

Image Processing - Exercise 4

Agam Hershko, id_214193831, 214193831

מבוא

התרגיל עוסק בבעיה שבה אנו מקבלים סרטון שבו המצלמה נעה בכיוון מוגדר (לרוב אופקי אך יכול להתמודד גם עם המקרה האנכי על ידי סיבוב הפריים) ורוצים ליצור פונרמות מהפריים שקיובלנו בתרגיל בדרכים שונות. מטרת התרגיל היא ליצור של סרטונים של פונרמות משתני סוגים. הסוג הראשון הוא מכונה **viewpoint** בו מרכיבים מסווג של פונרמות שהתקבלו על ידי שינוי נקודת המבט בסרטון. הסוג השני מכונה **dynamic** בו מרכיבים מסווג של פונרמות שהתקבלו על ידי שינוי נקודת המבט בסרטון. והסרטונים בו מרכיבים מפסים דינמיים - ככלומר שהחלק הרלוונטי הסרטון משתנה לפי הזמן ואין סטטי.

הטכניקות העיקריות אותן למדנו בכיתה שבהן השתמשתי בתרגיל הん אלגוריתם SIFT על מנת למצוא את נקודות העניין בכל פריים ואת המתארים (descriptors) שלו, אלגוריתם RANSAC כדי לקבל התאמה בין נקודות עניין בפריים עוקבים וForward Warping כדי לחתך רצועות (strips) מהתמונה המקורית ולממן אותן בצורה מדויקת בתוך הקבוס של הפונרמה (השתמשתי באינטראקטיבית כדי להבטיח שהתמונה המעובדת תהיה חלקה ולא חוריות).

אלגוריתם

להלן את מקד באלגוריתם המשותף של התרגיל שהשתמשתי בו עבור שני הסוגים של הפונרמות בתרגיל.

ראשית, אנו מקבלים מהמשתמש את הנתיב של סרטון הויידאו, את סוג הפונרמה שנרצה ליצור והאם צריך לסובב את הפריים ממצב אנכי לאופקי ושומרם את הפריים של הסרטון לטור רשימה (אם יש צורך נסובב את הפריים לפני הקלט של המשתמש).

עברתי בלולאה על זוגות של פריים עוקבים כדי לחשב את מטריצת ה **descriptor** של השינוי שביניהם. השתמשתי באלגוריתם SIFT כדי למצוא את נקודות העניין והמתארים של כל פריים ולאחר מכן הפעלת סינון עבור כל אחד עומדים מעל ערך סף שאprt עליון בהמשך. הקולט של אלגוריתם זה הוא פריים בודד שכן הוא מופעל לכל אחד מזוג הפריים העוקבים והפלט הוא רישימה של **keypoints** ורישימה של descriptors עבור כל תמונה מבין השתיים. להלן תופיע הממחשה ויזואלית של מציאת נקודות העניין הסרטון Kessaria, יש לשים לב שמספר הנקודות יכול לנوع בין עשרות אלפיים לאחר הסינון תלוי בתמונה (ראה תמונה שמאלית).



לאחר מכן, נפעיל את האלגוריתם RANSAC כדי למצוא את ההתאמנה בין שני פריים סמוכים. הפלט של האלגוריתם הקודם הוא הקולט הנוכחי והפלט הוא מטריצת $\text{p} \circ \text{rig}$ (נשים לב שם קיבלנו ערך לא תקין, כלומר לא נמצא התאמות שמהן ניתן לחשב מטריצה איזומטרית את האלגוריתם שוב ושוב). הסיבה שמחשבים את מטריצת $\text{p} \circ \text{rig}$ היא ההנחה שעלייה אנחנו מtabססים שהו שינויים בתזוזה של התאמות בין פריים לפריים (translation) וסיבוב (rotation) בלבד. כלומר אנחנו לא נדרשים להתחזות עם מצב שבו היה שינוי בגודל הפריים (scaling).

מצורפת המכחה ויזואלית בתמונה הימנית בסוף העמוד הקודם להתקנת הנקודות (צבע תכלת) מתוך סרטון Shinkansen. התקנות מוצלחות בין 2 נקודות (inliers) מסומנות על ידי קו יירוק והתקנות שגויות (outliers) מסומנות על ידי קו אדום. בחרתי בכמה תמונה עם מספר מצומצם של נקודות וקווים.

