

מגשים:

אופק חררי

שני ישראלוב

הבהרות על אופן העבודה שלנו: עבדנו בסביבת PyCharm, חוץ מתיקיית data שניתנה בתרגיל השתמשנו בתקיית my_data המכילה תמונות לחלקים 2-3, וידיאו ומודלים מאומנים לחלק 3. הקוד לחלק 2 נמצא בקובץ my_homography, לחלק 3 נמצא בקובץ my_arr ולחלק היצירתי בקובץ my_vid2vid_ext. בכל אחד מהסקריפטים הקוד שלנו מחולק לפונקציות עזר שמימשנו ולפונקציות שנדרשו לממש בתרגיל. ב-main אפשר להריץ פונקציות עזר לפי הסעיפים בתרגיל.

Part 1 - Theory

שאלה 1.1

לפי העקרונות של המטריצה F מתקיים $l = F^T x$ כלומר המטריצה הפונדמנטאלית היא התמרה מנקודה בקואורדינטות הומוגניות לישר בקואורדינטות הומוגניות. אנחנו נרצה להראות שאם הקואורדינטות של התמונה מנורמלות כך שהמקור שלהן מצביע לנקודה p אז האלמנט f_{33} מתאפס במטריצה הפונדמנטאלית.

ישר l מתקבל על ידי חיסור של 2 וקטורים בקואורדינטות הומוגניות:

$$l = \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} - \begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} = \begin{pmatrix} x_1 - x_2 \\ y_1 - y_2 \\ 0 \end{pmatrix}$$

עבור המשוואה $l = F^T x$ נקבל:

$$\begin{pmatrix} x_1 - x_2 \\ y_1 - y_2 \\ 0 \end{pmatrix} = \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

עבור הנקודה $c_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ נקבל:

$$\begin{pmatrix} x_1 - x_2 \\ y_1 - y_2 \\ 0 \end{pmatrix} = \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

נקבל בשורה השלישית אילוץ הדורש איפוס של f_{33} .

שאלה 1.2

נגדיר את נקודות המרכז של המצלמות בתור $c_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$, $c_2 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ כאשר הנקודה c_1 היא world origin ותנועת המצלמה היא לאורך ציר ה- x .

מכיוון שהתמונה המתקבלת מקבילה לציר ה- x ניתן לרשום $e' = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ ולכן נקבל:

$$F = [e']_x = \begin{pmatrix} 0 & -e_z & e_y \\ e_z & 0 & -e_x \\ -e_y & e_x & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$$

הקשר בין נקודות תואמות בתמונות מקיים $x' F x = 0$. נגדיר $\vec{x}' = \begin{pmatrix} x'_1 \\ y'_1 \\ z'_1 \end{pmatrix}$, $\vec{x} = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}$ להיות זוג נקודות תואמות.

$$x'Fx = \begin{pmatrix} x'_1 \\ y'_1 \\ z'_1 \end{pmatrix}^T \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} = \begin{pmatrix} x'_1 \\ y'_1 \\ z'_1 \end{pmatrix}^T \begin{pmatrix} 0 \\ -z_1 \\ y_1 \end{pmatrix} = 0$$

$$\Rightarrow \frac{y'_1}{z'_1} = \frac{y_1}{z_1}$$

על ידי שימוש בפיקסל \vec{x} בתמונה הראשונה נקבל בנקודה התואמת $\vec{x'}$ בתמונה השנייה את אותה הקואורדינטה בציר y . לכן נקבל שהקווים האפיפולרים של 2 המצלמות הם גם מקבילים לציר ה- x .

שאלה 1.3

יהיו קואורדינטות של אובייקט ב-3D $\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}$ להיות העמדה של הרובוט בזמן i . לכן מתקיים:

$$\begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = K \left(R_1 \begin{pmatrix} u \\ v \\ w \end{pmatrix} + t_1 \right)$$

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = R_1^{-1} \left(K^{-1} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} - t_1 \right) = R_1^T K^{-1} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} - R_1^T t_1$$

עבור העמדה ה-2 מתקיים:

$$\begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} = K \left(R_2 \begin{pmatrix} u \\ v \\ w \end{pmatrix} + t_2 \right) = K \left(R_2 (R_1^T K^{-1} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} - R_1^T t_1) + t_2 \right)$$

$$= KR_2 R_1^T K^{-1} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} - KR_2 R_1^T t_1 + K t_2$$

מכאן נקבל:

$$R_{rel} = KR_2 R_1^T K^{-1}$$

$$t_{rel} = -KR_2 R_1^T t_1 + K t_2$$

והמטריצות המבוקשות הן:

$$E = t_{rel} \cdot R_{rel}$$

$$F = (K^{-1})^T E K^{-1} = (K^{-1})^T (t_{rel} \cdot R_{rel}) K^{-1}$$

Part 2 - Planar Homographies

Part 2.A – Planar Homographies: Theory warm up

שאלה 2.0

מערכת המשוואות $p^i = Hq^i$ כתובה בקואורדינטות הומוגרפיות ולכן היא נכונה עד כדי סקאלה ולכן נציג אותה באופן הבא:

$$\alpha^i p^i = Hq^i$$

נרשום את מערכת המשוואות המתקבלת:

$$(1) \alpha^i p_x^i = h_1 q_x^i + h_2 q_y^i + h_3$$

$$(2) \alpha^i p_y^i = h_4 q_x^i + h_5 q_y^i + h_6$$

$$(3) \alpha^i = h_7 q_x^i + h_8 q_y^i + h_9$$

לאחר הצבה של α^i נקבל:

$$(1) h_7 q_x^i p_x^i + h_8 q_y^i p_x^i + h_9 p_x^i = h_1 q_x^i + h_2 q_y^i + h_3$$

$$(2) h_7 q_x^i p_y^i + h_8 q_y^i p_y^i + h_9 p_y^i = h_4 q_x^i + h_5 q_y^i + h_6$$

נעביר אגפים ונקבל :

$$(1) h_7 q_x^i p_x^i + h_8 q_y^i p_x^i + h_9 p_x^i - h_1 q_x^i - h_2 q_y^i - h_3 = 0$$

$$(2) h_7 q_x^i p_y^i + h_8 q_y^i p_y^i + h_9 p_y^i - h_4 q_x^i - h_5 q_y^i - h_6 = 0$$

נעת נציג את מערכת המשוואות הזו באופן של $A^i h$:

$$A^i h \triangleq \begin{pmatrix} -q_x^i & -q_y^i & -1 & 0 & 0 & 0 & q_x^i p_x^i & q_y^i p_x^i & p_x^i \\ 0 & 0 & 0 & -q_x^i & -q_y^i & -1 & q_x^i p_y^i & q_y^i p_y^i & p_y^i \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{pmatrix} = 0$$

כלומר קיבלנו 2 משוואות שתלויות עבור נקודה i ולכן עבור N נקודות נקבל $2N$ משוואות בלתי תלויות

$$Ah \triangleq \begin{pmatrix} A^1 \\ A^2 \\ \vdots \\ A^{N-1} \\ A^N \end{pmatrix} h = 0$$

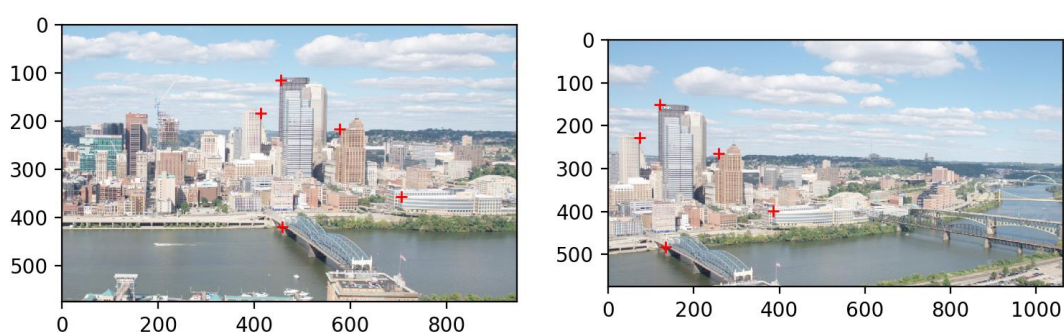
H מורכב מ-9 אלמנטים ומכיוון שאת h ניתן לנרמל לפי אחד האיברים (האיבר האחרון) אז נקבל שדרגת h היא 8 ולכן יש 8 איברים שיש למצוא . בכדי למצוא את האיברים האופטימליים נפתור את בעיית האופטימיזציה $\min_h \|Ah\|^2$

נשים לב שעבור זוג נקודות i תואמות מקבלים 2 משוואות ולכן בשביל למצוא את 8 הנעלמים ב- h נדרש לנו לפחות 4 נקודות ובנוסף נדרש שהמשוואות של הנקודות יהיו בת"ל.

