

תרגיל 5

הנחיות הגשה

1. העבודה היא ביחידים.
2. ההגשה היא עד ליום ראשון, בתאריך 17.01.21 בשעה 23:30.
3. יש להגיש את הקובץ myfunction.c בלבד. הגשת התרגיל תתבצע באמצעות submit. עזרה ניתן למצוא באתר:

<https://support.esc.biu.ac.il/submit/>

כדי למנוע בעיות, עדיף להגיש את הקובץ ישירות מחשבון הלינוקס שלכם.

שימו לב: העברת הקבצים דרך Windows יכולה לגרום לכך שלא יתקמפלו. במקרה זה **הציון יהיה 0** ללא זכות לערעור.

4. בשורה הראשונה (!) של כל קובץ אותו אתם מגישים, יש לרשום בהערה מס' ת.ז. ושם מלא. לדוג':

// 123456789 Eyal Cohen

5. בהצלחה ☺

רקע כללי

בתרגיל זה נתון לכם קטע קוד של מתכנת אחר ומטרתכם היא לייעל את זמן הריצה שלו ככל הניתן בעזרת כל הכלים והעקרונות אשר למדתם בקורס. המטרה היא להגיע לזמן ריצה של התוכנית מהיר ככל האפשר (ובכך לקבל את מירב הנקודות בתרגיל).

שימו לב: הקובץ היחיד אשר אתם רשאים לשנות הוא myfunction.c בלבד.

התקנת סביבת העבודה

בשורת הפקודה יש לרשום את הפקודה הבאה:

```
sudo apt-get install freeglut3-dev libxmu-dev libxi-dev
```

הקוד

בקובץ myfunction.c, הפונקציה myfunction מקבלת תמונה וערך בחירה נוסף.

1. אם הערך שמתקבל הוא 1, הפונקציה מבצעת החלקה ("טשטוש") של התמונה ולאחר מכן מבצעת חידוד (שיפור) של התמונה המוחלקת. גם התמונה לאחר ההחלקה וגם התמונה לאחר החידוד נשמרות בתור קבצי bmp. זה מתבצע ע"י הפונקציה writeBMP שבקובץ writeBMP.c (קוד שאותו לא משנים).
2. אם מתקבל ערך שונה מ-1, הפונקציה מבצעת החלקה ("טשטוש") עם פילטר של התמונה ולאחר מכן מבצעת חידוד (שיפור) של התמונה המוחלקת.
3. אם לא יתקבל ערך נוסף לאחר התמונה, הפונקציה תזרוק הודעת שגיאה ותסיים את התוכנית.

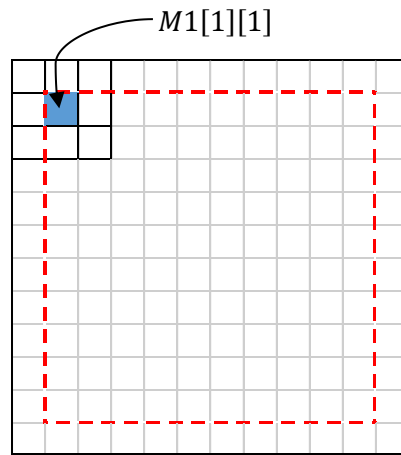
ייצוג תמונה במחשב

תמונה במחשב מיוצגת כמטריצה, שבה בכל תא שמור ערך הפיקסל המתאים. דרך נפוצה לייצוג תמונה צבעונית במחשב היא שיטת RGB: בכל תא שומרים 3 ערכים: Red, Green, Blue (ייצוג RGB). כלומר, כל צבע מיוצג ע"י שילוב של 3 צבעים- אדום, ירוק וכחול (נקראים גם "ערוצים").

החלקה וחיידוד של תמונה (פעולת קונבולוציה)

פעולת "החלקה" (smooth) של תמונה היא החלפת כל פיקסל בממוצע של כל הפיקסלים שמסביבו בתוך "חלון" בגודל 3×3 שבמרכזו נמצא הפיקסל. פעולת "החלקה" עבור תמונת RGB היא ביצוע החלקה עבור ערכי R בנפרד, ערכי G בנפרד וערכי B בנפרד. למשל, עבור ערוץ מסוים (אחד מ-R, G או B), בהינתן תמונה M1 בגודל $N \times N$, הפיקסל במקום ה-[1,1] יקבל את הערך הבא:

$$1. \quad M2[1][1] = \frac{\sum_{i=0}^2 \sum_{j=0}^2 M1[i][j]}{9}$$



כאשר M2 זו התמונה לאחר ה"החלקה". שימו לב שערכי הפיקסלים השכנים נלקחים מהתמונה המקורית.

