

תוכן עניינים:

| | |
|---------|---|
| 2 | משפטיים והגדרות: |
| 3..... | מצגת 1 : Intensity Transformations, Histogram Equalization, Quantization |
| | Convolution, Filters, Template Matching, Edge Detection, Gradient, |
| | Sobel, Laplacian Zero Crossing (LoG), Image Sharpening, Bilateral filter, |
| 6..... | מצגת 2 : Canny Edge Detection, Hough Transform |
| | Image Pyramids, Aliasing, Gaussian Pyramids, Laplacian Pyramids, |
| 14..... | מצגת 3 : Image Blending |
| 18..... | מצגת 4 : Optical flow, NCC, Lucas Kanade, Iterative LK, Horn-Schunck |
| | Homogeneous Coordinates, Image Warping, Forward Warping, |
| 26..... | מצגת 5 : Backward (Inverse) Warping |
| 30..... | מצגת 6 : Stereo |
| | Homography, Direct Linear Transforms (DLT), Image Rectification, |
| 33..... | מצגת 7 : Image Stitching |
| 38..... | מצגת 8 : SIFT, Parametric fitting, RANSAC |
| | Camera Model, Intrinsic/Extrinsic Parameters, Camera Calibration, |
| 42..... | מצגת 9 : Perspective-n-Point (PnP), Reprojection Error |
| | Reconstruction (In 3D Space), Cross Product, Essential Matrix, |
| 49..... | מצגת 10 : Epipolar Geometry, Fundamental Matrix, Rectification, N-Linearities |
| | Triangulation, Structure From Motion (SfM), Bundle adjustment, |
| 62..... | מצגת 11 : Robust Estimation |
| 66..... | מצגת 12 : Digital Pipeline |

משפטים והגדרות:

1. מטריצה הפיכה נקראת מטריצה רגולרית.
2. מטריצה אינה הפיכה נקראת מטריצה סינגולרית.
3. דרגה של מטריצה מסדר גודל $a \times m$ היא לכל היותר (n, m) min.
4. הגדרה: דטרמיננטה של מטריצה ריבועית, היא סקלר התליי ברכיבי המטריצה.
5. הגדרה: ערך עצמי (Eigenvalue) של טרנספורמציה ליניארית או של מטריצה הוא סקלר כלשהו, כך שקיים וקטור שונה מוקטור האפס (הנקרא וקטור עצמי) שהפעלת הטרנספורמציה עליו, או הכפלתו במטריצה, מכפילה אותו באותו סקלר.
6. במטריצה מסדר גודל $a \times n$ (ריבועית):
 - דרגת המטריצה שווה לח אם ורק אם המטריצה מטריצה רגולרית (הפיכה).
 - דרגת המטריצה קטנה מכך אם ורק אם הדטרמיננטה שווה לאפס.
 - הדטרמיננטה שווה לאפס אם ורק אם המטריצה אינה הפיכה.
 - אם המטריצה הפיכה (בנוסף לכך שריבועית) ומתקיים $A^{-1} = A^T$ אז היא נקראת אורתוגונלית
7. נורמה של וקטור היא פונקציה המקבלת וקטור ומחזירה מספר ממשי. מסומן ע"י $\|V\|$.
ישנים כמה סוג נורמות שיחסובן מתבצע בצורות שונות.
הנורמה מקיימת 3 תנאים:
 - א. חיוביות: $0 \geq \|V\|$. וגם מתקיים $0 = \|V\|$ אם ורק אם $V = 0$.
 - ב. הומוגניות: נניח $R \in a$ אז $\|aV\| = |a| \|V\|$.
 - ג. אי שוויון המשולש: $\|V_2 + V_1\| \leq \|V_2\| + \|V_1\|$ניתן לנורמל וקטור כך שנחלה את הווקטור בנורמה של עצמו. לאחר הנרמול נקבל וקטור ייחידה כך שהכוון שלו הוא אותו כיוון כמו הווקטור שנרמלנו
8. נורמה מוגדרת גם על מטריצות עם תנאים מסוימים.
9. פירוק מטריצה לערכים סינגולריים (SVD - Singular value decomposition) – ניתן לפרק מטריצה לשולשה רכיבים כך שברכיב האמצעי נקבל מטריצה המכילה את הערכים העצמיים באילסון הראשי וברכיב השולי נקבל את הווקטורים העצמיים המתאימים לערכים העצמיים של המטריצה. (גם לרכיב הראשי יש משמעות בפני עצמו. כמו לדוגמה כפ' שנראה במצגת 10 כשרצחה לחלץ [בין היתר] את t ממטריצת essential). נדבר על פירוק זה באלגוריתם DLT במצגת 7.

מצגת 1:

הקורס כולל אלגברה לינארית, מתמטיקה, הסתברות ותכנות.

דוגמא לכלים של עיבוד תמונה: Histogram equalization, Quantization.

עיבוד תמונה זה שיפור התמונה וראייה ממוחשבת זה הבנת הסצנות בתמונה.

ישנם קשיים בהבנת התמונה מכיוון שישנם הרבה רעשיות, חסימות, רקע שימושי, מרווחות בתמונה הנבעות מהתנועה מאוד מהירה של אובייקט ביחס לתהיל הצלום (לדוג' ציפור מנערת את הראש מהר) וכו'

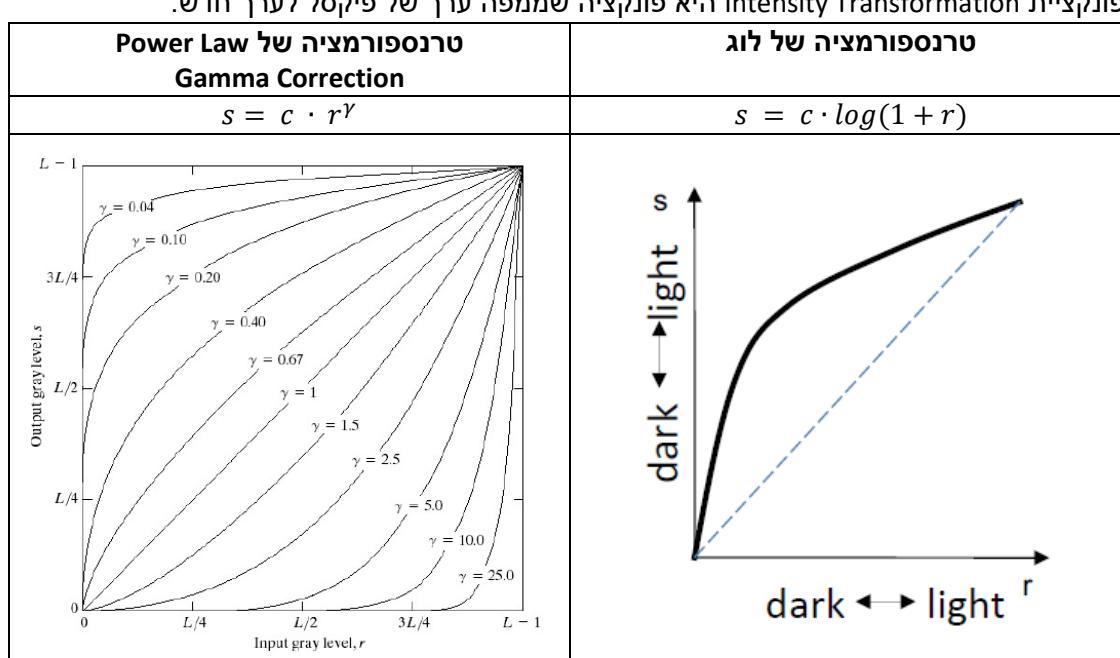
החור הקולט את הקורנים של התמונה נקרא aperture

חולוציה זה מרחב הדגימות, למשל, כמות הפיקסלים.

כמויות הערכים המקסימלית שפיקסל יכול להכיל נקראת Quantization (בדרכ' 255 שדה 8bit).

:Intensity Transformations

פונקציית Intensity Transformation היא פונקציה שmaps ערך של פיקסל לערך חדש:



הען יותר רגישה לצבעים כהים ולכן עדיף להשקיע יותר ביטים לצבעים כהים ופחות לצבעים בהירים.

:Histogram Equalization

ההיסטוגרמה היא תדריות הופעת כל גוון בתמונה. כלומר, מספר הפיקסלים בתמונה עבור כל גוון.

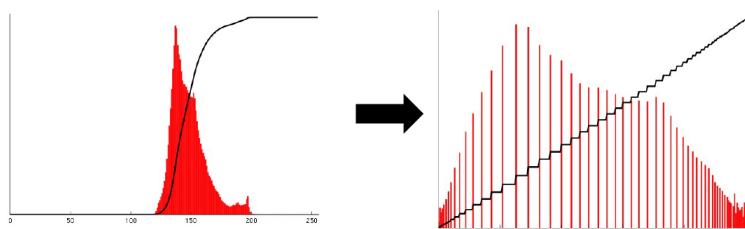
חלוקת הופעת גוון במספר הפיקסלים הכלול היא ההסתברות לקבל את אותו גוון.

אם רצים ניגודיות גבוהה צריכה להיות התפרוס ולא תהיה צפופה. במקרה אחרות זה אומר

יש שינוי חדים בין הצבעים בתמונה.

פונקציית ההצטברות של ההיסטוגרמה היא The Normalized Histogram (CDF).

אם נצליח לגרום לכך של פונקציית ההצטברות להתפרק מהצבע המינימלי עד למаксימלי נקבל ניגודיות גבוהה.



מאפיינים של השוואת היסטוגרמה:

1. ההייסטוגרמה החדשה היא מונוטונית עולה (כי נעשה שימוש בהיסטוגרמה המוצטברת)
 - הטרנספורמציה שומרת על היחס בין כל שני צבעים. (לדוגמא אם מס' הפיקסלים של גוון X גדול ממספר הפיקסלים של גוון Y, אז גם לאחר השוואת היסטוגרמה נקבל שמו' הפיקסלים של גוון X יהיה גדול ממנו הפיקסלים של גוון Y)
2. השוואת היסטוגרמה משנה את היסטוגרמה של התמונה פעמי אחד. אם נפעיל את האלגוריתם יותר מפעם אחת, נחזיר ונקבל את היסטוגרמה המשוועת שקיבלנו אחריו פעמי אחדות של האלגוריתם.
3. לאחר הרצת האלגוריתם, העוצמה של כל גוון היא בערך הסבירות המוצטברת.
 - המשמעות של פיקסל 127 היא שחצי מהפיקסלים קטנים ממנו.
 - לדוגמה, אם אקח את התמונה (אחרי השוואת היסטוגרמה), וכל פיקסל מתחת ל 64 יופיע שחור ומעל היפוך לבן - אך נקבל תמונה שרביע ממנה שחור ורביע ממנה לבן
4. מתי יכשל:
 - כאשר לא צריכים עשר של צבעים, לדוג' טקסט (נתחיל לראות רעש: הסיבים והקימוטים של הדף, נק', שהן מרוחקים בתוך הדיו)
 - אם בתוך תמונה יש הטעויות שונות (ז"א גם חלק של אור וגם חלק של צל)
 - אם בתמונה יש חלקים שנלקחו ממוקורות שונים (לדוג' שילוב בין טקסט לתמונה)
5. היסטוגרמה טוביה היא כאשר לכל רמת אפור יש אותה כמות של פיקסלים
6. אם לפני שינוי היסטוגרמה היי X פיקסליהם הנמוכים מפיקסל מסוים אז הכמות צריכה להישמר גם לאחר השינוי (אם לא כך אז שינויו לגמרי את התמונה)

אלגוריתם:

1. Compute the image histogram
2. Compute the cumulative histogram
3. Normalize the cumulative histogram (divide by the total number of pixels)
4. Multiply the normalized histogram by the maximal gray level value ($K-1$)
5. Round the values to get integers
6. Map the intensity values of the image using the result of step 5.
7. Verify that the minimal value is 0 and that the maximal is $K-1$, otherwise stretch the result linearly in the range $[0, K-1]$.

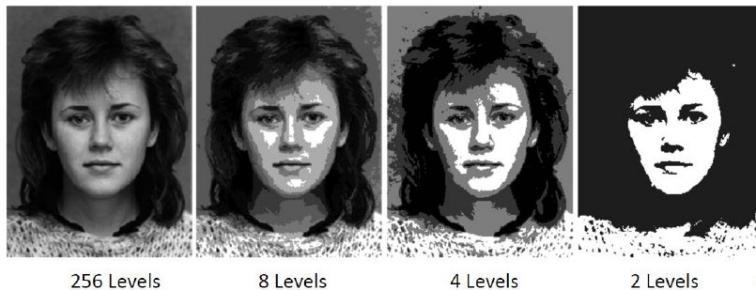
כדי לפתר בעיות של כשלון השוואת היסטוגרמה (שתי ארנו), אפשר לבצע את האלגוריתם הבא:

- רצים על כל הפיקסלים בתמונה
 - לכל פיקסל, מחשבים את היסטוגרמה לפי חלון סביבו (בגודל מסוים), ואז מפעלים את השווון היסטוגרמה רק על אותו פיקסל
 - זהים לפיקסל הבא וחוזרים על הצעד לעיל עבור כל פיקסל בתמונה
- השוואה גלובלית תיצור ממוצע מכל הצבעים מכל התמונה. השוואת חלון תיקח את הממוצע רק באזורי מסוים, ולכן תעבד אחרת על אזורים חשוכים ומוארים (כי הממוצע של כל אחד מהם שונה).



Quantization

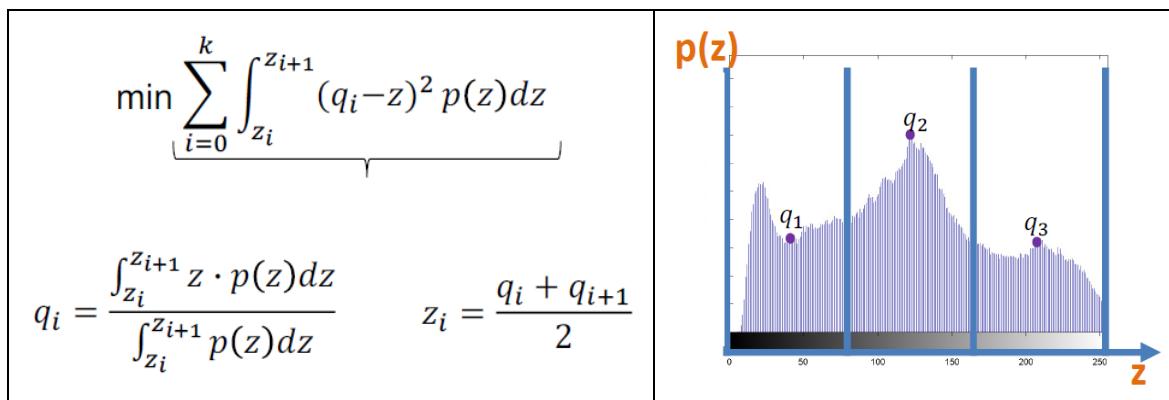
הורדת מספר הפיקסלים בתמונה.
הרעין: חלקה של ההיסטוגרמה ל- K קטעים. כל הגוונים שבאותו קטע ימופו לגוון אחד.



קוונטיזציה של 2 רמות טובת לטקסט (שחור על גבי לבן)
נניח כי ישנו ערci z שמייצגים גבולות כך שכל תחום (בין שני גבולות) ממופה לצבע q יחיד כלשהו
לדוגמא אם נחלק את 255 הצבעים לשולשה חלקים אז ישנו 3+1 גבולות (3 תחומים) ו3 צבעים
(צבע לכל תחום)

נרצה לחשב כל פעם את השגיאה ואז לחשב ולעדכן את q, z בהתאם עד להתקנות של השגיאה.
חישוב השגיאה: עוביים על כל פיקסל ומכפילים את ההסתברות שלו כפול ההפרש שלו עם הq אליו
הוא ממופה.

חישוב z: ממוצע של הצבע q שמתחתיו והצבע q שמעליו (על מנת למזער שגיאה)
חישוב q: הכפלת כל צבע (רק בתחום אותו הq מיפה) בהסתברות שלו ואז מנורמלים



נשים לב שעבור דרגות אפור שכיחות (היסטוגרמה גדולה) מקבל שגיאה גדולה יותר.
שגיאה שהיא אףו - מדובר בתמונה המקורית.

בנוסף, נשים לב כי אם יש תחום ללא פיקסלים אז בחישוב q המתאים מקבל חלוקה ב-0.
וגם, נשים לב ש q הוא ממוצע משוכל של כל גווני האפור באזורי שלו.

למה זו נקבעים ע"י ממוצע?
יש שני קטעים שמייצגים ע"י (i)q, (i+1)q. נניח שהקו (i)z בין שני הקטעים הוא לא באמצע.
אם יש פיקסל שהוא משמאלי ל (i)z אבל עדין יותר קרוב ל (i+1)q הוא יוצג עדין ע"י (i)q כי הוא
משמאלי ל (i)z אבל אמרור להיות מוצג ע"י (i+1)q כדי להקטין את השגיאה.

מצגת 2:

"שניהם כמה סוגים של פעולות על התמונה:

- **פעולות נקודתיות** (מצגת 1): **פעולות על ערכי פיקסל.** היסטוגרמה וכו'.
- **פעולות מרחביות** (מצגת נוכחית): **פעולות על ערכי פיקסל + קורדינטות פיקסל**
- **פעולות גיאומטריות:** **פעולות התליויות בקורדינטות של הפיקסל** (הוזזה וכו').

פילטרים:

פילטר באטען צבע חלון: שינוי פיקסל בהתאם לפיקסלים המקבילים אותו (בטווח החלון)
יכול לעזור בהורדת רעשים, מציאת קצוות, התאמת תבניות (Template Matching)

פילטר Max: לקיחת המקס' שבחלון.

פילטר Min: לקיחת המינימום שבחלון.

פילטר Median: מיען ערכי הפיקסלים לפי הגודל (עם חזרות) ולקיחת הערך האמצעי (כך אפשר להמנע בהשפעת ערכים ענקיים על התוצאה). לא ניתן לבצע באמצעות קונבולוציה (למטה).

פילטר ממוצע: לקיחת הממוצע של כל הפיקסלים בחלון (מטשטש את התמונה יותר ככל שהחלון גדול יותר)



מדיאן מזעיר את סכום ההפרשים האבסולוטיים (SAD)

$$\text{med}(\{I(m, n)\}) = \min_u \sum_{(m, n) \in N} |I(m, n) - u|$$

ממוצע מזעיר את סכום ההפרשים הריבועיים (SSD)

$$\text{mean}(\{I(m, n)\}) = \min_u \sum_{(m, n) \in N} (I(m, n) - u)^2$$

קונבולוציה:

אופרטור ליניארי אשר פועל סכום ממושקל על כל הפיקסלים בסביבה מסוימת ובכך ניתן ליישם פילטר על תמונה

קונבולוציה חד ממדית מתבצעת ע"י הפעכת הווקטור השני (ומשיכתו שמאליה כך שהאיבר הראשון בווקטור הראשון יהיה מול האיבר האחרון בווקטור השני [0 מול 0 כמו בתמונה למטה]) ואז סכום הכפלת ערכי הווקטורים הוא האיבר הראשון בתוצאה. לאחר מכן מזדירים את הווקטור השני (ההפור) ימינה ומבצעים את הסכום שניית לקבלת האיבר השני וכן הלאה נוסחה:

$$h(x) = (f * g)(x) = \sum_{i=-\infty}^{+\infty} f(x-i)g(i)$$

Signal/Image Mask, Filter, Kernel

דוגמה: (חישוב איבר ראשון)

$$\begin{array}{c} 0 \downarrow & & 0 \downarrow \\ f = (0 & 0 & 1 & 0 & 0) & g = (0 & 0 & 1 & -1 & 0) \\ \hline & f = (0 & 0 & 1 & 0 & 0) \\ & g = (0 & -1 & 1 & 0 & 0) \\ \hline \end{array}$$

את הסיגナル משאירים כמו שהוא.

לגביו התמונה (במקרה של חד ממד) ישן כמה גישות לגביו הערכים שמחוץ למערך:

1. ריפוד באפסים

2. מחזוריות של אותו מערך (משורשר לעצמו אינסוף פעם)

3. מראה (הבדיקה המערך ההפכי מימין וגם משמאלי)

(ישן דוגמאות בהמשך כשנדבר על קונבולוציה דו ממדית)

ישן כמה גישות לגביו הקצוות של הווקטור (גודל הפלט):

1. Same: וקטור התוצאה יהיה כגודל חלון התמונה עליה ביצעו את הפעולה

ImageWin + SignalWin - Full .2

ImageWin - SignalWin + 1 - Valid .3

- Option 1: "same" (size A)

$$\begin{array}{ccccccccc} 0 & 0 & 0 & | & 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 1 & | & 4 & 10 & 16 & 22 & 22 \\ \hline & & & * & 1 & 2 & 3 \\ & & & & 15 & 0 & 0 \end{array}$$

- Option 2: "full" (size A + size B + 1)

$$\begin{array}{ccccccccc} 0 & 0 & 0 & | & 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 1 & | & 1 & 4 & 10 & 16 & 22 \\ \hline & & & * & 22 & 15 & 0 & 0 \end{array}$$

- Option 3: "valid" (size A - size B + 1)

$$\begin{array}{ccccccccc} 0 & 0 & 0 & | & 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 1 & | & 10 & 16 & 22 & 22 & 15 \\ \hline & & & * & 0 & 0 & 0 & 0 & 0 \end{array}$$

מכיוון שאנו הופכים את הווקטור השני (סיגナル) אך מקבלים תכונות קומוטטיביות: $A^*B = B^*A$
אם לנחות נקבע קורולציה (מדד סטטיסטי המבטא את המידה שבה שני משתנים קשורים ליניארי)
(כלומר הם משתנים יחד בקצב קבוע) ואז משנה מי מימין מי משמאלי.

תכונות קונבולוציה:

1. קומוטטיביות: $A^*B = B^*A$

2. אסוציאטיביות: $H^*(G^*H) = (F^*G)^*H$

3. חלוקה: $F^*(G+H) = F^*G + F^*H$

דוגמאות:

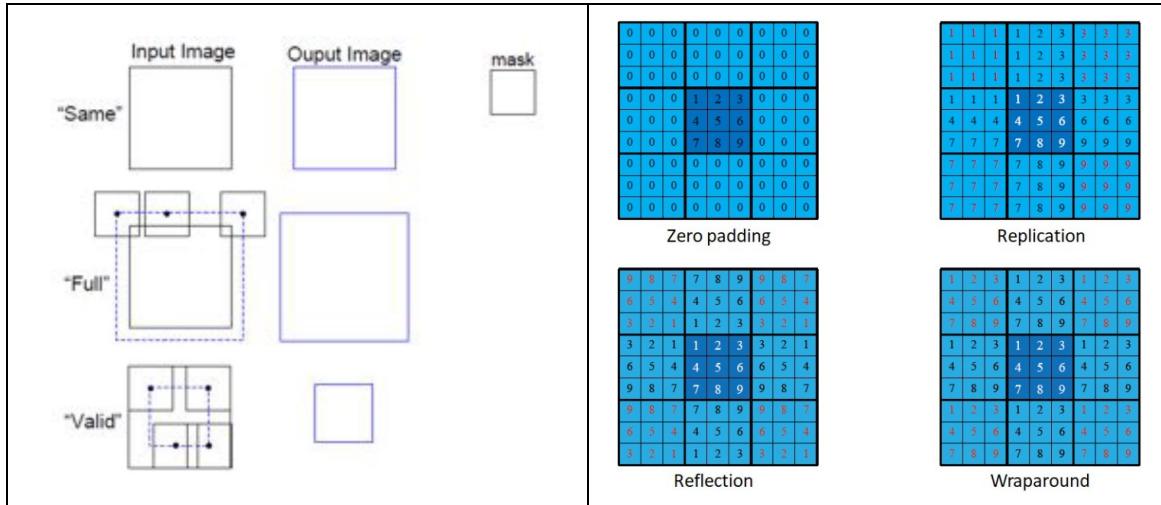
- $f * (1 1) * (1 1) = ? f * (1 2 1)$
- $f * (1 1) * (1 -1) = ? f * (1 0 -1)$
- $f * (1 1)^T * (1 1)^T = ? f * (1 2 1)^T$
- $f * (1 1)^T * (1 -1)^T = ? f * (1 0 -1)^T$

קונבולוציה דו ממדית:

$$f(x,y) * g(x,y) = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} f(x-i, y-j) g(i, j)$$

↑ ↑
Image Mask, Filter, Kernel

גם בד"ד ממד הגישות שהוזכרו לעיל (בחד ממד) קיימות:



Template Matching

בהינתן תמונה וסיגナル (אנו הכוונה לתת תמונה) נרצה לדעת היכן התת תמונה נמצאת בתמונה עצמה נרצה להתמודד עם רעש ותאורה שונה בין התמונה והתת תמונה.

נרצה מודל שמבצע סכימה/הכפלת.

לכן, נרצה להשתמש בחישוב Normalized Cross Correlation NCC שהוא כביכול מנורמל את התאורה כך שנוכל להתגבר על תאורה שונה / רעש.

| NCC בכתיבה רגילה (Zero Mean) | Zero Mean NCC (תוספת לNCC) מנוע שינויים בהאורה [כלומר, מונע שינוי החוספה] של סקלר ע"י הורדת הממוצע עבור כל חלון בהתאם לפניה ביצוע NCC | NCC בהיבט ווקטורית (כל ערך ווקטור אחד מוכפל בערך המקביל בווקטור השני) מנוע שינויים בגודל [הכפלת בסקלר] כי הכפלת 2 וקטורים בקבוע לא משנה זוויות |
|---|---|---|
| $\frac{\sum_{(i,j) \in W} I_1(i,j) \cdot I_2(x+i, y+j)}{\sqrt{\sum_{(i,j) \in W} I_1^2(i,j) \cdot \sum_{(i,j) \in W} I_2^2(x+i, y+j)}}$ | $\vec{\alpha} = \vec{\alpha} - \text{mean}(\vec{\alpha})$ Same for beta | $\cos(\theta) \stackrel{\text{def}}{=} \frac{\vec{\alpha} \cdot \vec{\beta}}{ \vec{\alpha} \cdot \vec{\beta} }$ מנורמים את אורך הווקטורי ע"י חילוק כל וקטור בונורמה (אורך) של רכש קובל ווקטור יחידה באותו כיוון. התוצאה היא הזרווית. |

NCC עובד גם אם התבנית אינה מתאימה לגמר.

