

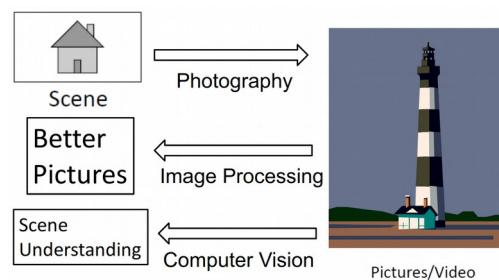
סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

ראייה ממוחשבת ועיבוד תמונה - הרצאה 1

הבדלים בין עיבוד תמונה לראייה ממוחשבת

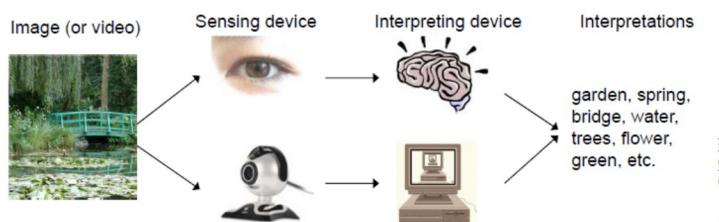


מטרתה של עיבוד תמונה בהינתן תמונה לשפר אותה.
 מטרתה של ראייה ממוחשבת היא להבין מה יש בתמונה: איזה אובייקטים יש בתמונה, איזה פעולות הם עושים, וכדומה.

"ישומי עיבוד תמונה:

- הוצאת רעש מהתמונה – denoising.
- ערבול – blending.
- חיבור תמונות לתמונה פנורמית.
- צביעת התמונה מחדש – image inpainting.

מה הכוונה בראייה ? זה לא פשוט ?
 רוב האנשים רואים .
 למעשה, כ 60% מהחומר האנושי מוקדשים עליון תהליכי הראייה .



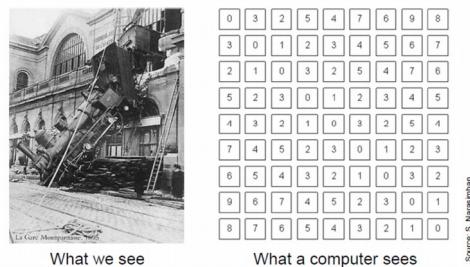
אנחנו קולטים דרך העין את הסביבה המוקה שלנו מעבד את התמונה ומפענח את מה שיש בה .
 בראייה ממוחשבת אנחנו רוצים לפענן את התמונה רק הפעם דרך המחשב .

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

אופן ייצוג התמונה במחשב



התמונות שאחנו רואים בעין, עברו המחשב הם בסך הכל מטריצה של מספרים. המטרה שלנו היא מתוך המספרים האלה להבין מהי התמונה.

ישומים בראייה ממוחשבת

- שחזור תלת מימד: גישה מבוססת גיאומטריה. נקרא גיאומטריה רבת מבטאים. מקבלים תמונה בדו מימד והמטרה היא ליצר מודל תלת מימדי. כלומר עברו כל פיקסל להגדיל מה העומק שלו בעולם האמיתי.
- סגמנטציה: הבנה של פריסת האובייקטים בתמונה. להפריד בין האובייקטים לבין הרקע שלהם. קיבוץ אחורים דומים.
- _recognition: אילו אובייקטים יש בתמונה.
- Class based: only general classes ○
Instance-based: a specific item ○



למה זה בכלל קשה ?



שינוי תוארה, בהירות



אותו אובייקט בכל מני זווית שונות – קשה לזהות שבדוחר באותו אובייקט.



scale

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

model based

תכנון מושג מודל, בהינתן בעיה יוצרים מודל חישובי המתאר את הבעיה בצורה מתמטית. יוצרים אלגוריתם הפותר את הבעיה המתמטית זו. הפלט של האלגוריתם יהיה למעשה פתרון הבעיה.

לדוגמה:

המטרה: למצוא קו שחור בודד בתמונה לבנה.

יוצרים מודל חישובי: $y = Ax + b$

אלגוריתם לישום המודל: חפש בכל הפיקסלים השחורים האפשריים לחבר אותם ובודק אם קיבלה את המשוואה שמלמעלה. פלט: פלוט את מיקום הקוו.

Deep learning based

בהינתן בעיה, אוספים כמות גדולה של תמונות עם בעיה דומה. מאמנים אלגוריתם (מודל) במטרה למצוא את הפרמטרים האופטימליים לשם פתרון הבעיה.

לדוגמה:

המטרה: מציאת קו שחור בתמונה לבנה.

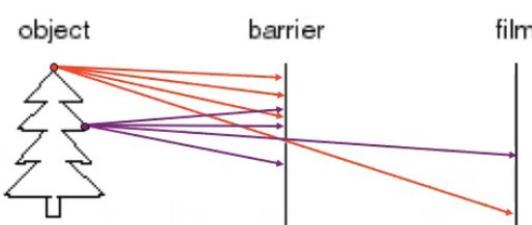
איסוף מאגר תמונות גדול של תמונות כאלה עם קווים שחורים במיקומים שונים.

אימון של אלגוריתם מוכר על מנת למצוא את הפרמטרים האופטימליים לפתרון המשימה. השתמש באlgorigthm + פרמטרים בהנתן תמונה חדשה על מנת למצוא את הקוו השחור בה.

From image processing to computer vision

- רמה פיזית: הסצנה במציאות (תאורה השתקפות).
- רמה פיזית: תנאי המצלמה (אופטיקה, חיישנים).
- עיבוד תמונה: קידוד (דחיסה), שיפור (שינוי צבעים, הוצאה רעש).
- עיבוד תמונה + ראייה ממוחשבת: זיהוי תכונות (אובייקטים, תנועות, פעולות).
- ראייה ממוחשבת: שחזור הסצנה במציאות (תלת ממד)
- ראייה ממוחשבת: סיוג, זיהוי אובייקטים.

Physical image model



מודל המצלמה מאפשר לנו למדل איך תמונה מתקבלת מהעולם. כאמור, התמונות מיוצגות במחשב על ידי מטריצה דו-ממדית של פיקסלים. כל תא במטריצה מכיל מספר המתאר את הגוון של הפיקסל. ככלומר, תמונה היא מיפוי דו-ממדי של עוצמת הצבע (intensity) בהינתן המיקום של סצנה בעולם האמיתי התלת-ממדי שלנו. בעולם האמיתי יש לנו איזשהו אובייקט כאשר חזורים קרני או מהאובייקט הם נקלטים בפイルם. הבעיה היא שם אנחנו נחשוף את הפילים ישירות אל העולם

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

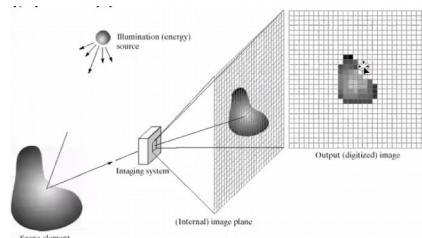
נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

הרבה מאוד קרני או יגעו לפילים. אנחנו מעוניינים שבכל נקודה באובייקט הגיעו ארכוך ורך נקודה אחת. מה שאנחנו עושים זה למשה לשום מיחסם כל שמכל נקודה באובייקט תכנס קרן או אחת. הפתוח במיחסם זהה נקרא (צמצם) aperture. התמונה המתקבלת היא תמונה פיזיקלית היא נובעת כתוצאה מכך שהאור פוגג בפילם. אנחנו יכולים לא לעבודים עם פילם אלא עם צילום דיגיטלי ולכן גם הקורס יתמקד בו.

Image formation: digital model

במקום שהאור יתקבל על פילם הוא מתקיים על ידי סנסורים. הסנסורים הם מערכת דו-מימדי שיכלجازור בו מכיל יחידת חומרה אחרת (שנקראת סנסור) שהיא יודעת לcompute כמה פוטונים מתקיים אליה בטווח זמן מסוים. כאשר אנו מצלמים מה שבעצם קורה זה שהמיחס נפתח, נכנס אור כל אחד מהSENSORS קולט את האור ושומר מה עוצמת האור שנכנשה. כל סנסור יכול לקבל ערך אחד בלבד, لكن, ככל שיש יותר סנסורים התמונה תהיה יותר אינטואיטיבית.