נרצה להתייחס אך ורק לפיקסלים שאפשר "להניח" עליהם את החלון, כלומר פיקסלים שיכולים להימצא במרכז חלון שכזה בגודל 3×3 . פיקסלים אלו הם הפיקסלים שבתוך התחום המסומן באדום. כלומר, אנחנו מתייחסים אך ורק לפיקסלים עם אינדקס שורה i המקיים $\frac{kSize}{2} \leq i < N - \frac{kSize}{2}$ ועם אינדקס עמודה j המקיים $\frac{kSize}{2} \leq j < N - \frac{kSize}{2}$, כאשר $kSize$ הוא גודל החלון (במקרה שלנו, $kSize = 3$).

לאחר סיום המעבר עם החלון על התמונה המקורית, התמונה המתקבלת נראית יותר מטושטשת, ולכן הפעולה נקראת "החלקה". ניקח מטריצה K בגודל 3×3 (נקרא לה "מסיכה", או kernel) ונעביר אותה על הפיקסלים של התמונה, כך שבכל פעם פיקסל אחר של התמונה המקורית יהיה במרכז המסיכה – כמו הפיקסל $M1[1][1]$ באיור הנ"ל. אם K היא מהצורה:

$$K = \frac{1}{9} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

אז נשים לב שנוסחא 1 הנ"ל שקולה לנוסחא הבאה:

$$M2[1][1] = \sum_{i=0}^2 \sum_{j=0}^2 M1[i][j] \cdot K[i][j]$$

למעשה, אפשר להכליל את הפעולה הנ"ל ולא להשתמש אך ורק בחישוב ממוצע, אלא לשנות את הערכים ב- K (נקראים גם "משקלים") ובכך לבצע סכום ממושקל אחר עבור הפיקסל שיהיה במרכז המסיכה. כך נוכל פעולות שונות על תמונה נתונה. למשל, עבור K מהצורה

$$K = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

מתקבלת תמונה "חדה" יותר. העברת מסיכה על תמונה וחישון⁸ הסכום הממושקל (בהתאם לערכים שבמסיכה) עבור כל פיקסל, נקראת פעולת קונבולוציה.

קימפול הקוד

לתוכנית מצורף makefile. כמובן שאין צורך לשנות אותו מכיוון שבבדיקה ייעשה שימוש ב-makefile המקורי. כרגיל, כדי למקפל את התוכנית יש להשתמש בפקודה make מהטרמינל.

הרצת התוכנית

לאחר קימפול תקין של התוכנית נוצר קובץ ריצה בשם showBMP. הרצת התוכנית מתבצעת בצורה הבאה:

```
./showBMP <imageName> <kernelOption>
```

לדוגמא:

```
./showBMP gibson_500.bmp 1
```

או

```
./showBMP gibson_500.bmp 24323
```

פקודות אלו יריצו את התוכנית עבור קובץ התמונה gibson_500.bmp אשר התקבל בקלט ויפעילו סוג קונבולוציה על בסיס הערך הנוסף שהתקבל כקלט.

שימו לב:

אתם יכולים להניח שקובץ התמונה שיתקבל ל-myfunction תמיד יכיל תמונה ריבועית (כלומר, מגודל $n \times n$) אך הקוד שלכם יבדק גם על קבצי תמונה בעלי n שונה (!)

הגשת התוכנית

השינויים שתבצעו בקוד ישפיעו על האופן בו מתבצעת הפעולה, אבל לא על התוצאה שלה – שימו לב שהקוד שלכם והקוד המקורי יוצרים את אותה התמונה.

מעבר להתבוננות בשתי התמונות, תוכלו להיעזר בעובדה שהרצת התוכנית כותבת לקובץ את התמונות שנוצרות. בתיקיית הקבצים הנלווים לתרגיל (ex5_files.zip שבאתר) תוכלו למצוא את קבצי התמונה שנוצרו עבור הפעלת הקוד המקורי של myfunction: Blur_correct.bmp עבור התמונה המוחלקת ו- Sharpen_correct.bmp עבור התמונה לאחר החידוד. היעזרו בפקודה cmp כדי להשוות בין הקבצים הללו לבין הקבצים שיצור הקוד שלכם.

הגשת התוכנית

יש להגיש את הקובץ myfunction.c בלבד. אין צורך להגיש את שאר הקבצים מכיוון שהקוד שלכם יקומפל עם תוכנית הבדיקה.

ציון התרגיל

בתרגיל זה נתונות שגיאות מכוונות אשר אתם מצופים לשפר אותן. אולם, פרמטר נוסף לקביעת הציון הסופי במטלה זו הוא זמן הריצה שלכם, כך שמומלץ מאד להמשיך ולשפר את הביצועים של התוכניות על מנת להבטיח את מקומכם הגבוה.

הערות

1. בתרגיל זה ייבדקו קלטים בגדלים שונים מהקלט הנתון.
2. יש צורך בהערות משמעותיות (להסביר את כיצד נעשה השיפור לקוד).
3. התרגילים ייבדקו על u2. הגשת התרגיל תחזיר פידבק שכולל זמני ריצה של 20 הרצות של התוכנית שלכם על התמונה המצורפת ב-ex5_files.zip (התעלמו מהציון שיתלווה לפידבק).

בהצלחה!