NCC פולט ערכים בין -1 ל 1.

כל שהערך יותר גבוה ההתאמה יותר גבוהה.

למעשה:

- ערך 0 מסמל שאין קו רציפה
- ערך קרוב ל 1+ מסמל שיש התאמה גבוהה
- ערך קרוב ל -1- מסמל אינדיקציה שתמונה אחת היא נגדית לשניה

כשעושים קונבולוציה על תמונה בעזרת קרNEL, נשים לב כי הפעולה מתבצעת על כל התמונה עם אותו קרNEL. לכן, כאשר חישוב כל פיקסל תלוי בפרמטרים אחרים (כלומר, צריך לחשב קרNEL ספציפי כל פעם) אז לא נשימוש בקונבולוציה.

Edge Detection

ניתן למצוא את הקצוות ע"י הבחנה בשינוי בין פיקסלים סמוכים

| תמונה מקורית | גזרה בציר ה-x | גזרה בציר ה-y |
|--------------|---|---|
| N/A | $\frac{\partial}{\partial \text{cols}} f(i, j) \approx f(i, j) - f(i, j-1)$ | $\frac{\partial}{\partial \text{rows}} f(i, j) \approx f(i, j) - f(i-1, j)$ |
| N/A | Implement convolution with $(1 \quad -1)$ | Implement: convolution with $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ |
| | | |

מציאת השיפוע (Gradient)
בתחילה מחשבים נגזרת לכל ציר בהתאם.

| נגזרת כיוונית (בהינתן הזרווית) זה בעצם העוצמה בזווית מסוימת | (a) דווית (a) | עוצמה (Magnitude) |
|---|--|--|
| $\cos(\alpha) \frac{\partial f}{\partial x} + \sin(\alpha) \frac{\partial f}{\partial y}$ | $\tan^{-1} \left(\left(\frac{\partial f}{\partial y} \right) / \left(\frac{\partial f}{\partial x} \right) \right)$ | $\sqrt{\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2}$ |

העוצמה המksamלית היא ה"נגזרת כיוונית" עם הזרווית בה יש את השינוי הכי גדול. כלומר, אם ככל שעולים ממעלה מתקבלים תמונה יותר בהירה אך העוצמה המקס' היא בזווית הפונה למטה.

חישוב העוצמה נתונים את הקצוות:

תחילת נחשב את הנגזרות (חישוב כל נגזרת בנפרד מימין)

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right) = \left(\begin{array}{c} \text{Image 1} \\ \text{Image 2} \\ \text{Image 3} \end{array} \right)$$

$$I_x = \text{Image 1} * \begin{bmatrix} 1 & 0 & 1 \end{bmatrix} = \text{Image 4}$$

$$I_y = \text{Image 1} * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} = \text{Image 5}$$

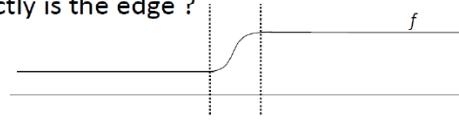
לאחר מכן, נחשב את העוצמה בהינתן הנגזרות (כמו שכתוב בנוסחה לעיל)

$$\sqrt{|I_x|^2 + |I_y|^2} = \text{Image 6}$$

* שימוש בקרנל $[1 \ 0 \ -1]$ מאפשר התחשבות בקצוות עבים יותר מאשר $[1 \ 1 \ -1]$

מה קורה אם גוזרים תמונה עם רעש?
הנגזרת היא השינויים שיש בתמונה, כיש רעש מקבלים נגזרת רועשת
בנגזרת ראשונה - הקצה נמצא בנקודות המקסימום
(zero crossing) בנגזרת שנייה - הקצה נמצא בנקודות האיפוס (zero crossing)

Where exactly is the edge?



Maximum of f'



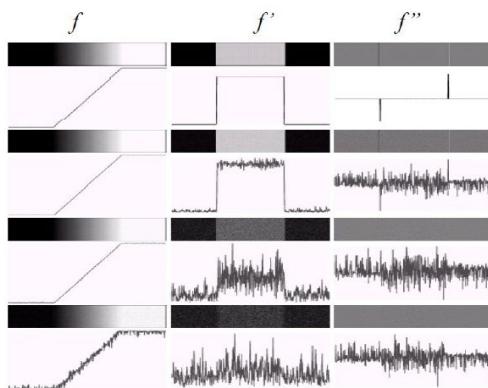
Zero crossing of f''



כל שיש רעש יש השפעה על הנגזרת הראשונה והשפעה יותר גדולה על הנגזרת השנייה:

Effect of Noise on Derivatives

Derivatives amplify noise



לכן, על מנת להימנע מרעש מרצו להחליק את התמונה באמצעות קרמל גאושיאני
מצד שני, לא נרצה להחלק את הקצוות (יקשה علينا למצאו אותם)
:**Sobel**

פילטר Sobel - פתרת הבעה הנ"ל ע"י חילוק התמונה בציר אחד וחיפוש הקצוות עם נגזרת שנייה
בציר השני (ולחפן).

לדוגמא, החילוק בציר ה- y ומציאת קצוות בציר ה- x :

$$f * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} = f * \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

בפילטר Sobel לא מכפילים את המטריצה (הימנית) ב-1/8 [אין צורך בכך כמשמעותם קצוות] אבל זה
נוצר אם נרצה לקבל את ערך השיפוע הנכון.

Laplacian Zero Crossing

מציאת הקצוות באמצעות הנגזרת השנייה לאחר החלקה גאוסיאנית.
כלומר, לאחר החלקה גאוסיאנית נבצע את הפילטר הבא: (אחד נגזרת שנייה של כל ציר בנפרד)

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

אלגוריתם LoG:

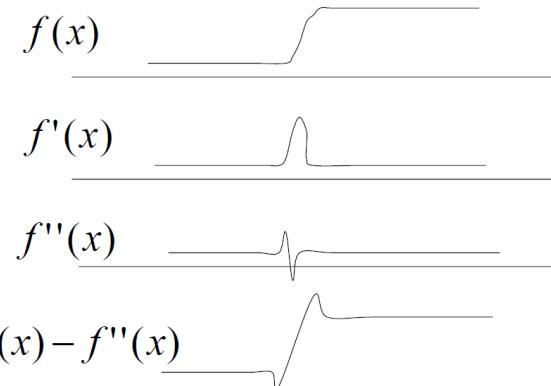
LoG: Laplacian of Gaussian

- Smooth with 2D Gaussian
- Apply Laplacian filter
- Look for patterns like {+,0,-} or {+,-} (zero crossing)
- Also known as the Marr-Hildreth Edge Detector

חסרון: התנוגות שגוייה בפיניות

Image Sharpening (חידוד תמונה)

חידוד תמונה מתבצע ע"י הבלטת הקצוות של התמונה.
לכן, חידוד תמונה יכול להתבצע ע"י החסרת הנגזרת השנייה מהתמונה



זאת מכיוון שכשהשינו בתמונה מתחילה לעלות איזה הנגזרת השנייה עולה בקיצוניות ולכן שמחסרים מהם מקובלים ערך ממש נמוך. לעומת זאת, כשהשינו בתמונה מתחילה לרדת איזה הנגזרת השנייה יורדת בקיצוניות ולכן לאחר שמחסרים מהם מקובלים ערך ממש גבוה

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

בעיות שיש בהחלקה תמונה:

Mean (Box filter): מטשטש תמונה, מסיר רעשים פשוטים, לא נשמרים פרטיים.

Gaussian: מטשטש תמונה, שומר פרטיים רק עברו ס קיטן.

Median: שומר על כמה פרטיים, טוב בהסרת רעשים חזקים.

פילטר איזוריים: Bilateral

מחליק איזוריים (מסיר רעשים) ובו זמנית שומר על הקצוות (אינו מחדד אלא רק שומר) הרעיון שעומד מאחורי זה הוא התחשבות ערכי הפיקסלים + המרחק מהם התחשבות בערכי הפיקסלים - משמר את הקצוות התחשבות במרחב של הפיקסלים - מסיר רעשים

$$I'(u) = \frac{\sum_{p \in N(u)} e^{-\frac{\|u-p\|^2}{2\sigma_c^2}} e^{-\frac{|I(u)-I(p)|^2}{2\sigma_s^2}} I(p)}{\sum_{p \in N(u)} e^{-\frac{\|u-p\|^2}{2\sigma_c^2}} e^{-\frac{|I(u)-I(p)|^2}{2\sigma_s^2}}}$$

$N(u)$ – The neighbor size of the filter support,

σ_c – The variance of the spatial distances,

σ_s – The variance of the value distances,

המכנה בביטוי לעיל נועד בשבייל לנורמל ל-1.

ניתן לפתור את הביטוי הנ"ל ע"י הכפלת בקרNEL מתאים

הקרNEL ישתנה עבור כל פיקסל בנפרד כאשר האמצע שלו הוא המקס'. סכום הkernel 1 בגל הנירמול.

Canny Edge Detection

הס'ינון הוא לינארי (עם פילטר). מניחים תוסף רעש גאוסיאני. מניחים שהקצוות מוארכים ומתחשכים.

החלוקת נוספת תשפר את היזהוי אבל תמנע לוקלייזציה

נרצה למצוא קצה (ולא רעש) שנמצא ליד קצה אמיתי. בנוסף, נרצה תגובה אחת לכל קצה. השוני בעחנון הוא מציאת קצוות מאורכים (שיור כל פיקסל לקצה מסוים)

אלגוריתם:

1. חלוקה גאוסיאנית
2. ייצרת נזרת לכל ציר x, y בהתאם
3. חישוב magnitude (עוצמה) וגם direction (כיוון) באמצעות נזרות
4. לכמת (quantize) את כיוונו (gradient) השיפוע (כלומר, לחלק לקבוצות בגל שיש רעש) [לדוגמא כל 45 מעלות זה יחשב ככוון אחד]
5. סינון קצוות ע"י Non Maximum Suppression מה שמייר כל קצה מוטשטש בתמונה הכוון (gradient) לקצה חד יותר.

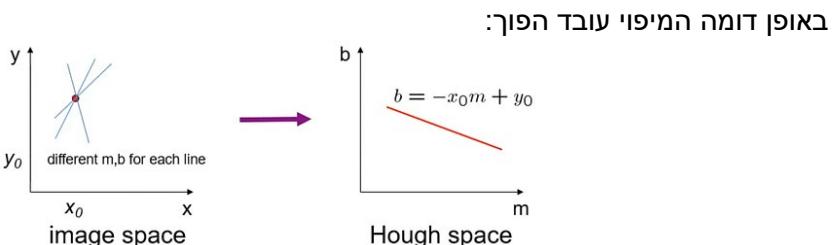
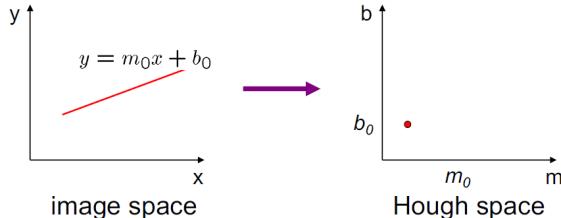
עושים זאת בצורה הבאה:

בתחילתה מעגלים את היזוית [לדוגמא, מודולו 45] (כיוון שעשינו כימות בשלב 4) אחר כך, משווים את ערך הפיקסל עם ערכי הפיקסלים השכנים שלו שנמצאים באותו כיוון. אם הפיקסל הוא maxima local אז נשאיר אותו (כי הוא מועמד להיות קצה), אחרת, נמחק (נאפס) אותו. (הפיקסלים הקרובים [שאינם קצוות] בכיוון הgradient יכולים לטעטש את הקצה).

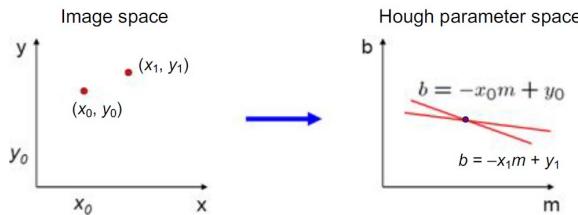
6. בשלב זה, רוצים לטפל בפיקסלים שנשארו. ככל, ננסה לקבל אינדיקציה על הפיקסלים בהתאם לעוצמה שלהם: עבורו על כל הפיקסלים עם טרשאולד $T_2 > T_1$. כל הפיקסלים שגבויים מהערך T_1 הם בהחלט קצוות. אחרת, כל פיקסליהם שהם גבוהים מהערך T_2 וגם מחוברים לקצה, יחויבו גם קצה. אחרת (ערךם קטן מהערך T_2), הם בהחלט לא קצה.

Hough Transform

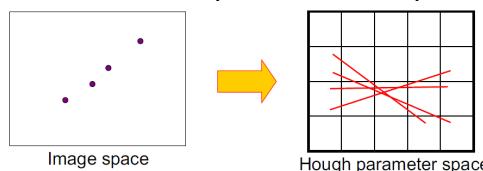
אפשר לזרות עצמים בתמונה כאשר העצמים תלויים בפרמטרים (לדוגמא, קו התלו依 ב-2 פרמטרים) נניח ואנו רצים למצאו קו בתמונה. אז נוכל לעבור על הקצוות של התמונה ואז בכל קצה נחשב את היזוית ובעזרתה לחשב מה הקו אותו היא יוצרת (מאונך לויזוית). לאחר מכן, אם הקו הינו $b + mx = y$ אז הפרמטרים המיצגים את הקו הינם m ו- b . את הפרמטרים היוצרים את הקו נשמר במרחב אחר (מצד ימין בתמונה למיטה) בנקודה (b, m). אם ישנו קו מתרחש בתמונה השמאלית אז נקבל local maxima בנקודה (b, m) וכך נוכל לדעת שמצאנו קו.



אם יש לנו 2 נקודות במרחב התמונה אז כל נקודה תיצור קו במרחב הימני. בנוסף, הפרמטרים של הקו (b, m) שחוצה את 2 הנקודות משמאל הם הנקודה של חיתוך 2 הקווים במרחב הימני.



באלגוריתם נחלק את הפרמטרים לשיבובים (נתיחה לערכים קרובים כאחד) בגל רעשיהם.
כל Bin שהוא local maxima מיצג קו. לדוגמא, עבור קו ספציפי:



חסרונות: לא תמיד אפשר ליזג קו ע"י b, m (נראה לדוגמה קו אנכי שמציריך m אינסופי).

“ציג קו בצורה אחרת: (+ פסודו אלגוריתם)

Basic Hough transform algorithm

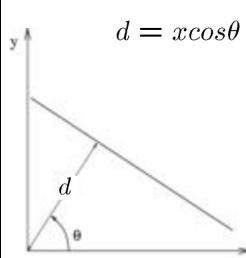
1. Initialize $H[d, \theta] = 0$
2. for each edge point $I[x, y]$ in the image
 - for $\theta = 0$ to 180

$$d = x\cos\theta + y\sin\theta$$

$$H[d, \theta] += 1$$
3. Find the value(s) of (d, θ) where $H[d, \theta]$ is maximum
4. The detected line in the image is given by

$$d = x\cos\theta + y\sin\theta$$

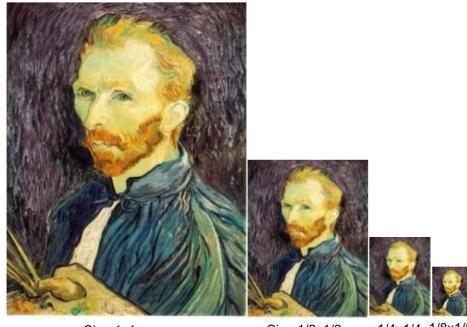
d is the perpendicular distance from the line to the origin
 θ is the angle this perpendicular makes with the x axis



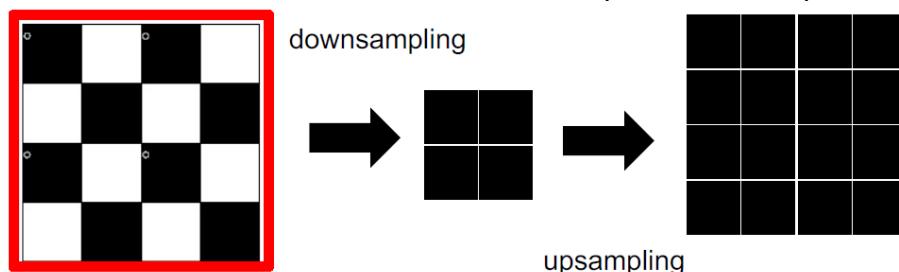
מצגת 3:

:Image Pyramids

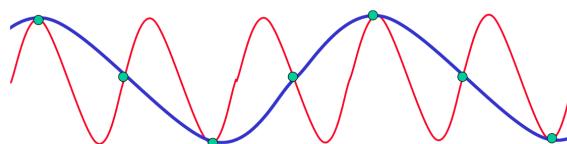
אוסף של תמונות בגודלים שונים (כל תמונה גדולה פי 2 באורך ופי 2 ברוחב מהקודמת)



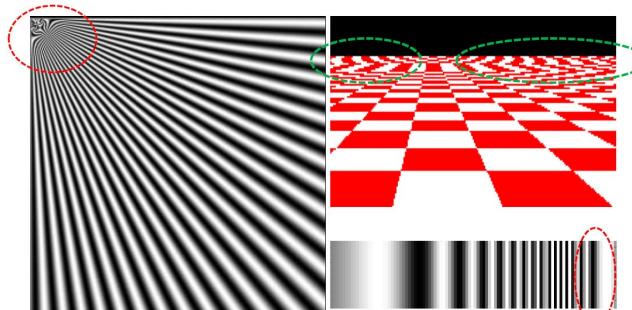
על מנת ליצור פירמידה נרצה לקחת כל פיקסל בדילוגים של 2 פיקסלים בעמודות וכן בדילוגים של 2 פיקסלים בשורות. הבעיה היא שמכיוון שהדgelות שונות ואורכי הגל בתמונה שונים אז קיבל תוצאה שאגיה (aliasing) אומר שאיננו יכולים לשחרר מששו דומה. לעומת זאת, כאשר משווה שלא קיים במצבים, תוצאה מתדר דגימה גדול מדי. מרחק הדגימה צריך להיות חצי מאורך הגל.



כאשר דוגמים, יש להתאים את הקצב כדי ללקוד את כל השינויים באות קצב הדגימה בכל רמה בפירמידות תמונה הוא $\frac{1}{2} \times \frac{1}{2}$. קצב הדגימה לאחר 3 רמות ביחס לתמונה המקורית הוא $\frac{1}{8}$ על $\frac{1}{8}$ לאחר מספר רמות, aliasing כמעט בלתי נמנע.



לדוגמה: המיציאות היא הסינוס האדום, דגמנו מתחכו את הנוקודות היירות בתדר דגימה גדול מדי, כך שאפשר להתבלבל ולהשוו שהפונקציה המקורית היא הסינוס הכחול.



לכן, נרצה שכל פיקסל אותו דוגמים, יוכל מידע של הפיקסלים הקרובים אליו (כולל הפיקסלים שלא דוגמים אותו) לפני שבכל דוגמים. זהה מאפשר בעזרת פירמידה גausian (Gaussian Pyramids) (להלן).

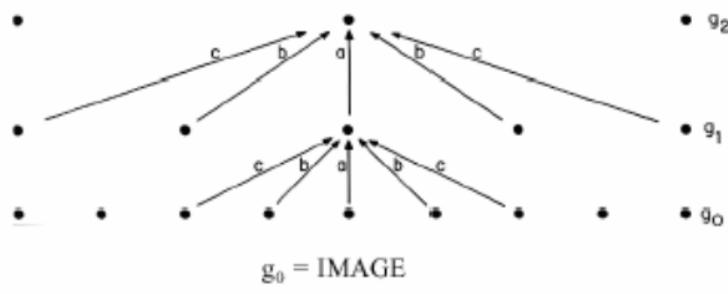
:Gaussian Pyramids

אלגוריתם:

- We define the Reduce operation
- Input: Image NxM, Output: Image N/2 x M/2
- The Reduce Operation:
 - Blur
 - Convolve with a 5×5 gaussian filter
 - Sub-sample
 - Select only every 2nd pixel in every 2nd row

כל פיקסל תורם מידע לפיקסלים ברמה הבאה

GAUSSIAN PYRAMID



כל רמה מחושבת בעזרת קונבולוציה אחת.
סיבוכיות המוקום: $N \times N (1 + 1/4 + 1/16 + \dots) = 4/3 * N \times N$
על מנת לשחרר תמונה גדולה מתמונה קטנה נctrar לעשות

The Expand Operation:

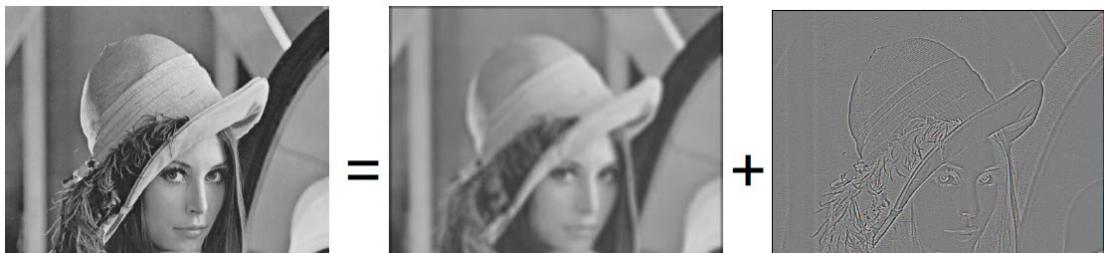
- Zero Padding (a1, 0, a2, 0, a3, 0, ...)
- Blur
 - Note: Blur needs different normalizations!
 - What is zero padding followed by blur with $1/2(1,2,1)$

לדוגמא: (חדר-ממד)

| | | | | | | | | |
|---|---|---|-----------|-----|-----------|---|---|---------|
| 2 | 3 | 4 | $4^{3/4}$ | 5.5 | $4^{3/4}$ | 3 | 2 | |
| 0 | 2 | 0 | 4 | 0 | 5.5 | 0 | 4 | $1/2$ |
| 1 | 2 | 3 | 4 | 5 | 6 | 5 | 4 | $* 1/4$ |

נרטול הkrnel מתבצע בצורה שונה בהתחשב בכך שישנם ערכים מאופסים.
בדוגמא לעיל הנרטול של הkrnel אינו $1/4$ אלא $1/2$ כיון שמכפילים את הנרטול ב $2 * (1/4 * 2) = 1/2$.

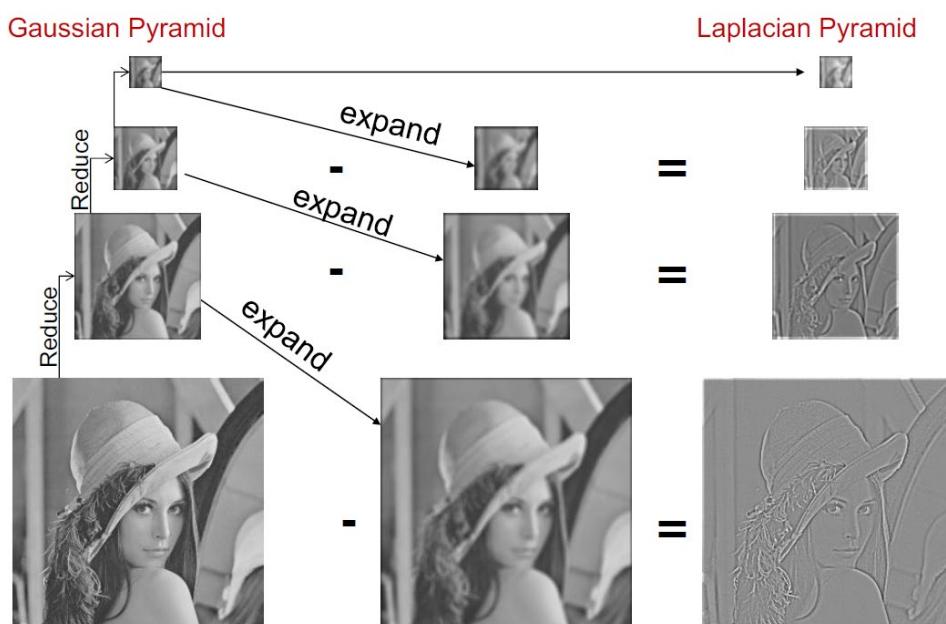
נגיד: Pyramid Level Difference - תמונה הפרש בין תמונה המקור בrama X לבין התמונה בrama 1-X (התמונה בrama 1-X קטנה יותר אז עושים לה Expand [כלומר, כעת שנייהם אותו גודל])



(כאן, התמונה השמאלית היא התמונה המקורי בrama X, התמונה האמצעית היא התמונה מהrama 1-X [לאחר Expand] והתמונה הימנית היא תמונה ההפרש)

כעת, בהינתן פירמידת גאוסיאן נוכל ליצור פירמידה של Pyramid Level Difference כאשר בrama הci' קטנה נשאיר את התמונה שהתקבל מפירמידת הגאוסיאן. הפירמידה המתקבלת נקראת לפולסיאן.

השחזר של התמונה יבוצע בצורה הבאה: (פירמידת לפולסיאן לתמונה המקורי)



דחיסת פירמידה:

אם ניקח את תמונות פירמידת הלפלסיאן ונעשה להם קוונטיזציה לכמה צבעים מצומצמת אז נקבל תמונה שדומה מאוד לתמונה המקורי.
ישיחס לא פרופורציוני בין צבעים שונים (בהסתברות גבוהה יש יותר אפסים) וזה אפשר לשימוש בהופמן ולשמור את התמונה בפחות מקום.

