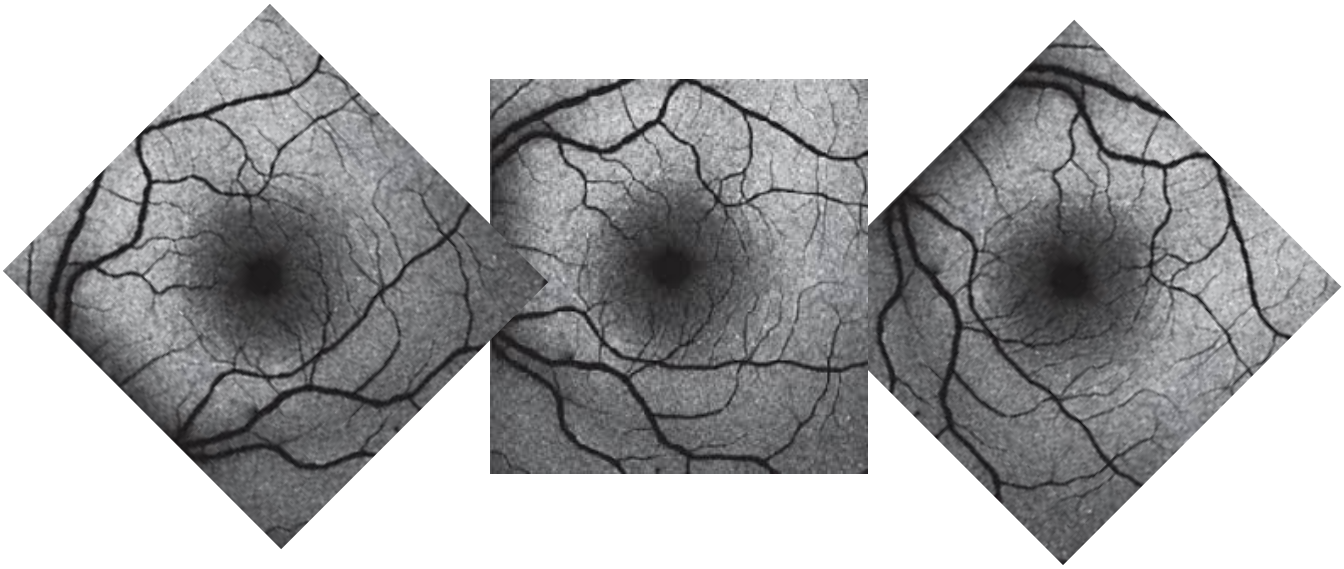
עבודה מספר 2 בקורס עיבוד תמונות רפואיות 67705

נושא: רגיסטרציה

פרופ' לאו יוסקוביץ - סמסטר א' תשע"ח

מתרגל: שסקין עדי adi.szeskin@mail.huji.ac.il



כללי

1. הגשת התרגיל היא ביחידים.
2. נא להגיש עד לתאריך: 24.12.18
3. בכל הסעיפים יש להניח כי שתי התמונות אשר עוברות רגיסטרציה הן באותו גודל בפיקסלים וכל פיקסל בעל אותם מימדים בכל הצירים.
4. על מנת לאפשר תרגול יעיל של החומר הנלמד סופקו לכם פונקציות עזר בקובץ utils.py.
5. את קוד ההרצה שלכם יש לכתוב אך ורק תחת הקבצים הנקראים Ex2_part[#].py .
6. יש להקפיד על הגשה מסודרת (10% מהציון בתרגיל): כותרות לגרפים, תיאורים ויחידות לצירים וגדלים נכונים בסקאלות הצירים.
7. עליכם להגיש דו"ח ובו תשובות לשאלות המילוליות ו/או עם תמונות בסעיפים בהם התבקשתם לכך. לנוחותכם, שאלות מילוליות וסעיפים בהם נדרש צירוף תמונה מסומנים בקו תחתון. בנוסף, עליכם להגיש את הקוד שכתבתם.
8. מבנה ההגשה: לכל אחד משני החלקים יש להגיש קובץ zip אחד. שם הקובץ יהיה <id>_ex2_part#.zip כאשר # הוא מספר החלק ו-<id> הוא תעודת הזהות כולל ספרת ביקורת. בתוך כל קובץ יימצא הדו"ח תחת השם Ex2_part[#].pdf. הדו"ח יכיל את שם המגיש , login ות"ז.
9. את הקבצים יש להעלות דרך המודל.

10. 10% מהניקוד מבוסס על הגשה מסודרת, לרבות קוד קריא ומתועד.

11. כל יום איחור יתוגמל בהורדה של 10 נקודות.