תמונה

- מערכת מספרים דו מימדי (מטריצה).
- פונקציה $I(x, y)$, נסמן ב: (y, x) .
- כל מיקום (y, x) נקרא פיקסל.
- הפונקציה מחשבת את עצמת הגוון במיקום (y, x) .
- הערכים האפשריים בכל מיקום יכולים לתאר סוגים שונים של תמונות:
 - שחור לבן (סיבית 1)
 - Gray-scale (8bit)
 - צבע (8bit*3)

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

2D real to 2D discrete: Sampling = how many (x, y) Resolution)

בראיה ממוחשבת נעבד עם תמונות דיגיטליות, עם ערכים בדידים. וכך נדרש לבצע שני דברים:

- לדגום (sample) את המרחב למטריצה דו-מימית עם ערכים בדידים.
- לכמת (quantization) כל דוגמה לעגל הקרוב ביותר, כי יש מספר סופי של גוונים.

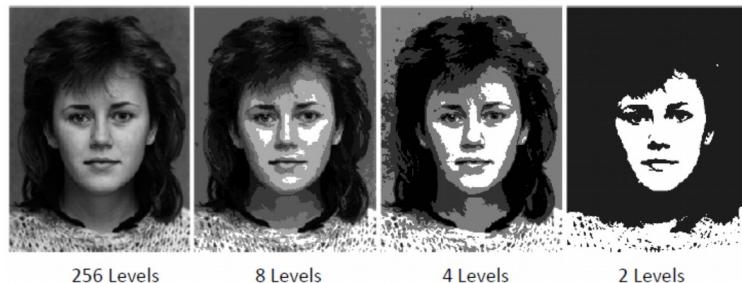
על מנת ליצג תמונה במחשב אנחנו חיבים להשתמש במספר סופי של פיקסלים. הדוגימה גורמת לנו לאבד אינפורמציה, ככל שאנו מוציאים פחות כנימות מאשר מאבדים מידע. כמוות הפיקסלים שאנו בוחרים בוחרם רוחולציה-למעשה מהי גודל המטריצה.



כימות Quantization

מה הם הערכים האפשריים של (x, y) ?

כמה ערכים אפשריים יש לי בכל כניסה (פיקסל), ככל שיש יותר ערכים אפשריים קיבל תמונה יותר עשירה.



Color images

- כל צבע הוא קומבינציה של 3 צבעים.
 - אדום, ירוק וכחול.
- אנחנו יכולים למשוך זאת על ידי מערך של תלת מידי של סנסורים.

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות ומיצגות של ד"ר גיל בן ארצי ומר שי אהרון

Intensity Transformations

Intensity transformation נלמד להפעיל טרנספורמציות על התמונה על מנת לשפר אותה.
 שינוי טווח עצמת הצבע בתמונה.

Intensity Transformation Function

אנחנו מקבלים מטריצה לכל כניסה במטריצה יש ערך. אנחנו יכולים להפעיל פונקציה שלוקחת ערך, ולהמיר אותו בעורף פונקציה לערך אחר.

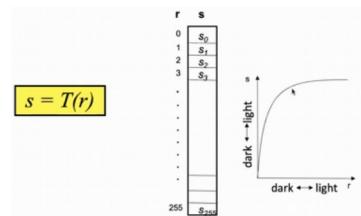
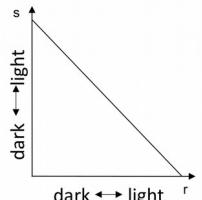


Image negative

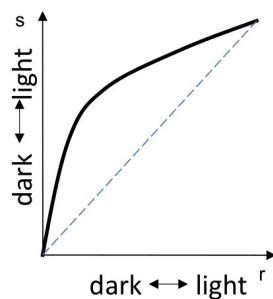
$$s = 1 - r$$



log transform

ניתן לנקוט גם פונקציות יותר מסוכבות, לדוגמה פונקציית הלוג. הפונקציה הזאת לוחקת חלק קטן של ערכים בחלק הכהה יותר בתמונה וממפה אותם להרבה יותר ערכים בתמונה היעד.

$$s = c * \log(1 + r)$$



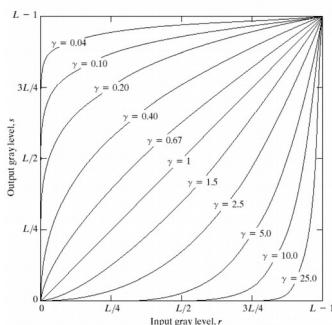
סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומיר שיאהרוני

Power Law Transform

ניתן להציג את הטרנספורמציות על ידי טרנספורמצית גמא.

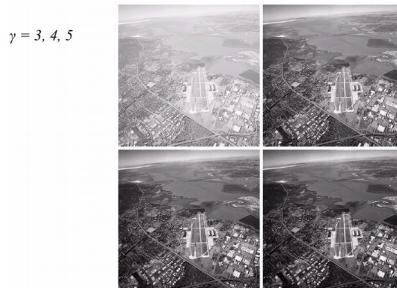


$$S = C \cdot r^\gamma$$

ניתן להסיק כי נקבל תמונה כהה יותר אם גמה < 1 , ובירה יותר אם גמה > 1 .

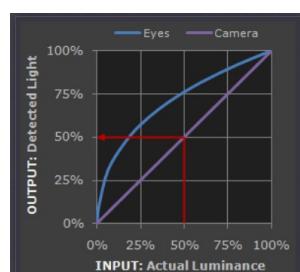
Gamma correction

שינוי ערכי גמה נקרא תיקון גמה. באמצעות תיקון גמה אנו יכולים לשלוט על בהירות התמונה.
 דוגמה:



בתמונה ניתן לראות שככל שהגמה גדולה יותר ההתקינה נעשתה יותר כהה.

העיניים שלנו אינם תופשют אוור כמו בצילמות. הסנסורים בצלמות קולטים אוור בצורה לא נראית. העין שלנו תופסת בהירות בצורה מאוד מיטימית. לצורך היישרות, הרובה יותר שימושי לראות טוב בלילה מאשר במהלך היום, ולכן אנחנו רואים הרבה יותר גוונים מהים מאשר גוונים בהירים. בהשוואה לצלמה, אנחנו רגשים הרבה יותר לשינויים בגוונים כהים מאשר לשינויים דומים בגוונים בהירים. אבל איך כל זה קשור לגמה? במקרה זה, גמה היא מה שמתרגם בין רגשות האור של העין שלנו לבין של המצלמה. כאשר נשמרת תמונה דיגיטלית היא "מקודדת גמה".



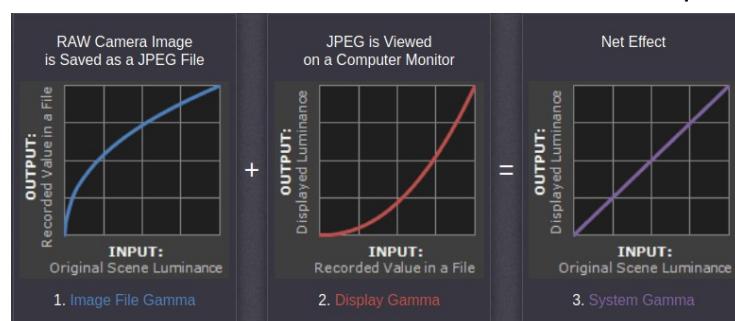
סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

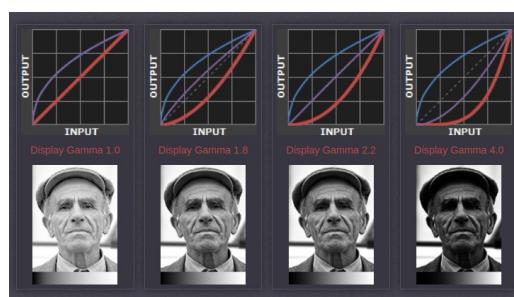
מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

תמונות מקודדות גמה מאחסנות גוונים בצורה עילית יותר. מכיוון שקידוד גמה קרוב יותר לאופן שבו אנו רואים את התמונה, נדרשים פחות ביטים על מנת לתאר טווח גוון נתון (גם ככה אנחנו לא מבחינים בשינויים בגוונים הבבירים). הגרדיאנט המקודד בדומה מפייצ את הצללים בערך באופן שווה על פני כל הטווח ("אחד מבחינה תפיסתית"). זה גם מבטיח כי עריכת תמונות, צבע והיסטוגרמות לאחר מכן מבוססים על גוונים טבאיים ואחדים בתפישה.