לאחר שמקבלים רשימה של מטריצות $\text{p} \circ \text{rig}$ (שהן בגודל 3 על 3 ומשתמשים בהן כדי להפעיל טרנספורמציה זו מימדית), נחשב את המטריצה היחסית לתמונה הראשונה על ידי חישוב מטריצה הופכית והכפלה במטריצה האחורונה שחוושבה.

לאחר מכן, נחשב את הגודל של הקנבס ואת היחסים של הקנבס בשני הצירים על ידי הפעלת הטרנספורמציה שמייצגת על ידי המטריצה היחסית לתמונה הראשונה על פינות התמונה וחישוב הערכים המינימליים והמקסימליים בשני הצירים. התוצאה תתקבל כאשר עברו הגודל של הקנבס נחשב את ההפרש בכל אחד מהצירם והיחסים יתהפכו על ידי הכפלת הערכים המינימליים בשני הצירים במינוס אחד.

להלן האלגוריתם מתפרק לשתי פונקציות נפרדות לפי סוג הפונורמות.

אם מדובר באלגוריתם מסווג point2point , נרצה שיוציאו פונורמות שונות מהסרטון שההבדל ביניהם הוא נקודות המבט. נחשב מיקומים ייחודיים ברוחב התמונה שיעזרו לנו לחשב את גבולות הרצועות. חישוב המיקומים היחסיים יבצע על ידי חלק משתנה של הקנבס בשני הצירים (המונה הוא אינדקס שאחנו מושנים והמננה הוא קבוע שהגדכנו עבור מספר נקודות המבט) עבור x וחצי מהגובה של הפריים עבור y ולאחר מכן נפעיל טרנספורמציה על המיקומים. המיקומים היחסיים יאפשרו לנו לקחת כל פעם רצועות מחילק אחר של הפריים (מתחלים מהרצועות בחלק השמאלי ועוביים לרצועות בחלק הימני- לא ניקח רצועות שקרובות מדי לקצות על מנת לא לקבל אפקטים בלתי רצויים בסיבוב התמונה כמו שלמדנו בכיתה).

גבולות של הרצועות (נקודות ההתחלה והסיום בציר x של הרצועות בקנבס) מחושבים באופן הבא, עבור הרצועה ה i נקודת ההתחלה של הרצועה מחושבת על ידי ממוצע של של המיקום שחושב שעבור הפריים i והפריים $i+1$ ונקודות הסיום של הרצועה מחושבת על ידי ממוצע של המיקום שחושב עבור הפריים $i+1$ והפריים $i+2$.

השלב הבא הוא ליצור פסיפס מכל הפריים של הפונרומה. ניצור קנבס ריק בגודלים שחושבו ונשנה את הגבולות של הרצועות במידת הצורך. ככה שמדובר פריים תיליך רצועה ברוחב פיקסל אחד לפחות. נשתמש בפונקציית warping שכתבתי (עליה אפרט בהמשך) כדי לקבל את הרצועה המתאימה ולשים אותה במקום המתאים בקנבס לפי גבולות הרצועה וגודלו היחס של הקנבס.

אם מדובר באלגוריתם מסווג dynamic , נשתמש באופן דומה באלגוריתמים ונשתחם באותו פונקציות עזר. להלן אפרט את השינויים הרלוונטיים. במקרים לחשב מיקומים ייחודיים של רוחב התמונה, נחשב את מרכזי הפריים ונפעיל על מרכזי הפריים את הטרנספורמציות שחושבו. לאחר מכן, נחשב את גבולות הרצועות באותו אופן תוך שימוש במרכזי הפריים. מקבל פונורמות שונות על ידי יצירת פסיפס עם shift

משתנה `shifit` לגבולות הרצעות. הערך `shifit` נע בין מינוס גודל מסוים שנגדיר עד לגודל בקפיצות של חלק יחסית מהגודל (יפורט בהמשך - הגודל הוא פונקציה של רוחב הפריים). כמו כן, נחשב א-מינימלי ומקסימלי של גבולות הרצעות ונשתמש בהזה כדי להתאים את גודל הקנבס לגבול הרצעות כך שבכל חלק מהסרטון נראה את הפסיפס בכל הפריים ולא רק בחלק הרלוונטי בקנבס.