Part 2.B – Planar Homographies: Practice

2.1 Manual finding corresponding points

הסבר המימוש: בסעיף זה השתמשנו בפונקציה `ginput()`, אנחנו מציגים את שתי התמונות אחת לצד השנייה, המשתמש צריך לסמן נקודה בתמונה השמאלית ולאחר מכן את הנקודה המתאימה בתמונה הימנית וכך הלאה עד שיש N נקודות. בתוך הפונקציה `getPoints` אנחנו רצים על הנקודות שסומנו בלולאה. נקודות עם אינדקס זוגי שייכות לתמונה השמאלית ונשמרות ב- p_1 , ונקודות עם אינדקס שלילי שייכות לתמונה הימנית ונשמרות ב- p_2 .



בתמונה הזאת בחרנו $N=10$. מסמנים נקודה בתמונה השמאלית ולאחר מכן את הנקודה המתאימה בתמונה הימנית, וכך הלאה עד שיש 5 זוגות.

2.2 calculate transformation

- הסבר על המימוש:** תחילה בדקנו שמספר הנקודות ב- p_2 שווה למספר הנקודות ב- p_1 כי הנקודות הללו מייצגות את הנקודות התואמות בשתי התמונות ולכן חייבים להיות שווים. לאחר מכן אנחנו רצים על N הנקודות ובונים את המטריצה הבאה:

$$A = \begin{bmatrix} -u & -v & -1 & 0 & 0 & 0 & ux & vx & x \\ 0 & 0 & 0 & -u & -v & -1 & uy & vy & y \end{bmatrix}$$

כאשר (u,v) הם קורדינטות של p_2 , (x,y) הנקודות מ- p_1 .
באמצעות SVD על מטריצה A אנחנו מוצאים את המטריצה V ומחלצים מתוכה את H .

- נראה כי הטרנספורמציה נכונה על ידי בחירת נקודות אקראיות בתמונה הראשונה והטלתן על התמונה השנייה. כאשר עושים את ההטלה מקבלים:

```
H=[e-01 -7.85256945e-02 3.65281381e+026.48941614 ]
[e-02 8.30355525e-01 -1.26312535e+016.30452664-]
[[-3.28582728e-04 -1.04228917e-04 1.00000000e+00]
```

והנקודות המוטלות הן:

```
[414.95256045 691.79475229 457.28170775 576.71599285 453.94346517]]
[[ 180.5189919 349.74006203 422.65128082 218.28580418 114.03541197]
```

- הערה: עשינו בדיקה ובדקנו הטלה של התמונה על עצמה, ציפינו לקבל את מטריצת היחידה אבל קיבלנו את המטריצה הבאה:

```
H=[e-03 6.33563035e-04 8.76063126e-019.13843568 ]
[e-03 1.20880614e-02 4.81790820e-011.35840814-]
[[-3.69730445e-06 2.00276464e-06 1.26145114e-02]
```

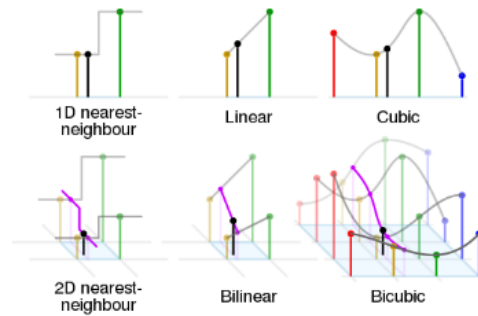
זוהי לא מטריצת היחידה. אנחנו מניחים שקשה לקבל את מטריצת היחידה מאחר שהסימון ידני והנקודות שנבחרות אינן מדוייקות.

2.3 image wrapping

זוהי התמונה השמאלית im1 שעברה wrap על מנת להתאים לתמונה הימנית im2.



- **הסבר על המימוש:** לפני ביצוע $wrapH$ לתמונה השמאלית, ביצענו Translation. פונקציית ה-Translation שלנו מקבלת בכניסה את התמונה im1 שעליה רוצים לבצע את ה- $wrap$ ואת מטריצת H שקיבלנו מפעולת ה- $compute$. הפונקציה הזאת באה להתמודד עם הבעיה שעקב פעולות על התמונה חלקים ממנה יכולים להיות בפיסקלים שליליים ואז אנחנו לא רואים את החלקים הללו. תחילה פונקציית ה-Translation קוראת לפונקציית $FindCorners$ פונקציה זו מוצאת את הפינות של התמונה המקורית ואת הפינות של התמונה לאחר כפל במטריצת הומוגרפיה. כפי שנאמר פיקסלים אלה עלולים להיות שליליים ולכן אנחנו מבצעים טרנסלציה (הזזה) של התמונה כך שכל התמונה תהיה רק בצירים החיוביים ושלא ייחתכו חלקים ממנה. פונקציה זו מחזירה לנו את H לאחר ההזזה, את הגודל של התמונה המאוחדת ומערך של גדלי הצירים. לאחר מכן אנחנו מבצעים את פונקציית ה- $wrapH$. אנחנו ממירים את התמונה למרחב צבעים LAB, מגדירים את הגודל של התמונה שעוברת wrapping לפי ה- $outside$ שחישבנו ב-Translation. לאחר מכן רצנו בלולאה על שלושת מרחבי הצבעים L,A,B. בכל מרחב כזה עושים אינטרפולציה באמצעות פונקציית $interp2d$ שמקבלת כקלט את הטווחים של גודל התמונה ומרחב הצבע הנוכחי. בחרנו להשתמש באינטרפולציה מסוג linear. לאחר מכן עשינו לולאה כפולה שרצה על ציר ה-x ועל ציר ה-y ומוצאת את הנקודות p_1 באמצעות כפל של מטריצת ההומוגרפיה בנקודות p_2 . לבסוף אנחנו חוזרים ממרחב הצבעים LAB למרחב הצבעים RGB.
- סוגי האינטרפולציה קשורים בפונקציה שמחברת בין הנקודות אינטרפולציה. באינטרפולציה ליניארית הפונקציה היא ממעלה ראשונה, ב-cubic ממעלה שנייה וכך הלאה. אנחנו ראינו שאינטרפולציה ליניארית נתנה תוצאות טובות ולכן נשארו איתה. ניתן לראות בתמונה הבאה מתוך ויקיפדיה:



- בחרנו לעבוד עם מרחב צבעים LAB כי יש לו gamut (פלטת צבעים) רחב יותר, דומה ליותר לתפיסה האנושית, בפרט הרכיב ה-L דומה לתפיסה האנושית של תאורה. מרחב צבעים זה מתייחס לשחור ולבן כאל ערוצים נפרדים. השימוש ב-LAB מסייע לנו ב-color balance איכותי.