למרות כל היתרונות הללו, קידוד גמה מוסיף שכבה של מרכיבות לכל תהליך הצגת התמונות. על תמונה מקודדת גמה צריך לעשות "gamma correction" בעת הצפייה – על מנת להמיר אותה בחזרה למצב המקורי. במקרים מסוימים מטרת קידוד הגמה היא לשמר את התמונה – ולא להציג התמונה. לרובו הזמן השלב השני "display gamma" מבוצע באופן אוטומטי על ידי הציגים שלנו וכרטיסי המסך.



1. מתאר תמונה במרחב הצבע של sRGB (המקודד באמצעות גמה של כ-2.2).
2. מתאר גמה לתצוגה השווה לסטנדרט של 2.2.



מימוש

```
# apply gamma power-low transform
gammaCorrectedImg = np.array((img / 255) ** gamma, dtype='float64')
```

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

ראייה ממוחשבת ועיבוד תמונה – תרגול 1

מה זה opeCV ?

- ספריית קוד פתוח הכללת פונקציות שמכונות בעיקר לראייה ממוחשבת ועיבוד תמונה.
- מקורו במחקר של אינטל בשנת 1999.
- גרסה 1.0 יצאă בשנת 2006. כיום הגרסה היציבה الأخيرة היא 4.2.0.
- תומך במספר שפות כגון C / C++, Python, Java, Matlab ועוד.

בקורס נכתנת בפייתון.

דרישות לקורס: python3.7 ומעלה, numpy1.16.4 ומעלה.

openCV

```
import cv2 as cv
# read image:
img = cv.imread("lena.jpg")
# print image shape(height, width, channels)
print(img.shape)
# display the image:
cv.imshow("image", img)
# save the image
# cv.imwrite("lena_copy.jpg", img)
# If we want to see each channel by itself
# we can simply display only one channel
cv.imshow("disp", img[:, :, 0])
# show the image, and waits until any key pressed
cv.waitKey(0)
```

(512, 512, 3)

Matplotlib

```
import matplotlib.pyplot as plt
import cv2 as cv
import numpy as np
# read image:
img = cv.imread("lena.jpg")
# display the image:
plt.imshow(img)
plt.show()
# openCV use BGR format and matplotlib use RGB
# To get the colors right:
img_rgb = cv.cvtColor(img, cv.COLOR_BGR2RGB)
plt.imshow(img_rgb)
plt.show()
# the 'plot' function
plt.plot(55, 22, '*') # for a point
plt.imshow(img_rgb)
```

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

כתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

```
plt.show()  
x = np.random.randint(0, 500, (10, 2))  
plt.plot(x[:, 0], x[:, 1], '*') # multi-points  
plt.imshow(img_rgb)  
plt.show()  
# plt.plot(x[:, 0], x[:, 1], '-*') # use '-' for lines
```

אוף יציג התמונה

ניתן לייצג תמונות כ- $[0, 255]$ Uint8 או $[0, 1]$ float .double

שgiaה נפוצה היא שיש תמונה עם ערכים בין 0 ל- 255, אך סוג המטריצה (dtype numpy) הוא float, ולכן כשלומרים או מציגים את התמונה, התוצאה היא תמונה לבנה.

מרחבי צבע

יש כמה דרכים לייצג תמונות, הידוע ביותר הוא RGB.
כל ערוץ מכיל את רמות העוצמה של צבע אחד (אדום, ירוק או כחול).
קיים מרחבי צבע נוספים, לכל אחד מהם שימוש אחר.



YIQ

ויאו מרחב הצבעים המשמש את מערכת הטלוויזיה הצבעונית NTSC, אשר בשימוש בעיקר בצפון ומרכז אמריקה, וביפן.
הערוץ הראשון (Y) הוא ערוץ העוצמה (טולם הזרה)(luminance scale).
שלא כמו ה- RGB, שם הבירות מפוצלת בין שלושת הערוצים.
.The other two contain the chrominance information

: Orange <-> Blue
Q: Purple <-> Green

ה- I וה- Q הם אורתוגונליים ובכך משתרעים על הצבעים האפשריים. מערכת הראייה האנושית רגישה הרבה יותר לשינויים בציר ה- I מאשר בציר ה- Q,
ומאפשרת להעביר את ציר ה- Q בפחות נאמנות, וכך לשמור על רוחב הפס.

ויאו שימש לשידור בטלוויזיות ישנות בשחור לבן וטלוויזיות צבעוניות חדשות בו זמינות, היסנות פשוט השתמשו בערוץ Y, בעוד שהחדשנות שילבו את השניים האחרים כדי לקבל תצוגת צבע.

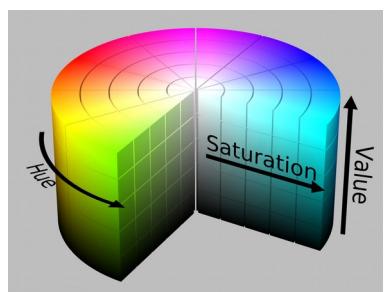
HSV

מרחב צבע נוסף הוא HSV, הפרמטרים שבו הם:

Hue – Color, Range $[0, 360)$ – גוון

Saturation – Strength of the color, $[0, 1]$ – רזונה

Value – Brightness, $[0, 1]$ – ערך בהירות



HSV מאוד שימושי כשאנו צריכים לקבל פיקסלים בטווח צבע

מסויים, שלא כמו RGB או YIQ ואחרים, בהם הצבע נמצא

בספקטרום רציף.

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

ראייה ממוחשבת עיבוד תמונה – הרצאה 2

Histogram Equalization

המטרה שלנו:

This is what we have



This is what we want

אנחנו נרצה להעלות את הניגודיות (contrast) של התמונה.

Contrast

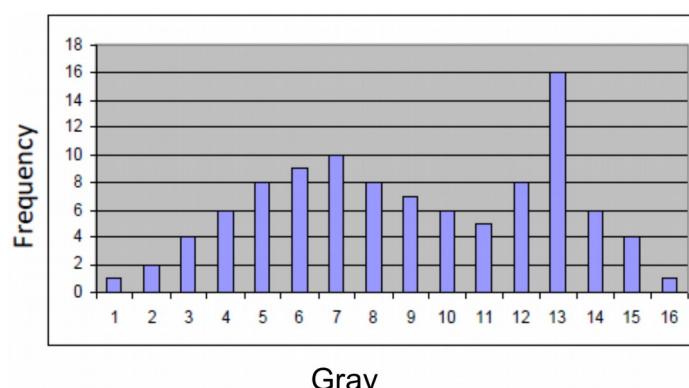
- הכוונה בניגודיות בתמונה היא:
- היחס בין האזורי הכהים בהיר להכינים.
- ההבדל בין הפיקסל המינימלי למקסימלי בתמונה.
- זה קשור להיסטוגרמה של תמונה.

Image histogram

היסטוגרמה היא מאפיין של תמונה, ממנו ניתן ללמוד על איזותה. היא מוגדרת כפונקציה שכיחות של רמות האפור, מבלתי להתייחס למיקום הפיקסלים בהם מופיעות רמות אפור אלו.

לדוגמה: נניח ובתמונה מסוימת ערך הפיקסלים נעים בין 1 ל 16.