מיזוג שתי תמונות בצדקה חלקה:
Image Blending



כשנמzag 2 תמונות נרצה למנוע **seams** (תפרים) בין 2 התמונות:

$\text{window} \geq \text{size of largest prominent feature}$

[אם יש שניי ממשמעות נרצה לטשטש את אחד ולא חלק ולכן גודל החלון גדול ממנו]

כשנמzag 2 תמונות נרצה למנוע **ghosting** (חפיפה גדולה מדי) בין 2 התמונות:

$\text{window} \leq 2 * \text{size of smallest prominent feature}$

[נרצה גודל חלון קטן פי 2 מהשיני הכי קטן (קשרו לתדרי התמונה אמר שלא יכנס לזה)]

(באופן כללי, גודלי החלונות תלויים בתדרים, אמר שזה קשרו לתורת הדגימה ולא יכנס לזה).

נרצה לדעת מה גודל החלון האידיאלי עבור המיזוג הספציפי זהה בהתאם לתדרים של התמונה

שימוש בפירמידות יכול לעזור לנו לעשות מיזוג (Blending)

אלגוריתם:

- Given two images A and B , and a binary image mask M
- Construct Laplacian Pyramids L_a and L_b
- Construct a Gaussian Pyramid G_m
- Create a third Laplacian Pyramid L_c where for each level k

$$L_c(i, j) = G_m(i, j)L_a(i, j) + (1 - G_m(i, j))L_b(i, j)$$

- Reconstruct all levels L_c in to get the blended image

למה זה באמת עובד?

עושים פירמידה לפולסיאנית (קצוות) לתמונה A

עושים פירמידה לפולסיאנית (קצוות) לתמונה B

עושים פירמידה גאוסיאנית (טשטוש) לMask

נשים לב (בתמונה של Lena לעיל) שככל שיורדים יותר ברמות (תמונות קטנות יותר) בפירמידה של הפלסיאן, קיבל את הקצוות הכי משמעותיות.

לעומת זאת, בפירמידה הגאוסיאנית של MASK כל שיורדים יותר ברמות קיבל יותר טשטוש.

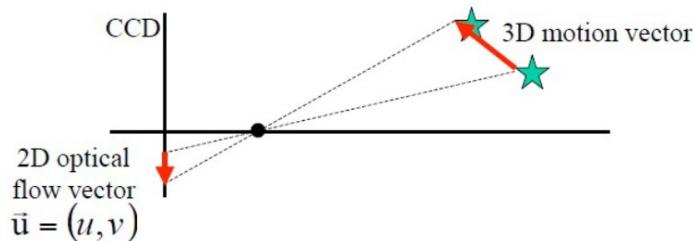
ואז יוצא שנקבל חפיפה גדולה יותר עבור קצוות משמעותיים יותר (חלון גדול יותר עבור קצוות משמעותיים יותר)



מצגת 4:

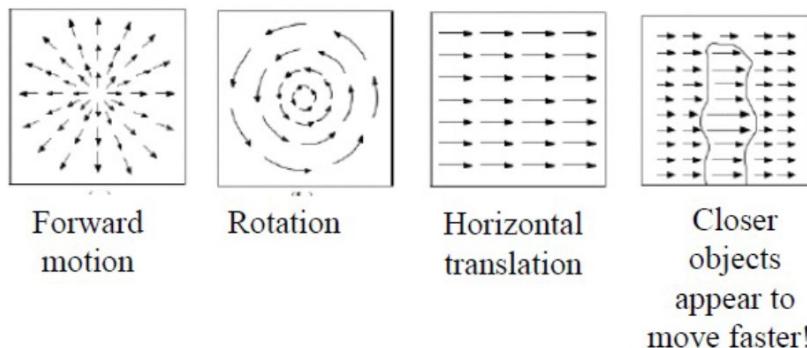
:Optical flow

זרימה אופטית (Optical flow) - תנועת פיקסלים בתמונה, שהיא תוצאה של הקרןת שדה תנועה



בහינתן 2 תמונות המציגות תנועה, עברו על פיקסל יהיה וקטור המציג את ציוון התנועה שלו ואת המהירות.

תמונות מגיבות לתנועות מסוימות בצורה שונה:



ישנן מספר שיטות לחישוב תזוזה בין שתי תמונות:

1. שיטות ישירות:

- CC (קורוס קורלציה), NCC (גרסה מנורמלת)

Lucas Kanade

- אלגוריתם Horn-Schunck (HS) (הרחבת לאלגוריתם LK)

2. מציאת פיצ'רים: (נדבר על זה במצגת 8)

- SIFT (חישוב נק' עניין + יצרת descriptors) [לא הרחכנו בהרצאה]
- RANSAC (לבחור מה ההתאמות הנכונות בין נק' העניין ומה ההתאמות הלא נכונות)

אם נניח שהבהירות בין 2 התמונות אינה משתנה (Brightness constancy) אז נוכל למצוא תזוזה ביצירים המוצגים ע"י (u, v) באמצעות שיטה ישירה.

:NCC

תחליה, השתמש בשגיאת הריבועית (SSD):

$$E(u, v) = \sum_x \sum_y (I_1(x, y) - I_2(x+u, y+v))^2$$

[ונכל גם למצוא סיבוב של התמונה. כלומר, נחפש (u, v, α) .]
מכיוון שמתק"ים:

$$(a-b)^2 = a^2 - 2ab + b^2$$

ונכל לכתוב את משוואת SSD לעיל בצורה הבאה:

$$E(u, v) = \sum_x \sum_y I_1^2 - 2 \sum_x \sum_y I_1(x, y) \cdot I_2(x+u, y+v) + \sum_x \sum_y I_2^2$$

כעת, נשים לב כי הפרמטר הראשון והאחרון כמעט קבועים ולכן למזער את SSD נרצה למקסם את הביטוי:

$$C(u, v) = \sum_x \sum_y I_1(x, y) \cdot I_2(x+u, y+v)$$

שזהו ההסכמה Cross Correlation (כפי שראינו במצגת 2 בנושא).
על מנת להימנע מרושן הנובע מהכפלת בסקלר נוספת סקלר נורמל ונשתמש בZero Mean:

$$NC(u, v) = \frac{\sum (I_1(x, y) - \hat{I}_1) \cdot (I_2(x+u, y+v) - \hat{I}_2)}{\sqrt{\sum (I_1(x, y) - \hat{I}_1)^2} \sqrt{\sum (I_2(x, y) - \hat{I}_2)^2}}$$

שזהו NCC Zero Mean (כפי שראינו במצגת 2 בנושא).

בהינתן 2 תמונות נחפש באופן בדיד כל אפשרות של תזוזה. הסיבוכיות משתנה בהתאם לפרמטרים:
אותם אנו מחפשים, ולכן:

Translation: (u, v) Complexity is N^2

Rotations: (u, v, α) Complexity is N^3

Zoom: (u, v, α, s) Complexity is N^4

:Lucas Kanade

אם נסיף הנחה שהתמונה משתנה מעט מאוד בסדר גודל של עד פיקסל אחד (די הגיוני מכיוון שבדרך כלל ישנו 30 תמונות בשנייה ולכן התזוזה בין כל 2 תמונות היא קטנה), אז יוכל לשפר את החיפוש עם קירוב טיילור.

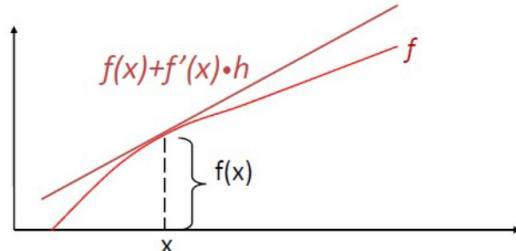
תזכורת טור טיילור (פיתוח סביב x_0):

$$f(x) = f(x_0) + f'(x)(x - x_0) \approx f(x_0) + f'(x_0)(x - x_0)$$

השווות קירוב טיילור (1D מול 2D):

- Local Taylor approximation in 1D:

$$f(x+h) \approx f(x) + f'(x) \cdot h$$



- Local Taylor approximation in 2D for images:

$$f(x+u, y+v) \approx f(x, y) + \frac{\partial f}{\partial x} \cdot u + \frac{\partial f}{\partial y} \cdot v$$

כעת, נרצה למזער את השגיאה (MSE) ע"י קירוב טיילור. כלומר, נרצה למזער את:

$$E(u, v) = \sum_x \sum_y [I_2(x+u, y+v) - I_1(x, y)]^2$$

ולכן לאחר שנงזר, ע"פ קירוב טיילור נקבל:

$$\begin{aligned} E(u, v) &= [I_2(x+u, y+v) - I_1(x, y)]^2 \approx \\ &[I_2(x, y) + \frac{\partial I_2}{\partial x} \cdot u + \frac{\partial I_2}{\partial y} \cdot v - I_1(x, y)]^2 = \\ &(I_x \cdot u + I_y \cdot v + I_t)^2 \end{aligned}$$

$$\text{where } I_x = \frac{\partial I_2}{\partial x}; \quad I_y = \frac{\partial I_2}{\partial y}; \quad I_t = I_2 - I_1;$$

כלומר, נרצה למזער את:

$$E(u, v) = \sum_{x,y} (I_x u + I_y v + I_t)^2$$

ומכאן, המשוואה אותה אנו מ Chapman לפתרון היא:

$$I_x u + I_y v = -I_t$$

ונכל לפטור בצורה הבאה:

$$I_x u + I_y v = -I_t \quad \rightarrow \quad \begin{bmatrix} I_x & I_y \\ & \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -I_t$$

כאשר, נזכיר כי:

I_x : The x derivative of image I_2

I_y : The y derivative of image I_2

I_t : The image difference $I_2 - I_1$

צורת פתרה ע"י חלון 5X5:

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}$$

$$\begin{matrix} A & d \\ 25 \times 2 & 2 \times 1 \\ & 25 \times 1 \end{matrix}$$

$$\begin{matrix} b & \\ 25 \times 1 & \end{matrix}$$

אבל נרצה לפטור בצורה כללית [затת בעית Least Square (least square השגיאה בין האגפים)]
כלומר בהינתן:

$$\begin{matrix} A & d \\ 25 \times 2 & 2 \times 1 \\ & 25 \times 1 \end{matrix}$$

פתרו בצורה הבאה: (הופכים למטריצה ריבועית מסדר 2X2)

$$\begin{matrix} (A^T A) & d \\ 2 \times 2 & 2 \times 1 \\ & 2 \times 1 \end{matrix}$$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

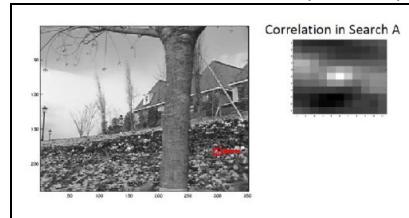
$$\begin{matrix} A^T A & \\ & A^T b \end{matrix}$$

המשוואות יהיו פתרות כאשר המטריצה $A^T A$ הפיכה (ואז ניתן לבדוק את וקטור הנעלמים).

נבדוק את הערכים העצמיים של $A^T A$ (מטריצה 2×2 ולכן ישנו מוקס' 2 ערכים עצמיים)
אם הע"ע קטנים (לדוג' קטנים מ-1) זה אומר שאין שיפוע לכל הכיוון וסביר להניח שהוא מסתכלים על פיקסלים בשטח פתוח (כמו שמיים וכו') שאינו מראה שום קורלציה [מקרה זה יכול לנבוע מרושע]
אם הע"ע מראים יחס גדול מיידי (אחד גדול ואחד קטן) זה מראה שכנהרא אנו מסתכלים על קצה

| קורלציה בשטח פתוח | קורלציה לאורך קצה |
|--|---|
| <p>נקבל קורלציה טובה לאורך כל הכוונים שנת החומוגני = מטריצה סינגולרית (דרגה 0)</p>   | <p>נקבל קורלציה טובה לאורך כל הקצה (הגג) קצה = מטריצה סינגולרית (דרגה 1)</p>   |

לכן, המקרה הכיווני טוב הוא אם נחשש במקומות בו יש קצנות בכל הכוונים (לדוג' האדמה בחלק התחתון) כי במקומות אלה המטריצה הפיכה! (דרגה 2)



מכאן, קיבלנו צורת פתרה גנריית:

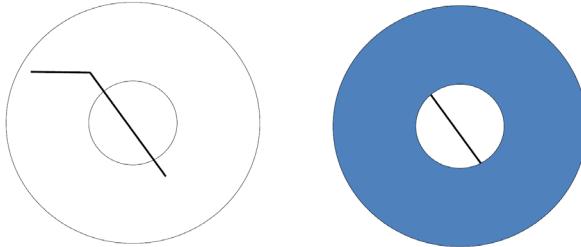
$$\left\{ \begin{array}{l} \frac{\partial E}{\partial u} = \sum_{x,y} I_x \cdot (I_x \cdot u + I_y \cdot v + I_t) = 0 \\ \frac{\partial E}{\partial v} = \sum_{x,y} I_y \cdot (I_x \cdot u + I_y \cdot v + I_t) = 0 \end{array} \right.$$



$$\left[\begin{array}{cc} \sum_{x,y} I_x \cdot I_x & \sum_{x,y} I_x \cdot I_y \\ \sum_{x,y} I_y \cdot I_x & \sum_{x,y} I_y \cdot I_y \end{array} \right] \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum_{x,y} I_x \cdot I_t \\ \sum_{x,y} I_y \cdot I_t \end{bmatrix}$$

אלגוריתם LK פותר בצורה פשוטה את בעיית הביריות הקבועה (brightness constancy) דרך הפתרון מתאפשרת לנארית מכיוון שההנחה היא שהחיזוק קטנה. חסרון של LK: באזוריים עם חיפויה קטנה (לדוג' פנורמה) תהיה התכנסות קטנה.

בניהם לב, כי אנו מחפשים לפתרו משווהאת אחת אבל יש לנו 2 געלים. וכך נוכל לומר באופן אחד משמעי מה הייתה התזוזה. **מבחן האלגברית אין רק תשובה אחת אלא יכולות להיות כמה!** מבחן פיזיקאי בעיה זו ניתן להשוואה עם בעיית הצלצל (The Aperture Problem) בעית הצלצל מתיחסת לעובדת שלן ניתן לקבוע את התונועה של מבנה מרובבי חד-מדדי, כגון קצה, באופן אחד משמעי אם רואים אותו דרך צמצם קטן כך שקצבות הגירוי אינם נראים. לעומת זאת, במקרה שבתמונה שמאל למטה אנו מזינים את האובייקט השחור ימינה, אז אולי היה חסם כמו בתמונה ימין) זה היה נראה כאלו הוא זו בוצרה אלכסונית (ימינה-למעלה)



מה אם נרצה למצוא גם סיבוב (בנוסף להזזה)?
נפתרו לצורך זהה אך נכתבו את המשוואות בצורה אחרת: (כלומר, v, u יהיו שונים הפעם)

$$x_2 = \cos(\alpha) \cdot x_1 - \sin(\alpha) \cdot y_1 + dx \approx x_1 - \alpha \cdot y_1 + dx$$

$$y_2 = \sin(\alpha) \cdot x_1 + \cos(\alpha) \cdot y_1 + dy \approx \alpha \cdot x_1 + y_1 + dy$$

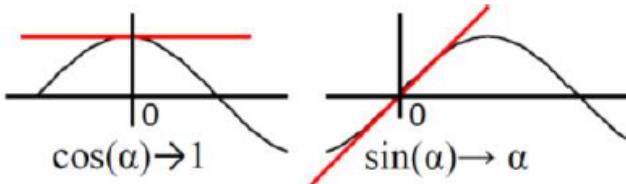
$$\sin(\alpha) \rightarrow \alpha \quad (\text{Small } \alpha)$$

$$\cos(\alpha) \rightarrow 1 \quad (\text{Small } \alpha)$$

$$u = x_2 - x_1 = -\alpha \cdot y_1 + dx$$

$$v = y_2 - y_1 = \alpha \cdot x_1 + dy$$

$$E(dx, dy, \alpha) = \sum_{x,y} (I_x \cdot [-\alpha \cdot y + dx] + I_y \cdot [\alpha \cdot x + dy] + I_t)^2$$



נשים לב כי α צריך להיות קטן על מנת לאפשר את הפתרון הלינארי הנ"ל!
כלומר, רק אם α קטן נוכל להשתמש במשוואות הנ"ל אך הערכים החדשים u, v, x, y נקבעים ע"י
המשוואות שמסומנות בצהוב (אין לינאריות).
מכאן, נגזר ונקבל את המשוואות:

$$E(dx, dy, \alpha) = \sum_{x,y} (I_x \cdot [-\alpha \cdot y + dx] + I_y \cdot [\alpha \cdot x + dy] + I_t)^2$$

$$\frac{\partial E}{\partial dx} = 0 = \sum_{x,y} (I_x \cdot [-\alpha \cdot y + dx] + I_y \cdot [\alpha \cdot x + dy] + I_t) \cdot I_x$$

$$\frac{\partial E}{\partial dy} = 0 = \sum_{x,y} (I_x \cdot [-\alpha \cdot y + dx] + I_y \cdot [\alpha \cdot x + dy] + I_t) \cdot I_y$$

$$\frac{\partial E}{\partial \alpha} = 0 = \sum_{x,y} (I_x \cdot [-\alpha \cdot y + dx] + I_y \cdot [\alpha \cdot x + dy] + I_t) \cdot (I_y x - I_x y)$$

- Iterations: Solve with "small α assumption"

- Warp with full accuracy of sin and cos.

- Pyramids: Angle remains the same...

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y & \sum (I_y I_x x - I_x I_y y) \\ \sum I_x I_y & \sum I_y I_y & \sum (I_y I_y x - I_x I_y y) \\ \sum I_x (I_y x - I_x y) & \sum I_y (I_y x - I_x y) & \sum (I_y x - I_x y)^2 \end{bmatrix} \begin{bmatrix} dx \\ dy \\ \alpha \end{bmatrix} =$$

$$= \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \\ \sum (I_y x - I_x y) I_t \end{bmatrix}$$

בדרכ זו ניתן למצוא עוד שינויים (לדוגמא, למצוא גם שינוי של scale [גודל התמונה]).

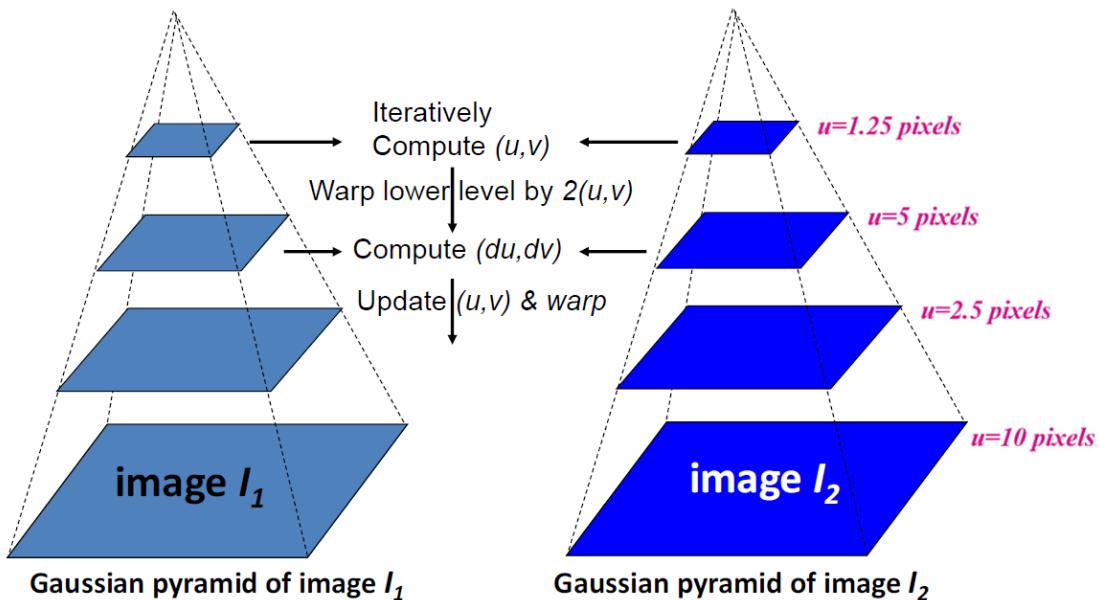
Iterative Lucas Kanade

نبצע את האלגוריתם כל עוד יש התכנסות:

- Compute image derivatives I_x, I_y . Set u, v to 0.
- Compute once $A = \begin{bmatrix} \sum I_x \cdot I_x & \sum I_x \cdot I_y \\ \sum I_y \cdot I_x & \sum I_y \cdot I_y \end{bmatrix}$
- Iterate until convergence ($I_t \approx 0$):
 - compute $b = \begin{bmatrix} \sum I_x \cdot I_t \\ \sum I_y \cdot I_t \end{bmatrix}, I_t(x, y) = I_2(x, y) - I_1(x+u, y+v)$
 - Solve equations to compute residual motion $A \cdot \begin{bmatrix} du \\ dv \end{bmatrix} = -b$
 - Update total motion with residual motion: $u+=du, v+=dv$
 - Warp I_2 towards I_1 with total motion (u, v) .**

מה קורה אם התנועה גדולה יותר מפיקסל?

ונכל להשתמש בפירמידות (Iterative & Multiscale approach). כלומר, אם נמצא תנועה של פיקסל בrama מסוימת אז תנועה זו מייצגת תנועה גדולה יותר בתמונה הגדולה יותר:



יתרונות:
מחשב את נגזרות התמונה פעמי אחת בלבד.

- יש שני שלבים בכל איטרציה: שערוך התנועה (שערור v, u) ואז Warping (הוזזה)
- עובד גם עם אומדן תנועה לקיי, כל עוד זה מפחית את השגיאה
- הוזזה תמונה אחת לכיוון השניה נעשה מהתמונה המקורית תוך שימוש בערכי V, U
- גLOBליים, ולא עם התמונה הקודמת תוקן שימוש בערכי V, U החלקיים (כי עיות חזר ונשנה מטושטש!)

שימוש ב- LK איטרטיבי מomin שאננו נמצאים בקרבת הפתרון.
נשים לב כי אם התנועה גדולה מדי היא עשויה לא להתכנס!

Horn-Schunck (HS) Optical Flow:
 שיפור לאלגוריתם LK (התמודדות עם רעש).
 שיטת לוקאס-קנדה מבוססת על אזורים מקומיים. באלגוריתם HS יש התחשבות בתזוזה של כל פיקסל. ולכן, באלגוריתם HS ישנו מרכיב v , ו לכל פיקסל בהתאם המיצגים את התזוזות. ההנחה באלגוריתם HS היא שסביר להניח שפיקסל זה באותו כיוון עם שכניו (הנחה זו נקראת regularization או smoothness) [הנחהנו כהה גם ב-LK אבל פר בлок ולא פר פיקסל].
 באלגוריתם HS הגדרנו את השגיאה כך:

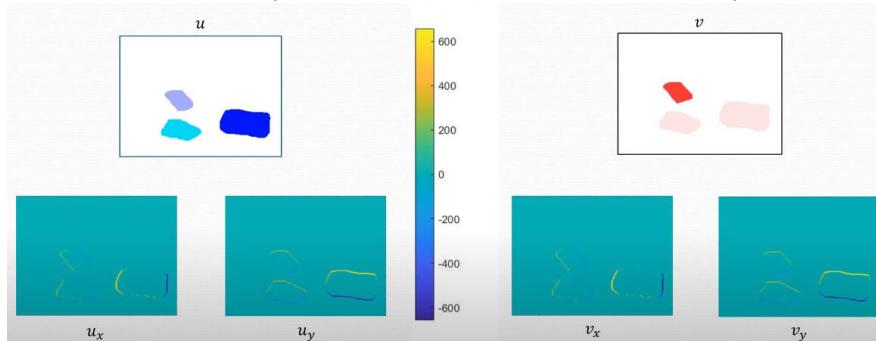
$$\sum_{\Omega} [u \cdot I_x + v \cdot I_y + I_t]^2$$

באלגוריתם HS מוסיפים אילוץ נוסף לשגיאה:

$$\sum_{\Omega} u_x^2 + u_y^2 + v_x^2 + v_y^2$$

שזה בעצם האילוץ של **smoothness**

זהו סכום השינויים (הנגזרת הראשונה של I , u [גוזרים אנכיא אופקי בכל אחד מהם]) מעליים כל שינוי בריבוע (כי לא מעוניין אותנו אם השינוי חיובי/שלילי)

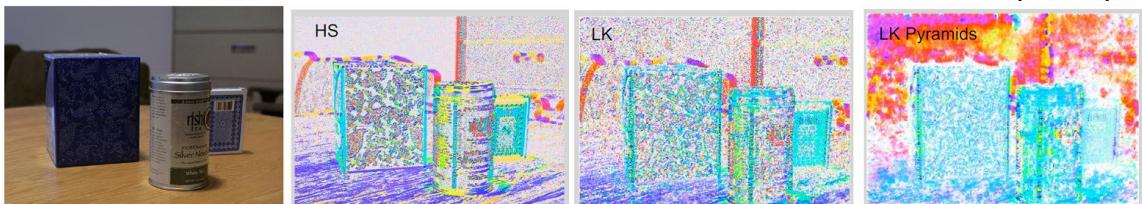


נחבר את האילוצים ונקבל כי חישוב השגיאה ע"פ HS הינו:

$$\sum_{\Omega} [u \cdot I_x + v \cdot I_y + I_t]^2 + \lambda \sum_{\Omega} u_x^2 + u_y^2 + v_x^2 + v_y^2$$

אשר ג' הוא איזון בין מידת החישוב של השוואת ערכי הפיקסלים ביחס לsmoothness.

ניתן להבחין בתוצאות השוואת האלגוריתמים בתמונה הבאה:



מצגת 5:

במצגת זו נלמד (גיאומטריה) על טרנספורמציות דו ממדיות ועל Image Warping.

במצגת 2 למדנו על פילטרים שמשנים את ערכי הפיקסלים.

במצגת הנוכחית נלמד על פילטרים שמשנים את מיקומי הקואורדינטות של הפיקסלים.