2.4 panorama stitching

ביצענו את התפירה לאחר בחירת נקודות תואמות בצורה ידנית וקיבלנו את התוצאה הבאה:



- **הסבר על המימוש:** מימשנו את התפירה באמצעות פונקציית מטפסת panoramaTwoImg שמבצעת את ה-wrapH על התמונה הימנית ואז קוראת לפונקציית getScaled. באמצעות פונקציה זו אנחנו מבצעים את ההתאמות לקראת התפירה. כאשר באמצעות פונקציית translation ביצענו את ההזזה של התמונה שעברה wrapping יש צורך להזיז בהתאמה את התמונה המיושרת למקום הנכון. פונקציה זו מקבלת ארגומנט שנקרא wrap_is_left מכיוון שיש הבדל בהתאמות אם התמונה שעברה wrapping נתפרת מימין או משמאל לתמונה המיושרת. לאחר מכן אנחנו קוראים לפונקציית ה-imageStitching שמקבלת את התמונות כאשר הימנית זו התמונה המיושרת והשמאלית זו התמונה שעברה wrapping. בפונקציה זו אנחנו שמים בתמונת הפנורמה את שתי התמונות לאחר ההתאמות ולוקחים לכל פיקסל את הערך המקסימלי מכל תמונה. לאחר מכן ממירים לערך uint8 ומחזירים את תמונת הפנורמה.

2.5 autonomous panorama stitching using SIFT

- SIFT מוצא את ה-feature matches בין התמונות שנתפרות. ואז אנחנו מוצאים באמצעות descriptor matching algorithm מסוג knn את ה-k-nearest-neighbors.
- **הסבר על המימוש:** עבדנו על התמונות ב-gray scale. לכל תמונה יצרנו sift ובאמצעות פונקציית detectAndCompute מצאנו את ה-Key points ו-descriptors. יצרנו BFMATCHER שמחזיר רשימה של DMatch objects. השתמשנו בשיטת knnMatch כדי למצוא את k הזוגות התואמים הטובים ביותר. לאחר מכן עשינו בדיקת ratio לזוגות. כל זוג שהמרחק שלו קטן מ-0.4 נשמר וכל השאר נמחקו.

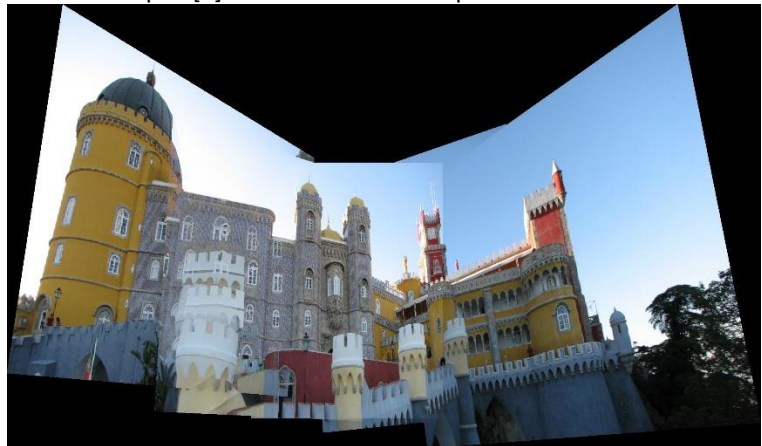


2.7 compare SIFT and Manual Image selection

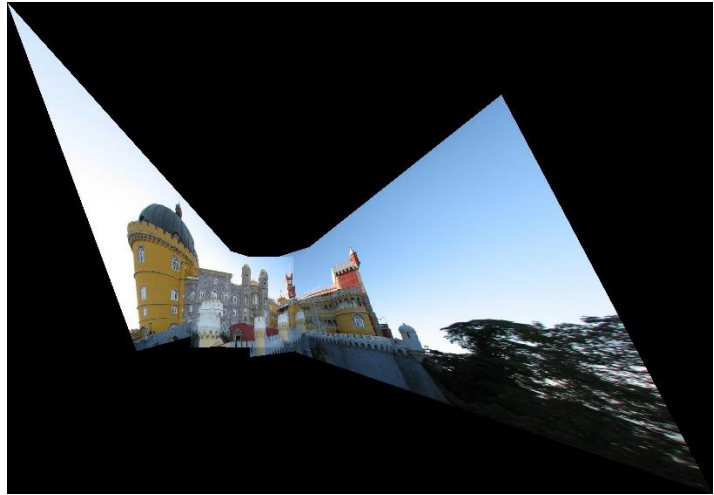
כעת יש תפירה של חמש תמונות. התחלנו מתפירה של תמונות 2+3 כאשר התמונה השלישית נשארת מיושרת והתמונה 2 עוברת wrapping. לאחר מכן לתמונה שהתקבלה אנחנו מחברים את תמונה 4 כאשר היא התמונה שעוברת wrapping כדי להתאים לתמונה שנוצרה מ-2+3. לאחר מכן מוסיפים את 1, ולבסוף את 5. נציין כי לתמונות של סינטרה ביצענו הקטנת רזולוציה פי 4 וקיבלנו תוצאות טובות על אף הקטנת הרזולוציה. בשביל לעשות את תמונת הפנורמה של beach סובבנו את התמונות 90 מעלות ותפרנו בצורה דומה לתמונת ה-sintra. בתמונות הים הקטנת רזולוציה פגעה בתוצאות עד כדי איבוד מידע ולכן נשארו עם הרזולוציה המקורית, בעקבות רזולוציה גבוהה בחיבור התמונה החמישית קיבלנו שגיאת זיכרון, לעומת זאת בשימוש עם RANSAC כן הצלחנו לחבר את חמשת התמונות כנדרש.

תמונות sintra:

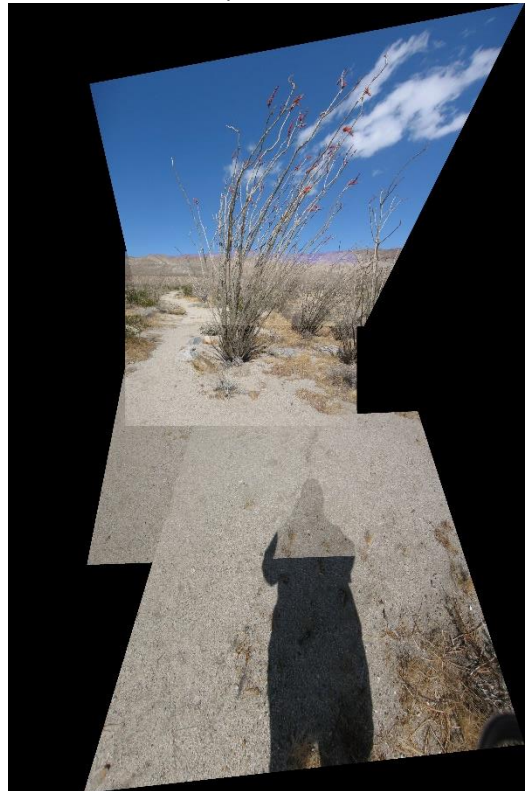
החיבור באמצעות SIFT: התהליך בתמונות נמצא בנספח [1] בסוף הדו"ח.



החיבור באמצעות סימון נקודות ידני: התהליך בתמונות נמצא בנספח [2] בסוף הדו"ח.



תמונת beach:
 החיבור באמצעות SIFT: התהליך בתמונות נמצא בנספח [3] בסוף הדו"ח



החיבור באמצעות סימון נקודות ידני:



השוואה בין 2 השיטות:

בחירת הנקודות בצורה ידנית, החסרונות הבולטים הם החוסר דיוק בהתאמה מושלמת של הנקודות והמאמץ מצד המשתמש. הנקודות צריכות להיבחר בסדר מסוים כדי שהנקודות p_1, p_2 יישמרו בסדר הנכון. בנוסף, שמנו לב כי אם הבחירת נקודות לא הייתה מדויקת מאוד קיבלנו תוצאות לא טובות. מאותה סיבה אנחנו עלולים לא לקבל תוצאות זהות מהרצה להרצה.

בשיטת ה-SIFT אין צורך לסמן ידנית והוא מבצע התאמה טובה יותר בין הנקודות וזה יתרון משמעותי על פני השיטה הראשונה.

2.8 RANSAC

השתמשנו ב-RANSAC כי שיטת חישוב ההומוגרפיה לא חסינה בפני outliers. אם יש match לא טוב זה יכול לקלל את כל ההומוגרפיה.