היסטוגרמה עונה על השאלה: כמה פיקסלים בתמונה הם עם ערך 3/1/2/.....16. כלומר, היסטוגרמה עונה על השאלה כמה פיקסלים בכל דרגת אפור יש ל.



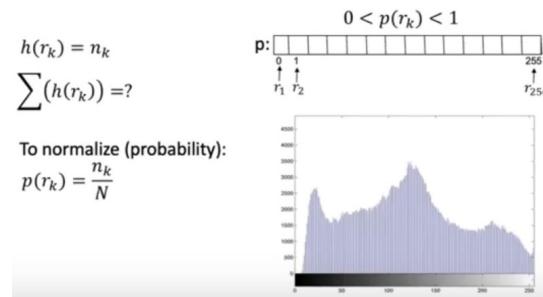
סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

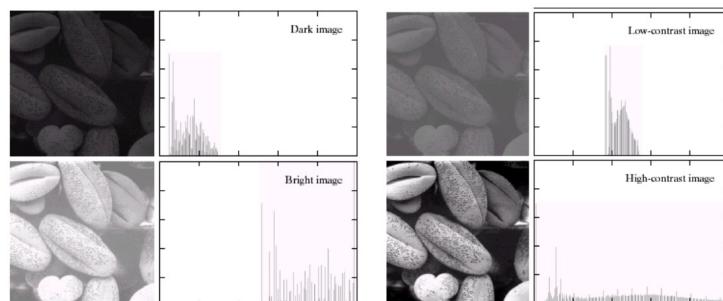
Histogram to probability

אם יש לנו היסטוגרמה, אז אנחנו יכולים להפוך את ההיסטוגרמה כך שתציג לנו את ההסתברות לקבל פיקסל מערך כלשהו בתמונה. ניתן לקבל את ההסתברות של הפיקסלים על ידי נרמול ההיסטוגרמה.



- ז – מיצג את דרגת האפור (a_k)
- $h(r_k) = a_k$ – כמה פיקסלים מאותו דרגת אפור.
- סכום כל r_k h שווה למספר הפיקסלים בתמונה.

Sample of image histograms



נשים לב שיש קשר בין איר שההיסטוגרמה נראית לבין הניגודיות בתמונה. בתמונה כהה ההיסטוגרמה תהיה מרוכזת מצד שמאל, בתמונה בהירה בימין, בתמונה בעלת ניגודיות גבוהה ההיסטוגרמה נפרשת על פני כל גווני האפור.

איך נגשים לביעית הניגודיות

- המטרה: שיפור הניגודיות בתמונה.
- איך אנחנו יכולים לעשות זאת?
- הבנת הפרמטרים של הבעיה.
- בניית מודל (נוסחה) המתאר את הבעיה על ידי שימוש בתכונות/mafynim/פרמטרים.
- הגדרת פתרון המבוסס על פורמלת המודל.
- בניית אלגוריתם כדי לפתור את נוסחה.
- הצעדים המפורטים מעלה הם גנרים עבור רוב הבעיות.
- לעיתיםначילה מבעה ספציפית נפתר אוותה, ולאחר מכן נכליל אותה.

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

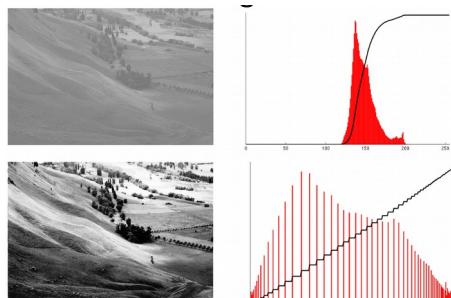
כתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

שלב ראשון: מציאת מודל המתאר את העולם שלנו.

Image parameterization: histogram
איך ההיסטוגרמה מתנהגת? מהי ההיסטוגרמה "טובה" ומה "רעה"?

היחסים בין ניגודיות להיסטוגרמה:



כאשר ההיסטוגרמה מתרכזת אך ורק בטווח קטן אז הניגודיות היא נמוכה.
כאשר ההיסטוגרמה נפרשת על פני גווני האפור, אז הניגודיות של התמונה יותר טוביה.

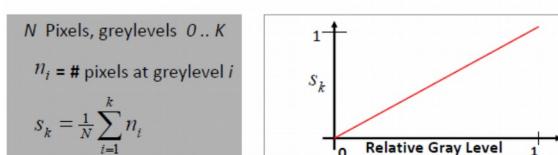
שלב שני: הגדרת הבעיה והפתרון הרצוי.

הבעיה: התפרשות לא אחידה של גווני אפור.
הפתרון: שימוש אחד בכל רמות האפור שיש.

שלב שלישי: יצירת אלגוריתם לפתרון הבעיה בשימוש המודל

תחילה נחשב את ההיסטוגרמה של התמונה (כמה פיקסלים מכל גוון אפור יש ל').
לאחר מכן נחשב את הרסתברות לכל גוון אפור בתמונה, על ידי נרמול ההיסטוגרמה:

Normalize to range 0..1



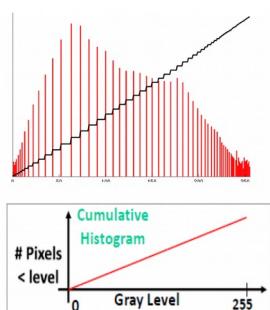
s_k היא פונקציה המחזירה מה הסיכוי לקבל ערך עד לערך פיקסל מסוים.

Good contrast = linear cumulative histogram

הReLU הוא כזה, אנחנו לא רק רוצים לדעת מה הסיכוי שלי לקבל ערך מסוים.
אלא מה הסיכוי לקבל ערך כלשהו שנמצא בטווח ערכים.
• נרמל את הערכים.

עבור ערך מסוים את הערכים הקטנים ממנו.

כלומר, חישוב עבור גוון אפור 2 יהיה סכימה של הרסתברות לקבל 2, 1, 0.



סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

- יעד: שיפור הניגודיות.

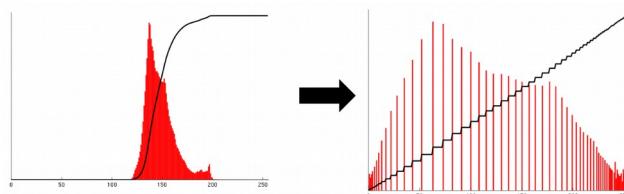
מודל: ניגודיות לפי היסטוגרם תמונות.

בעיה: פיזור לא אחידת של גווני אפור.

פתרון: נעשה שימוש שווה בכל רמת אפור.

כיצד: Uniform histogram \leftrightarrow linear cumulative histogram

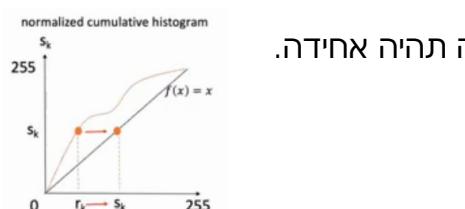
לכן בעזרת המודל שלנו, המטרה לשנות היא להפוך היסטוגרמה מצטברת לא טובת להיסטוגרמה מצטברת לינארית.



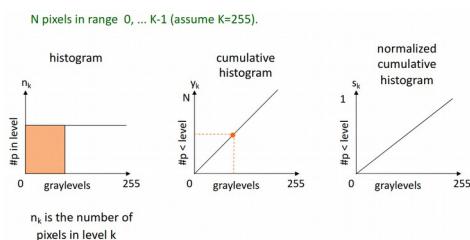
- אבל איך עושים זאת?

Proposed solution

- נרצה לעשות normalized cumulative intensity transform כך שההתוצאה תהיה כמו שיותר קרובה ל $y = f(x)$ (פונקציית הזוזות). זאת המטרה שלנו.
- نمפה כל גוון אפור r_k ל s_k .
- התוצאה תהיה שההיסטוגרמה של התמונה תהיה אחידה.