ישן 2 סוגי טרנספורמציות:

1. לokaלità - שונה עבור כל מיקום ומתוארת בדרך כלל כשדה וקטור
2. גlobality – אותה פורמה של טרנספורמציה עבור כל הפיקסלים (רק צריך לאותל את מטריצת הטרנספורמציה עם פרמטרים ואפשר לרוץ על הכל)

מטריצת טרנספורמציה M מתוארת בצורה הבאה:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = M \begin{bmatrix} x \\ y \end{bmatrix}$$

קואורדינטות הומוגניות:

| | |
|--|---|
| <p>represent coordinates in 2 dimensions with a 3-vector</p> $\begin{bmatrix} x \\ y \end{bmatrix} \longrightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ <p>Scalar multiplication does not change the point:</p> $\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \cong \begin{pmatrix} 2x \\ 2y \\ 2 \end{pmatrix} \cong \begin{pmatrix} 3.5x \\ 3.5y \\ 3.5 \end{pmatrix} \cong \begin{pmatrix} \lambda x \\ \lambda y \\ \lambda \end{pmatrix}$ <p>are all equivalent to $\begin{pmatrix} x \\ y \end{pmatrix}$</p> | <p>Add a 3rd coordinate to every 2D point</p> <ul style="list-style-type: none"> • (x, y, w) represents a point at location $(x/w, y/w)$ • $(x, y, 0)$ represents a point at infinity • $(0, 0, 0)$ is not allowed |
|--|---|

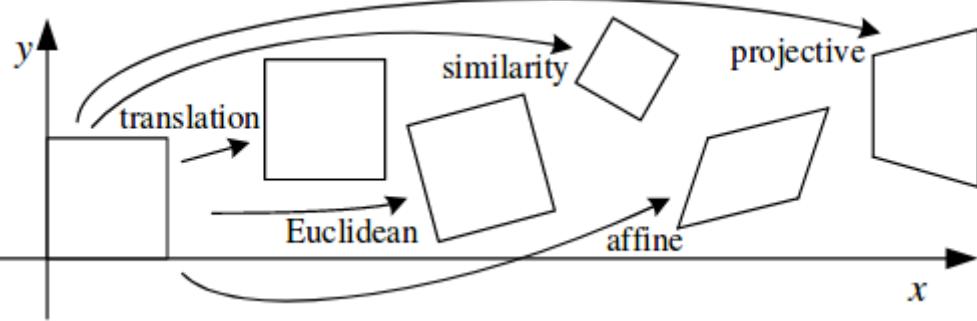
מערכת קואורדינטות זו נוחה לייצג טרנספורמציות שימושיות רבות.
לדוג', קואורדינטות הומוגניות מאפשרות הוספת סקלר לכל קואורדינטה (שימושי לביצוע

סוגי טרנספורמציות:



| שם | תכונות | מטריצה |
|---|---|--|
| Translation (הויזזה של התמונה כמו שהוא על צירים (x, y)) | חייב להשתמש בקואורדינטות הומוגניות כיון שאין לנו אוריינטציה. במטריצה 2×2 . דרגות חופש: 2 | $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$ <i>Homogeneous Coordinates</i> |
| Scaling (הקטנת/הגדלת התמונה) | דרגות חופש: 2 או 1 אם אחיד | $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ |
| Rotation (סיבוב התמונה) | $x = r \cos(\phi)$ $y = r \sin(\phi)$ $x' = r \cos(\phi + \theta)$ $y' = r \sin(\phi + \theta)$ <i>Trig Identity...</i> $x' = r \cos(\phi) \cos(\theta) - r \sin(\phi) \sin(\theta)$ $y' = r \sin(\phi) \cos(\theta) + r \cos(\phi) \sin(\theta)$ <i>Substitute...</i> $x' = x \cos(\theta) - y \sin(\theta)$ $y' = x \sin(\theta) + y \cos(\theta)$ דרגת חופש: 1 | $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_R \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ הטרנספורמציה ההפוכה היא באמצעות הזווית $\theta - \pi$. $R^{-1} = R^T$ אך מתקיים $\det(R) = 1$ ו- |
| Mirror (הפיכת התמונה כמו מראה) | דרגת חופש: 0 | $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ מראה עבור ציר ה- y : $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ מראה עבור $(0,0)$: |
| Shear (הויזזה התמונה בצורה מקבילית) | דרגת חופש: 2 | $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ |

הכללה של מספר הדזות ביחד:



| שם | תכונות | מטריצה |
|---|--|--|
| Rigid או [Euclidean] (הוזזה + סיבוב) | שומר על הכלול (זוויות, אורךים) פרט לכיוון/orientacija של התמונה דרגת חופש: 3 | $\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & t_x \\ \sin(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$ |
| Similarity (הוזזה + סיבוב + (Uniform Scale) | שומר זוויות ופורופורציה בין שני קטעים, אבל לא שומר על כיוון/orientacija ואורךים דרגת חופש: 4 | $\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} s \cdot \cos(\theta) & -s \cdot \sin(\theta) & t_x \\ s \cdot \sin(\theta) & s \cdot \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$ |
| Affine (הוזזה + סיבוב + + Uniform Scale (Shear) | שומר על מקבול (קטעים מקבילים) ישארו מקבילים) ופורופורציה בין שני קטעים דרגת חופש: 6 | $\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$ |
| Projective או Perspective [Homography] (המרת 2D אל 2D) | שומר רק על קווים ישרים דרגת חופש: 8 | $\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$ נוצר רצף לחלק את u, v בפרמטר השלישי, נקבל: $x' = u/w$ $y' = v/w$ |

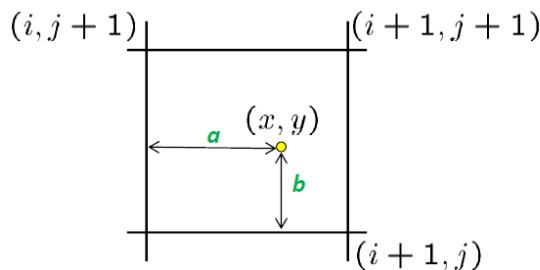
ניתן לשרשר כמה טרנספורמציות ביחד
שים לב! הכפלת מטריצות אינה קומוטטיבית ולכן יש חשיבות לסדר הכפלת המטריצות
* לגבי טרנספורמציה Affine וטרנספורמציה Projective (הומוגרפיה) נרחב במצגת 7.

:Forward warping

ביצוע Forward warping זה מה שבערך הבנו עד כה. לוקחים כל פיקסל ומנסים את הקואורדינטות שלו לקואורדינטות החדשות המתקבלת מהכפלת הטרנספורמציה על הקואורדינטות הישנות Splatting: אם הפיקסל החדש נשלח למיקום שהוא בין כמה פיקסלים אפשר לחלק את ערך הצבע שלו בין שכניו [נקרא "splatting"]

:Backward (inverse) warping

מאתחלים מטריצה יעד ואז עברו כל פיקסל ביעד מחשבים את קואורדינטות המקור שלו עם המטריצה ההופכית של הטרנספורמציה. כמובן, עם מטריצה T^{-1} ומשם לוקחים את ערכו Interpolate: אם הפיקסל מגע "מתוך" בין פיקסלים (קורה כאשר לא קיבלנו מספרים עגולים של קואורדינטות) [נקרא "Interpolate"] nearest neighbor, bilinear, bicubic, Gaussian.



$$\begin{aligned} f(x, y) = & (1 - a)(1 - b) f[i, j] \\ & + a(1 - b) f[i + 1, j] \\ & + ab f[i + 1, j + 1] \\ & + (1 - a)b f[i, j + 1] \end{aligned}$$

השווי בין השיטות:

- Inverse warping יוצרת תמונה ללא חורים
- מצריכה העתקה T הפיכה (זה לא תמיד קיים)

באופן כללי, Inverse warping (אם אפשרית) עדיפה על פני Forward warping.

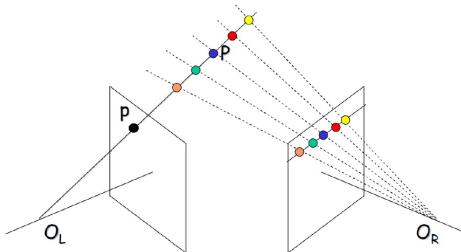
אם ניקח לדוגמה את אלגוריתם LK Iterative, אז אנו יודעים שיש שימוש Warpimg על מנת להשווות בין החלונות של התמונות ובהתאם לכך לשערך כל פעם את התזוזה בין התמונות. לכן, כדאי להציג את התמונה שלאחר ההזזה לכיוון תמונה המקור (inverse) ולא את תמונה המקור לכיוון התמונה שלאחר ההזזה (forward). כמו כן, נזכיר שນctrer לטפל במקרים בהם צריך אינטראפולציה.

מצגת 6:

כעת, נלמד על (Stereo, Homography, Image Stitching (Mosaic/Panorama) .Features, RANSAC).

:Stereo

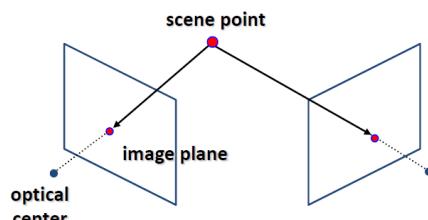
מבנה ועומק הם דו מושגים ייחודיים באופן מהותי כאשר מסתכלים מנוקודות מבט מסויימות תמונה זו ממדית מתיחסת לכל קרטיס יחיד ולכן לא יודע באיזה עומק בקרן נמצא האובייקט סטריאו עוזר לפטור זאת



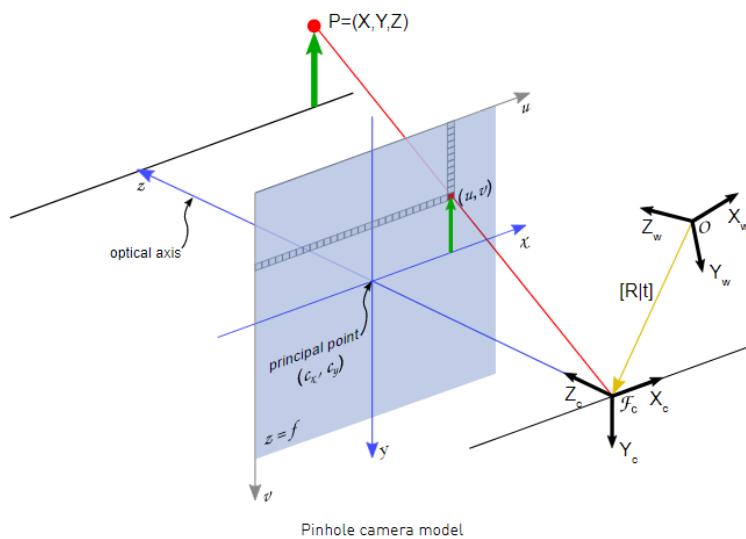
סטריאו עוזר לנו לאבחן את צורת האובייקט בהינתן התנועה (במילים אחרות, בהינתן 2 תמונות שצולמו מזוויות שונות)

אנחנו נצטרך לדעת כמה דברים:

- מיקום המצלמה ("calibration")
- ההתאמנה בין 2 נקודות (כל נקודה בתמונה אחת עם הנקודה המתאימה בתמונה השנייה)



קצת מושגים:



P בתמונה היא נקודה בעולם

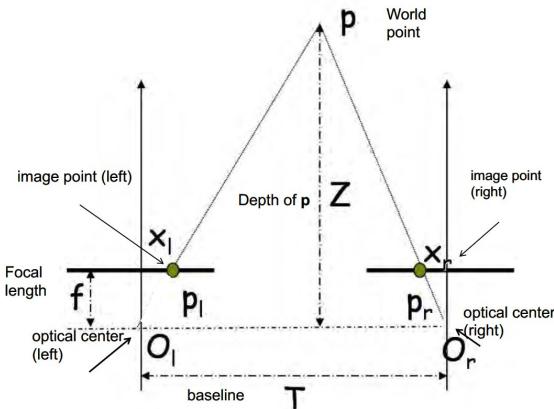
[נשים לב כי ציר z הוא הציר האופטי (עומק) ולכיוון מישור התמונה]

center of projection (נקרא גם Optical Center) – המרכז האופטי בター המצלמה מייצג את הנקודה שבא כל קרטיס האור (היצירות את התמונה) מצטלבות. בתמונה מסומן עם F_c .
Effective Focal Length – המרחק בין Center of projection אל מישור התמונה.
Baseline – כאשר יש לנו 2 מצלמות איזה המרחק בין COP שלהם נקרא Baseline

נניח כי יש לנו 2 תמונות המציגות תנועה (כל תמונה מצלמה אחרת)
נדיר את הבעיה כך שהתזוזה בין התמונה היא רק על מישור X (יש תזוזה רק לצדדים)



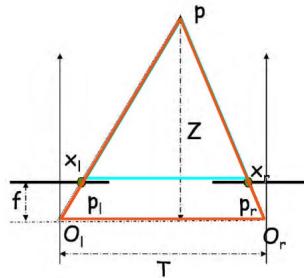
כעת, נביט בפרמטרים הבאים:



(הקו הימני האנכי זאת מצלמה ימין והקו השמאלי האנכי זאת מצלמה שמאל.
(בנוסף, לכל מצלמה יש ציר משלה כך שהצד השמאלי הוא החיבוי)

נניח כי אם יודעים את הפרמטרים הפנימיים של המצלמה (ניתן לדעת ע"י קליברציה כפי שנראה במצגת 9). לכן, בפרט אנו יודעים את ערכו של f (focal length) ואת ערכו של T שהוא Baseline .

כעת, נוכל להבחן בהתאם בין מושלים דומים:



$$\frac{T+x_l-x_r}{Z-f} = \frac{T}{Z}$$

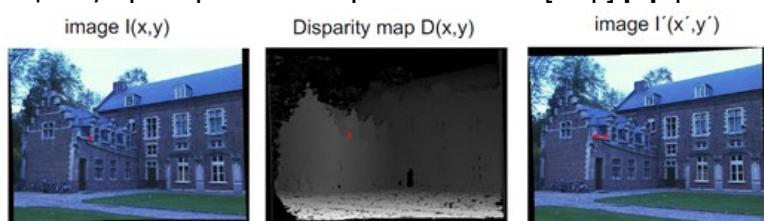
(בחישוב המשוואות נזכיר שלכל מצלמה יש ציר משלה וגם שהצד השמאלי הוא החיבוי)
ומכאן:

$$Z = f \frac{T}{x_r - x_l}$$

disparity

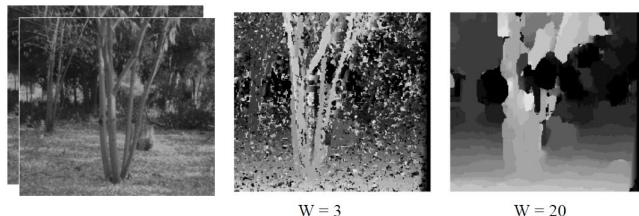
כאשר ההפרש בין ה- x -ים (שזה ההפרש בתמונה) נקרא **disparity**.

כלומר, אם נדע את הפרש התזוזה (disparity) אז נדע את העומק לאובייקט שהוא Z נשים לב כי ישיחס הפוך בין disparity לעומק. אם disparity גדול אז העומק קטן והופך אם נשים אצבע למרחק קטן [קרוב] לעניינים ונסגור עין אחת לסתורין נבחין ב disparity גדול).



כעת, אם נמצא התאמות בין 2 פיקסלים (כל אחד מתמונה) נוכל לחשב את disparity ולשחזר עומק הדרך למצוא התאמות בין פיקסלים ע"פ הגדרת הבעה לעיל (позזה רק בציר X) היא שעבור כל חלון בתמונה 1 נעבור על אותו הקו (עליו נמצא החלון) בתמונה 2 ונחשש את ההתאמה הכי טובה. ניתן לבצע זאת כפי שביצענו Template Matching. כמובן, בעזרת SSD או NCC.

חלונות בגודלים שונים יניבו תומנות disparity שונה:



חלון קטן - יתן לנו יותר פרטיהם אבל יותר רעש (זה בעצם הטריד-אוף)
חלון גדול - יתן לנו פחות רעש (תומנת disparity חלקה יותר) אבל פחות פרטיהם
במקרים בהם ישנים כמה התאמות (לדוגמא, משטח שחור גדול קר שישנם הרבה התאמות) אז נקבל הרבה רעש בתומנת disparity. זאת מכיוון שכל חיפוש של התאמה געשה עצמאית להתקאה מוביל להתחשב בשאר הסביבה. לכן, נרצה להכין אילוץ נוסף על מנת לפתור בעיה זו.

הגדרות:

- uniqueness - לכל נקודה בתמונה אחת חייבת להיות לכל היוטר התאמה אחת בתמונה השניה (כל נקודה מגיעה מכאן ספציפית לכך עבור התאמה [זוג נקודות] מתקיים שהקרניים של שתי הנקודות נחתכות). כלומר, ההנחה שהקרניים נחתכות פעמי אחת).
- ordering - הנקודות המתאימות צרכות להיות באותו סדר בשתי התוצאות [סדר האובייקטים אותם צילמנו] (לא מתקיים תמיד).
- smoothness - ההנחה היא שההפרשים (disparity) משתנים לאט.

כעת, נוכל להגיד התאמה בעזרת ערכי הפיקסלים (Match quality) וגם בעזרת Smoothness (smoothness) נקבל פונקציית שגיאה התלויה בשנייהם: (ובכך מכנים אילוץ שיכל לבדוק את התוצאה)

$$f(x) = \sum_{i=1}^n m_i(x_i) + \sum_{i=2}^n \phi(x_{i-1}, x_i)$$

quality of match uniqueness, smoothness

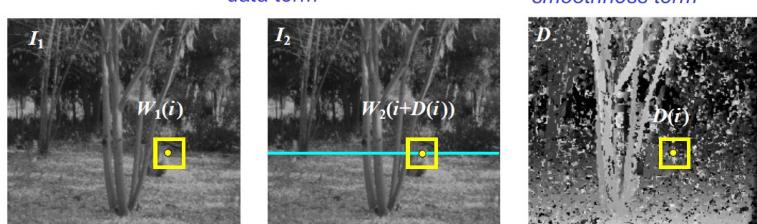
casar, נוכל להגיד את פונקציית Match quality כך:

$$m(x) = \alpha(1 - \text{NCC})^2$$

כאשר נשים לב כי ככל שהNCC יותר גדול [התאמה גבוהה] נקבל שגיאה נמוכה יותר.
לא הרחיב, אך הזכיר שניתן לפטור את הבעה הנ"ל בצורה דינמית במצבת 6 שוקופית 30, בנווסף, בשוקופית 31 ישנה השוואה מבחינת סיבוכיות ובשוקופית 32 מזוכר שבעזרת תכונת דינامي במרחב 2D מקבלים אי דיווק בציר ה-Y כי אנחנו התייחסנו רק לציר ה-X מבחינת Smoothness].
(ישנה שיטה יותר מתקדמת שמתיחסת בSmoothness גם בציר ה-Y אבל אמר שלא למד את זה)

לסיכום, כפי שאמרנו, משתמש בנוסחה:

$$E(D) = \underbrace{\sum_i (W_1(i) - W_2(i+D(i)))^2}_{\text{data term}} + \lambda \underbrace{\sum_{\text{neighbors } i,j} \rho(D(i) - D(j))}_{\text{smoothness term}}$$



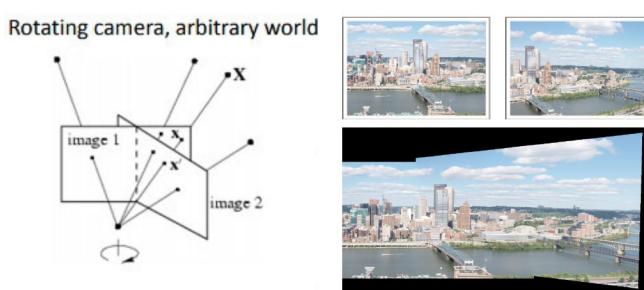
מצגת 7:

Homography

כבר הזכרנו במצגת 5 (בסיכום זה) את הטרנספורמציות הгалובלית Projective ו-Affine. Homography היא שם אחר לטרנספורמצית Projective. שהיא טרנספורמציה גלובלית מתמונה 2D לתמונה 2D.

המשמעות של הוזמת הומוגרפיה היא קבלת תמונה כאילו צילמו מזווית אחרת את אותן פיקסלים זהה מתקיים רק כאשר מצלמים משתח Ci אם נצלם אובייקטים עם עומק (תלת ממד) אז כשנצלם אותן מקומות אחר בהכרח לא נקבל את אותן פיקסלים חשוב מאוד לדעת שזה נובע משנהו במיקום של המצלמה לכן, אם נשאיר את המצלמה במיקום כך שהCOP לא צ'כל (אין translation) אך הוזמת של המצלמה הוא שונה אנחנו כן נוכל למפות את התוכנות בזוויות שונות של המצלמה רק שלא תהיה חיפוי מלאה

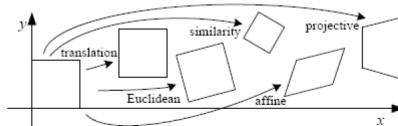
[לא יהיה שינוי בעומק של האובייקטים Ci לא הוזמו את המצלמה]



ככה בעצם ניתן לבנות תמונות פנורמה (מצלמיים באותו מקום [COP קבוע] אבל בזוויות שונות).

תזכורת לטרנספורמציות:

- translation: $x' = x + t \quad x = (x, y)$
- rotation: $x' = R x + t$
- similarity: $x' = s R x + t$
- affine: $x' = A x + t$
- perspective: $\underline{x}' \cong H \underline{x} \quad \underline{x} = (x, y, 1)$
(\underline{x} is a homogeneous coordinate)



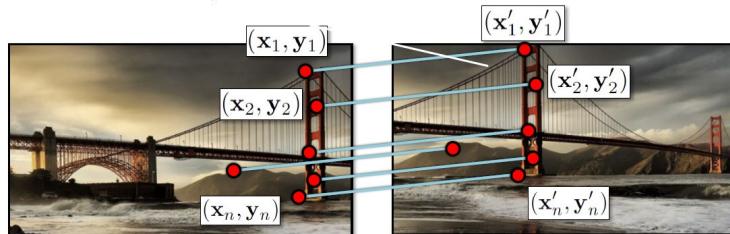
טבלת השוואת דרגות החופש ומבנה המטריצות:

| Name | Matrix | # D.O.F. | Preserves: | Icon |
|-------------------|----------------------------|----------|-------------------|------|
| translation | $[I \mid t]_{2 \times 3}$ | 2 | orientation + ... | |
| rigid (Euclidean) | $[R \mid t]_{2 \times 3}$ | 3 | lengths + ... | |
| similarity | $[sR \mid t]_{2 \times 3}$ | 4 | angles + ... | |
| affine | $[A]_{2 \times 3}$ | 6 | parallelism + ... | |
| projective | $[H]_{3 \times 3}$ | 8 | straight lines | |



air נמצאת הטרנספורמציות?

פתרון ראשון: ממוצע (עבור סט של התאמות נחלק את ההיסט [הפרש] שלהם במספר ההתאמות)



$$\text{Displacement of match } i = (\mathbf{x}'_i - \mathbf{x}_i, \mathbf{y}'_i - \mathbf{y}_i)$$

$$(\mathbf{x}_t, \mathbf{y}_t) = \left(\frac{1}{n} \sum_{i=1}^n \mathbf{x}'_i - \mathbf{x}_i, \frac{1}{n} \sum_{i=1}^n \mathbf{y}'_i - \mathbf{y}_i \right)$$

אבל במקרים כאלה יכולות להיות לנו הרבה משוואות וקטצ געלמים (נקרא over constrained) וזה בעיה כמו שראינו בKA כשלקחנו חלון 5x5 (שיצר 25 משוואות). כמובן, זאת בעיה כי קיבל מערכת משוואות "מוגדרת יתר על המידה", כלומר, יותר מדי משוואות מנעלמים וזה גורר פתרונות המכילים אפסים שזה מה שאנו רוצים.

פתרון שני:

נניח שיש нам התאמות של זוגות פיקסלים בין 2 תמונות (כל פיקסל מתאים לפיקסל בתמונה השנייה) אז אם לדוגמא נרצה לחפש טרנספורמציה הומוגנית נצטרך למצוא 8 געלמים ולכן נצטרך 8 משוואות מכאן, אם כל נקודה תעניק לנו 2 משוואות אז מספיק לנו 4 נקודות (התאמות).
באופן דומה, עבור טרנספורמציה של translation מספיקה לנו נקודה (התאמה) אחת כי כל נקודה מעניקה 2 משוואות:

$$\begin{aligned} \mathbf{x}_i + \mathbf{x}_t &= \mathbf{x}'_i \\ \mathbf{y}_i + \mathbf{y}_t &= \mathbf{y}'_i \end{aligned}$$

מכאן, ניקח את סט המשוואות ונפתר בעזרת Least Squares (כמו שעשינו בKA)

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ \vdots \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} x'_1 - x_1 \\ y'_1 - y_1 \\ x'_2 - x_2 \\ y'_2 - y_2 \\ \vdots \\ x'_n - x_n \\ y'_n - y_n \end{bmatrix}$$

$$\mathbf{A}_{2n \times 2} \quad \mathbf{t}_{2 \times 1} = \mathbf{b}_{2n \times 1}$$

כמובן, נפרמל את המשוואות כך שיהיה לנו מטריצה של נתונים כפול וקטור של געלמים שווה לוקטור/מטריצה של נתונים.

נרצה לחפש קירוב לפתרון של המשוואה לעיל ולכן ננסה למצער את הביטוי:

$$\|\mathbf{At} - \mathbf{b}\|^2$$

ומכאן, כמו בKA נחשב:

$$\mathbf{A}^T \mathbf{At} = \mathbf{A}^T \mathbf{b}$$

$$\mathbf{t} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}$$

זה הוא **pseudo inverse**. כל בעיה מהצורה $\mathbf{Ax} = \mathbf{B}$ נוכל לפתור בצורה דומה)

נרחיב את החישוב הנ"ל ונראה אותו עבור מציאת טרנספורמצית Affine:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

ונקבל:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ \vdots & & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_n \\ y'_n \end{bmatrix}$$

$$\mathbf{A}_{2n \times 6} \quad \mathbf{t}_{6 \times 1} = \mathbf{b}_{2n \times 1}$$

טיפ: כשרצאה לפתור בעזרת Least Square אז תמיד נתחל בכר שנרשום את וקטור הנעלמים.