לבסוף, נשמר את הפריים של הפונרמות שקיבלנו כסרטון ונסובב את הפריים הסרטון אם יש צורך לפי קלט המשתמש.

פרטי מימוש

להלן את מקד בפרט המימוש של האלגוריתם המשותף של התרגיל שהשתמשה בו עבור שני הסוגים של הפונרמות בתרגיל.

השתמשה כאן ב `cv2` שביל פעולות מתמטיות בין היתר ובמספריה `cv` (בדיקה זה ATIICHOT למספריה `cv`) על מנת לבצע פעולות על תמונות. בחרתי בה כיון שהיא ספריה מקיפה, מהירה ונוחה מאוד למשימות הקשורות בעיבוד תמונה. השתמשתי ב `cv2` לצורך שמירת פריים הסרטון (`cv2` כתיבת סרטון) - יכולתי להשתמש גם כאן אבל הפונקציה `cv` יותר טוב יותר.

את שם הקובץ קיבל כארגומנט של התוכנית ואת הערכיהם הבוליאניים עבור סוג האלגוריתם והאם יש צורך בסיבוב נקבע מה `cv` על ידי שימוש בפונקציית `cv2`.
נשתמש בפונקציה `cv2.get_frames_from_video` שמייחת כדי לקבל את הפריים מהסרטון על ידי שימוש בפונקציה `cv.VideoCapture` ומעבר על הפריים תוך שימוש בפונקציית `cv.read` של ערך החזרה של אותה פונקציה. סיבוב הפריים במידת הצורך מבצע בפונקציה `cv2.rotate_video` ע"י שימוש בפונקציה `cv2.rotate`.

עברית בלולאה על זוגות של פריים עוקבים ונשמר אותו במשתנים `previous_frame` ו- `current_frame`. נשתמש ב `cv2.SIFT_create` כדי ליצור אובייקט מסווג `sift` `detector` וב `sift.detectAndCompute` כדי למצוא keypoints כדי לנណ נקודות התאמה לפי החשיבות/חזקקה של נקודת העניין באמצעות הערך `kp.response` המקיים. בחרתי בערך סף כיון שעבורו קיבלת מס' נקודות עניין מתאים.

השתמשתי ב `(cv.NORM_L2, cv.BFMatcher(cv.NORM_L2, crossCheck=True))` כדי להתאים מתאים (Descriptors) בין תמונות על בסיס מרחק אוקלידי (`NORM_L2`) ולזוזה התאמה הדידית באמצעות Cross-Check. נמצא מטריצת `cv2.RANSAC` על ידי שימוש באלגוריתם RANSAC להתאמה של נקודות.

להלן את המימוש שלי לאלגוריתם RANSAC שראינו בכתיבה. מספר האיטרציות של האלגוריתם (אותו נעדכן תוך כדי ריצה אך נגיד מספר מקסימלי של איטרציות- 100) הוא `log` של אחת פחות ההסתברות `k` (שנגדיר אותה קבוע שווה ל 0.99) שהיא ההסתברות שההתאמה טובה לאחר מספר כשלשו של איטרציות חלק `log` של אחת פחות אומגה בחזקת מספר הנקודות שצורך כדי לחשב את המודל (במקרה של `log` שווה ל 2). אומגה הוא ההסתברות לקבלתliers (נקבע את הערך ההתחלתי

לחצי ונדען תוך כדי הריצה). בכל איטרציה נבחר שתי נקודות שונות ונחשב מטריצת p_{ij} . נמיר את הקורדינטות למטריצה של 3 על 3 ונכפיל את הקורדינטות במטריצת p_{ij} ונחשב את המרחק האוקלידי בין קואורדינטות המקור שuberו טרנספורמציה לקואורדינטות היעד ונגדר את ה $\text{d}_{\text{inliers}}$ כאשר המרחק קטן מקבוע מסוים שנגדר (הגדרתי אותו כ-1 כיון שמצוות שזה ערך שונה ל-1' תוצאות טובות).