איך מייצרים cumulative histogram



- מקבלים היסטוגרמה, שאומרת לנו כמה פיקסלים יש לנו מכל גוון.
- מייצרים cumulative histogram על ידי כך שעבור כל ערך סוכמים את כל הערכים לפני כן.
- ונורמל בכמות הפיקסלים ומקבלים את ה CDF.

דוגמה:

h cumulative:

1	50	255							

$N = 10,000$

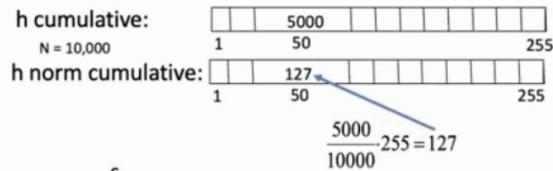
סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

אנחנו מקבלים קומולטיב היסטוגרמה ואנו רואים שמחצית מהפיקסלים נמצאים בטווח גוני האפור של 50-0. השאלה היא איפה הייתה רצחה לאחר התיקון שהי 5000 פיקסלם האלה היה ?
 5000 פיקסלם מייצגים לנו 50 אחוז מסך כל הפיקסלם, אנחנו רוצחים שהCDF יהיה לינארי ולכן נרצה ש 5000 הפיקסלם האלו יהיו עד 127, ואחריהם עוד 5000.

איך עושים זאת ?



Full Formulation

- היסטוגrama טובה:
- בכל דרגת אפור יש לנו אותה כמות פיקסלים
- על מנת לשמר על תוכן המידע של התמונה, יש לשמור על הסדר היחסי בין פיקסלים.
- כל הפיקסלים שהיו יותר מפיקסל עם ערך אפור R צריכים להישאר כהים יותר מפיקסל זה כאשר פיקסל זה מקבל ערך חדש S.
- זה אומר שלכל ערך R, מספר הפיקסלים עם ערכי האפור הנמוכים אמרות להיות זהה למספר הפיקסלים עם ערכים אמורים נוספים מ- S, כאשר S הוא הערך אליו מMOVE R.

Formulation by CDF

- תמונה בקנה מידת אפור ($g = 0 \dots 255$)
- עבור כל רמת פיקסל R יש לנו את ההיסטוגרמה המנורמלת:

$$cdf(R) = \sum_g^R p(g)$$

G-gray levels, R-previous pixel level, S-new pixel level •

◦ אז יש לנו את התנאים הבאים:

1. אנו הופכים רק R ל- S חדש (לא ידוע):

$$\sum_{t=0}^S p_{new}(t) = \sum_{g=g_{min}}^R p_{old}(g)$$

2. הסתברות החדשה צריכה להיות שווה:

$$p_{new}(t) = 1/G$$

$$g_{new} = \left\lceil G \sum_{g=g_{min}}^{g_{old}} p_{old}(g) - 1 \right\rceil \quad • \text{ הפתרון:}$$

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

- הערות:
 - g_{new} הוא לא ידוע, g_{old} הוא ידוע.
 - $g(g_{\text{old}})$ הוא ההיסטוגרמה של דרגת אפור g , לא ה

Histogram Equalization: Algorithm

1. חשב את ההיסטוגרמה של התמונה.
2. חשב את ה cumulative histogram .
3. נormal את ה cumulative histogram :
normalization.
- חלק במספר הפיקסלים.
- הכפל בערך ה gray level המקסימלי (1-K).
4. עגל את התוצאות על מנת לקבל מספר שלם.
5. מפה את ה intensity values של התמונה על ידי התוצאה של שלב 5.
6. ודא כי הערך המינימלי הוא 0 והמקסימלי הוא 1-K.

מימוש

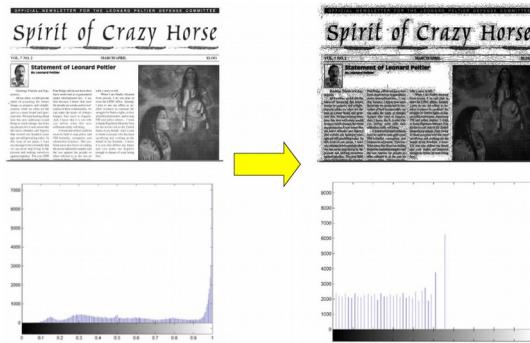
```
# find the histogram of the original image
histOrig, _ = np.histogram(imgOrigInt.flatten(), 256, range=(0, 255))
cumsumOrig = np.cumsum(histOrig) # calculate cumulative-sum of the original image
cdfNorm = (cumsumOrig * 255 / cumsumOrig[-1]).astype("uint8") # normalize cumulative histogram
imgEq = cdfNorm[imgOrigInt] # apply the transformation

# bounds checking:
imgEq[imgEq < 0] = 0
imgEq[imgEq > 255] = 255
```

מתי האלגוריתם לא עובד ?

- כאשר התפלגות רמות האפור הרצויה אינה אחידה (למשל טקסט).
- כאשר תוכן מקורות שונים מושווה יחד (למשל שני תמונות ממוקרות שונים).

אם נפעיל את האלגוריתם שלנו על התמונה מימין נקבל את התמונה משמאלו, שכאמור היא לא טובה זהה לא למה שציפינו.



סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

Adaptive Histogram Equalization

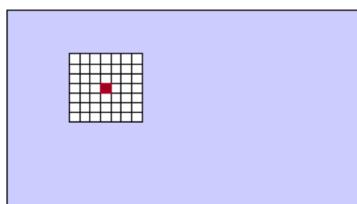
- נשותמש כאשר בתמונה ישנו התפלגויות שונות של רמת האפור.

- לדוגמה: אזורים שימושיים ואזורים מוצלים.

- Poor result for histogram equalization

- להפעיל את האלגוריתם על האזורים השונים בנפרד.

- פתרון: חשב את ההיסטוגרמה עבור אזורים מקומיים מסביב לכל פיקסל.



- עבור כל פיקסל

- חשב את ה-LUT equalization באזור מקומי.

- בצע טרנספורמציה על ידי LUT רק את הפיקסל המרכז.

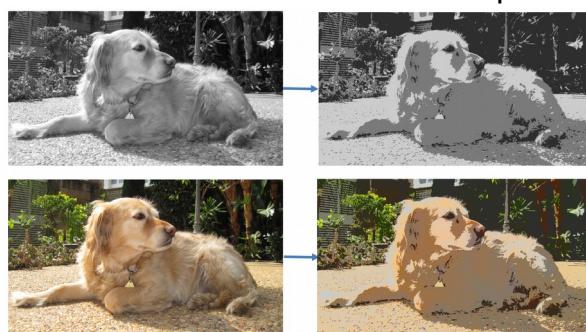
- עבור לפיקסל הבא.

Quantization

המרת ערכים אנלוגיים (אותות, תדרים) לערכים דיגיטליים (BITS) דורשת עיגול של הערכים האנלוגיים הרציפים לערכים בדידים. השיטה היא לדגם מספר אותות, לצרף אותם לעיגול אותם לערך אחד הנקרא קוונטיזציה. תהליך זה נקרא קוונטיזציה. לדוגמה, קוונטיזציה מתיחסת למספר דרגות האפור שיש לי, כ Schnitzel להקטין אותם נעשה קוונטיזציה.

Quantization procedure

דחיסת טווח ערכים לערך יחיד



ככל שמרידים את טווח הערכים האפשרי לייצוג פיקסל אנחנו מגבלים את יכולת שלנו לשינוי צבעים. ככל מרידים נקבל תמונה פחות אינטואיטיבית.

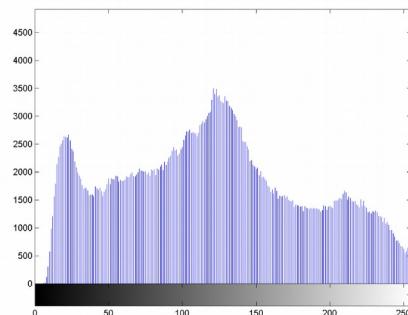
למה שבכל נרצה לעשות זאת? לעיתים יש לנו חומרה מוגבלת ואני רצימ לדחוס על מנת להקטין את נפח התמונה.