הסבר על המימוש: רצנו בלולאה לפי מספר האיטרציות שהגדרנו. יצרנו באופן רנדומלי קבוצת נקודות, חישבנו את H לפי הזוגות p_1, p_2 כאשר לקחנו רק את הזוגות לפי הקבוצת נקודות שהגרלנו. יצרנו דגימות של p_1 באמצעות כפל של H עם הנקודות p_2 . נתנו לכל קבוצת נקודות כזאת ציון שמבוסס על המרחק בין p_1 לדגימות שיצרנו. הציון הגבוה ביותר הוא ה- H שאנחנו מחזירים. אנחנו בחרנו $niter=4000$ ו- $tol=5$. כאשר $niter$ מספר האיטרציות עד למציאת H אופטימלי, tol המרחק בין p_1 לדגימותיו.

תוצאות החיבור לאחר שימוש ב-RANSAC:

Sintra בצורה האוטומטית באמצעות SIFT ו-RANSAC: תהליך התמונות בנספח [4] בסוף הדו"ח:



בצורה בחירת נקודות ידנית: בצורה זאת יצאו תוצאות לא טובות כתלות בבחירת הנקודות. התקשנו לסמן את הנקודות כך שתתקבל תוצאה סבירה.

Beach

החיבור באמצעות SIFT ו-RANSAC: התהליך בתמונות נמצא בנספח [5] בסוף הדו"ח



החיבור באמצעות סימון נקודות ידני ו-RANSAC:



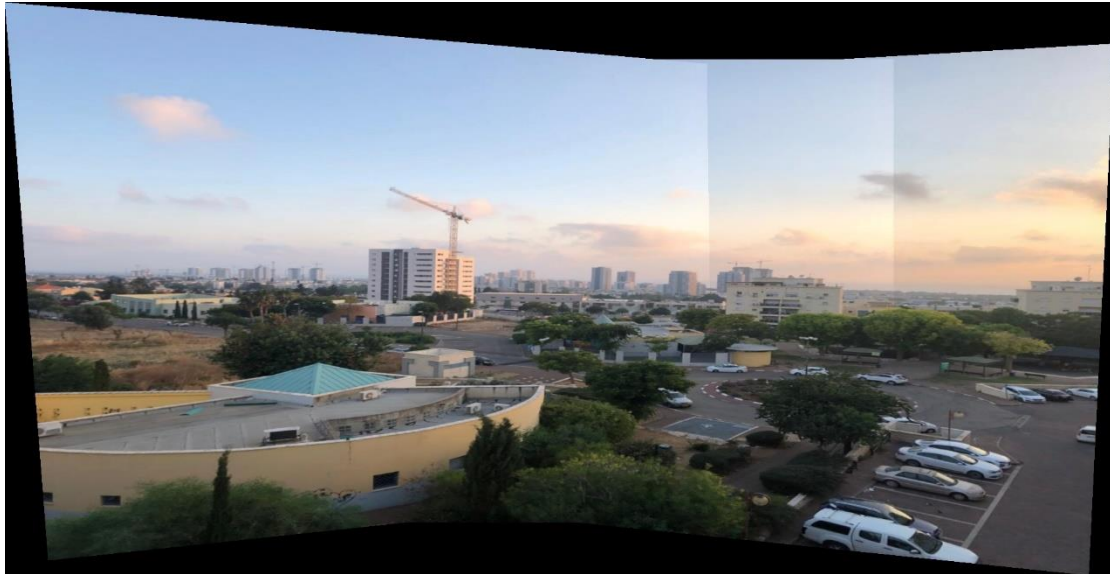
כמו שציינו בהשוואה בין SIFT לידני התוצאות לא אחידות עבור הידני וקשה לקבל תוצאות סבירות במקרים מסויימים. במקרה הזה העלנו רק את הפנורמה החלקית כי הסימון הידני של החיבורים הבאים נתן תוצאות לא טובות אחרי מספר נסיונות.

- **האם יש שיפור עבור הידני ועבור ה-SIFT?** לדעתנו ההשפעות של ה-RANSAC במקרים שבחנו הן מינוריות. במקרה של הידני ה-RANSAC לא משפיע וזה הגיוני כי יש לנו מספר קטן של זוגות של נקודות. לעומת זאת, במקרה של SIFT יש לנו מספר גדול יותר של נקודות ואז ה-RANSAC "מסנן" נקודות לא טובות. ואכן בתמונת ים קיבלנו תוצאה טובה יותר, ניתן לראות שהתמונה ב-RANSAC חיברה נכון את התמונה של האיש כך שהצבעים לא משתנים, בנוסף החיבור היה מהיר יותר.

Blending 2.9 – לא עשינו

2.10 Be Creative

ביצענו תפירת תמונות פנורמה של נוף מהמרפסת, חיברנו באמצעות SIFT כי לדעתנו הוא הביא תוצאות טובות בצורה פשוטה. בחרנו את התמונה האמצעית להיות התמונה הישרה ותפרנו אליה את התמונה הימנית ולאחר מכן לתמונה המחוברת חיברנו את התמונה השמאלית. השתמשנו ב-SIFT ללא RANSAC וקיבלנו תוצאה מצויינת.

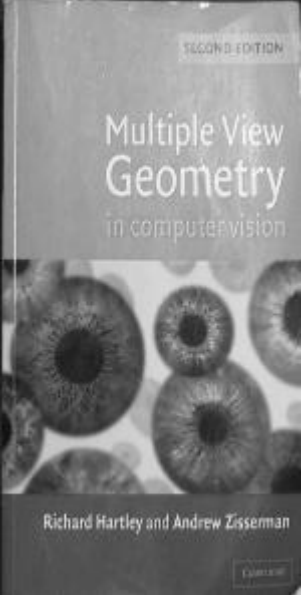
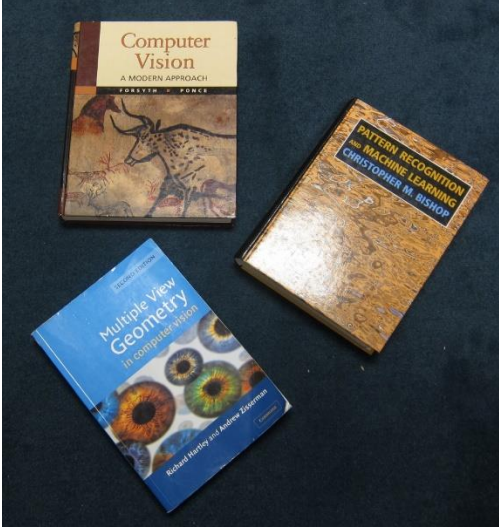


2.11 Affine vs Projective – לא עשינו

Part 3 – Creating your augmented reality application

3.1 Create reference model

הסבר על המימוש: השתמשנו בפונקציית `getPoints` הידנית, ובקלט שלנו הכנסנו את התמונה המבוקשת פעמיים. מהפונקציה הזאת אנחנו רוצים לקבל אך ורק את p_1 הנקודות שסומנו בתמונה השמאלית (בעלי אינדקס זוגי). הגדרנו את p_2 בתור 4 הפינות של ריבוע בגודל `out_size=[300,150]`. חישבנו באמצעות `computeH` את מטריצת ההומוגרפיה, השתמשנו בפונקציית `Translation` ולאחר מכן בפונקציית `wrapH`. תוצאת הביניים זו התמונה המקורית כך שהאובייקט שסימנו, במקרה שלנו, הספר האהוב עלינו `Multiple view geometry` כעת מקביל לצירים x, y . עכשיו עלינו לבצע חיתוך על מנת לבדוד את הספר משאר הרקע, ביצענו את החיתוך וקיבלנו את התוצאה המבוקשת.