чисוב מטריצת p_{ij} מתבצע על ידי חישוב הזווית בין 2 הנקודות שלחנו מכל פריים וחישוב הפרש הזווית ביניהן. נגדר את מטריצת h כמו שלמדנו לפני הזווית ונחשב את הנקודות מפריים המקורי. לאחר מכן, נחשב את וקטור h הפרש של הנקודות בתמונה היעד לנקודות המשובצות של תמונה המקור. השתמש בערכים אלו כדי ליצור מטריצת p_{ij} כמו שלמדנו בכיתה.

בשלב הבא נחשב את המטריצה היחסית לתמונה הראשונה על ידי חישוב מטריצה ההפיכה והכפלה במטריצה האחורונה שוחשבה וכך למצוא מטריצה ההפיכה בשונקציה `alg.linalg.inv()`.
cut Apart את מימוש האלגוריתם שתיארתי למציאות גודל הקבב וההיסטים. השתמש בגודל וגובה התמונה כדי לשמר ברשימה `corners` את ערכי הפינות ונחשב את הטרנספורמציות של הפינות על ידי שימוש בשונקציה `cv.transform`.

להלן האלגוריתם מתפצל לשתי פונקציות נפרדות לפי סוג הפונרמות. כתע אתאר את המימוש של הפונקציות הרלוונטיות.

אם מדובר באלגוריתם מסווג `dynamic`, נחשב את מרכזי הפריים באמצעות הפונקציה `calculate_frames_centers`. נחשב את גבולות הרצועות באמצעות `calculate_strip_borders` על מרכזי הפריים באמצעות `cv.transform`. נחשב את גבולות הרצועות באמצעות `shift_size` כדי לחשב את ה `shift` שנוסף לגבולות הרצועות (נשלח את זה בחלק הקודם (נוסיף דגש שעבור גבולות הרצועות אם המינימום גדול שווה למקסימום אז נחליף בינהם). השתמש במשתנה `shift_size` כדי לחשב את ה `shift` שנוסף לגבולות הרצועות (נשלח את זה כפרמטר ל `create_mosaic`) שמחושב באופן הבא- הוא שווה לחצי מרוחב הפריים בהחזרת רוחב הפריים חלק קבוע (שווה ל 10). ה `shift` נע בין `size - shift - 1` ל `shift_size` שנגדר עד ל `size` חלקי `SHIFTS_NUMBER` (קבוע שהגדרתי ל-30).

האלגוריתם של `create_mosaic` תואר בחלק הקודם, רק אציין ש `shift` זה הגודל שלחנו בשונקציה הקודמת ו- `is_dynamic_mosaic` שנאותחל כ `True`. ה `canvas` שאתחולט אוטומטית על ידי הפונקציה `zeros`. המשתנים `x_min, x_max, canvas_width, canvas_width` יאותחלו בערכים `0, 0, 1000, 1000` לפי הגבולות של הרצועות (`max_x_strip, min_x_strip`) כאשר יתקבל מינימום חדש בכל איטרציה. נקרא לפונקציה `warp_strip` ונתחל את הקבב לפי איך שהוסבר קודם.

להלן אתאר את המימוש של `warp_strip`. השתמש בשונקציות `ravel`, `np.stack`, `np.meshgrid` כדי ליצור רשת של קואורדינטות (`y, x`) עבור פסים בתמונה ולהמיר אותם לקואורדינטות הומוגניות (`1, y, x`) כדי לאפשר חישוב טרנספורמציה ההפוכה. כך, ניתן לבצע עיוות על הפסים ולהרכיב את התמונה הסופית בפונרמה. הקוד מNORMAL את הקואורדינטות ההומוגניות (`1, y, x`) על ידי חילוקה בקואורדינטה ה-3 (`z`) כדי להחזיר את הקואורדינטות לפורמט `2D`. לאחר מכן, הוא משנה את הקואורדינטות המפולגות לרשת דו-ממדית של `x` ו-`y`, כך שניתן להשתמש בהן ל `warping` של התמונה. לבסוף, השתמש ב-`cv.remap` כדי לעשות Backward Warping החלק של התמונה לפי הקואורדינטות החדשות `y_frame` ו-`x_frame` ולבצע אינטרפולציה לינארית בין פיקסלים כדי לחשב ערכים חדשים (משתמשים במצב גבול קבוע כדי להחסיר פיקסלים מחוץ לתמונה- ערך 0).