נרחיב את החישוב הנ"ל ונראה אותו עבור מציאת טרנספורמצית Homography

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

במידה ויש לנו מטריצת טרנספורמצית Homography עם 9 פרמטרים היא תיתן את אותן פתרון אם אחד מהפרמטרים יהיה המספר 1. לצורך העניין ניקח את האיבר האחרון h_{22} .
אם נחלק את כל המטריצה במספר 22 נקבל שהאיבר האחרון הוא 1. המטריצה המתתקבלת תיתן את אותה התוצאה כמו המטריצה הקודמת.

למה זה נכון?

כפי שהגדכנו את הנקודה ההומוגנית (שלשה כאשר הערך האחרון הוא 1) אז אמרנו שהכפלת בסקלר לא משנה את הנקודה [כי לבסוף כשרצאה לעבור למשור דו ממד תמיד נחלק בפרמטר השלישי ונקבל את אותה תוצאה]
גם כאן, לאחר החלוקה באיבר האחרון זו אותה הומוגרפיה
(ישנן סיבות למה לחلك דואק באיבר האחרון. אמר שלא נלמד את זה)

הנקודות שנתקבלו הן:

$$x'_i = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$y'_i = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

(מחלקים באיבר השלישי כי עוברים לדו ממד)

נעביר אגפים:

$$x'_i(h_{20}x_i + h_{21}y_i + h_{22}) = h_{00}x_i + h_{01}y_i + h_{02}$$

$$y'_i(h_{20}x_i + h_{21}y_i + h_{22}) = h_{10}x_i + h_{11}y_i + h_{12}$$

ונפתרו עם (pseudo inverse Least Square): (נוסף כמו למיטה וונפתרו עם

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x'_i x_i & -x'_i y_i & -x'_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -y'_i x_i & -y'_i y_i & -y'_i \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

:Direct Linear Transforms (DLT)

זהות שיטה נוספת לפתרון Least Square של מערכת הומוגנית. כלומר, מהצורה $Ah=0$ (הווקטור האחרון הוא אפסים).

ישנה בעיה במקרים מסוימים מהצורה הנ"ל והוא שלווקטור הנעלמים h תמיד יש פתרון טריוויאלי והוא $h=0$ (ווקטור אפסים).

אנו לא רוצים את הפתרון זה כי אומנם הוא נכון אלגברית אבל הוא לא נכון גיאומטרית.

ונכל להימנע מבעיה זו ע"י זה שנבחר את h כך שהנורמה שלו תהיה שווה ל-1.

הסיבה שמנכל לעשות זאת היא מכיוון שווקטור h הוא הומוגרפי ולכן איןנו רגיש לכפל בסקלר ובאופן דומה הוא גם לא רגיש לחילוק ולכן נוכל לחלק את h כך שהנורמה שלו תהיה שווה ל-1.

והפתרון יעשה בצורה הבאה:

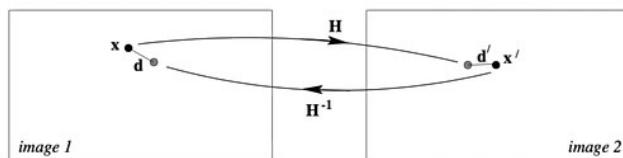
1. נבחר את הווקטור העצמי המתאים לערך עצמיici קטן (השלישי מתוך השלשה) של $A^T A$. ניתן לעשות זאת באמצעות SVD (לא הרחכנו על זה): נפעיל את SVD על A ונקבל את השלשה שהיא $[t, s, u]$. איז הווקטור שאנו רוצים הוא העמודה האחורונה של u .

2. נעשה לווקטור reshape ליהיות 3×3 (הווקטור הנ"ל תמיד עם 9 מספרים)

3. המטריצה המתקבלת (3×3) היא הפתרון הסופי [נחלק באיבר האחרון כיוון שכפי שהסבירנו לעיל המטריצה נdana עד כדי הכפלה בסקלר. ישנה דרך נוספת להתגבר על זה והוא ע"י דרישת שהנורמה של ווקטור הנעלמים (h לפי הסימון כאן) תהיה 1]

ונכל להעריך את איקות ההומוגרפיה שמצאו בצורה הבאה:

$$\sum_i d(\mathbf{x}_i, \mathbf{H}^{-1}\mathbf{x}'_i)^2 + d(\mathbf{x}'_i, \mathbf{Hx}_i)^2$$



(מדדית ההפרשין בין המקור [אחרי הפעלת ההומוגרפיה] ליעד והפור)

:Planar Image Rectification

תיקון תמונה מישורית.



בנייה 4 התאמות (מספיק 4 נקודות בשבייל הומוגרפיה) כך שההתאמות הן מיפוי של קודקודוי תמונה

המקור (שאיינה ישרה [תמונה שמאל לעיל]) לקודקודוי היעד [תמונה ימין לעיל].

כעת, נמצא את ההומוגרפיה בהינתן ההתאמות הנ"ל ונפעיל Warping.

ונכל גם לתקן משטח מישורי:



Image Stitching

תפירת תמונה אחת לשניה.

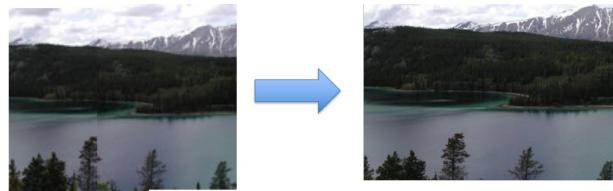
תפירת תמונה מתבצעת ע"י לקיחת סט תמונה כך שכל תמונה חופפת בצורה מסוימת לתמונה אחרת (בשערתת) וחיבור כל התמונות על מנת ליצור תמונה גדולה (בעודים):

- לוקחים סט של תמונות כפי שתיארנו
- עברו כל 2 תמונות מחשבים את התזוזה ביןין (או ע"י חישוב הרוטציה של המצלמה)
- עושים Warp מהתמונה השנייה לראשונה (כל תמונה צולמה מזוית אחרת אז לאחר Warp נקבל תמונה גדולה שככイル צולמה מזוית [הزاوية של התמונה הראשונה])
- אם ישנו עוד תמונות חוזר על הפעולות

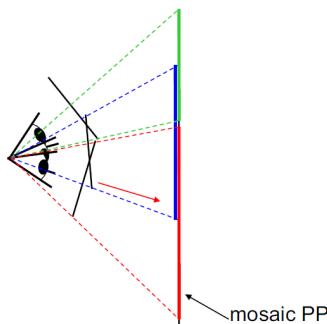
לפי הצעדים שתיארנו הרשוואות נעשות בהשוואה לתמונה הראשונה. בדרך"כ הרשוואה נעשית לתמונה האמצעית.

הערה: כשהתחלנו לדבר על הומוגרפיה הזכרנו כי בהינתן שניי בזווית של המצלמה ללא תזוזה (COP קבוע) נוכל למזג את התמונות כי אין שניי בעומק. אז גם כאן ההנחה היא שמדובר בתמונות שאכן ניתן למזג ביניהן.

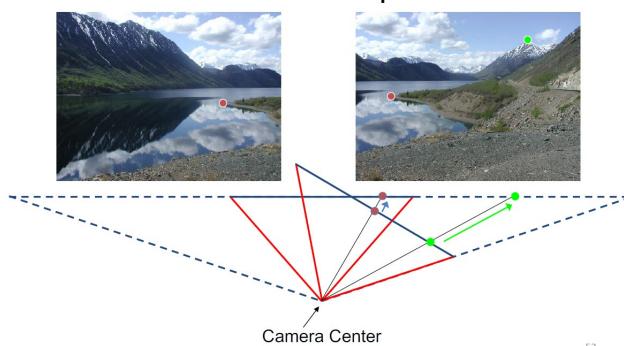
לאחר ביצוע Warp לפעים נקבל תמונה שאינה סימטרית: לכן, נצרך לעשות ערבות (blend) על מנת ליצור פסיפוס (mosaic):



ניתן לבצע Image Stitching (תפירת תמונות) ע"י מיפוי של כל תמונה למישטח משותף



כלומר, אנו מטילים כל תמונה למישטח משותף



כאמור, מכיוון שכאר המצלמה מסתובבת בלבד (שינוי בזווית) ללא תזוזה, אז מבנה התלת ממד לא משתנה ונוכל להפעיל הומוגרפיה.

מצגת 8:

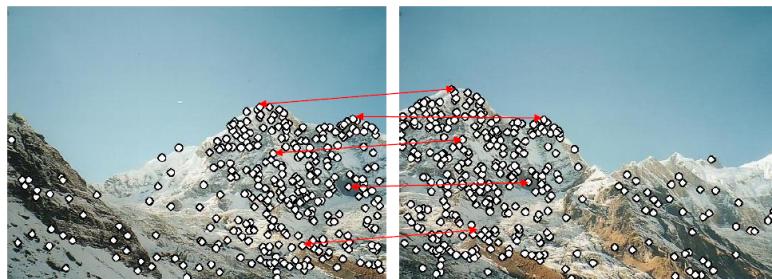
במצגת 4 ראיינו שיטות ישירות לחישוב תזוזה בין שתי תמונות: NCC, LK.
כעת נלמד על שיטות לחישוב תזוזה המבוססות על מיציאות פיצ'רים בתמונה. כאמור, ע"י מיציאות נקודות תאומות

חסרונות השיטות הישירות:

- פחות מתאים לתמונות חופפות חלקית (כמו שמלמים עברו פנורמה)
- טווח מוגבל של התכנסות

אלגוריתם התאמה ע"י מיציאות פיצ'רים:

1. זיהוי נקודות עניין בשתי התמונות
2. בניית מתאר (descriptor) מכל נקודה [descriptor זה יציג מיוחד של הנקודה]
3. מיציאת זוגות מתאימים (נקודה מכל תמונה)
4. כעת, משתמשים בזוגות לחישוב התזוזה (כך ניתן לחבר אותן וליצור לדוג' פנורמה)



נקודות העניין שימושיות עבור:

- יישור תמונה (הומוגרפיה), חישוב מטריצת fundamental fundamental learned בהמשך וכו')
- שחזור תלת ממד (3D Reconstruction)
- ועוד...

בעיה 1: זיהוי הנקודות באופן עצמאי בשתי התמונות



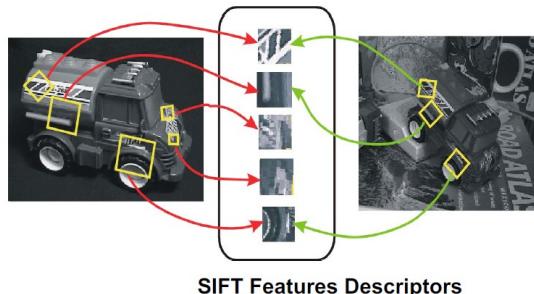
מעט בלתי אפשרי מכיוון שיקולות להיות הרבה התאמות לכל נקודה

בעיה 2: גם אם נמצא את הנקודות יהיה קשה לקבוע איזה נקודה מתאימה לאיזה נקודה

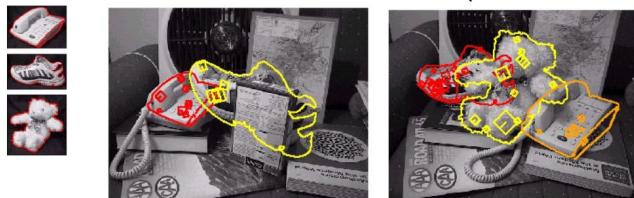


:SIFT

אלגוריתם התאמת התאמה ע"י מציאת פיצ'רים (לא הרחיב כ"כ על האלגוריתם) מוצא התאמות בצורה מאוד טובה גם אם התמונות צולמו מזווית שונות וגם עם סיבוב, scale ופרמטרי תמונה אחרים:



מכיוון שאלו מתארים (descriptors) מקומיים מבוסס חלקיים, הם מתפקדים היטב גם כאשר חלקים מסוימים חסרים (כלומר תחת חסימה):

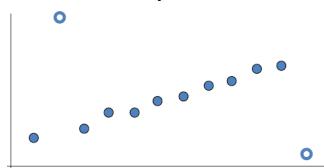


אי דיק בהתקנות יכול לבוא מרעיש, חסימות (אובייקט חוסם אובייקט) וגם מטעויות בחישוב גם אלגוריתם זה נותן התקנות שגיאות ولكن נרצה שיטות רובוטיות (מתמודדות עם רעש)

:Parametric fitting

בנייה צורה פרמטרית בהינתן נקודות.

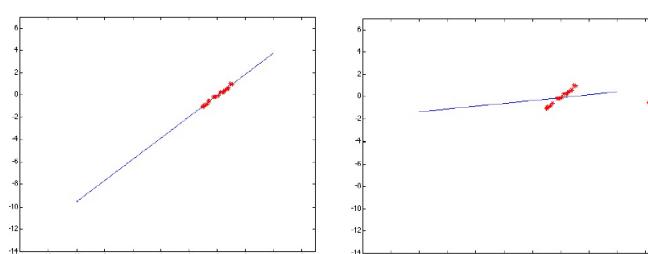
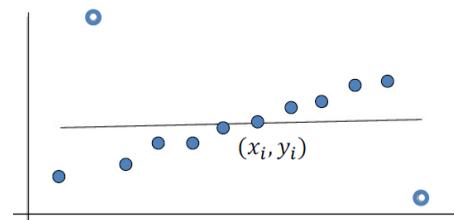
לדוגמה, בהינתן נקודות נרצה לקבל פרמטרים של קו החוצה אותו



דוגמא נוספת היא הומוגרפיה כי בהומוגרפיה אנחנו מקבלים פרמטרים על בסיס התקנות. ביצוע parametric fitting יכולות לתרחש גם בעיות בחישוב. כלומר, התקנות שגיאות (בהרבה) הנקראות :Outliers

$$\text{distance from point } (x_0, y_0) \text{ to line } y=Ax+By+c \quad d = \frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}}$$

Least Square (LS) error is not robust
one outlier point may cause a very
big mistake

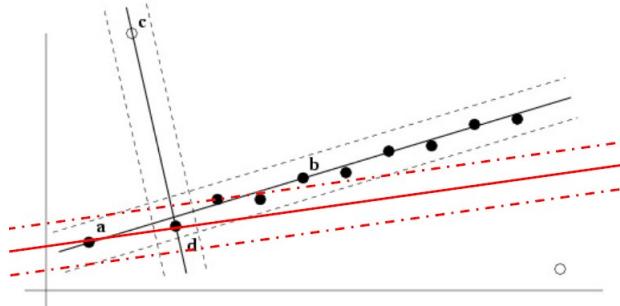


(נשים לב איך נקודת Outlier אחת בתמונה ימין משפיעה על הקו המתkeletal).

:RANSAC

אלגוריתם RANSAC (RANDOM Sample Consensus) הוא אלגוריתם הסתברותי הפועל בשיטה איטרטיבית להערכת פרמטרים של מודל מתמטי מסוים מוגדר קבוצה של נקודות (במקרה שלנו נקודות המכונים חריגים (outliers)).
לכן, ניתן להשתמש באlgorigthm RANSAC על מנת לחשב פרמטרי טרנספורמציה בין תМОנות.

Support (inliers) - בהינתן מודל מתמטי מסוים, ה- Support מוגדר להיות קבוצת הנקודות הנמצאות במרחב של ככל היוטר סוף מסוים מהמודל. שאר הנקודות יחשבו כנקודות outliers.
דוגמא (ubo koo):



אם נשתכל על המודל האדום (במקרה זה המודל הוא קו) אז רק נקודות a, b הם Support.

רעיון האלגוריתם:

- עבור מספר N איטרציות:
 - בחר הסתברותית מספר מינימלי של נקודות הנוצר בשבי המודל (לדוגמא, עבור הומוגרפיה צריך 4 נקודות)
 - בנה את המודל עם ההתאמות (נקודות) וחשב את ה- Support שלו
 - בסוף האלגוריתם בחר את המודל שהניב את ה- Support המקסימלי

אבל אנו רוצים לבנות מודל מדויק יותר וכן לשמור את ההתאמות (נקודות) עצמן כך שמההתאמות שהניבו את ה- Support המקסימלי (בסוף האלגוריתם) נבנה את המודל מחדש (כלומר מכל נקודות ה- Support).).

האלגוריתם: (עבור הומוגרפיה)

RANSAC loop N times:

1. Select four feature pairs (at random)
2. Compute homography H (exact)
3. Compute inliers where $\|p_i' - H p_i\| < \epsilon$

Finish:

- Keep largest set of inliers
- Re-compute least-squares H estimate using all of the inliers

(תזכורת: כשמחשבים את הומוגרפיה צריך לפתור Least Square [עם DLT או pseudo inverse].)

כמה איטרציות צריך בשבייל לקבל תוצאה טובה?
אנו רוצים שבהתברות k לפחות דגימה אחת תהיה ללא שום outlier (לדוגמא, $p=99\%$, $\epsilon=1\%$).
נניח כי ϵ הוא ההסתברות לקבל outliers וונני שדגמים קבוצה של S התאמות
אז מספר הדגימות N הנדרש הוא:

$$(1 - w^s)^N \leq 1 - p$$

$$N \geq \frac{\log(1 - p)}{\log(1 - w^s)}$$

השוואות של מספר הדגימות:

$$p = 0.99 \quad \epsilon = 1 - w$$

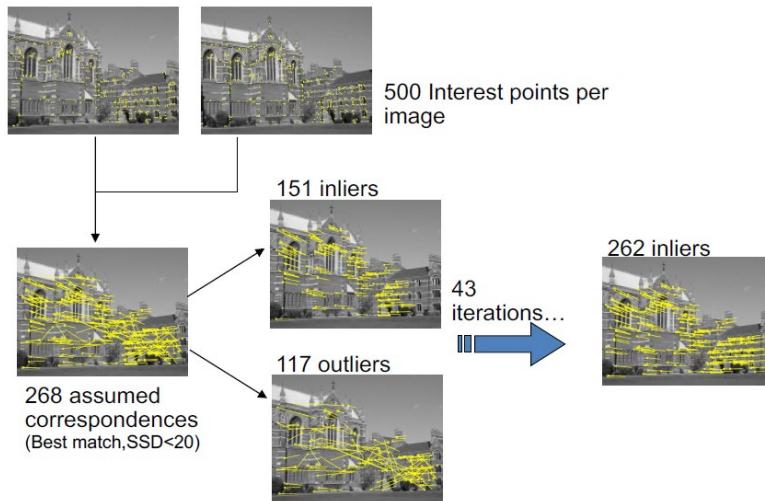
w : inliers probability,

$1-w$: outlier probability

| Sample Size s | proportion of outliers ϵ | | | | | | | |
|------------------|-----------------------------------|-----|-----|-----|-----|-----|------|--|
| | 5% | 10% | 20% | 25% | 30% | 40% | 50% | |
| 2 | 2 | 3 | 5 | 6 | 7 | 11 | 17 | |
| 3 | 3 | 4 | 7 | 9 | 11 | 19 | 35 | |
| 4 | 3 | 5 | 9 | 13 | 17 | 34 | 72 | |
| 5 | 4 | 6 | 12 | 17 | 26 | 57 | 146 | |
| 6 | 4 | 7 | 16 | 24 | 37 | 97 | 293 | |
| 7 | 4 | 8 | 20 | 33 | 54 | 163 | 588 | |
| 8 | 5 | 9 | 26 | 44 | 78 | 272 | 1177 | |

(לדוגמא, בהומוגרפיה [4] דגימות [4] outliers 50% ו- 72 איטרציות לקבלת דגימה טובה
בהתברות 99%)

השוואה:



(בתמונה ניתן לראות שאם מפעלים על כל ההתאמות נקבל 151 אינליירס, אך בשיטת האיטרציה
נקבל [אחרי 43 איטרציות] 252 אינליירס)

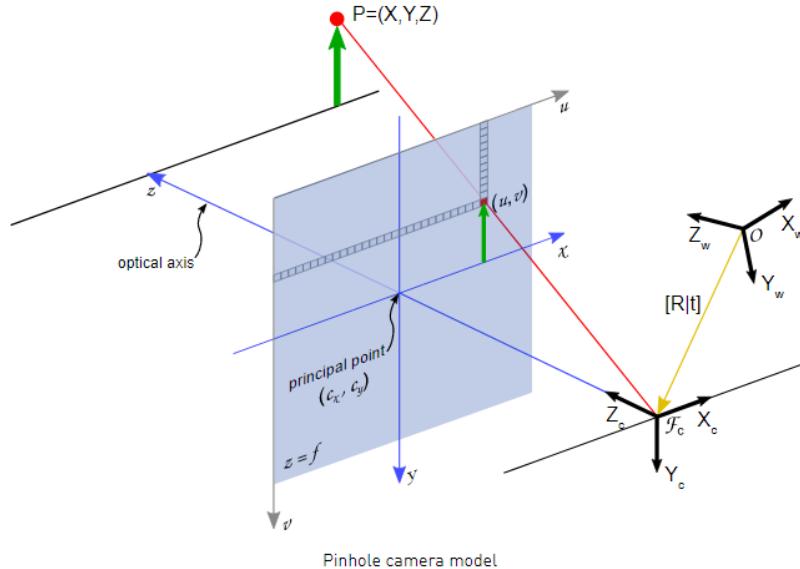
יתרונות: אלגוריתם פשוט, יעיל, ניתן להפעיל עבור מספר רחב של מודלים, ניתן למדוד את השגיאה
חסרון: נדרש הרבה איטרציות עבור יחס גבוה של אוטליירס ביחס לאינליירס
החיסרון הנ"ל משמעוני כי ישנים הרבה בעיות עם הרבה אוטליירס
[ניתן להתגבר על החיסרון עם בחירה סלקטיבית של נקודות]

מצגת 9:

nenich וקיימת נקודה (0,0) בעולם (בשביבת אותה אנו מצלמים)
אז נרצה לדעת איפה המצלמה שלנו נמצאת ביחס לעולם.

מודל המצלמה:

תזכורת מצגת 6:

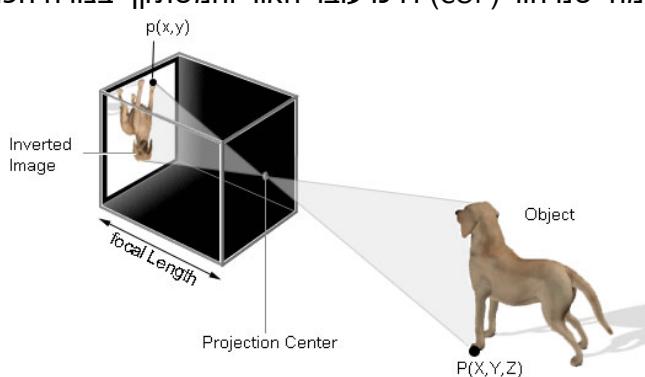


P בתמונה היא נקודה בעולם

[נשים לב כי ציר z הוא האופטי (עומק) ולכיוון מישור התמונה]

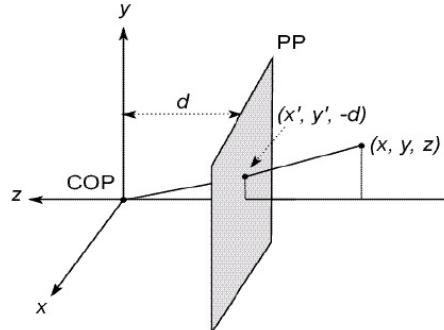
(נקרא גם Center of projection [Cop] – המרכז האופטי בתוך המצלמה מייצג את הנקודה
שהה כל קני האור (הויצרת את התמונה) מצלבות. בתמונה מסומן עם F_c .
המרחק בין COP למשתך – Effective Focal Length
המשתך בין COP לנקודת המצלמה – Baseline

בתמונה לעיל רואים שמשיר התמונה נמצא **לפני COP** (כלומר, בין COP לבין COP).
בפועל, במודל המצלמה ישנו חור (COP) דרךו עובר האור והמשתך בצורה הפוכה **לפנים** המצלמה



תיאורית על מנת ליזוג את המודל אנו משקפים את התמונה חזירה להיות לפני COP (כלומר,
شمישור התמונה יהיה בין העולם לבין COP) ולכן מציגים את המודל כפי שהוא בתמונה
הראשונה (השיקוף גם הוא במשתך focal length מהה COP)

אנו נרצה להמיר נקודה תלת ממדית בעולם לנקודה בתמונה (לפייקסל):

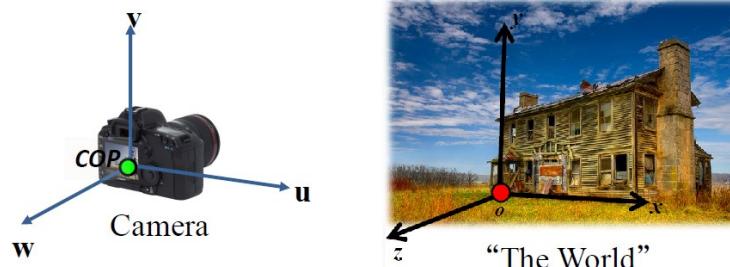


$$(x, y, z) \rightarrow \left(-d \frac{x}{z}, -d \frac{y}{z} \right)$$

[מדובר בווקטורים, لكن הכפלת/חילוק משנים את אורך הווקטור.
במילים אחרות, הכפלת "מוסיפה" הפרש וחילקה "מחסרת" הפרש,
לכן, מחולקים בז על מנת לחסר את ההפרש מהאובייקט בעולם עד COP
ואז מכפילים בז שהוא d על מנת להוסיף את ההפרש מהCOP למישור התמונה]
יש 3 חלקים בהם נרצה להתחשב:
1. קואורדינטות העולם
2. קואורדינטות המצלמה (על COP עצמו כמו בתמונה הראשונה)
3. קואורדינטות מישור התמונה (הקרניים בCOP משתקפות למישור התמונה [לפייקסלים])

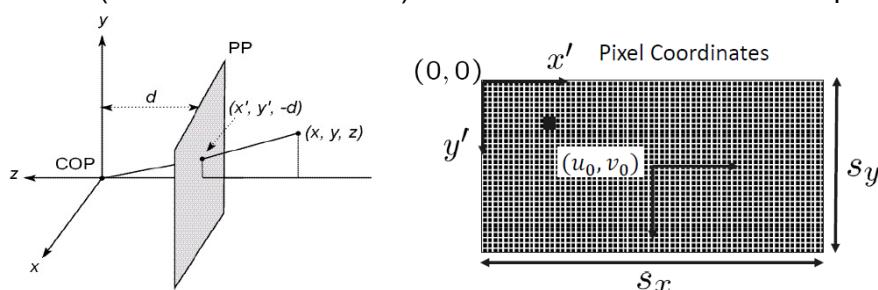
לכן נצטרך:

1. להמיר מקואורדינטות העולם לקואורדינטות המצלמה (בעזרת **פרמטרים חיצוניים**)



השינוי נובע מהזזה + רוטציה (rigid) של המצלמה וכן ההמרה תלויות בפרמטרים אלו.