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומර שי אהרון

שאלה: איך נבחר את הקוונטיזציה האופטימלית?



Given: K - the number of values we want
Goal: Find the boundaries (z) and the values themselves (q)

הינו רצים של אחר המיפוי שנעשה, ההבדלים בין לפני המיפוי יהיה מינימליים.

- פונקציית השגיאה שלנו תהיה MSE weighted

$$\sum_{i=1}^n w_i (x_i - \bar{x})^2$$

x_{bar} : זה הערך הנבחר.

x : רמת האפור.

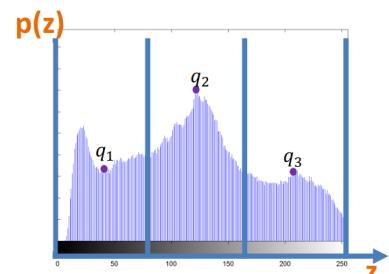
w_i : ההסתברות של x_i .

Quantitation – solution

Goal: quantize the image to 3 intensities

$$\min \sum_{i=0}^k \int_{z_i}^{z_{i+1}} (q_i - z)^2 p(z) dz$$

$$q_i = \frac{\int_{z_i}^{z_{i+1}} z \cdot p(z) dz}{\int_{z_i}^{z_{i+1}} p(z) dz} \quad z_i = \frac{q_i + q_{i+1}}{2}$$



Given: K - the number of values we want

Goal: Find the boundaries (z) and the values themselves (q)

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

ראייה ממוחשבת ועיבוד תמונה – הרצאה 3

פעולות על תמונות

- פעולות נקודה: פעולות על ערך פיקסל מסוים, אין תלות בסביבת הפיקסל, לא צריך לזכור נתונים. (שינוי היסטוגרמות, השיעורים הקודמים)
- פעולות מרחביות: פעולות התלוויות בערך הפיקסל ובקוואורדינטות שלו. פעולות אלה הן תלויות בסביבה של הפיקסל.
- פעולות גאומטריות: פעולות על הקואורדינטה וערךו של הפיקסל. (נתמקד בהן בשיעור זה).

Filtering

הפעלת אותה פונקציה על כל חלון אפשרי בגודל קבוע בתמונה.

120	124	119	116	113	112	114	116	117	114	121
129	110	106	107	108	110	114	110	121	109	112
124	116	109	108	106	110	113	118	120	115	119
109	111	114	108	106	109	113	116	119	120	120
105	114	112	110	109	111	114	116	124	124	124
114	116	114	111	109	109	110	112	114	127	125
124	119	116	112	109	108	109	111	112	126	123
127	120	118	113	110	108	108	109	110	123	120
125	121	118	114	110	108	107	108	110	120	117
129	115	107	99	98	102	105	103	98	117	118
125	127	119	110	108	112	114	110	105	111	114
119	132	122	113	110	112	113	109	103	102	107
113	125	116	106	102	104	105	100	94	96	102

Filtering – sliding window

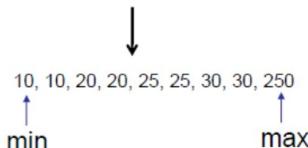
- עבור כל פיקסל בתמונה:
 - קח חלון בגודל KxK.
 - חשב את הפונקציה עבור הפיקסלים שבחלון.
 - שנה את ערך הפיקסל שבמרכז החלון, לפולט הפונקציה.
- הפעולה הזאת נקראת פילטר.
- העפולה הזאת נקראת sliding window מאחר ואנו מפעילים את הפונקציה עבור כל הפיקסלים בתמונה.

The Min/Max filters

עבור כל חלון לוקחים את הערך המינימלי/מקסימלי.

30	10	20
10	25	250
20	25	30

$$\text{Min filter: } f'(x, y) = \min(\{f(m, n)\})_{(m,n) \in N(x,y)}$$



$$\text{Max filter } f'(x, y) = \max(\{f(m, n)\})_{(m,n) \in N(x,y)}$$

סיכון קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

Noise model



Original Image

$$f_n(x, y) = \begin{cases} f(x, y) & \text{with probability } p \\ 255 & \text{with probability } (1-p)/2 \\ 0 & \text{with probability } (1-p)/2 \end{cases}$$



Salt & Pepper Noise



The median filter

מ민ים את עוצמת הפיקסלים בחלון הנוכחי ולוקחים את הערך האמצעי.
 (נשים לב כי החזיות אינם מושפע מערכיהם חריגים (ולכן הוא שימושי במקרים אלו))

$$f'(x, y) = med(\{f(m, n)\})_{(m, n) \in N(x, y)}$$

10, 10, 20, 20, 25, 25, 30, 30, 250
 ↑
 median

30	10	20
10	25	250
20	25	30

החזיות מזער את סכום ההפרשנים בערך מוחלט:

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

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות ומיצגות של ד"ר גיל בן ארצי ומר שי אהרון

תוצאות של פילטר חציו בגודל 3x3



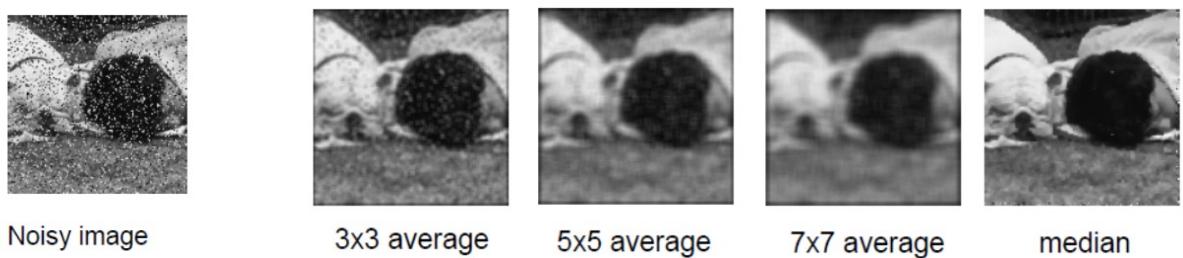
The Average filter

סוכמים את כל הגוונים בחלון ומחלקים במספר הפיקסלים בחלון, ממוצע.

$$f'(x, y) = \text{mean}(\{f(m, n)\})_{(m, n) \in N(x, y)} = \frac{1}{|N|} \sum_{(m, n) \in N(x, y)} I(m, n)$$

הממוצע מזעיר את סכום ההפרשיות בריבוע.

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



Noisy image

3x3 average

5x5 average

7x7 average

median

קונבולוציה

קונבולוציה זו פעולה מתמטית בין שתי פונקציות או סדרות ערכים. מקובל לסמן קונבולוציה באמצעות הסימן: *

קונבולוציה הוא אופרטור לינארי: הוא יכול להיות מיוצג על ידי מטריצה. תכונות הקונבולוציה:

- אסוציאטיביות (חוק הקיבוץ): $a^*(b^*c) = (a^*b)^*c$
- קומוטטיביות (חוק החילוף): $a^*b = b^*a$
- דיסטריביטיביות (חוק הפילוג): $c^*(a+b) = c^*a + c^*b$

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

קונבולוציה 1D

מבצעת את הדברים הבאים:

- בהינתן וקטור.
- תהפוך אותו כר שהסוף יהיה ההתחלתה. (flip)
- עברו אליו על הוktor השני.
- הכפל את הערכים.

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

Signal/Image Mask, Filter, Kernel

מבחינה מתמטית לשם ההגדרה הגבולות הם אינסוף ומינוס אינסוף. בפועל נחשב את המספרים שמעבר לגבולות הוktor C-0.