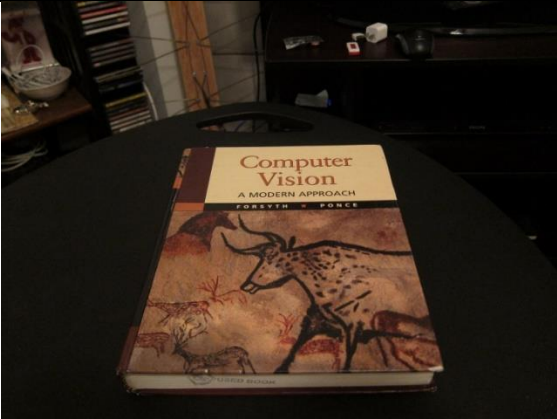
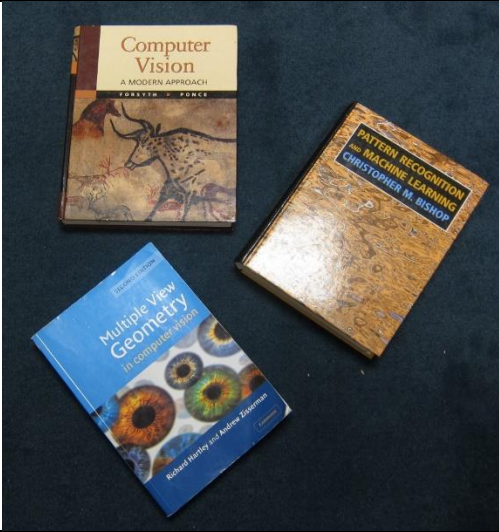
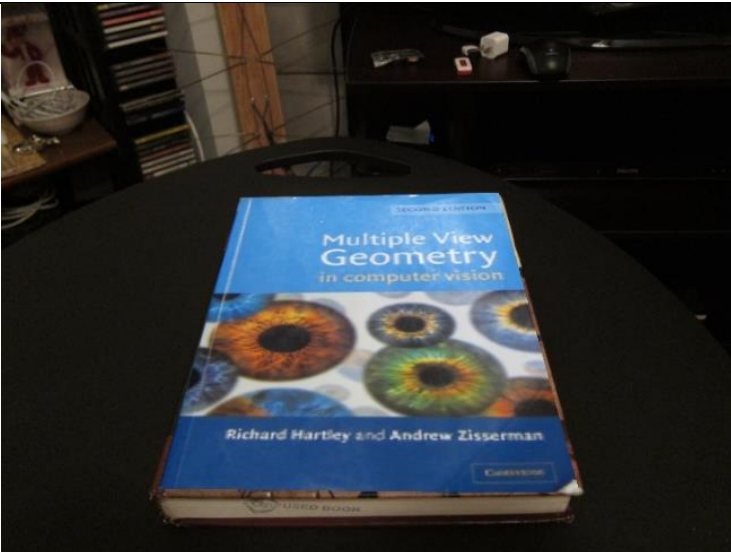
תמונת הרפרנס הרצויה	התמונה המקורית
	

3.2 implant an image inside another image

הסבר על המימוש: יצרנו פונקציה שנקראת `im2im()` שמקבלת בכניסה את ה- `scene_path` ואת ה- `im_path`. כאשר הארגומנט הראשון זו התמונה שאליה רוצים להדביק את התמונה בארגומנט השני. מ- `im_path` אנחנו מקבלים את תמונת הרפרנס באמצעות הפונקציה שמימשנו בסעיף הקודם. כאשר אנחנו משתמשים בפונקציה `getPoints` על תמונת הסצנה ומחלצים את p_2 בלבד. מגדירים את p_1 כפינות של תמונת הרפרנס. מחשבים את H בין התמונות

ומבצעים wrapH לתמונת הרפרנס. לאחר מכן אנחנו מחלצים את האינדקסים של תמונת הרפרנס לאחר wrapping ומבצעים הזזה לאינדקסים הללו, לאחר מכן אנחנו פונים לתמונת הסצנה ומדביקים באינדקסים הרלוונטים שמצאנו את תמונת הרפרנס.

ניתן לראות דוגמא אחת כאן ושתי דוגמאות נוספות בנספחים [6] ובתקיית output.

דוגמא ראשונה: ספר גיאומטריה מרחבית מחליף את מקומו של ספר ראייה ממוחשבת	
Scene	Reference
	
התמונה החדשה:	
	

3.3 video time – לא עשינו

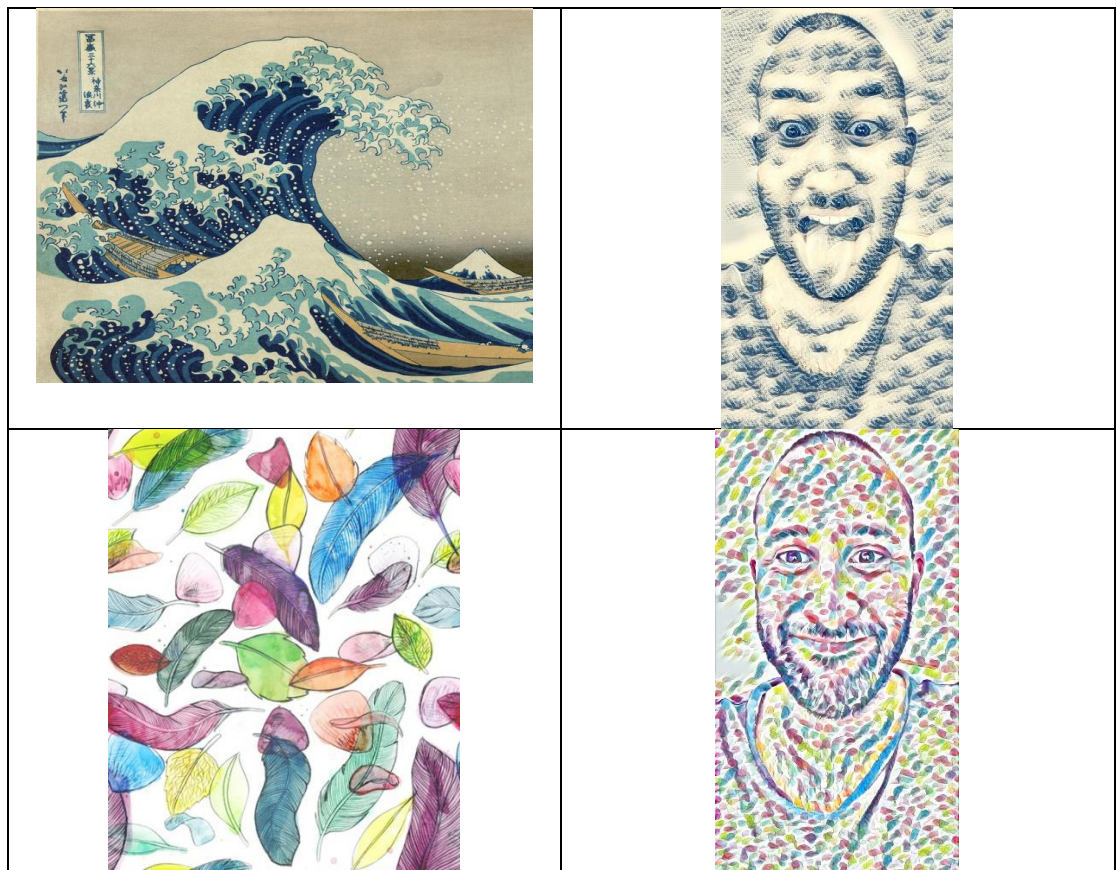
3.4 Creativity section

השתמשנו ברשת נוירונים מאומנת שמפעילה סגנונות אומנותיים על תמונות. אחרי האימון, הרשת מעצבת תמונות במהירות. השתמשנו ברשת מתוך הקישור הבא: [icjohnson](#). המודלים מבוססים על מאמר [neural algorithm of artistic style](#). מתוך המאמר קראנו על תהליך בניית הרשת CNN, תמונה מוצגת כסט של תמונות מפולטרות בכל שלב ברשת CNN. בונים את תמונת הכניסה מחדש כאשר יודעים רק את תגובת הרשת בשכבה מסויימת. במאמר מציגים בניית feature space ששומר סטייל של תמונת כניסה, לדוגמא ציור של וואן גוך, מחשבים את הקורלציה בין פיצ'רים שונים בשכבות ואז משחזרים משכבות שונות ברשת זה יוצר תמונות שמתאימות לסטייל של הציור הנתון. קישור לוידיאו שלנו: [smile](#). השתמשנו בקוד עזר שהופיע בגיט והתאמנו לתוצאה הרצויה שלנו. קרדיט מופיע בקוד תחת הקובץ my_vis2vid_ext.