2. להמיר מקואורדינטות המצלמה למישור התמונה (בעזרת **פרמטרים פנימיים**)



לכל מצלמה יש פרמטרים פנימיים משללה מהתיבות הבאות:

- יתכן שמרכז התמונה במצולמה מסוימת אינו ב(0,0)
- יתכן שהיחס בין האורך והרוחב של התמונה שונה
- יתכן שפיקסל אינו מרובע
- יתכן שההפרש focal length שונה

מסיבות אלו יתכן שבמצולמות שונות קרטינים ימופו לפיקסלים שונים.

מטריצת הפרמטרים החיצוניים (Extrinsic) מミירה מקואורדיינטות העולם לקואורדיינטות המצלמה
מטריצת הפרמטרים הפנימיים (Intrinsic) מミירה מקואורדיינטות המצלמה למשור התמונה

נראה כיצד המטריצות בנויות.
הנקודה בעולם היא תלת ממדית. נרצה לבצע עליה פעולות על מנת להמיר אותה לדו ממד וכן
נשתמש בקואורדיינטות הומוגניות. למשל, נקודה בעולם היא מהצורה:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

מטריצת הפרמטרים הפנימיים (Intrinsic):
נסמן: $K =$ מטריצת הפרמטרים הפנימיים.

שלב א': טיפול בהפרש של focal length (מוסמן ב-f)

$$x = K[I \quad 0]X \rightarrow w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

(כאשר I הוא המטריצה באדום וכאשר 0 הוא וקטור האפסים מימינו)

שלב ב': טיפול בהפרש של (0,0) של התמונה ביחס ל-COP (כלומר, אם הנקודה לא באמת באמצעות)

$$x = K[I \quad 0]X \rightarrow w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & u_0 & 0 \\ 0 & f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

(זה בעיקנון translation ביחס לעומק [שהוא Z])

שלב ג': טיפול ביחס הגודל של הפיקסל (לודג' אם פיקסל אינם מרובע ויש לו יחס שונה)

$$x = K[I \quad 0]X \rightarrow w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

(מוסיפים את הפרמטרים כמו בשלב א')

שלב ד': טיפול בעיות מקבילי (shear/skew)

$$x = K[I \quad 0]X \rightarrow w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

(בדרכ' shear זה עם 2 פרמטרים. נראה כן זה אחד בגלל שזו נטיה לצד אחד
מכאן, ניתן לראות כי מטריצת פרמטרים פנימיים היא עם 5 דרגות חופש).

מטריצת הפרמטרים החיצוניים (Extrinsic):

תזכורת: $K =$ מטריצת הפרמטרים הפנימיים.

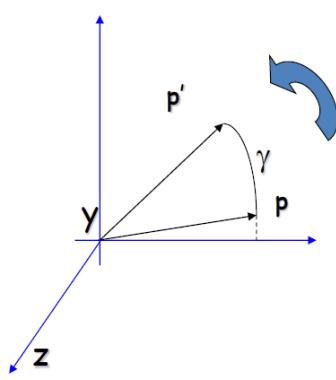
את מטריצה זו נרצה לשרשר בין מטריצת הפרמטרים הפנימיים לעולם.
כאמור, מטריצת הפרמטרים החיצוניים מושפעת מתזוזה של המצלמה בעולם ומהרטציה שלה
מכאן, מכיוון שהמצלמה יכולה להיות ב-3 ממדים וגם לעשות רוטציה ב-3 ממדים אז בהכרח מסוף
הפרמטרים ממנו המטריצה מורכבת הוא 6. כמובן, 6 דרגות חופש.

שלב א': טיפול בהזזה של המצלמה בעולם (ב-3 צירים)

$$\mathbf{X} = \mathbf{K}[\mathbf{I} \quad \mathbf{t}] \mathbf{X} \rightarrow w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

(מסומן בד' בהתאם לכל ציר)

שלב ב': טיפול ברטציה של המצלמה בעולם (ב-3 צירים)



$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ולכן:

$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

לסיכום:

$$\mathbf{x} = \mathbf{K}[\mathbf{R} \quad \mathbf{t}] \mathbf{X}$$

↓

$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

ישנים 5 דרגות חופש
ישנים 6 דרגות חופש

פרמטרים פנימיים
של מצלמה

פרמטרים חיצוניים
של מצלמה

והמטריצה השלמה של הפרמטרים הפנימיים והפרמטרים החיצוניים הינה:

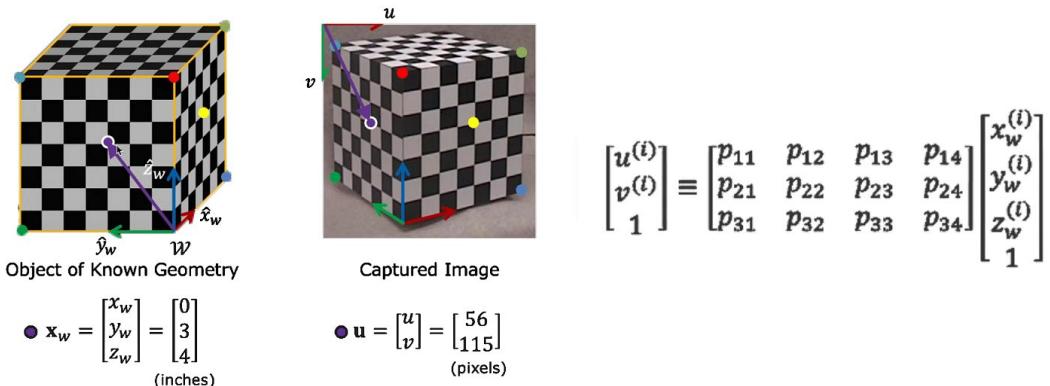
$$\mathbf{x} = \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \Pi \mathbf{X} = \mathbf{K}^{\text{rotation translation}}_{\text{intrinsic}} [\mathbf{R} \quad \mathbf{t}] \mathbf{X}$$

וזהו מטריצת המצלמה (projection matrix). [זה"כ 11 דרגות חופש].
בעזרת מטריצה זו ניתן למפות כל נקודה בעולם לפיקסל ספציפי בתמונה **אפילו מבלי לצלט בפועל**.

Camera Calibration

כיוון מצלמה (Camera Calibration) היא פעולה שנרצה לעשות על מנת למצוא את מטריצת המצלמה ברגע שמצאנו את המטריצה אין צורך לחשב אותה שוב עבור המצלמה ונוכל למפות כל נקודה בעולם לפיקסל בתמונה בהנחה שלא נשנה את מקום המצלמה (כי זה משנה את הפרמטרים החיצוניים \mathbf{R} , \mathbf{t})
(במצגת זו אנו לומדים קליברציה של מצלמה ביחס לעולם. בהמשך נראה קליברציה של 2 מצלמות)

הרעיון הוא ללקחת אובייקט תלת ממדי כלשהו כך שאנו יודעים מה המרחקים בין הנקודות עליו, לבסוף, לאובייקט זה צריך להיות משור X, משור Y, משור Z (אסור שככל הנקודות יהיו על משור אחד, צריך לפחות 2 משוריים כדי שהמשוואות לא יתנוונו). ומכאן נמפה נקודות הנמצאות על האובייקט התלת ממדי לנקודות המתאימות במשור התמונה (דו ממד)



תנאים לקואורדינטות:

- אנחנו יכולים להגיד את נקdot המרכז בעולם $(0,0,0)$ של קואורדינטות העולם כראינו
- אסור שככל הנקודות יהיו דו-משוריות
- קואורדינטות הומוגניות -> נוכל להשתמש בקואורדינטות יחסיות לכל נקודה

מכאן, נציב כל קוואורדינטה בעולם ולאיזה פיקסל הוא ממופה לאחר הכפלת מטריצת המצלמה לאחר שקיבלנו סט משוואות (ישנם 11 נעלמים ולכן לפחות 6 נקודות) נוכל לפתור least square (עם פתרון לינארי כמו DLT או אחרית):

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 & -v_1 \\ & & & & \vdots & & & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_n \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{Ax=0 \text{ form}}$$

(זה מאד דומה להומוגרפיה פרט לכך שהומוגרפיה זאת מטריצה מסדר 3x3 וכאן 4x3)

יתרונות:

- דרך קלה לפתורן
- תוצאות לא רעות

חסרונות:

- לא מביא את הפרמטרים עצם אלא את הפתרון הסופי
- אין התחשבות בחסכנות radial distortion (ישנים קווים שמתעקמים כתוצאה מצלום וכאן אין התמודדות לכך)
- לא ניתן להכניס פרמטרים ידועים כמו focal length (f) בצורה ידנית
- לא מגדיר שגיאה (כלומר, מכיוון שכן זה לינארי מקבלים תשובה חד משמעית ולא מתבצעת אופטימיזציה במשתנה ויש בדיקת שגיאה)

(לא הרחיב אך אמר שישנן שיטות לא לינאריות שמתגברות על כמה מהחסרונות הנ"ל)

[\[משתיר לקליברציה\]](#): Extracting Intrinsic and Extrinsic Parameters
בاهינתן מטריצת המצלמה, ישנה דרך לחץ את הפרמטרים (החיצוניים) t, R, וכן את K (פנימיים)
דיבר במדויקות ולא הרחיב על האלגוריתם. שכתבתי את מה שאמר

הרעיון הוא ש玆 פרקרים את מטריצת המצלמה ל 3 מטריצות ומפעלים פירוק מסויים שיעזר לחץ

[מצאים את הוקטור שמאפס שהוא C] Camera Center [ומבצעים RQ וכו']:

1. Let M is left 3x3 matrix of P, then
 $P = P \cdot \text{sign}(\det(M))$
2. Find camera center C, s.t. $P \cdot C = 0$
 - SVD of P, C is the right singular vector corresponds to the smallest singular value
3. Apply RQ decomposition of M
4. K is the upper diagonal matrix and R orthogonal matrix
5. T is $-R \cdot C$

או בצורה הבאה:

1. Extract local features
2. Establish 3D to 2D correspondences $\{\tilde{X}_i, \tilde{u}_i\}$
3. Estimate P from $\{\tilde{X}_i, \tilde{u}_i\}$
4. Let M be the first 3×3 matrix of P
5. Normalize P by multiplying all elements with $\text{sign}(\det(M))$
6. Compute the camera center C by SVD
7. QR-decomposition of M gives us $M = \hat{R}\hat{K}$
where \hat{K} is an upper triangular like K and \hat{R} is an orthogonal matrix like R
8. Define $D = \text{diag}(\text{sign}(K_{11}), \text{sign}(K_{22}), \text{sign}(K_{33}))$
9. Then $K = \hat{R}D$ and $R = D\hat{R}$ and $t = -RC$

בשלב 3 שאנו משורים את P ואז מפרקים על פי השערור
(יש הפניות ומימוש לאלגוריתם במצגת)

ישנו עוד אלגוריתמים לביצוע קליברציה
Zhang's Calibration Method:

שיטת נוספת לביצוע קליברציה

במקרה לצלם תמונה בודדת עם 3 מישורים בשילוב ביצוע קליברציה (כמו שראינו לעיל):

- משתמש במספר תמונות של לוח שחמט א-סימטרי
- כאשר כל תמונה צולמה במקומם ובכיוון שונה
- ניתן לפרק את הקשר בין המצלמה ללוח השחמט גם על ידי הומוגרפיה
- בכל אחד, $h=0-Z=2$ בקואורדינטות העולם
- באמצעות עובדה זו נוכל לכלי (קליברציה) יישורת כדי לחוץ K

(גם כאן, בכל התמונות המצלמה נשארת במקום ורק הזרות משתנה)

(PnP) (Pose Estimation): (נקרא גם Chose-PnP)

מציאת פרמטרי t , R , t ממציאת פרמטרים מהיצורן, ונניח שיש לנו נקודות בעולם ונתן ידיעות לאן הן מוטלות במשור התמונה הפרמטרים מהיצורן, וכך ניתן למצוא את הפרמטרים החיצוניים של המצלמה (t , R) אזי נוכל למצוא את הפרמטרים החיצוניים יש דרגת חופש 6 וכן באופן מינימלי מספיק לנו 3 נקודות וזאת כפי שלמדנו, לפרמטרים החיצוניים יש דרגת חופש 6 ולכן מינימלי מספיק לנו 3 נקודות וזאת גם הסיבה שהוראים לשיטה גם בשם "PnP". [האלגוריתם רגייש לטעויות]. בעזרה אלגוריתם זה נוכל לחבר סצנה תלת ממדית גם אחריו תזוזה של המצלמה

Reprojection Error:

הערכת האיכות של מטריצת המצלמה המשוחזרת P (מטריצה שקיבliśmy מאלגוריתם

P^1, P^2, P^3 are the rows of P .

The reprojection error is

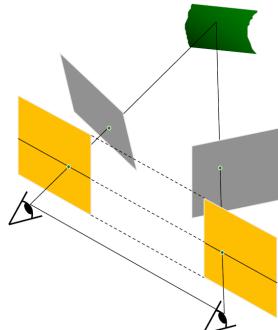
$$\left\| \begin{pmatrix} x_1 - \frac{P^1 \mathbf{X}}{P^3 \mathbf{X}}, & x_2 - \frac{P^2 \mathbf{X}}{P^3 \mathbf{X}} \end{pmatrix} \right\|^2$$

גם כאן, כמו בחישוב השגיאה של הומוגרפיה, מחשבים את ההפרש (לוקחים נורמה) בין הנקודה בעולם לאחר הפעלת הטרנספורמציה לבין הנקודה המתאימה במשור. (במילים אחרות נקבל כמה הנקודה "רוחקה" מהין שהיא אמורה להיות). נשים לב כי בחישוב הנ"ל מחלקים כי מדובר בהומוגניות אך ניתן לבצע גם עבור נקודה לא הומוגנית.

מצגת 10:

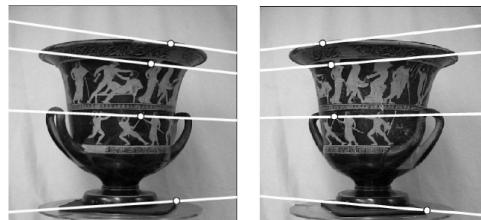
:3D Reconstruction

שחזור תלת ממד. נדבר על שחזור מ2 מצלמות (Two Views Geometry) ובהמשך נכליל. כשלמדנו על סטראו הנחנו שככל נקודה בתמונה אחת מתאימה לנקודה הנמצאת **על אותו קו** בתמונה השנייה. כמובן, נובע מכך שמשטחי הצלומים מקבילים. בפועל, משטחי הצלום של 2 המצלמות אינם בהכרח מקבילים:



(ניתן לראות כי משטחי התמונות של המצלמות [האפורים] אינם מקבילים)

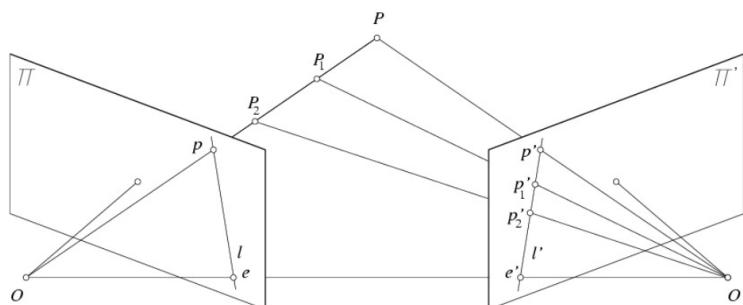
נרצה למצוא התאמות של נקודות בהתאם לחישוב הגיאומטרי של המשטחים



גם במקרה זה ישנה תconaה בה לכל נקודה בתמונה אחת ישנו קו ספציפי בתמונה השנייה עליו הנקודת המתאימה נמצאת. כמובן, לא יודעים היקן הנקודת המתאימה, אך יודעים על איזה קו. מכאן, לכל ההתאמות [זוג נקודות] בין התמונות, ישנו זוג קווים בהתאמה הנקראים "קווים אפיופוריים" את היחס בין הקווים ניתן להגדיר באמצעות הכפלת מטריצה הנקראת "Essential Matrix" על מנת למצוא את המטריצה נctrar לדעת את מיקומי המצלמות בעולם

באופן אינטואיטיבי, כל נקודה בתמונה אחת (שמאל בתמונה למיטה) התקבלה מקרן אוור קר שלא ידוע מאייזה אובייקט [הנמצא בעומק מסוים] הגיע האור. מנגד, בתמונה השנייה יתכן וישנם הרבה דגימות **מאותה** קרן אוור, ומוחוק גיאומטריה נקל שכל הדגימות יהיו על גבי קו ספציפי בתמונה **השנייה** (ימין בתמונה למיטה).

רק הדגימה שנלקחה מאותה אובייקט היא המתאימה לנקודה בתמונה שמאל:

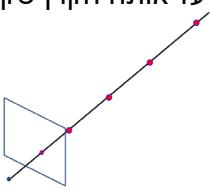


(נשים לב שנקודה בתמונה שמאל מייצרת אוסף של נקודות בתמונה ימין המתפרשים על קו אחד)

Euclidean Geometry: מרחב אוקלידי הוא כמו שŁMDNÓ בבית ספר עם פיתגורוס וכו'. קלומר יש משמעות למרחך.
אינוריאנטיים: אורכיים, זוויות ומקבילות, שכיחות (נקודות)

Projective Geometry: במרחב זה אין חשיבות למרחך. לדוגמה, זה כמו שŁMDNÓ תמונה של 2 מסילות רכבות סמוכות, אך בתמונה זה נראה שהן מתקרבות באינסוף אבל זה לא נכון. זה גם מה שקרה בהומוגרפיה, לאחר ביצוע טרנספורמציה הומוגרפית כל קו נשאר קו אך אין שמירה על מקבילות.
אינוריאנטיים: קווים ונקודות

כאן, בProjective Geometry Two Views אנו מוחשים את הקשר בין הנקודות ע"פ גיאומטריה זו, נבע כי כל הנקודות על אותה הקרן שקולות:



(הכפלת נקודה מסוימת בסקלר מאריכה את הווקטור ונקבל נקודה אחרת על אותה הקרן)

עבור כל נקודות בתמונה אחת נרצה למצוא את הנקודות המתאימות בתמונה השנייה
קו מוגדר ע"י פרמטרים c, b, a בצורה הבאה:

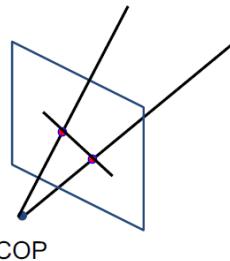
$$ax + by + c = 0 \text{ or } (a, b, c)^T (x, y, 1) = 0$$

(בדרכ"כ מהצורה $c + ax = y$ אך כאן ממשיים l_0)

ולכן, כל נקודה הומוגנית $(1, y, x)$ המאפשרת את המשוואה הקו [כאשר הקו מוגדר ע"י $c + ax + by = 0$].

נמצא על הקו. כמובן, מתקיים: $0 = l_0^T l$.

בහיבט של תלת ממד, הנקודות והקו מאונכים:



(כאן מסתכלים מבחינה גיאומטריה קוולקטיבית [קואורדינטות הומוגניות])

מכיוון שניינו להגדיר קו ע"י 2 נקודות (מתיחת קו בינם) אז בהינתן 2 נקודות, נוכל למצוא את הקו
טכנית אם נחפש שלישיה $(c, b, a) = l$ כך שמתקיים:

$$l_1^T \cdot l = 0 \text{ and } l_2^T \cdot l = 0$$

כלומר, 2 הנקודות מfasoot, שהה אומר ש2 הנקודות נמצאות על הקו.

ניתן למצוא את הוקו הנ"ל באמצעות פוליה אלגברית הנקראת Cross Product (שימושי בגיאומטריה):

Cross Product

בහינתן 2 וקטורים V_1, V_2 , פוליה Cross Product עליהם נותנת וקטור חדש W הנמצא בשנייהם.

כלומר, עבור הפעולה $W = V_1 \times V_2$ נקבל וקטור W הנמצא בכל המשטח.

ומכיון שהוא וקטור W ניצב לשניים אחד הכפלה של W בכל אחד מהווקטורים יהיה שווה ל-0.

נוסחת החישוב בתלת ממד:

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \times \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} v_2 \cdot w_3 - w_2 \cdot v_3 \\ v_3 \cdot w_1 - w_3 \cdot v_1 \\ v_1 \cdot w_2 - w_1 \cdot v_2 \end{bmatrix}$$

(נשים לב שגם החיסרה של כל צמד מהווקטורים בהצלבה)

נוסחת Cross Product הנ"ל מתקיימת ב-3D.

במקרה לביצוע Cross Product ישירות ע"פ הנוסחה לעיל, נוכל להמיר את הנקודהה הראשונה למטריצה מסוימת כך שבהכפלה רגילה של המטריצה בנקודהה השנייה נקבל את אותה תוצאה:

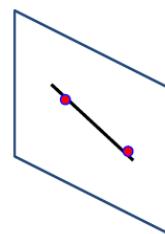
- **Cross Product in matrix notation**

$$- p_1 \times p_2 = [p_1]_x \cdot p_2 = \hat{p}_1 \cdot p_2$$

$$[p_1]_x = \begin{bmatrix} 0 & -z_1 & y_1 \\ z_1 & 0 & -x_1 \\ -y_1 & x_1 & 0 \end{bmatrix}, p_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}$$

מכאן, בהינתן הנקודות P_1, P_2 הפעולה \vec{l} תיתן:

$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \times \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} y_1 z_2 - z_1 y_2 \\ z_1 x_2 - x_1 z_2 \\ x_1 y_2 - y_1 x_2 \end{pmatrix}$$



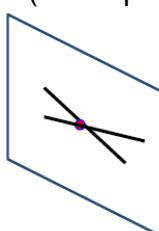
כאשר (כפי שאמרנו לעיל) מתקיימים:

$$p_1^\top \cdot p_2 = 0 \text{ and } p_2^\top \cdot l = 0$$

אם נבצע Cross Product על שני קווים $\vec{l}_1 \times \vec{l}_2$ נקבל נקודה P כך שמתקיים:

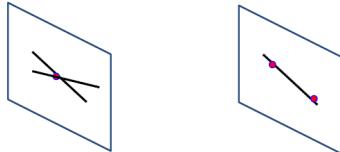
$$p_1^\top \cdot l_1 = 0 \text{ and } p_2^\top \cdot l_2 = 0$$

כלומר, הנקודהה נמצאת על שני הקיימים: (בחיתוך שלהם)



בහינתן קו כלשהו, בהכפלה של נקודה הנמצאת על הקו בקו עצמו קיבל 0. כמו כן, ע"פ מה שלמדנו, עבור $l = P_1 \times P_2$ הקו l הוא קו עליון נמצאות 2 הנקודות ולכן נובע $0 = P_1(P_2)$.

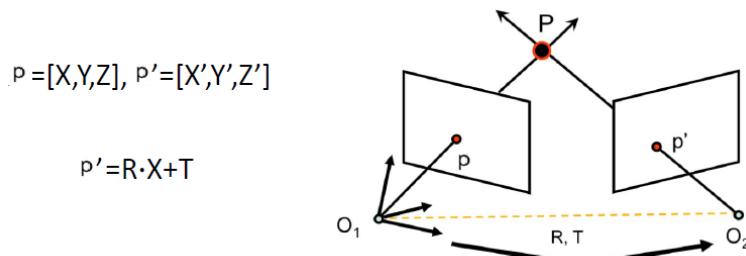
בהינתן $W = U \times U$ אז גם אם לא נדע אם U הם קווים או נקודות היחסים בניהם ישמרו. זה נקרא **duality** בין הקווים לנקודות בגיאומטריה קוילקטיבית. לומר כל המשפטים יהיו נכונים בין אם הם נקודות ובין אם הם קווים (קרניים).



מטריצה יסודית (Essential Matrix)

מטריצת Essential היא מטריצת המיקום היחסי של המצלמה (ההסבר למטה) ההנחה היא שאנו יודעים את מטריצת המצלמה (עבור כל מצלמה בהתאם)

מטריצת Essential מאפשרת לנו לדעת עבור מיקום של נקודה במצולמה אחת מה מיקום המתאים במצולמה השנייה. למשל, מעבר ממערכת קואורדינטות של נקודה במערכת הצירים של מצלמה אחת למערכת קואורדינטות של נקודה במצולמה השנייה (**לא מדובר בקואורדינטות של הפיקסל עצמו, בשביל זה צריך את מטריצת הפרמטרים הפנימיים**):



ניתן לראות שנוכל לדעת את מיקום הנקודה במצולמה اليمنית באמצעות המיקום של הנקודה במצולמה השמאלית ע"י T, R שהם הפרמטרים החיצוניים (כמו הרזזה ורוטציה של המצלמה השנייה מהראשונה).

באופן תיאורטי מה צריך לעשות זה לחת את הנקודה (לדוגמא השמאלית) ולבטא אותה באמצעות קרן. לאחר מכן נצטרך לדעת מה הפרמטר החופשי שהוא הסקלר שאומר איפה הנקודה ישבת בפועל על הקרן (עומק).