שיטות לקביעת ערכים מחוץ לגבולות

- Option 1: Zero padding

$$\begin{matrix} 0 & 0 & 0 & \boxed{1} & 2 & 3 & 4 & 5 & 0 & 0 & 0 \\ & & & \downarrow & & & & & & & \\ & & & 4 & 10 & 16 & 22 & 22 \end{matrix}$$

- Option 2: Wrap around

$$\begin{matrix} 3 & 4 & 5 & \boxed{1} & 2 & 3 & 4 & 5 & 1 & 2 & 3 & 4 & 5 \\ & & & \downarrow & & & & & & & & \\ & & & 19 & 10 & 16 & 22 & 23 \end{matrix}$$

- Option 3: Reflection

$$\begin{matrix} 3 & 2 & 1 & \boxed{1} & 2 & 3 & 4 & 5 & 5 & 4 & 3 & 2 \\ & & & \downarrow & & & & & & & \\ & & & 7 & 10 & 16 & 22 & 27 \end{matrix}$$

IMPLEMENTATION

```
def conv1D(inSignal: np.ndarray, kernel1: np.ndarray) -> np.ndarray:  
    """  
    Convolve a 1-D array with a given kernel  
    :param inSignal: 1-D array  
    :param kernel1: 1-D array as a kernel  
    :return: The convolved array  
    """  
  
    signal_len, kernel_len = inSignal.size, kernel1.size  
    pad_signal = np.pad(inSignal, (kernel_len - 1)) # padding with zeroes  
    flip_kernel = np.flip(kernel1) # flip the kernel  
    # mode= "full"-size of output vector is: signal_len + kernel_len - 1  
    return np.array([np.dot(pad_signal[i:i + kernel_len], flip_kernel) for i in range(signal_len + kernel_len - 1)])
```

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

correlation

קורלציה היא אותו הדבר כמו קונבולוציה רק ללא הפיכת הוקטור. לכן במקרה שהוקטור הוא פלינדרום (סימטרי) הקורלציה והקונבולוציה זהות.

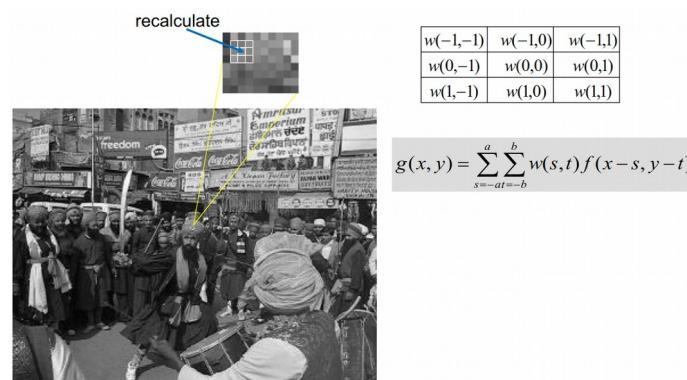
קונבולוציה 2D

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

↑ ↑
Image Mask, Filter, Kernel

פילטר מרחבי:

- רצים על כל פיקסל בתמונה:
 - בוחרים את האיזור כר שמרכזו הוא הפיקסל הנוכחי.
 - עושים באיזור מכפלה פנימית בין ערכי הפיקסלים באיזור לבין W.
 - מעדכנים את מרכז האיזור בהתאם של המכפלה הפנימית.



חשבות ה- \otimes בקונבולוציה

The diagram shows the element-wise multiplication (Hadamard product) of two matrices, $F(x, y)$ and $H(u, v)$, to produce the result matrix $G(x, y)$. The matrices are:

$F(x, y)$ (Input):

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

$H(u, v)$ (Kernel):

a	b	c
d	e	f
g	h	i

$G(x, y)$ (Output):

0	i	h	g					
0	f	e	d					
0	c	b	a					

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

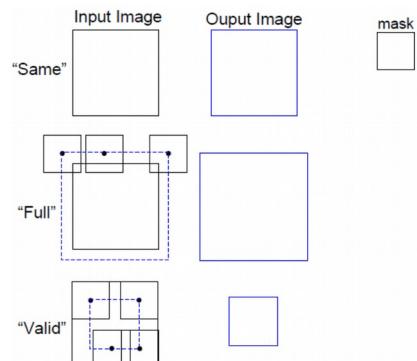
נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

Borders

- Option 1: "same" (size A)

$$\begin{matrix} 0 & 0 & 0 & \boxed{1} & 2 & 3 & 4 & 5 & 0 & 0 & 0 \\ 0 & 0 & 1 & \boxed{4} & 10 & 16 & 22 & 22 & 15 & 0 & 0 \end{matrix} * \begin{matrix} 1 & 2 & 3 \\ \hline 1 & 2 & 3 \end{matrix}$$



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

$$\begin{matrix} 0 & 0 & 0 & \boxed{1} & 2 & 3 & 4 & 5 & 0 & 0 & 0 \\ 0 & 0 & 1 & \boxed{4} & 10 & 16 & 22 & 22 & 15 & 0 & 0 \end{matrix}$$

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

$$\begin{matrix} 0 & 0 & 0 & \boxed{1} & 2 & 3 & 4 & 5 & 0 & 0 & 0 \\ 0 & 0 & 1 & \boxed{4} & 10 & 16 & 22 & 22 & 15 & 0 & 0 \end{matrix}$$

מימוש

```
def conv2D(inImage: np.ndarray, kernel2: np.ndarray) -> np.ndarray:
```

```
    """
    Convolve a 2-D array with a given kernel
    :param inImage: 2D image
    :param kernel2: A kernel
    :return: The convolved image
    """

    # get image and kernel shape
    image_height, image_width = inImage.shape[:2]
    kernel_height, kernel_width = kernel2.shape[:2]
    pad_image = image_padding(inImage, kernel_height, kernel_width) # pad the image
    return np.array([np.sum(pad_image[i:i + kernel_height, j:j + kernel_width] * kernel2)
                    for i in range(image_height)
                    for j in range(image_width)]).reshape((image_height, image_width))
```

החלקה Smoothing

נשים לב כי ממוצע גווני האמור לאחר ההחלקה נשאר אותו הדבר, כי סכום המשקלות הוא 1.

1	1	1
1	1	1
1	1	1

1/9 *

1	2	1
2	4	2
1	2	1

1/16 *

מחליף כל פיקסל בממוצע משוקל של הסביבה, שלו.

כאשר הערך המקורי הוא הדומיננטי ביותר, וישנה חישובות לקרבת הפיקסלים.

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון



משמאל התמונה המקורי, מימין התמונה לאחר החלקה עם קרנל גדול.

מימוש

```
def blurlImage1(in_image: np.ndarray, kernel_size: int) -> np.ndarray:  
    """  
    Blur an image using a Gaussian kernel  
    :param in_image: Input image  
    :param kernel_size: Kernel size  
    :return: The Blurred image  
    """  
  
    gaus_1d = get_gaus_1d(kernel_size)  
    return conv2D(in_image, np.dot(gaus_1d, np.transpose(gaus_1d)))  
  
def get_gaus_1d(kernel_size: int) -> np.ndarray:  
    """  
    my implementation to gaussian 1D kernel according opencv doc  
    :param kernel_size: size of the kernel  
    :return: gaussian 1D kernel  
    """  
  
    gaus_1d = np.array([[np.exp(-(np.square(x - (kernel_size - 1) / 2)) / (2 *  
        np.square(get_gaus_sigma(kernel_size))))  
        for x in range(kernel_size)]])  
    gaus_1d /= gaus_1d.sum()  
    return np.transpose(gaus_1d)  
  
def blurlImage2(in_image: np.ndarray, kernel_size: int) -> np.ndarray:  
    """  
    Blur an image using a Gaussian kernel using OpenCV built-in functions  
    :param in_image: Input image  
    :param kernel_size: Kernel size  
    :return: The Blurred image  
    """  
  
    return cv2.filter2D(in_image, -1, get_gaussian2D(kernel_size, get_gaus_sigma(kernel_size)))
```

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

```

def get_gaus_sigma(kernel_size: int):
    """
    :param kernel_size: size of kernel
    :return: value of sigma factor in gaussian filter
    """

    return 0.3 * ((kernel_size - 1) * 0.5 - 1) + 0.8
def get_gaussian2D(size: int, sigma: float) -> np.ndarray:
    """
    :param size: kernel size, It should be odd.
    :param sigma: gaussian standard deviation.
    :return: gaussian 2D filter in shape ksizexksize
    """

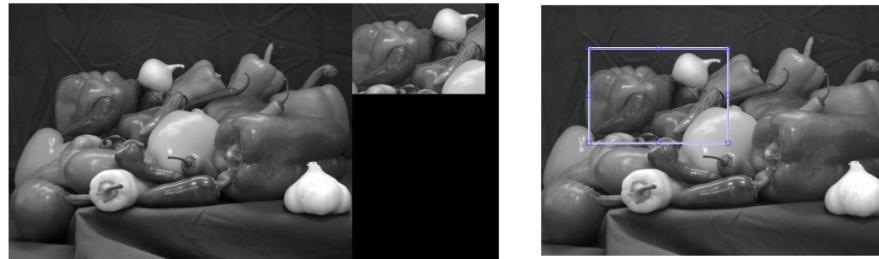
    gaus_kernel = cv2.getGaussianKernel(size, sigma)
    return np.dot(gaus_kernel, np.transpose(gaus_kernel))

```

ראייה ממוחשבת ועיבוד תמונה – הרצאה 4

Template matching

בהינתן תמונה/סיגナル ותבנית של חלק מהתמונה/סיגナル, נרצה למצוא את המיקום של התבנית בתמונה.