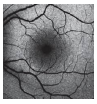
מבוא

באופן כללי, בין שתי סריקות של אותו פציינט בזמנים שונים אין התאמה טובה. בין שני מועדי הסריקה הפציינט נשם, זז, ואולי גם בילה כמה חודשים בבית. למרות זאת, לעיתים קרובות נרצה שהתמונות יהיו בהתאמה מושלמת, לצרכי השוואה ומעקב, או לטובת טיפול מונחה הדמייה (הקרנה, החדרת מחט) בנקודה הנכונה ע"פ תוכנית טיפול שגובשה על גבי תמונת הסריקה הראשונה. אם האיבר שמעניין אותנו הוא קשיח, כמו המוח לדוגמא, אזי ניתן למצוא את התנועה הזו בעזרת רגיסטרציה ריגידית.

רגיסטרציה ריגידית היא תהליך של מציאת סיבוב והזזה בין זוג תמונות. לפיכך בתמונות תלת מימדיות, כמו MRI או CT למשל, רגיסטרציה ריגידית מאופיינת בשישה פרמטרים: שלושה פרמטרי הזזה, ושלוש זוויות סיבוב. סט של הזזות וסיבובים כאלו נקרא "טרנספורמציה". בתהליך רגיסטרציה אנו מחפשים את הטרנספורמציה בין שתי תמונות, ולעיתים אנו נדרשים גם להפעיל את הטרנספורמציה על אחת התמונות על מנת ליישר אותה כלפי התמונה השנייה.

אלגוריתמים למציאת רגיסטרציה ריגידית מתחלקים באופן כללי לשני סוגים: מבוססי תמונה, או מבוססי נקודות עניין. באלגוריתמים מבוססי נקודות עניין אנו תחילה מזהים אזורים קטנים בשתי התמונות שמתאימים האחד לשני, כלומר מייצגים כנראה את אותה נקודה באיבר הנסרק. ברגע שיש לנו מספיק התאמות כאלה ניתן לחשב את הטרנספורמציה.
באלגוריתמים מבוססי תמונה עושים את ההיפך: תחילה מנחשים טרנספורמציות כלשהן, מפעילים אותן על התמונות (כלומר מיישרים תמונה אחת כלפי השנייה), ואז מודדים את הדמיון בין התמונות אחרי היישור, ובוחרים את הטרנספורמציה בעלת הדמיון החזק ביותר. לעיתים קרובות אלגוריתמים כאלו הם איטרטיביים, כלומר הם משתמשים בטרנספורמציה טובה על מנת למצוא טרנספורמציה טובה יותר וחוזר חלילה.

חלק 1. אלגוריתמים מבוססי נקודות עניין ומבוססי תמונה

מטלה א' 60% מכלל התרגיל: בחלק זה נמצא רגיסטרציה ריגידית בין שתי תמונות אופטיות **דו ממדיות** כאשר בתור תמונות דו-מימדיות בחרנו בסוג חדש של תמונות שעדיין לא נתקלתם בהם שנקראות FAF - Fundus Autofluorescence  .
למעוניינים יש חומר נוסף כאן. ספיצפית המקרים הם של חולים במחלה הנקראית age related macular degeneration **-**AMD.

רקע: אלגוריתמים מבוססי נקודות עניין מוצאים רגיסטרציה בין שתי תמונות במספר שלבים אוטומטיים:

- זיהוי נקודות עניין.
- מציאת התאמות (pairing) בין נקודות עניין בשתי התמונות.
- סינון נקודות outliers.
- חישוב הרגיסטרציה מתוך ההתאמות הנכונות.

בתרגיל זה נניח שנקודות העניין וההתאמות ביניהן בשתי התמונות כבר נבחרו (ושהן עלולות לכלול outliers) ונכתוב פונקציה אשר משתמשת בהן כדי לחשב את הרגיסטרציה הריגידית בין שתי התמונות. נשתמש בסכמת RANSAC כדי לסנן את ה-Outliers בחישוב הטרנספורמציה ולסמן מי הם ה-inlier ומי הם ה-outliers.

אנא הורידו מ: https://drive.google.com/open?id=1qAL93qYRRuR6cFAtFdJVNF_OrCA_WqD0

את הקבצים utils.py ואת התמונות משם.

1. 5% טענו את התמונות BL01.tif,FU01.tif והשתמשו בפונקציה

```
[BLPoints, FUPoints] = getPoints('no_outliers')
```

שתחזיר נקודות דו מימדיות במערכת צירי התמונה בתור שתי מטריצות בעלות N שורות ושתי עמודות כל אחת, כך שההתאמות הן בין שורות מתאימות בשתי המטריצות. הציגו את ההתאמות בעזרת צילום מסך של subplot עם שתי התמונות בעזרת פונקציית עזר שתכתבו. היעזרו בפונקציה annotate על מנת להציג את מס' הנקודה ליד סימן המציין את מיקומה. מהו מספר ההתאמות השגויות (כלומר, שנראה בבירור מהתמונות שצמד הנקודות לא מתאר את אותו מבנה אנטומי בתמונה)?