בפונקציית `neural_style_transfer` אנחנו מכניסים את הpath לוידיאו ואת המודל. אנחנו יוצרים את הרשת מתוך המודל באמצעות פונקציית `dnn` (deep neural network) של `cv2`. יוצרים blob מהתמונה ומבצעים `forward pass` ברשת.

לאחר מכן יצרנו פונקציה שמשתמשת בפונקציית הזו כדי להפעיל את המודל על הפריימים מהוידיאו שלנו. תחילה השתמשנו בפונקציה שניתנה לנו על ידי סגל הקורס כדי לפרק את הוידיאו לפריימים, לאחר מכן לכל פריים הפעלנו את המודל לפי חלוקה לזמנים. כלומר פריימים עד שנייה 20 עברו שינוי סטייל לפי המודל הראשון שבחרנו, יותר מ-20 מודל 2, יום מ-45 מודל שלישי ויותר מ-60 מודל רביעי. המודלים נמצאים בתיקיית `my_data` בתוך תיקיית `models`. בחרנו אקראית 4 מודלים. ניתן לראות את היצירות שעל פיהם נכתב המודל ואת הפריימים מהוידיאו שלנו אחרי מעבר ברשת `dnn` עם המודל המתאים.

פריים מתוך הוידיאו המקורי	
	
היצירה שממנה נוצר המודל	פריים לאחר סטייל מהמודל
	
	



נספחים:

[1] תפירת פנורמה של תמונת (Portugal) pena national palace באמצעות SIFT:

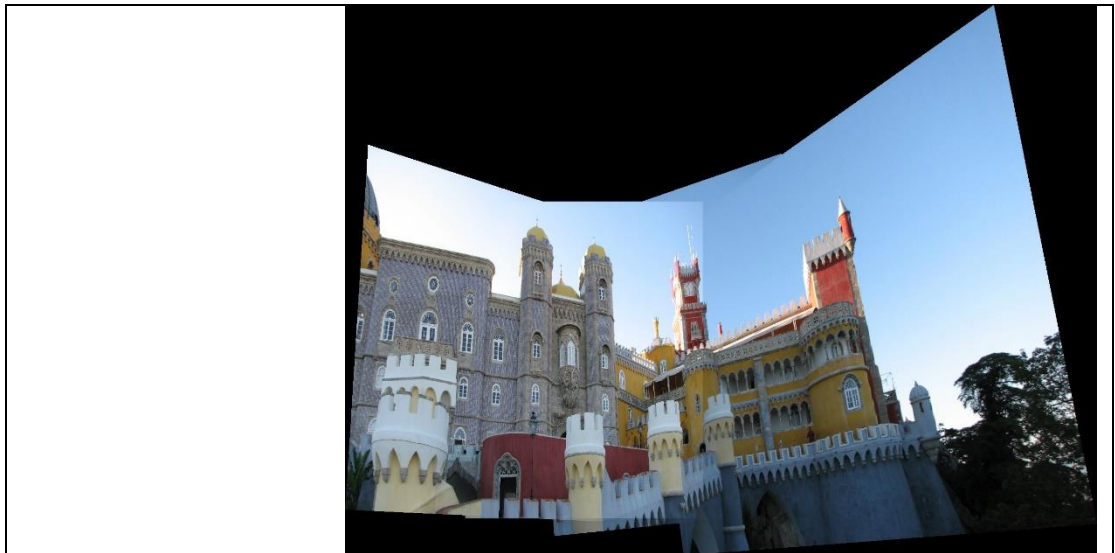
2+3



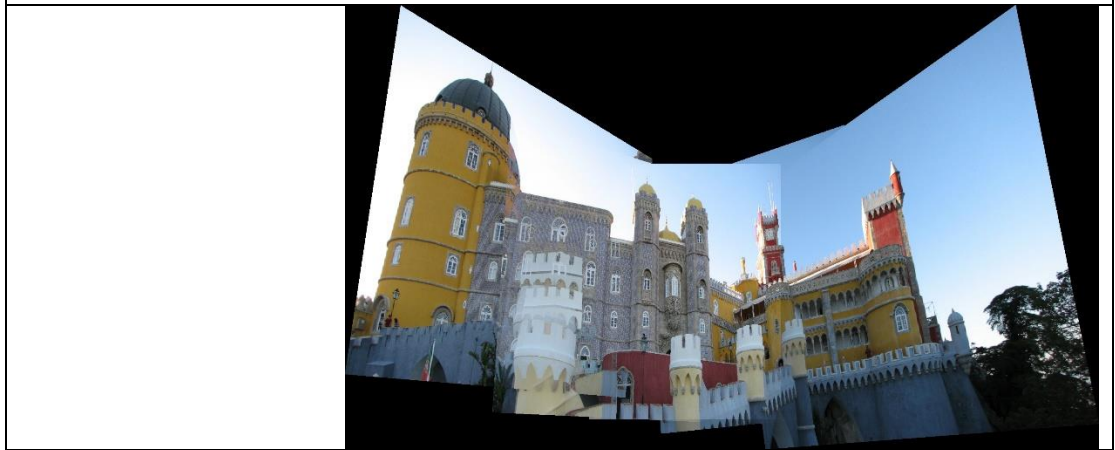
2+3+4



2+3+4+1

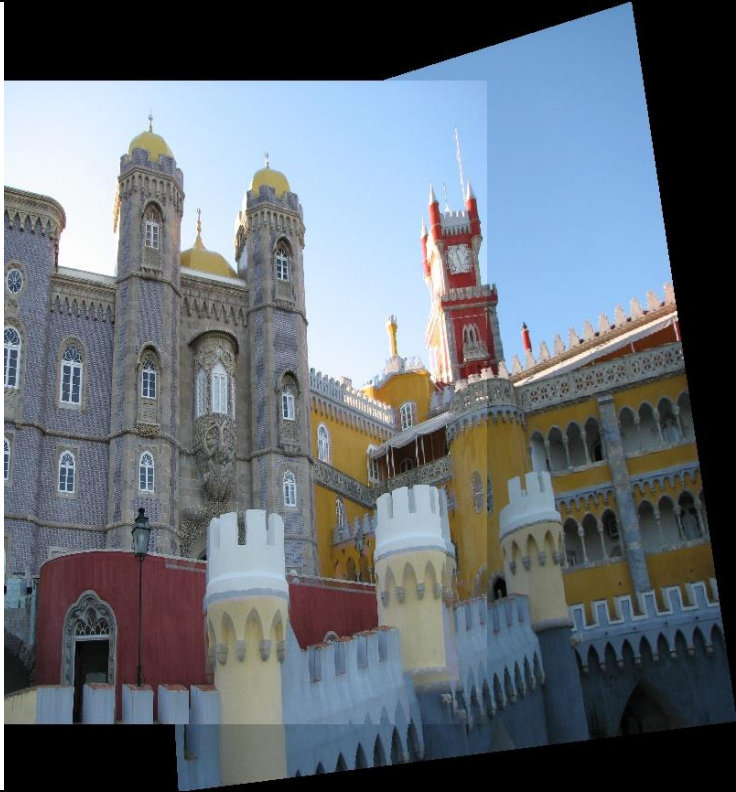


2+3+4+1+5

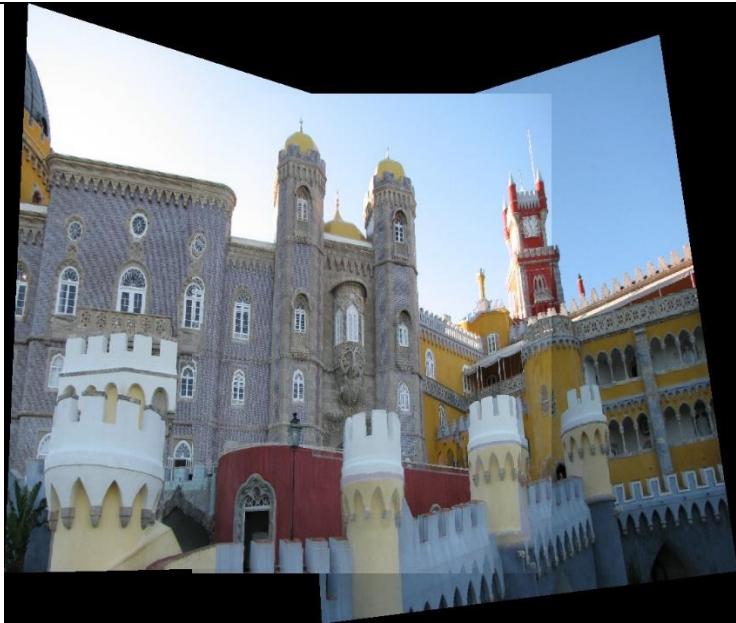


[2] תפירת פנורמה של תמונת pena national palace (Portugal) באמצעות בחירת נקודות ידנית:

