונים אופקיים ואנכיים.

השפעה על התוצאות: מפיק תוצאה מאוזנת לקצוות גם בכיוון האופקי וגם בכיוון האנכי.