עיבוד תמונה – תרגיל 4

אוריין חסידים ~ oryan.hassidim ~ 319131579

# מבוא

בתרגיל זה נִדָּרֵשׁ לבצע Image Alignment – קיבלנו שתי תמונות מקבילות, שבאחת מהן הורדה הרזולוציה ובאחת מהן בוצע חיתוך ו-warping, ואנו נדרשים ליישר את החתוכה אל התמונה עם הרזולוציה הנמוכה.

נשתמש בכלים אותם למדנו בכיתה, וספיציפית בכיוון של MOPS: חילוץ פי'צרים ע"י זיהוי פינות באלגוריתם Harris, חילוץ דסקריפטורים כחלון מרזולוציה נמוכה שמיושר לפי הגרדיאנט, אלג' RANSAC למציאת התאמה של פרמטרים לדגימות, ויישור תמונה. נשתדל עד כמה שאפשר לממש את האלגוריתמים המעניינים בעצמנו ולא להשתמש במוכנים של OpenCV.

# אלגוריתם

## חילוץ פיצ'רים – זיהוי קצוות ע"פ אלג' Harris // מקבל תמונה, מחזיר רשימת קצוות

1. המרה לגווני אפור;
2. עבור גדלי חלון שונים (משתנה לפי אופי התמונה):
   1. חישוב תגובת הפינה של הפיקסלים עבור חלון בגודל זה ע"פ הנוסחה שלמדנו, ונרמול לגודל החלון:
3. עבור כל פיקסל מציאה של התגובה המקסימלית;
4. בחירת כל התגובות מעל לסף מסויים (סף יחסי ביחס לתגובה המקסימלית על התמונה);
5. סינון הפיקסלים עבורם מתקבל מקסימום לוקאלי בתגובת הפינה;
6. מציאת האינדקסים של הפינות שנמצאו.

## חישוב דסקריפטורים – מציאת חלונות מאפיינים // מקבל תמונה ורשימת נקודות (קצוות), ומחזיר רשימת חלונות מאפיינים

עבור כל פינה:

1. חישוב קירוב לגרדיאנט בפיקסל ע"פ גווני האפור של התמונה;
2. חישוב הזווית של הגרדיאנט;
3. לקיחת חלון בגודל מסביב לפינה זו;
4. סיבוב החלון כדי ליישר את הגרדיאנט כלפי מעלה;
5. לקיחת חלון בגודל ;
6. הורדת שתי רמות בפירמידה;
7. נרמול ע"פ ממוצע וסטיית תקן.

## מציאת התאמות בין דסקריפטורים // מקבל שתי רשימות חלונות מאפיינים, מחזיר רשימה של זוגות אינדקסים מתאימים

עבור כל חלון מתמונה אחת:

1. מציאת שני החלונות הכי דומים (במרחק אוקלידי) בתמונה השנייה;
2. אם היחס בין המרחקים הוא מתחת לסף שבחרנו – הכרזה על הזוג כמתאימים .
3. סינון ומיון מערכי הפינות לפי התאמות.

## RANSAC // מקבל שתי רשימות קצוות מסודרות לפי זוגות מתאימים, מחזיר מטריצת יישור מיטבית (רנדומי)

עבור מספר איטרציות:

1. הגרלה של 4 התאמות רנדולמליות;
2. חישוב מטריצת ההומוגרפיה לפי 4 ההתאמות המוגרלות;
3. בדיקה כמה מההתאמות האחרות, המרחק בין הציפייה למיושרת ע"פ מטריצת ההומוגרפיה שמצאנו קטן מסף כלשהו.

החזרת המטריצה עבורה היו התאמות רבות ביותר עם מרחק קטן מהסף.

## חישוב מטריצת הומוגרפיה מ-4 נקודות // מקבל 4 זוגות נקודות, מחזיר מטריצת יישור

אנחנו מחפשים מטריצה מהצורה:

שתתקיים עבור כל 4 הנקודות. אחרי כמה חישובים אנו מקבלים את המערכת הנ"ל ופותריים אותה.