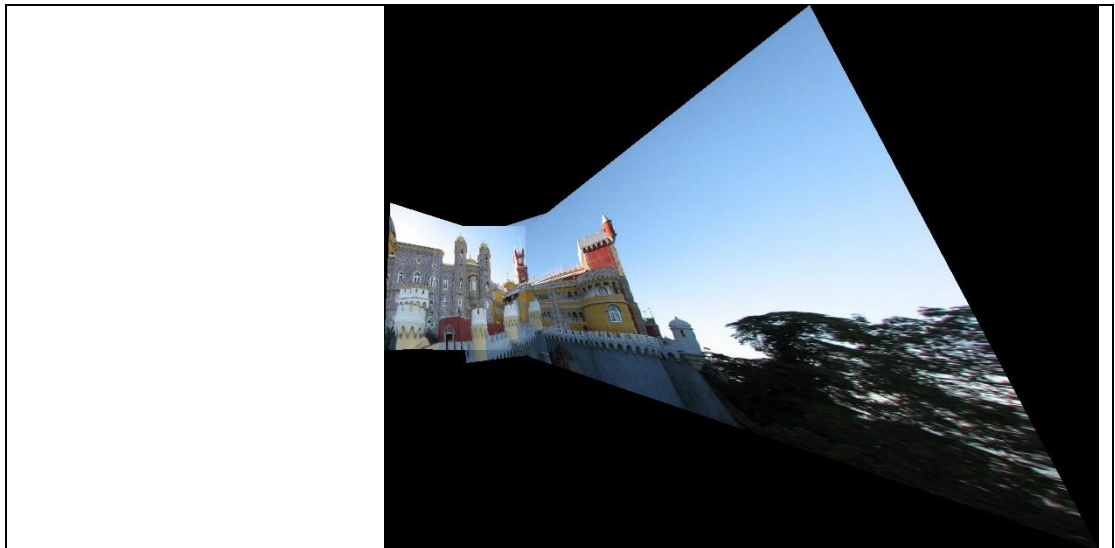
2+3



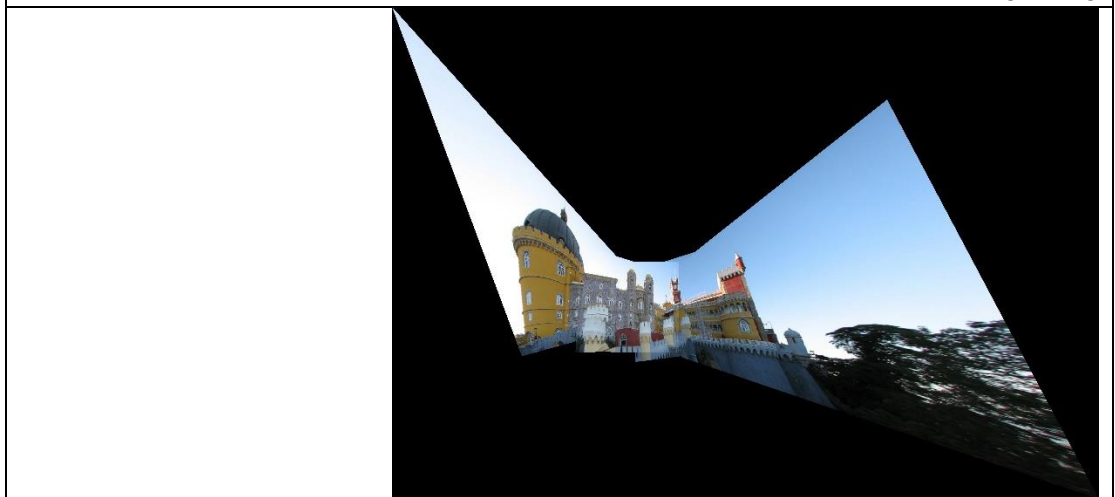
2+3+4



2+3+4+1



2+3+4+1+5

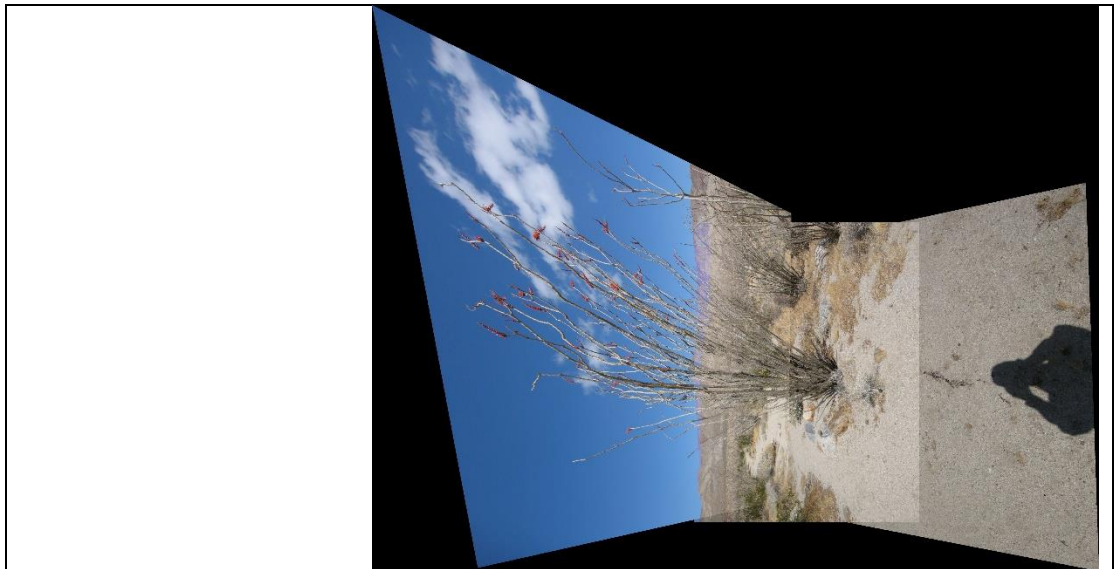


[3] תפירת פנורמה של beach באמצעות SIFT ובלי RANSAC

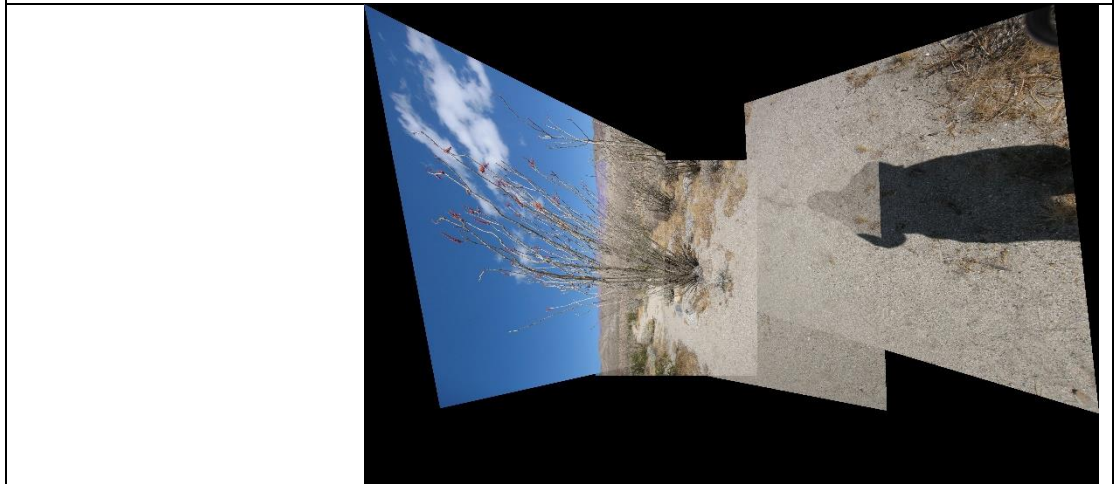
2+3



2+3+4

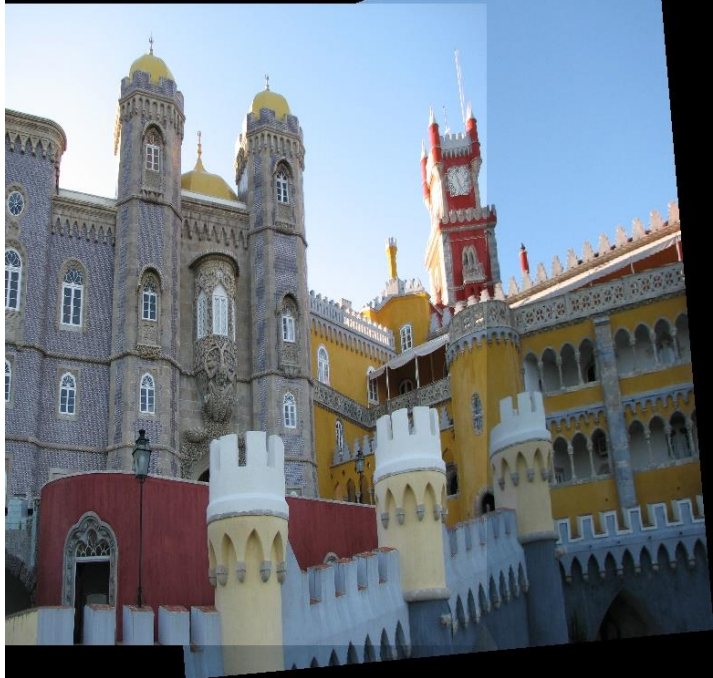


2+3+4+1



[4] תפירת פנורמה של תמונת pena national palace (Portugal) באמצעות SIFT ו-RANSAC:

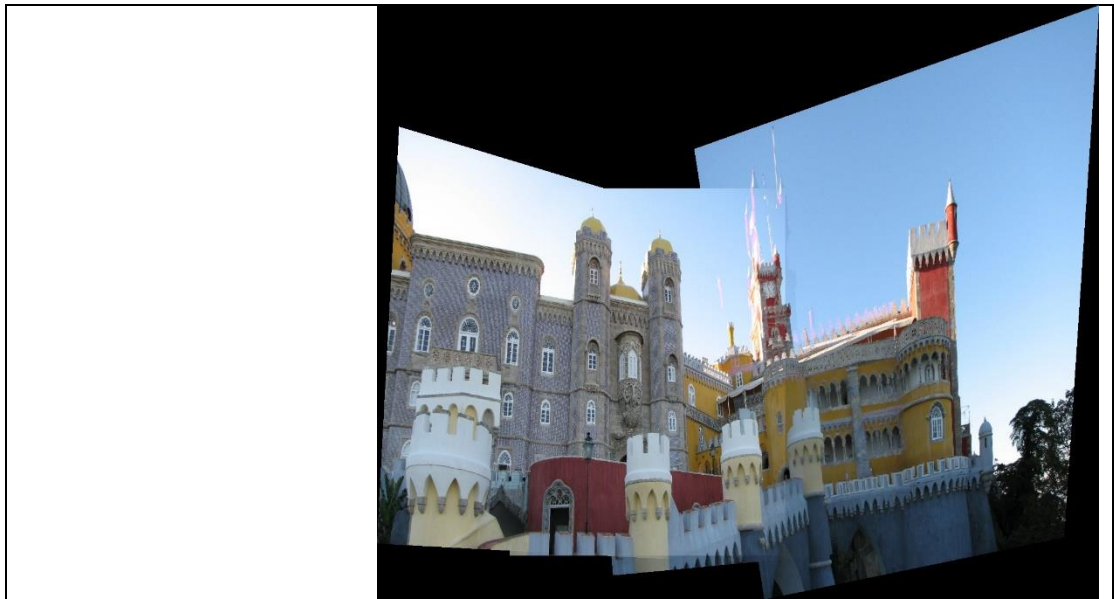