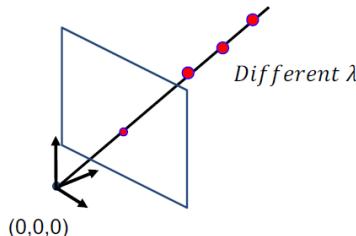
נניח כי הנקודה הינה:

$$X = [X, Y, Z] \in \mathbb{R}^3$$

אז, אם הערך focal length הוא 1 קיבל את מיקום הנקודה במישור התמונה:
 $x = [x/z, y/z, 1]$

(בגלל שהערך focal length הוא 1 אז מכפילים את הקואורדינטות ב-1)
 ומכאן, נוכל לקבל כל נקודה על הקרן בצורה הבאה:

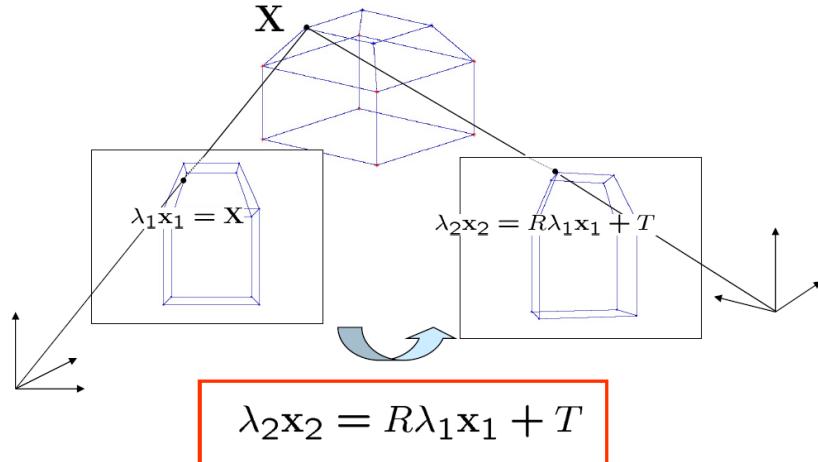
$$\lambda x = \lambda [x/z, y/z, 1]$$



מכאן, בהינתן 2 נקודות x_1, x_2 מattaיות, אזי אם נדע את הפרמטרים R, T בין המצלמות אז יוכל למצוא את λ_1, λ_2 כר' שמתקיים:

$$X = R \cdot X + T \Rightarrow \lambda_2 x_2 = R \lambda_1 x_1 + T$$

כלומר, נמצא את העומק (מל' נקודה) בהם הקרןיהם "פגשות".



פיתוח: (Essential Matrix)

$$\begin{aligned} \lambda_2 x_2 &= R \lambda_1 x_1 + T \\ \hat{T} \lambda_2 x_2 &= \hat{T} R \lambda_1 x_1 + \hat{T} T \\ \hat{T} x_2 \lambda_2 &= \hat{T} R x_1 \lambda_1 + \hat{T} T \\ \hat{T} x_2 \lambda_2 &= \hat{T} R x_1 \lambda_1 + 0 \\ x_2^T \hat{T} x_2 \lambda_2 &= x_2^T \hat{T} R x_1 \lambda_1 \\ 0 &= x_2^T \hat{T} R x_1 \lambda_1 \\ 0 &= x_2^T \hat{T} R x_1 \end{aligned}$$

הסביר:

נכפול כל אגף משמאלי ב \hat{T} (Cross Product עם T) ולכן הביטוי $\hat{T} \hat{T}$ יתאפשר (כי הם מאונכים). לאחר מכן, נכפול משמאלי כל אגף ב x_2^T ולכן אגף שמאלי יתאפשר (כי $x_2^T \hat{T}$ מאפס גם את T וגם את x_2 [מאונך אליו]). וזה אומר שנתקבל ביטוי שווה ל0). מכאן, נקבל את הביטוי המופיע בסוף (התעלמנו מ λ_1 בשורה האחורונה מכיוון שהביטוי הימני שווה ל0 ולכן נוכל להתעלם מסקלר).

ולסיום נקבל את הביטוי:

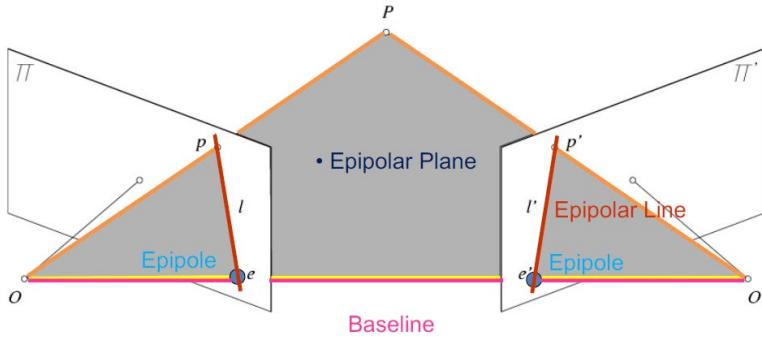
$$x_2^T \underbrace{\hat{T} R}_{E} x_1 = 0$$

כאשר באמצעות הביטוי נקבל את מטריצת ה-essential (המטריצה בסדר גודל של 3×3)

$$x_2^T \begin{bmatrix} e_1 & e_2 & e_2 \\ e_4 & e_5 & e_6 \\ e_7 & e_8 & e_9 \end{bmatrix} x_1 = 0$$

נשים לב למשמעות המטריצה: בהינתן המטריצה ונקודה אחת (x_2), אזי כל נקודה שניצב בתווך x_1 שתאפשר את המשוואה היא נקודה הנמצאת על הקוו שמוסטל מהנקודה x_2 .
ומכאן אם x_2 נמצאת על תמונה אחת, אוסף הפתרונות (הנקודות המאפשרות) הם קו בתמונה השנייה.

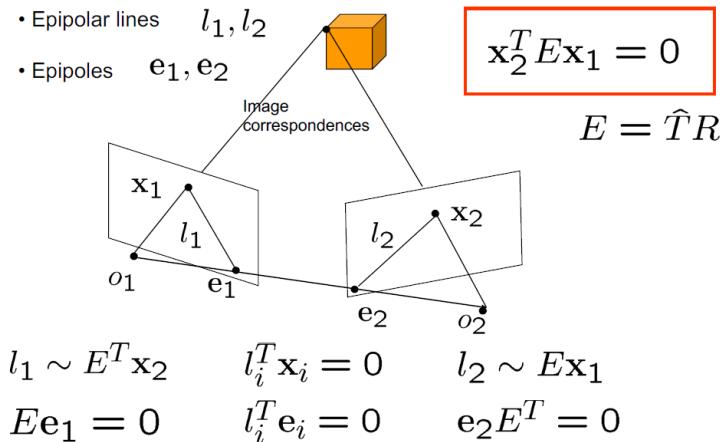
גיאומטריה אפיולרית:



Baseline: המרחק בין ה-COP-ים של 2 המצלמות
Epipole: נקודת החיתוך של Baseline עם מישור התמונה
Epipolar plane: משטח משולש הנוצר ע"י נקודה בעולם P והענין (כביס המשולש)
Epipolar line: החיתוך של הענין של התמונה עם מישור Epipolar plane

נשים לב כי הענין Epipolar plane אינו יחיד כי כל נקודה בעולם P יוצרת Epipolar plane שונה.
 כל הקווים האפיולריים נחתכים עם האפיולרי.
 המישור האפיולרי חוצה כל מישור תמונה (בהתאם) בקו אפיולרי.
 בשורה התחתונה, הקו האפיולרי זה כל ההתאמות.

ובע מכל האמור לעיל:



כאשר:

- $l_1 \sim E x_2$ מכיוון שהכפלת מטריצתessential מטילה את הקו l_1 (כל ההתאמות) בתמונה השנייה (סימון "תילדה" ולא "שווה" מכיוון שהוא עד כדי סקלר).
- $l_2 \sim E^T x_1$ מכיוון שמדובר בקו הומוגני וגם אם מכפילים בסקלר הוא נשאר אותו הדבר.
- $E e_1 = 0$ אם היינו יודעים את הנקודה התלת ממדית הינה יכולם לדיק גם בסקלר עצמן).
- $l_i^T e_i = 0$ מכפלת כל קו בנקודת שוכנתה עלינו מניב 0 כמו שראינו בתחילת המציגת
- $l_1 \sim E^T x_2$ (ובע במידה דומה כפי שהסבירנו את $l_2 \sim E x_1$)
- $E e_1 = 0$ נבע מתרגיל מעוניינ – נקודת האפיול נמצאת על כל קו האפיול ולכן מאפסת את כל הקווים. מכיוון שכך, הוא מופיע גם את E כי כל הקווים נוצרים מ- E .
- $e_2 E^T = 0$ (ובע במידה דומה כמו שהסבירנו את $E e_1 = 0$).
- $l_i \sim E^T e_i = 0$ מכיוון שכפי שאמרנו נקודת האפיול מאפסת את כל הקווים.

:Estimating Essential Matrix

נרצה לשערר מהי מטריצת ה

לעיל קיבלנו את המשוואה $\mathbf{0} = \mathbf{x}_2^T \hat{\mathbf{T}} \mathbf{R} \mathbf{x}_2$

ניתן לרשום אותה בצורה הבאה: (לא הרחיב איך מפורמלים לצורה זו)

- Denote $\mathbf{a} = \mathbf{x}_1 \otimes \mathbf{x}_2$

$$\mathbf{a} = [x_1 x_2, x_1 y_2, x_1 z_2, y_1 x_2, y_1 y_2, y_1 z_2, z_1 x_2, z_1 y_2, z_1 z_2]^T$$

$$\mathbf{E}^s = [e_1, e_4, e_7, e_2, e_5, e_8, e_3, e_6, e_9]^T$$

ונקבל את המשוואה $\mathbf{0} = \mathbf{a}^T \mathbf{E}^s$ אותה ניתן לפתור עם DLT (פתרית least square).

מכאן נרצה למצאו:

$$\min_{\mathbf{E}^s} \| \mathbf{A} \cdot \mathbf{E}^s \|$$

כאשר \mathbf{A} היא מטריצה שכל שורה בה היא אילוץ מנוקודה (כל נקודה זה \mathbf{a} אחד שראינו לעיל) **כאן כל נקודה מייצרת משווה אחת ולכן לא מספיק 4 נקודות אלא ציריך 8.**

מבחינה אלגברית \mathbf{E} מכיל 9 פרמטרים אבל בגלל שמדובר בהומוגניות אז בחלוקת באיבר האחרון נקבל שישנים 8 געלמים.

בפועל, מבחינה גיאומטרית יש 3 דרגות חופש עבור רוטציה ו-3 דרגות חופש עבור translation. בנוספ', בגלל שזו מטריצה הומוגנית מתקיים שאם נכפיל אותה בכל מספר היא עדין תיתן לנו את מה שציריך ולכן נוכל לחלק באיבר האחרון וכן להוריד דרגת חופש אחת וכך נקבל 2 דרגות חופש בלבד עבור translation. ומכאן סה"כ דרגות החופש הוא **5** בלבד.

ישנה דרך נוספת לנកראת "5-points algorithm" שהיא ניתנת לפתור בעזרת 5 נקודות בלבד (לא הרחיב). אבל נוכל לפתור כמו קודם עם 8 נקודות.

משפטים הנוגעים למטריצת ה

Theorem - Essential Matrix Eigenvalues

A matrix \mathbf{E} is an essential matrix iff its SVD: $\mathbf{E} = \mathbf{U} \Sigma \mathbf{V}^T$ satisfies: $\Sigma = \text{diag}([\sigma_1, \sigma_2, \sigma_3])$ with $\sigma_1 = \sigma_2 \neq 0$ and $\sigma_3 = 0$ and $\mathbf{U}, \mathbf{V} \in SO(3)$

Theorem - Pose Recovery

There are two relative poses (\mathbf{R}, \mathbf{T}) with $\mathbf{T} \in \mathbb{R}^3$ and $\mathbf{R} \in SO(3)$ corresponding to a non-zero matrix essential matrix.

כאשר (3) SO היא חבורת המטריצות מסדר גודל 3×3 כאשר הערך העצמי השלישי שלהם הוא 0 ושני הערכים עצמאיים האחרים שווים.

מאפייני מטריצת ה הנגזרים מהמשפטים הנ"ל:

- יש לה ערך עצמי אחד שהוא 0 (הآخرן).
- הDeterminant של המטריצה הוא 0 ולכן אינה הפיכה.
- שני הע"ע הראשוניים שלה שווים.
- יש לה שני מרחבים (poses) שמתאימים לה [כי כיש 2 תמונות אז תמיד יהיו 2 זוויות שונות המביאות את היחס בינהן] (בחירה המערכת הקואורדינטאות של המטריצה קובע את המנה כמו שנראה בהמשך).

כאמור, אנו יכולים לפטור בעזרת DLT את המשוואה:

$$\min_{E^s} ||A \cdot E^s||$$

אבל לא תמיד נקבל את המאפיינים של הע"ע לעיל. לכן, נאיץ זאת בצורה הבאה:

1. SVD: $E^s = A\Sigma B^T$ where $\Sigma = diag(r, s, t)$
2. $\Sigma' = diag(1, 1, 0)$
3. $E = A\Sigma'B^T$

כלומר, נשנה את האלכסון של Σ המכיל את הע"ע (МОוטר לנו לעשות זאת שירוטית כי זה עד כדי כפל בסקלר)

מציאת E וחילוץ t : R:

- Obtain $E = U\Sigma'V^T$ from proper form
 - If $\det(UV^T) = -1$ we need to take $-E$ instead of E
 - Right hand vs. left hand coordinate system
- Recover Translation
 - Take the 3rd column of U
 - Normalize it to length 1: $T = \frac{u_3}{\|u_3\|}$
- Recover Rotation:

$$R_1 = UWV^T, R_2 = UW^TV^T \text{ with } W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

כלומר, בוחרים את E בהתאם למערכת הקואורדינטות [ימנית / שמאלית] (לא הרחיב) וממנה גם אפשר לחזור את t . [נזכיר כי $E = \hat{T}R$].

בפתרון המשוואות שעשינו קיבל כמה פתרונות. הדרך לבחור את הפתרון הנכון היא לחפש את הפתרון בו הנקודה נמצאת לפני המצלמה:

- The sign of E cannot be known since it is in homogenous coordinates
- It follows that the translation vector can be either positive or negative and it is a valid solution
- The four possible solutions are:
 - $(R_1, T), (R_1, -T), (R_2, T), (R_2, -T)$
- The way to recover the correct solution is to do 3D reconstruction of a point in the scene
 - In only one solution the point will be in front of both cameras

(מטריצה בסיסית): Fundamental Matrix

מטריצה זו דומה למטריצת essential מבchnint המטרה. השוו בינהם הוא:

1. מטריצת essential מאפשרת לחשב נקודה מתאימה לנקודה מסוימת מבchnint המיקום במערכת הצירים של המצלמה. لكن, לאחר מכן חיברים את מטריצת הפרמטרים הפנימיים כדי להמיר את הנקודות מערכות הצירים של המצלמה למערכת קואורדינטות של פיקסלים.
2. כאן במטריצת fundamental, ההנחה היא שגם ל偶像יות הפרמטרים הפנימיים של המצלמה لكن, נהיה מוכרים להשיג מטריצה שמיירה מערכות קואורדינטות של פיקסלים של מצלמה אחת למערכת קואורדינטות של פיקסלים בצלמה השנייה (מקרה יותר גנרי)

[תזכורת: בקואורדינטות מצלמה 0,0 יכול להיות במרכז או בצד ימין למיטה. אבל לאחר המרת מקואורדינטות פיקסלים נקבל שהו,0 נמצא בצד שמאל למעלה כמו שהוא מכירם]

- We do not know the K's
- We have matching pixels $x \leftrightarrow x'$
- Define: $\hat{x} = K^{-1}x$, pixel to camera coordinate
- The Essential Matrix is $\hat{x}'^T E \hat{x} = 0$
- And in terms of pixels: $x'^T K'^{-T} E K^{-1} x = 0$
- We have 3x3 matrix F: $K'^{-T} E K^{-1}$, which relates each pixel in 1st image to a line in the 2nd
- The Fundamental Matrix constraint: $x'^T F x = 0$

(אם לוקחים נקודה במישור התמונה ומכפילים משמאלי ב [מטריצת פרמטרים פנימיים] נקבל מעבר מקואורדינטות מצלמה [מישור התמונה] למקואורדינטות פיקסלים. אבל אם נכפיל ב K^{-1} זו המרת ההפוך, כמובן, מקואורדינטות פיקסלים למקואורדינטות מצלמה [מישור התמונה])

מטריצת fundamental היא מדרגה 2 (לא מטריצה הפיכה כי מטריצה 3x3 צריכה דרגה 3 בשבייל להיות הפיכה).

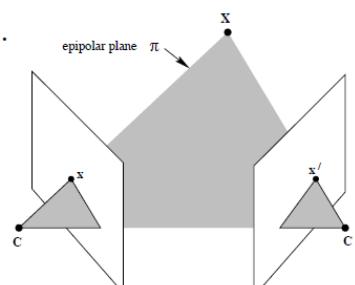
מטריצת fundamental זהה למטריצת essential מבchnint ההגדרות:

Epipolar lines:

- ◊ $l' = Fx$ is the epipolar line corresponding to x .
- ◊ $l = F^T x'$ is the epipolar line corresponding to x' .

Epipoles:

- ◊ $Fe = 0$.
- ◊ $F^T e' = 0$.



הבדל היחיד הוא שמספר דרגות החופש המינימלי בessential הוא 5, וכך, לעומת fundamental מספר דרגות החופש המינימלי הוא 7 (אם פותרים בדרך פולינומית עם מציאת שורשים וכו'. ניתן לפתור עם דרך לינארית עם 8 כפוי שנראה [במטריצה יש 9 מספרים אך אחד קבוע וכן נשאר עם 8]).

נניח ויש לנו התאמה של 2 נקודות במערכת הקואורדינטות של ה פיקסלים :

$$\mathbf{x}'^T \mathbf{F} \mathbf{x} = 0, \quad \mathbf{x}_i \leftrightarrow \mathbf{x}'_i$$

$$\mathbf{x} = (x, y, 1)^T \quad \mathbf{x}' = (x', y', 1)^T \quad \mathbf{F} = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$$

(כאשר \mathbf{F} לא ידוע לנו)

אז כל התאמה נוכל לבנות משווה בצורה הבאה:

$$x'x f_{11} + x'y f_{12} + x'f_{13} + y'x f_{21} + y'y f_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0$$

כל נקודה מעניקת משווה אחת, לכן, נדרש 8 משווות.

בעtron הפתרון המינימלי מצרי 7 נקודות אך אנחנו לוקחים 8 נקודות (אמר שלא נכנס לזה עצה נקבע):

$$\begin{bmatrix} x'_1 x_1 & x'_1 y_1 & x'_1 & y'_1 x_1 & y'_1 y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \\ x'_n x_n & x'_n y_n & x'_n & y'_n x_n & y'_n y_n & y'_n & x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

כלומר, צריך למצער את הביטוי:

$$\|\mathbf{Af}\|$$

בנוסף, ב-DL מוסיפים תנאי נוסף והוא שהנורמה של \mathbf{f} צריכה להיות 1.

אחד ממאפייני המטריצה \mathbf{F} הוא שדרגתה 2 ולכן אינה הפיכה (Determinant ≠ 0).

לפעמים בغالל שגיאות בההתאמות הנקודות, לא תמיד נקבל \mathbf{F} היא מדרגה 2.

לכן, לאחר הרצת SVD נאלץ זאת ע"י איפוס הערך הסינגולרי השלישי:

Let $\mathbf{F} = \mathbf{U}\Sigma\mathbf{V}^T$ where

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix}, \text{ let } \Sigma' = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

then $\mathbf{F}' = \mathbf{U}\Sigma'\mathbf{V}^T$ is the solution.

האלגוריתם הינו: (8-point algorithm)

```
% Build the constraint matrix
A = [x2(1,:)'.*x1(1,:)' x2(1,:)'.*x1(2,:)' x2(1,:)' ...
      x2(2,:)'.*x1(1,:)' x2(2,:)'.*x1(2,:)' x2(2,:)' ...
      x1(1,:)'           x1(2,:)'           ones(npts,1) ];

[U,D,V] = svd(A);

% Extract fundamental matrix from the column of V
% corresponding to the smallest singular value.
F = reshape(V(:,9),3,3);

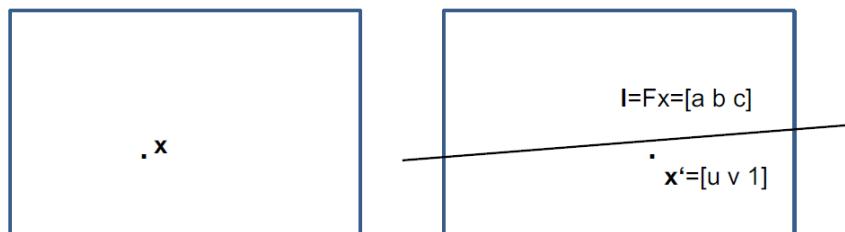
% Enforce rank2 constraint
[U,D,V] = svd(F);
F = U*diag([D(1,1) D(2,2) 0])*V;
```

בשאיפית 35 הסביר (לא הרחיב) שմבחן גנריות כשפוגרים כפי שהסבירו לעיל אז יכולות להיות הרבה שגיאות כי יש הבדלים של סדרי גודל בנקודות (התאמת שגואה של נקודות שה הפרש ביניהם גדול מאוד). אז ישנה טרנספורמציה שמנורמלת את הנקודות ע"י התחרשות בשונות ואז יוצא שמבחן גנריות יותר קל לפתור.
 בשאיפית 46 הסביר (לא הרחיב) שניתן לחץ את מטריצת המצלמה P' , P מטריצה F

אם קיבלנו מטריצת F או E . איך יודעים כמה הוא טוב?
 נשים לב כי לא בדיק מספיק להשוות בין $'x$ (שזהו נקודה ספציפית בתמונה הראשונה) אל Fx כי Fx הוא קו שטוח בתמונה השנייה (מוטל ע"י x) ולכן, אנו בודקים מרחק של הנקודה $'x$ מהקו
 כלומר, נקבל פונקציית שגיאה: (כמו פון השגיאה של ההומוגרפיה רק שכאן בודקים מרחק מקו)

$$\sum_i d(x'_i, Fx_i)^2 + d(x_i, F^T x'_i)^2$$

 נוכל לפתח ולקבל פורמלציה שונה של הנוסחה: (לא נראה איך מפתחים)
 Distance of a point to epipolar line



$$d(l, x') = \frac{|au + bv + c|}{\sqrt{a^2 + b^2}}$$

The final formula: $\sum_i (x'^T F x_i)^2 \left(\frac{1}{(Fx_i)_1^2 + (Fx_i)_2^2} + \frac{1}{(F^T x'_i)_1^2 + (F^T x'_i)_2^2} \right)$

נקרא Symmetric Epipolar Distance כי הבדיקה זו כיוונית. נוסחה זו נראית יותר מהירה אך הנוסחה הקודמת גם נכוןה.

הראנו בשיעורים קודמים שבהינתן סט התאמות ישנו אלגוריתם הנקרא RANSAC המאפשר לבנות מודל תור מזעור השגיאות (הנובעות מנקודות outliers).

גם כאן, נוכל להשתמש ב-RANSAC בשביל למצוא את מטריצת F :

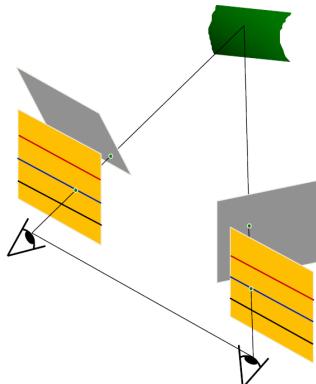
- בחר אקריאת 8 התאמות
- חשב את F בעזרת DLT
- חשב את השגיאה (Symmetric Epipolar Distance)
- חוזר על הפעולה N פעמים ובחור את מטריצת F הכי טובה

לסיכום, השוואת Essential מול Fundamental

1. גיאומטריה אפיולרית:
 - בessential צריך לדעת ערכים של קליברציה של המצלמה (פרמטרים פנימיים)
 - בfundamental אנו מחשבים גם אותם (מספריק לעבור רק עם הפיקסלים)
2. הנחות:
 - בessential צריך התאמות + פרמטרים פנימיים של המצלמה
 - בfundamental צריך התאמות
3. מספר נקודות מינימלי:
 - בessential צריך 5 נקודות (בפתרונות פולינומית. ניתן גם לינארית עם 8 נקודות [המטריצה עם 9 מספרים אבל 1 קבוע ולכן נשאים עם 8])
 - בfundamental צריך 8 נקודות (בפתרונות לינארית. ניתן גם עם 7 נקודות פולינומית) [בפתרונות לינארית המטריצה עם 9 מספרים אבל 1 קבוע ולכן נשאים עם 8]
4. דרכי פתרון:
 - בessential: RANSAC + Linear
כאמור, אפשר לפתור עם פתרון פולינומי מינימלי (5 נקודות) כאשר על הנקודות מפעילים RANSAC ואז את הפתרון הפולינומי.
 - בfundamental: RANSAC + DLT + Nonlinear
כאן מדובר על פתרון לא לינארי שנותן תוצאות מאוד מדויקות. הבעה בפתרונות לא לינאריים היא שאין רק פתרון אחד ולכן נקודות ההתחלה חשובות. מסיבה זו מקדים פתרון עם DLT RANSAC ו-DLT RANSAC + Linear.
5. תוצאות:
 - בessential: מציאת התאמות. בנוסף, ניתן למצוא את R ואת T (את K יודעים)
 - בfundamental: מציאת התאמות. בנוסף, ניתן גם למצוא את מטריצת המצלמה.
 - ניתן לבצע: Rectification, Triangulation, Structure from motion עליהם נדבר כעת

(糾正: Rectification)

כשדיברנו על סטראו ההנחה הייתה שככל נקודה נמצאת על אותה שורה בתמונה השנייה
אך הזכרנו שזה לא תמיד ככה:



ביצוע Rectification מאפשר יישור התמונות כך שכל נקודה המתאים לנקודה הנמצאת בקו מקביל בתמונה השנייה. זה מתבצע באמצעות טרנספורמציה Warping (לא הרחיב עלייה).