## יִשּׁוּר פרספקטיבי של תמונה ע"פ מטריצה // מקבל תמונה ומטריצת יישור, מחזיר תמונה מיושרת

עבור כל פיקסל בתמונת המטרה:

1. מחשבים את מיקום המקור ע"פ המטריצה ההפכית של מטריצת היישור;
2. מחשבים את ערבול הצבעים של הפיקסלים ביניהם נפלנו, ע"פ משקול השטח הכלוא המרוחק מהם (בילינארי):
   1. מחשבים את כל האופציות לשילובי עיגול כלפי מעלה ועיגול כלפי מטה בכל קוארדינטה,
   2. דוגמים את הפיקסל המעוגל.
   3. משקללים לפי מכפלת המרחקים מהעיגולים ההופכיים לאלו שלקחנו.

## הדבקת שתי התמונות // מקבל שתי תמונות ומטריצת יישור, מחזיר תמונה ממוזגת

1. טעינת ערוץ השקיפות של התמונה עם הרזולוציה הגבוהה;
2. יישור שלו ע"פ מטריצת היישור שמצאנו;
3. קביעת המסכה כמעבר סף של ערוץ השקיפות המיושר;
4. סכימה בינארית ע"פ המסכה הבינארית הנ"ל של שתי התמונות: התמונה עם הרזולוציה הגבוהה, שיישרנו ע"פ היישור שמצאנו, והתמונה בעלת הרזולוציה הנמוכה.

## מציאת התאמה בין שתי תמונות // מקבל שתי תמונות, מחזיר תמונה ממוזגת בהתאמה מקסימלית (רנדומי)

1. עבור כל תמונה:
   1. טשטוש גאוסיאני;
   2. מציאת פינות;
   3. חלונות מאפיינים לכל פינה;
2. מציאת התאמות בין חלונות משני התמונות;
3. RANSAC למציאת מטריצת היישור הטובה ביותר;
4. הדבקה של התמונה המיושרת לתמונה עם הרזולוציה הנמוכה.

# פרטי מימוש

כרגיל, השתמשתי ב-NumPy וב-OpenCV, משום שהם מאוד נוחים, מאוד פופולריים, ועם דוקומנטציה נהדרת.

## חילוץ פיצ'רים – זיהוי קצוות ע"פ אלג' Harris

בחרתי להשתמש ב**קירוב** של האריס ע"י הפונקציה שתוארה לעיל באלגוריתם ולא ע"י הע"ע של מטריצת הנגזרות החלקיות מרמה שנייה, עקב היות הקירוב קל בהרבה לחישוב ועשה עבודה מעולה על התמונות שניתנו.

לתמונה הראשונה גדלי החלונות היו , לתמונה השנייה בלבד. סף לתמונה הראשונה – 0.1, ולתמונה השנייה – 0.3. מקסימום לוקאלי ע"פ חלון בגודל (רק מי שיש להם פאה או קדקד משותף).

מומש עצמאית.

אחרי המון סבבים עם תוצאות מוזרות למדי, גיליתי שהפיתרון הוא כמו בתרגילים הקודמים – המרה ל-int32🙄.

## חישוב דסקריפטורים – מציאת חלונות מאפיינים

גודל 44 לחלון לפני הסיבוב כדי שבכל מקרה אחרי הסיבוב יהיו חלון 32 מהקטע. .

הורדת רמה בפירמידה מספריית OpenCV (מימשנו ידנית כבר בתרגיל 3). כל השאר מומש ידנית.

## מציאת התאמות בין דסקריפטורים

לתמונה הראשונה סף יחס נמוך יחסית (בררן יותר) – 0.6, ולתמונה השנייה סף יחס גבוה – 0.9. בדיעבד מסתבר שבטעות לא שמתי כלל סף ראשוני ושמתי סף יחסי בלבד, מה שהביא לאי אלו התאמות תמוהות משהו, אבל ה-RANSAC פיצה על זה בכל מקרה.

מומש ידנית.

## RANSAC

לא כל פעם הייתי מבסוט מהתוצאה אז לפעמים הרצתי שוב... אז יש תמונות שזכו לפחות או יותר 10000 הרצות. סף המרחק המקסימלי המותר היה 5 לשתי התמונות. מומש ידנית.

## חישוב מטריצת הומוגרפיה מ-4 נקודות

השתמשתי בפונקציה המוכנה של OpenCV. לשים לב שהסטנדרט שלהם לגבי x ו-y הוא הפוך אז היינו צריכים להפוך לפני העבודה איתה.