2+3



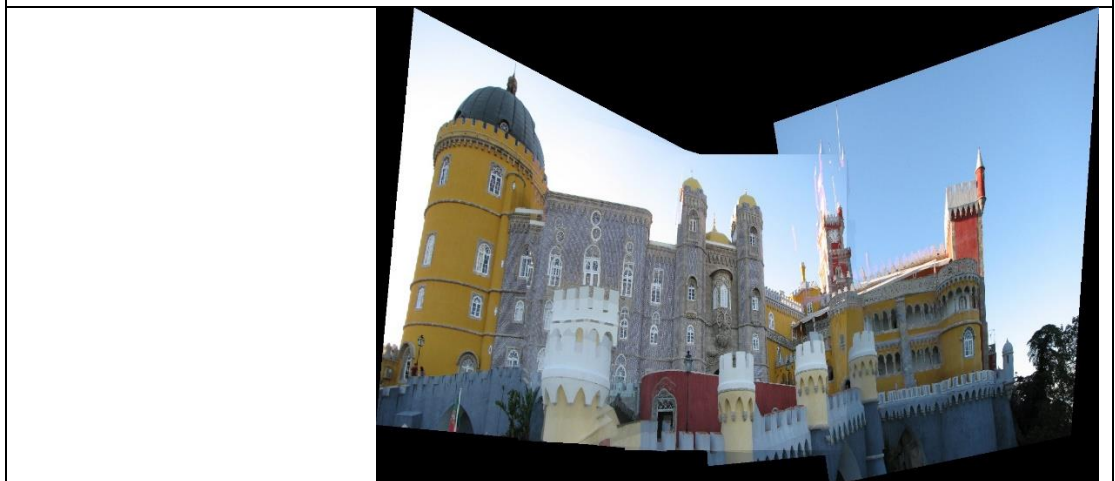
2+3+4



2+3+4+1



2+3+4+1+5

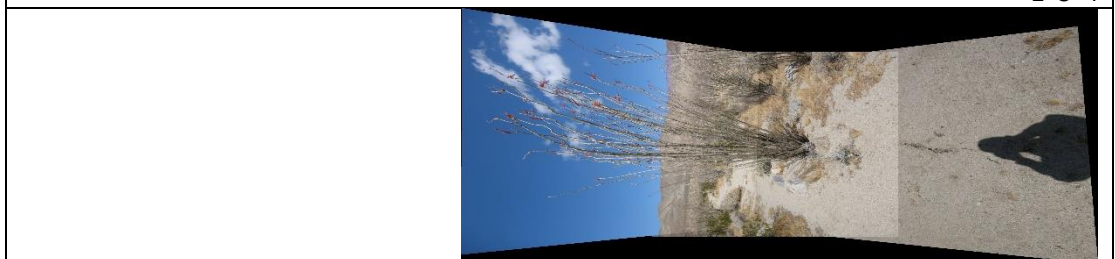


[5] תפירת פנורמה של תמונת beach באמצעות SIFT ו-RANSAC:

2+3



2+3+4



2+3+4+1



נספח [6] שתילת תמונה בתוך תמונה אחרת

דוגמא שנייה: הווארד וולוויץ ושלדון קופר רבים על מהדורה ישנה של קומיקס על משפחת ווסט

scene	Reference
התמונה החדשה:	



דוגמא שלישית: ביל גייטס הצעיר מדגמן בינג' "חברים" במחשב	
scene	Reference
התמונה החדשה:	