2. 20% ממשו את

```
def calcPointBasedReg(BLPoints, FUPoints)
```

המחזירה את פרמטרי הרגיסטרציה מתוך התאמות נתונות. rigidReg היא מטריצה 3x3 שנחזיר. הערכים יהיו כאלה שאם מפעילים את הטרנספורמציה על FUPoints, מקבלים נקודות קרובות ככל האפשר לנקודות BLPoints במובן Least-Squares. העזרו בהסבר כאן על מנת לבצע את החישוב ע"י SVD. התוצאה שלכם צריכה לקיים בקירוב:

```
[FUPoints ones(N,1)] * rigidReg ==
[BLPoints ones(N,1)]
```

כאשר N הוא מספר הנקודות. שימו לב שמבנה המטריצה rigidReg מורכב ממטריצת סיבוב עם וקטור הזזה:

```
[R(2x2) 0;
T(1x2) 1]
```

3. 5% ממשו את

```
def calcDist(BLPoints, FUPoints, rigidReg)
```

המחשבת ומחזירה את ה**מרחק של כל נקודה שעברה טרנספורמציה מהנקודה אליה היא אמורה להתאים בפיקסלים. חשבו את ה Root Mean Square Error -** מדד השגיאה של הרגיסטרציה מתוך הפלט של הפונקציה שכתבתם (הפונקציה מחזירה וקטור באורך N, אך בסעיף זה יש לחשב מוקטור זה גם ערך סקלרי יחיד ביחידות של פיקסלים). מהו ערך השגיאה בפיקסלים עבור ההתאמות שנטענו בסעיף א'?

4. 30% ממשו פונקציה שטוענת את שתי התמונות ואת הנקודות שבחרתם, מחשבת את הרגיסטרציה ביניהן, מפעילה את הטרנספורמציה על התמונה FU ומציגה אותן זו על גבי זו באופן שיאפשר לראות שכעת הן נמצאות ברגיסטרציה זו ביחס לזו. רמז: ניתן להציג את אחת התמונות עם edges בלבד, ו\או כל אחת בצבע שונה. צרפו לדו"ח את שתי התמונות ברגיסטרציה.

5. 5% כעת טענו את

```
[BLPoints, FUPoints] = getPoints('with_outliers')
```

וחזרו על סעיפים 1 ו-4. האם התמונות ברגיסטרציה? מהו ערך השגיאה? מדוע זה כך?

6. 30% קראו את התיעוד והדוגמאות של utils => ransac . עליכם להשתמש בה על מנת לכתוב פונקציה

```
[rigidReg, inliers] = calcRobustPointBasedReg(FUPoints, BLPoints)
```

שתצליח לחשב את הטרנספורמציה למרות הימצאות אחוז מסוים של outliers לא ידועים מראש ברשימת ההתאמות. יש לספק ל-ransac שתי פונקציות אותן היא תפעיל על הנתונים: אחת לחישוב טרנספורמציה מתוך תת-קבוצה נתונה של התאמות, ואחת לחישוב השגיאה של הטרנספורמציה על כל אחת מההתאמות המקוריות. היעזרו בפונקציות שכתבתם בסעיפים הקודמים.

7. 5% חזרו על סעיף 5 אך השתמשו בפונקציה הרובוסטית לחשוב הרגיסטרציה. הדגישו בעזרת צבעים שונים/צורות שונות אילו מהנקודות הן inliers ואילו הן outliers.

8. בדקו את עצמכם ע"י יצירת סט נקודות מלאכותיות, ביצוע טרנספורמציה ידועה עליהן לקבלת סט נוסף של נקודות, הוספת רעשים אקראיים קטנים וגם התאמות שגויות שהן outliers, ושחזור הטרנספורמציה המקורית מההתאמות בעזרת הפונקציה שכתבתם. אין צורך להגיש קוד או תשובות בסעיף זה.

פונקציות שימושיות לחלק זה:

```
from skimage import transform as tf
AffineTransform
warp
```

מטלה ב 40% מכלל התרגיל: בחלק זה, כהכנה לתרגיל הבא ממשו 2 פונקציות

```
SegmentBloodVessel(Image)
```

כאשר ניתן להניח שהכתם השחור במרכז הוא לא חלק מהסגמנטציה וניתן להוריד את הכיתוב של התאריך למטה. .

וכמו כן ממשו שיטה למצוא Features דומים מתמונה נתונה/סגמנטציה, ניתן להיעזר באלגוריתמים כגון SURF,SIFT,ORB

עבור חלק זה הוספתי עוד 2 מקרים.

FindRetinaFeatures(Image)