(בפועל אם נישר את התמונות אחרי הטרנספורמציה אז הקווים כבר לא יהיו מקבילים)

באופן פשוט יותר, נוכל לעשות זאת ע"י (Epipolar rectification) בצורה הבאה:

- מעבור על כל נקודה בתמונה הראשונה
- את הקו האפיפולרי המתאים לנקודה (שמוטל בתמונה השנייה ולא בהכרח באותו גובה של הנקודה) נשים בשורה אחת בתמונה חדשה (על אותו צ)
- (ניתן למצוא את הקו האפיפולרי המתאים באמצעות F)
- חזר מעבור כל הנקודות עד ליצירת תמונה מלאה
- (נשים לב כי בסיום כל נקודה מישרת באותו גובה עם הקו האפיפולרי של עצמה)

:N-Linearities - Multiple View Tensors

עד כה דיברנו על Two-Views. כמובן, על קשר בין 2 תמונות.

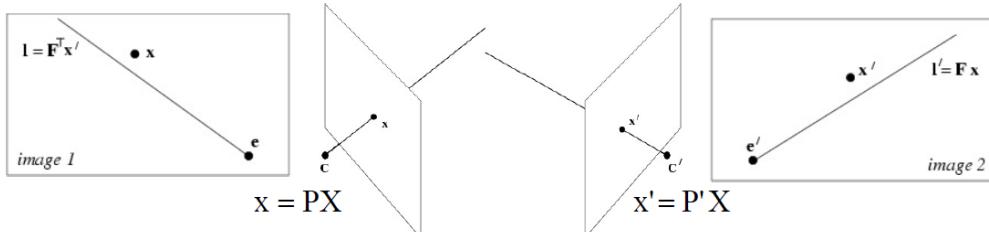
ניתן להרחיב את הקשר זהה עבור מספר N גובה יותר של תמונות.

אמר שלא ניתן לזרע ארכוסביר באופן כללי זהה מתבצע באמצעות דטרמיננטות קר שנתקבל פורמלistically שנכונה לכל כמות תמונות שהיא וניתן את הקשרים מכל נקודה לכל אחד מהקווים. ההבדל הוא שמקודם קיבלנו קשר מנוקודה לקו, כאן מכיוון שישנם כמה קווים אז זה מצלב ונקבל קשר מנוקודה לנקודה.

מצגת 11:

:Triangulation

שחזור נקודה תלת ממד בהינתן פיקסלים ומטריצות מצלמה P, P' (מmirות מנוקודה בעולם לפיקסל)
כלומר, נרצה לדעת היקן הקרןים (המוטלות מ2 הנקודות) נגשות וכך נוכל לשחזר עומק
הבעיה היא שיש רוש ונקודות שכטיכול אמורות להיות מתאימות לא באמת יפגשו (יחתכו)



(ניתן לראות בתמונה ימין ובתמונה שמאל את המרחקים של הנקודות מהקוויים האפיופולרים)

נניח ונרצה לשחזר נקודה X בעולם, אז נוכל למדל את הבעיה בצורה הבאה:

$$\mathbf{X} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad \mathbf{x}' = \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \mathbf{p}_3^T \end{bmatrix} \quad \mathbf{P}' = \begin{bmatrix} \mathbf{p}_1'^T \\ \mathbf{p}_2'^T \\ \mathbf{p}_3'^T \end{bmatrix}$$

$$\begin{array}{ll} \mathbf{x}' = \mathbf{P}'\mathbf{X} & \mathbf{x} \times (\mathbf{P}\mathbf{X}) = 0 \\ \mathbf{x} = \mathbf{P}\mathbf{X} & \mathbf{x}' \times (\mathbf{P}'\mathbf{X}) = 0 \end{array} \quad \Rightarrow \quad \mathbf{A}\mathbf{X} = \mathbf{0} \quad \mathbf{A} = \begin{bmatrix} \mathbf{u}\mathbf{p}_3^T - \mathbf{p}_1^T \\ \mathbf{v}\mathbf{p}_3^T - \mathbf{p}_2^T \\ \mathbf{u}'\mathbf{p}_3'^T - \mathbf{p}_1'^T \\ \mathbf{v}'\mathbf{p}_3'^T - \mathbf{p}_2'^T \end{bmatrix}$$

כלומר, נדרש קיום של נקודה בעולם X כך שלאחר הכפלת מטריצת המצלמה הראשונה נקבל את
הנקודה x המתאימה (במישור המצלמה הראשונה) וגם נדרש את אותו קיום עבור המצלמה השנייה
[רצים שכל נקודה המייצגת פיקסל תהיה שווה לביטוי שמורכב מהכפלת מטריצת המצלמה
(המתאימה) בנקודה X . לכן, מכיוון שהם שווים, ביצוע Cross Product בניהן יניב 0 (נובע מהנוסחה
שראיתנו במצגת 10)]

מסיבה מסוימת לאחר ביצוע Cross Product של כל משווהו לוקחים רק את 2 השורות הראשונות
מההתוצאה (ישנם שלושה שורות לפני הנוסחה) [+] וזה הסיבה שלמטריצה A יש 4 שורות ולא 6.
מכאן, מטריצת A היא המטריצה המתבקשת לאחר שנבצע $A = P \times x$
(נשים לב כי מספר העמודות של מטריצה A הוא 4, וכך, במקרה שלנו A מטריצה 4×4).

הפיתוח הנ"ל הוא בדיק ואוטו פיתוח כמו במטריצת Essential, Fundamental והומוגרפיה
כי גם בהומוגרפיה הפיתוח מtabased על הדרישת: $0 = Hx = x' \times H^{-1}x' = x' \times x$ או באופן דומה $0 = x \times x'$

ומכאן, לאחר שיש לנו את A , ניתן לפתור עם SVD:

Given $\mathbf{P}, \mathbf{P}', \mathbf{x}, \mathbf{x}'$

1. Precondition points and projection matrices
2. Create matrix \mathbf{A}
3. $[\mathbf{U}, \mathbf{S}, \mathbf{V}] = \text{svd}(\mathbf{A})$
4. $\mathbf{X} = \mathbf{V}(:, \text{end})$

כשמקבלים מטריצה לפעמים יש הבדלים בין הערכים של המטריצה (ערכים קטנים וערכים גבוהים)
הפרשים אלו משפיעים על אופטימיזציה מבחינה גנטית כי כל שגיאה קטנה בחישוב מניבה שגאה
גדולה בתוצאה וכן מנורמלים (ע"י הורדת mean [unit variance] זה נקרא pre-condition (pre-condition
ישנים נוסחות pre-condition של DLT).

יתרונות:

- קל לפירמול
- ניתן להקליל עבור כמה תמונות שנרצה ע"י שטוטיף תנאים ונקבל מטריצה A אורך יותר

חסרונות:

- זה לא משתנה באופן שלכתי (H<->P). כלומר, אם נכפיל את הפיקסלים ואת נקודות התלת ממד במטריצה הפיכה אנו אמרוים לקבל את אותה תוצאה וכן התוצאה לא גנריית אלא עם ערך אחד ספציפי

נשים לב, שכשכנינו את A ופתרנו לינארית הפכנו את הבעיה אלגברית כך שבפתרון אי-התיחסות גיאומטרית על הטלות. ولكن התוצאה לא כci מדוייקת.

נוכל לשפר זאת ע"י פתרון שאין לינארי: נדרש שההסתrema של הנזודות תהיה ע"י מטריצת F המקיים שהנקודות לא יהיו רוחקות מהקוויים האפיולרים:

Minimize projected error

$$d(x, \hat{x})^2 + d(x', \hat{x}')^2 \text{ subject to } \hat{x}^T F \hat{x} = 0$$

or equivalently subject to $\hat{x} = P \hat{X}$ and $\hat{x}' = P' \hat{X}$



לא הרחיב, אך אמר שישנו פולינום מדרגה 6 שאפשר לפתור בעזרתו את הבעיה, כלומר מזער את:

$$d(x, l(t))^2 + d(x', l'(t))^2$$

בעזרה פולינום זה ניתן למצאו את F המתאימה שתיתן נזודות יותר מדויקות.

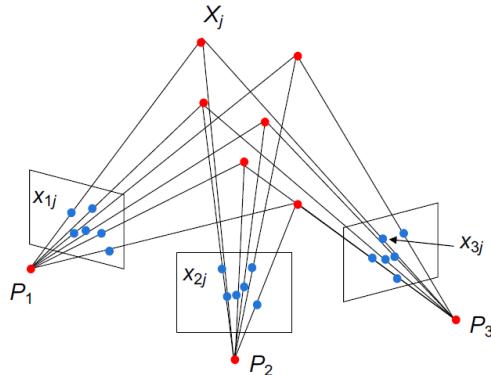
Projective Ambiguity: נניח ומוצאו נקודה X בעולם ונקודה x במישור, אז מטריצת המצלמה הממפה בנים אינה חד משמעית:

$$X = PQ^{-1}QX$$

נשים לב שעבור כל מטריצה Q הפיכה נקבל עוד פתרון.
 ויתר מזה, גם אם נדע את P הפתרון יהיה נכון עד כדי הכפלת בסקלר.

:Structure From Motion (SfM)

בנייה תלת ממד כתוצאה של תנועת המצלמה (שהיא הוזזה וווצעה של המצלמה בעולם).
נניח ויש לנו n תמונות של m נקודות בעולם:



כשיש לנו יותר מ-2 תמונות נקבל תוצאה יותר שלמה מכיוון שלא מכל זווית רואים הכל.
בנוסף, נקבל תוצאה יותר מדויקת כי בין כל 2 תמונות יש רוש.

לכל נקודה בעולם יש n התאמות במישורים (כלומר, n פיקסלים המתאימים לה)
ה"מבנה" הוא הנקודה בעולם. ה"תנועה" זה התנועה של המצלמה בעולם (הוזזה וווצעה) הניתנת
לחילוץ ממטריצת המצלמה.

אם אין לנו את מטריצת המצלמה זאת בעיה, ונוכל לשחרר אותה עד כדי מטריצה הפיכה Q כפ'
שראיינו (Projective Ambiguity) בעזרת קליבורציה. אבל זה לא משחרר את R ואת T כי שדריבנו על
קליבורציה אמרנו שהמטריצה שימושית כל עוד לא זדים.

ונכל לעשות זאת ע"י כך שנורשים את כל המשוואות ונפתרו עם Least Square כמו בסרטון Two-View:

$$\mathbf{x} \times (\mathbf{P}\mathbf{X}) = 0 \quad \mathbf{x}' \times (\mathbf{P}'\mathbf{X}') = 0 \quad \mathbf{x}'' \times (\mathbf{P}''\mathbf{X}'') = 0$$

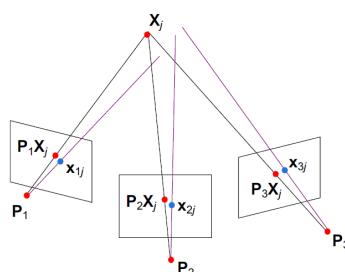
Sequential Sfm: הדרך איתה פותרים עבור כל הנקודות. בתחילת טריאנגולציה (Triangulation) כפי שראינו
עם Two-Views (מחלצים מטריצת מצלמה, מחלצים תלת ממד וכו') ואז מוסיףים כל פעם מצלמה
ומוסיףים מידע חדש (נקודות).
[כל פעם שמוסיפים מצלמה עושים אופטימיזציה כדי שיתאים לכל המערכת, ככלומר, גם למידע החדש].

:Bundle adjustment

שיטת לא לינארית לביצוע structure and motion ע"י מזעור השגיאה:

$$E(\mathbf{P}, \mathbf{X}) = \sum_{i=1}^m \sum_{j=1}^n D(\mathbf{x}_{ij}, \mathbf{P}_i \mathbf{X}_j)^2$$

כלומר, מחפשים מטריצת מצלמה \mathbf{P} , נקודה בעולם \mathbf{X} , ונקודה במישור \mathbf{x} כך שהמפרק בין הטלת \mathbf{X}
לנקודה \mathbf{x} יהיה מינימלי.



מתי M_{sf} יכשל?

- כאשר לא תהיה מספיק חפיפה בין התמונות (לא יוכל למצוא התאמות)
- תנוצה שהוא רוטציה ללא הזזה (כי Triangulation לא מאתרת תנוצה שהוא רוטציה)
- נזכיר שבפיתוח של מטריצת fundamental הנחנו שיש R וגם T . ואם אין את T לא יוכל לגלות תנוצה שהוא סיבוב למרות זהה מקרה יותר קל.

:Robust Estimation

כל מה שהסבירנו עד כה יותר נכון תיאורתי ופחות מעשי מכיוון שתמיד יהיה רעש (אוטולירט, נקודות שלא מתאימות וכו') אבל ניתן לפתור זאת עם RANSAC עבור כל אחד מהמודלים שלנו:

- Camera Calibration: K
- Camera Pose: R, T
- Essential Matrix
- Fundamental Matrix
- Homography
- Triangulation
- Structure from Motion

נctrיך אלגוריתם כללי להערכת מודל בהתחשב ברעש הנובע מਆוטולירט (לדוגמא, עבור קליברציה (כאשר הנתונים שלנו זה אוסף של התאמות בין פיקסלים ונקודות בתלת ממד) נרצה למצוא מטריצת מצלמה):

We need

- Model: P matrix
- Data: matching 3D and 2D(pixels): $\{(X_i, x_i)\}_{i=1..N}$
- Equations based on the data $x = Px$
- Error function: given the model P , how can we estimate its quality? for (X, x) , $|Px - x|$ should be minimum

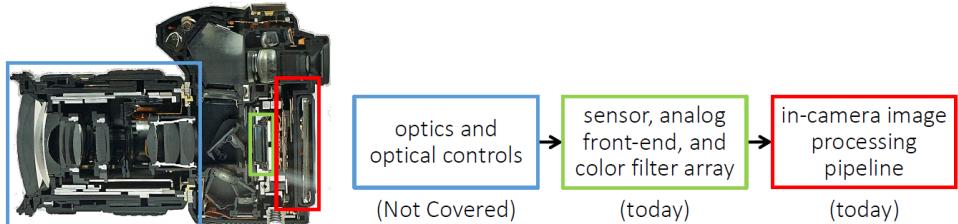
We can then use RANSAC

- Sample minimal set of data points
- Infer model
- Evaluate its quality using the error function
- Do it N times and select the best model

(נזכיר שRANSAC מקבל כמות מינימלית של נקודות הנוצרות למודל [אותם דוגמים מהכלל])

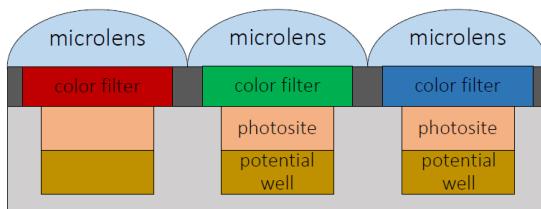
מצגת 12:

(את רוב המציגת זו בעיקר שכתבתי מהרצאה שלי להרחב בהසרים)
(מבנה המצלמה): Digital Pipeline



במצלמה יש תריס שקולט פוטונים (חלקים פיזיים שיש באור) ולוקדים אותם מהר עד שהתריס נסגר.
אלברט איינשטיין גילה קשר בין כמות הפוטונים הנקלטים (מה שעובר עם האור) לבין האלקטרונים (זרם) הנפלטים
כל סנסור מורכב מארבעה דברים: microlens → color filter → photosite → potential well: זה בעצם ה

pipeline



כשmagן אור נוכל להעביר אותו דרך הסנסור ולהמיר לאלקטרונים ואז להעביר אותו להלה עם מעגלים חמליים כדי לדעת מה הערך של הפיקסל אבל תהlixir ההמרה טיפה מורכב ולכן הכמות אותה הצלחנו להמיר נקראות quantum efficiency. השם photosite מmir את הפוטונים.

אם יש תאורה גבוהה מדי או נמוכה מדי יהיה קשה לתפוס פוטונים גם כשחושך יש כמות קטנה של פוטונים אבל דעת אם זה רעש או באמת מידע שכן מספיק אור זה נקרא under-exposure (זה כיש יותר מדי פוטונים ואז אפשר לקלוט עוד כיש הרבה אור זה נקרא over-exposure (זה כיש יותר מדי פוטונים ואז אפשר לקלוט עוד לכל מצלמה יש כמות אלקטרונים שה

sensory

 יכול לשומר לפני ביצוע רויה (saturation))

בחזרה לתהlixir:

microlens מכניס את האור (אוסף את כל הקרניים. כמו שיתור אור בשבייל לקבל אינדיקציה על הנקודה). האור עובר דרך photosite ופולט את האלקטרונים.

לפעמים ישנו רכיב OPLF (optical lowpass filter) שנותן טשטוש קיטן כי יש לפעמים תדרים מאוד גבוהים שלא קיימים בעולם שמעוניים טיפה את התמונה (תדרים גבוהים יותר נתונים יותר פרטימ בתמונה ותדרים נמוכים נתונים את הדברים היותר משמעוניים בתמונה)
ברכיב ה"נ"יל ישנו חומר שמקבל את כל הקרניים אבל לא מעביר תדרים מאוד גבוהים כיש הרבה מאוד שינויים בczęיפות גבוהה ואין לנו מספיק סנסורים בשבייל לקלוט את כל השינויים אז מקבלים עיוותים (aliasing).

אנו תמיד נצטרך מספיק סנסורים כדי לצלם שטח מספיק גדול.
ישנים 2 סוגים של חיישנים המmirים מפוטונים לאלקטרונים: CCD, CMOS.
החיישן CCD הוא יותר רגיש אבל CMOS הוא יותר זול.
בשניהם יכול להיות over-exposure אבל CCD יכול להיות גם מריחות בתמונה.
כל סנסור ממופה לפיקסל.

בעדשה של הסנзорים ישנים סנסורים המוחכבים בקצבות מסוימות במקום שהאור לא מגיע לשם כלל הסיבה לקיום סנסורים אלו (נקראים Black Level) היא שהSENSORS הם רכיב אלקטרוני שיכול לפלוט אלקטرونים גם ללא קבלת אור ולכן נרצה לנמל את כל הסנסורים בעזרתם ע"י החסירה של כמות האלקטרונים הנפלטים מהlevel Black (ambilי לקבל או).

לאחר קבלת האלקטרונים צריך להעביר אותם עיבוד. אם כמות האלקטרונים קטנה (quantum efficiency נמוך) יוכל להגבר את הכמות בעזרת התהיליך הבא
analog amplifier → analog to digital converter (ADC) → look up table (LUT)
כאשר analog amplifier הוא הגדלת הכמות. analog to digital זה המרת מאנלוגי (אלקטرونים) לדיגיטלי (BITS).

בגלל תופעה פיזיקלית הפיקסלים שבמרכז מקבלים יותר אור מהפיקסלים בקצוות ולכן התמונה המתקבלת היא עם מסגרת שחורה מטושטשת (נראית כמו פילטר מעוצב) אבל העיבוד מתקן זאת.

שוב, איך זה עובד?
נפתח השטול שקוולט פוטונים. אח"כ נכנסים פוטונים (למערך של הסנзорים). אח"כ הפוטונים מומרים לאלקטרונים בעזרתphotosite והו well שמחזיק את כל האלקטרונים. אח"כ מתבצע reads out, כלומר, האלקטרונים מומרים לזרם אנלוגי. אח"כ מתבצעת צבירה של הזרם ואז הזרם האנלוגי מומר לדיגיטלי. לבסוף מתבצעים תיקוני רעש וכו'.

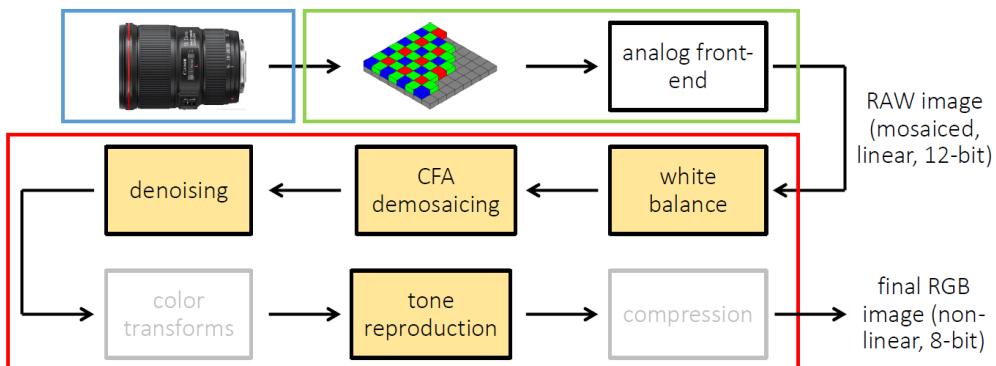
על `color filter` לא דיברנו. אנו קוראים "צבע" לתגובה של אורכי גל מסוימים אותו אנו מקבלים. כל מיני תאורות שונות יכולות להשפיע לאורכי גל שונים. תמונה מורכבת מכמה אורכי גל לכמות הצבע (תדרים, גוון כלשהו) שיש מכל אורכי גל קוראים (SPD) Spectral Power Distribution הSENSOR מגיב לכל אורכי גל (צבע) בצורה שונה, לפעמים פולט יותר אלקטרונים ולפעמים פחות התגובה היא (SSF) Spectral Sensitivity Function ניתן לעשות אינטגרל (צבירה) על המכפלה של SPD עם SSF והתוצאה היא התגובה

בעין האנושית מחלקים ל-3 חלקים את התאים שיש לנו במערכת הראייה (כביכול אלו הסנзорים שלנו) אנו בדר"כ מגיבים לאורכי גל ארוך ב-64% (אדום). ומגיבים לאורכי גל בינוני ב-32% (ירוק)

הfilter color מאפשר רק לאורכי גל מסוימים לעبور (לדוגמא, רק אדום) ישנו כל מיני סידורים של פיקסלים. אחד מהם הוא Bayer (נגוץ) והוא שכל פיקסל מכיל 2 ריבועים של י록 אחד של אדום ואחד של צהוב. הסתבה שיש 2 י록ים היא שהעין רגישה יותר לירוק וגם זה נותן יותר אינדיקציה לגבי האור. הרעיון של הסידור הוא לאסוף כמה שיותר מידע עם כמה שפחות סנסורים ישנו גם סידורים מיוחדים כמו להשים את color filter אחד על השני וכו' כשאין מספיק אור מקלים רעש/מריחות.

לאחר שעברנו את כל התהיליך אנו יודעים بصورة דיגיטלית את הערכים שקיבלו מכל סנסור לפי הסידור שהשתמשנו בו (לדוג' Bayer). ערכים אלו נקראים RAW. פעמים כשמצלמים במצב לילה וכדומה עובדים יישורות על RAW. כלומר, לפני תיקונים כלשהם.

כל התהליך של העיבוד נקרא (ISP) image signal processor



* ההמרה RAW ל-JPG וכו' לפעמים נקראת rendering.

תהליך העיבוד: (לפי הסדר כמו בתמונה לעיל)

1. White Balance - אמן המוח יודע לסווג צבעים בתמונה למורות שזה לא באמת הצבע

בתמונה. لكن אנו רוצים להראות את התמונה בצורה האידיאלית.
מצבעים זאת ע"י הצבע הלבן כי הci קל לומר איך הוא אמר לו להיות.

ניתן לתקן ע"י Presets מוכנים מראש בהתאם לתנאים. ואפשר בצורה אוטומטית ע"י כר שמכירחים את הממוחצע להיות אפור או להזכיר את הצבע הci בהיר להיות לבן או לבצע היסטוגרמה.

לגביו שתי השיטות הראשונות מניחים שהירוק נותן את התואrho ומשאירים אותו שהוא. שיטת האפור: את האדום/כחול מחולקים בממוצע של עצמו ומכפילים בממוצע של הירוק

שיטת הלבן: את האדום/כחול מחולקים במקס' של פיקסל בהינתן הפסיפס (כל סנסור בודק משאו ספציפי ולא שלשה) וכן ציריך אינטראפולציה.

2. CFA Demosaicing - בשבייל לדעת את הRGB של כל פיקסל בהינתן הפסיפס (כל סנסור בודק bilinear interpolation: ממוצע של 4 שכנים (של כל צבע בנפרד)

noise shot: פוטונים שבאים לא בצורה רציפה אבל כביר כל בקבוקות בגודלים שונים בגל
שאנו תופסם אותם ברכיב קטן (אם יש מספיק אוור נמנע מבוע זוז)

dark shot noise: תמיד יש פליטה של פוטונים גם כשותך (גרוע יותר כשההנטסור חם)

read noise: רעש הנובע מהמירה מאנלוגי לדיגיטלי

ביצוע של denoise קלאלי הוא באמצעות פילטר mean או median תלוי בריש. salt and peper Mean

3. Denoising - רעש (כשאין מספיק אוור) הנוצר מכמה סיבות:

noise shot noise: פוטונים שבאים לא בצורה רציפה אבל כביר כל בקבוקות בגודלים שונים בגל

קצרים (כהיים). העין רגישה לטון יותר כהה ומחכינים יותר בשינויים קטנים. כפי שהזכרנו בתחילת הקורס שאם נקודד תמונה עם פחות צבעים נקודד באזורי היותר כהים.

אם יש ערפל או יותר בהירות או פחות בהירות משתמש בחוסnton חום gamma correction (ביחסו ל-RAW).

4. Color transforms - (לא דיבר על זה)

Tone Reproduction - כפי דיברנו על זה בחושTON gamma correction העין רגישה יותר לטווויים קצרים (כהיים). העין רגישה לטון יותר כהה ומחכינים יותר בשינויים קטנים. כפי שהזכרנו בתחילת הקורס שאם נקודד תמונה עם פחות צבעים נקודד באזורי היותר כהים.

5. Gamma correction - (לא דיבר על זה)

לפעמים ביצוע של דברים פיזקיים נדרש לקבל את RAW שהוא הci קרוב לסנסור בשבייל לקבל כמה שיותר מידע למורות שהעיבוד פולט תוצאה שנראית טוב יותר לעין
וגם בעיצוב יותר קל לעבד עם RAW
הчисרון RAW הוא קבצים יותר גדולים
אם יש לנו תמונה שאינה RAW קשה להחזיר אותה ל-RAW. רוב המכצלמות היום מאפשרות ליצא RAW ובפיתוח יש ספרייה שבעזרתה ניתן להציג קובץ RAW ולעשות עליו פעולות.