כאשר מדובר באלגוריתם מסווג viewpoint, מה שחייב לציין זה שה shift שאנו חnage כפרטט הוא 0 ושבסוף מוסיף לרשימת הפריים את אותה רשימה בסדר הפוך כדי לקבל את האפקט הרצוי.

לבסוף, כדי לטפל בשמיירת הפריים לסרטון נוצרה להמיר את הצבעים בין הפורמטים השונים של הספריות (מ BGR של cv2 openCV ל RGB של imageio). כמו כן, השתמש בrotate_video שוב במידה ויש צורך לפי קלט המשמש וכך נוצר לשנות את גודל הפריים למידדים זוגיים (اشתמש בפונקציה adjust_frame_size ככה ספרית imageio). הפונקציה המרכזית בחלק זה היא saveimageio עבורה נגיד גודל של 8 פריים לשנייה.

הकשיים העיקריים אתם התמודדתי במהלך הבנת השימוש של ה Warping בחלק של הפסיפס והבנת חישוב shifting בחלק האלגוריתם הדינامي.

תוצאות ויזואליות

להלן ציג את התוצאות של האלגוריתם מסווג viewpoint- עברו הסרטון של הסירה (הפריים הראשון, והאחרון מוצגים לפי הסדר). ניתן לראות את ההשפעה של אפקט ה parallax, כלומר גופים קרובים ורחוקים לא נעים באותו האופן.



להלן ציג את התוצאות של האלגוריתם מסווג dynamic- עברו הסרטון של העצים. ניתן לראות שבדומה לסרטון של המפלים, הפונורמות מוצגות באופן הדרמטי ולא נכנסות בתוך פריים אחד (כלומר צריך לייצב את הסרטון). הפריים מוצגים משמאלי לימי.



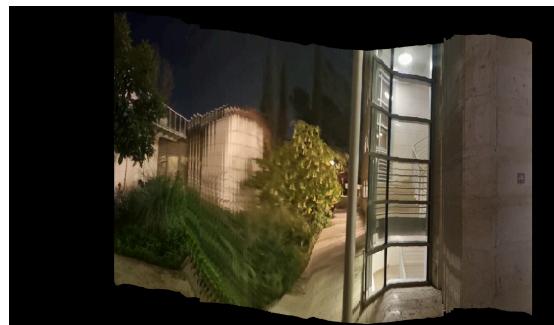
להלן אציג את הסרטונים שצילמתי. הנה הקלט של הסרטון עבורי התקבלו תוצאות טובות (לפי הסדר מימין לשמאל). כיוון התנועה בסרטון הוא מימין לשמאל.



cut אציג את התוצאות של אותו סרטון (לפי הסדר משמאל לימין)



להלן ציג את התוצאות של האלגוריתם עבורי התקבלו תוצאות לא טובות (הסיבה היא שונה הזום של הסרטון במהלך הסרטון שמתבסס על קר שלא יהיה scaling). בסרטון ניתן להבחן בקלות יותר בעיות ביצירת הפיסוף. קלט לעליה ופלט למיטה, משמאלי לימין (כיוון התנועה משמאל לימין).



מסקנות

למדי אלגוריתמים שימושיים כגון SIFT, RANSAC ושיטות כמו Warping. באמצעות כלים שימושיים אלה אנו יכולים ליצור פסיפס של תמונות וליצור סרטון של פנורמות לפי הסוגים השונים של האלגוריתמים, שנוצרים על ידי שינוי נקודת המבט (viewpoint) או שינוי ה shift (ב dynamic).

ניתן לשפר את התוצאות שייצאו לנו בתרגיל באמצעות ייצור סרטון במרקחה של ה dynamic והתייחסות באלגוריתמים לציר ה u בתמונה ולא רק לרצועות בציר ה x על מנת לצמצם את האפקט של ה parallax.