## יִשּׁוּר פרספקטיבי של תמונה ע"פ מטריצה

מאחר וכבר היינו בתענוג מעין זה בתרגיל 6 של קורס אינטרו (הפתרון שלי ב-[github](https://github.com/Oryan-Hassidim/Intro2CSEx/blob/master/Ex6/cartoonify.py#L146)), השתמשתי בפונקציה המוכנה של OpenCV. בכל מקרה תיארתי לעיל את הקווים הכלליים של האלגוריתם.

## הדבקת שתי התמונות

בחרתי לעשות את המסכה כסף של ערוץ השקיפות כדי להימנע מהקו השחור בקריאת הקובץ כ-RGB רגיל. ערבול פירמידה לא עזר כאן בגלל התדר הגבוה של הקו השחור הזה. ערבול פירמידה טוב באזורים דומים. הסף הוא בערך 240 (מתוך 255). מומש ידנית.

## מציאת התאמה בין שתי תמונות

מומש ידנית באמצעות כל הנ"ל.

# תוצאות ויזואליות

## חילוץ פיצ'רים – זיהוי קצוות ע"פ אלג' Harris

תמונה שמכילה צילום מסך, אשה

התיאור נוצר באופן אוטומטי 

במבט ראשון נראה שאלו המון המון "פינות" שלאו דווקא נכונות. מהתבוננות בפאטצ'ים (להלן) נראה שהם אכן מספיק מעניינות ומצדיקות יחס, ופשוט צריך רזולוציה גבוהה מספיק כדי לשים לב לפינתיות שם.

כמו"כ, מצער הזיהוי של המסגרת של האובייקט בתמונה עם הרזולוציה הגבוהה, אבל במקום לנסות לסנן אותם ידנית, ניתן למציאת ההתאמות לטפל בזה.

## מציאת התאמות בין דסקריפטורים

דוגמאות לזוגות מתאימים:

תמונה שמכילה צבעוני, ריבוע, מלבן, צילום מסך

התיאור נוצר באופן אוטומטי תמונה שמכילה צילום מסך, ריבוע, צבעוני, מלבן

התיאור נוצר באופן אוטומטי תמונה שמכילה צילום מסך, צבעוני, ריבוע, מלבן

התיאור נוצר באופן אוטומטי   
(כל הצבעים מנורמלים, אז החלונות לא מופיעים בצבעים הללו במקור)

תמונה שמכילה שמיים, דיונות, חול, דיונה

התיאור נוצר באופן אוטומטי תמונה שמכילה דיונות, חול, בחוץ, סהרה

התיאור נוצר באופן אוטומטי

הימנית – כל ההתאמות, בשמאלית – התאמות שקצרות מ-400, שזה מה שהגיוני פה (לתצוגה בלבד). בעצם רוב הקווים המאונכים הם בגדול אלו שאנו מחפשים.

## הדבקת שתי התמונות

תמונה שמכילה טקסט, צילום מסך, צללית

התיאור נוצר באופן אוטומטי

(אין פה משהו מעניין או מרגש מדי, זו המסכה ע"פ ערוץ השקיפות המיושר)

## מציאת התאמה בין שתי תמונות

להלן תוצאות סופיות:





# מסקנות

בעבודה זו מימשנו אלגוריתמים מסורתיים להתאמת ויישור תמונות (כן אני יודע שזה נשמע ניסוח של בינה מלאכותית, מבטיח שממש כתבתי את זה בעצמי, זו באמת התחושה שלי).

אחד הדברים המעניינים הוא שאע"פ שבדיעבד גיליתי שהיו המון אי דיוקים במימוש שלי (האריס לפי קירוב ולא לפי ע"ע של מטריצת נגזרות סובל מרמה שנייה כמו במימוש של OpenCV, חישוב גרדיאנט לא לפי סובל אלא רק על 4 הפיקסלים הסמוכים (סובל לוקח 6), סף התאמה יחסי בלבד בלי סף קבוע) והמון מקומות שהיו יכולים לשפר מאוד את זמן הריצה ואת איכות ההתאמות, בעצם אלגוריתם ה-RANSAC חִיפָּה על כל אלו והצליח להביא לתוצאות מרשימות אע"פ שהקלט אליו לא היה מיטבי.

גם כאן, כדרכי בתרגילים הקודמים, היו כל מיני טעויות טכניות (טיפוסים ב-NumPy, קונבנציית x,y ב-OpenCV) שאתגרו אותנו, אבל בסוף הצלחנו להתגבר.

תודה רבה!