נציג את התבנית כמטריצה. האתגר שיכל להיות פה הוא שהතבנית עשויה להיות מעט שונה מהתמונה עקב: רעש בתמונה, תאורה שונה.

נרצה להשתמש ב:Normalized cross correlation

$$\cos(\theta) \stackrel{\text{def}}{=} \frac{\vec{\alpha} \cdot \vec{\beta}}{|\vec{\alpha}| \cdot |\vec{\beta}|}$$

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

נגידר את הווית בין שני הוקטורים כמכפלה פנימית של הוקטורים (אלפא ובטא) וחילוקה בנוינהם. נשים לב כי הווית אינה תלויות באורך הוקטורים (משום שהוקטורים מנורמלים). וכך שוניים בתאורה לא יהו פקטור.

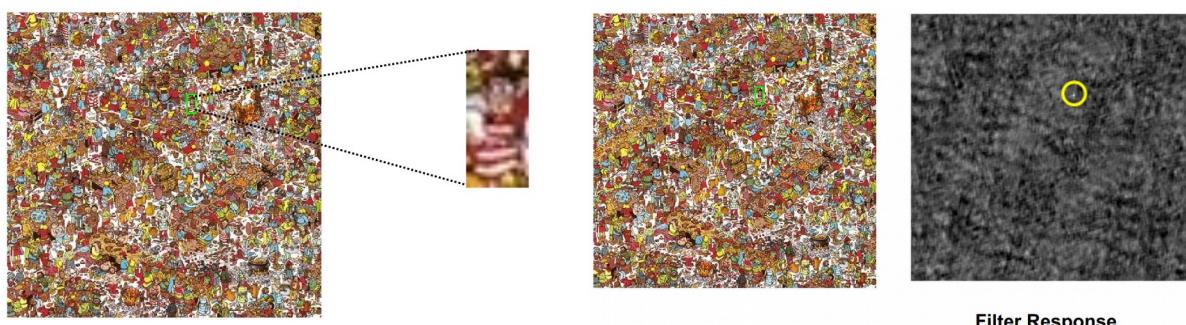
Algorithm NCC

קלט: מטריצה דו-ממדית של התמונה $A_{n \times n}$, ותבנית דו-ממדית $K_{k \times k}$.

פלט: מיקום התבנית בתמונה.

אלגוריתם:

- עבור כל חלון $K_{k \times k}$ אפשרי בתמונה הפעל CCC בין התבנית והחלון.
- המיקום בו התקבל הערך הגבוה ביותר הוא מיקום התבנית המשוער.
- החזר את מיקום התבנית המשוער.



Filter Response

Edge detection

אנחנו רוצים להמשיך לאין דברים חשובים בתמונות. דיברנו על זיהוי תבניות, וicut נדבר על edge detection, גילוי קצוות, קווי מתאר של אובייקטים בתמונה. בשונה מזיהוי תבניות שם אנו מקבלים התבנית של חלק מהתמונה ואצלינו למצוא את התבנית בתוך התמונה, כתעט לא נקלט מראש התבנית צזו. עדין, נוכל לחילץ מהתמונה את הקווי המתאר של האובייקטים שבה. ההנחה הבסיסית שלנו בזיהוי קווי המתאר של האובייקטים היא שני משמעותי בגווני האפור בין האובייקט עצמו לבין הרקע של האובייקט.



זה נשמע הגיוני, אבל בהינתן תמונה, איך נוכל למצוא את השינויים הללו? נמצא את השינוי בתמונה כפי שאנחנו מוצאים את השינוי בפונקציה – באמצעות נגזרת או במקרה שלנו באמצעות גרדיאנט.

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

שימוש לחישובי נגזרת בתמונה:

- לראות את מתאר האובייקטים.
- לראות את הגבולות והקצוות של האובייקטים.

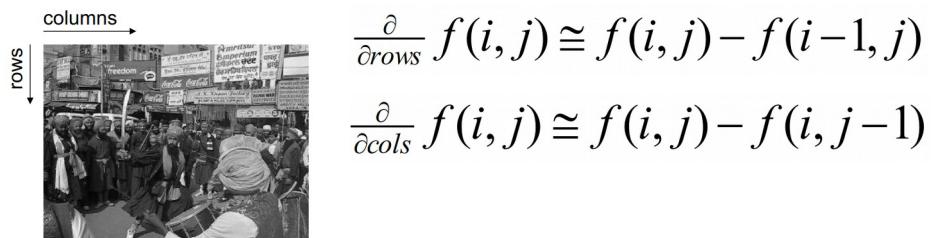
עבור פונקציה בעלת שני משתנים x, y הנגזרת החלקית מוגדרת כ:

$$\frac{\partial f(x, y)}{\partial x} = \lim_{\varepsilon \rightarrow 0} \frac{f(x + \varepsilon, y) - f(x, y)}{\varepsilon}$$

אבל בغالל שאנו מדברים על תמונות אנחנו נמצאים בעולם דיסקרטי, لكن נשתמש בקירוב:

$$\begin{aligned}\frac{\partial f(x, y)}{\partial x} &\approx \frac{f(x+1, y) - f(x, y)}{1} \\ &\approx f(x+1, y) - f(x, y)\end{aligned}$$

Edge detection using derivative approximation



נמצא את הנגזרת באמצעות קרナル שמותאם במיוחד כדי למצוא את הנגזרת:

$$\begin{pmatrix} 1 & -1 \end{pmatrix}$$

נגזרת לעמודות

$$\begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

נגזרת לשורות

סיכום קורס ראייה מוחשבת ועיבוד תמונה

<https://github.com/shaynaor> נכתב על ידי: שי נאור

mbos על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

נשים לב שמאחר והתמונה לא רציפה, חישוב של נגזרת באמצעות קונבולוציה הוא קירוב בלבד.



The gradient

גרדיאנט זהו וקטור המכיל את הנגזרות החלקיות בציר ה- x ובציר ה- y . וקטור הגראדיאנט הוא הcy.
בו יש את השינוי הכי גבוהה בערכיו הפונקציה (בעוצמת הגונים), והגודל שלו זה כמות השינוי. וזה
בדוק מה שהיינו צריכים לשם מציאת הקיצות.

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{pmatrix}$$

$$\nabla f = \left[\frac{\partial f}{\partial x}, 0 \right] \quad \nabla f = \left[0, \frac{\partial f}{\partial y} \right] \quad \nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$



$$|\nabla f| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

$$\alpha = \tan^{-1} \left(\left(\frac{\partial f}{\partial y} \right) / \left(\frac{\partial f}{\partial x} \right) \right)$$

$$\cos(\alpha) \frac{\partial f}{\partial x} + \sin(\alpha) \frac{\partial f}{\partial y}$$

Derivative gradient magnitude

אורן הגרדיאנט יכול לשמש כגלאי קצאות פשוט:

$$I_x = \star \begin{pmatrix} -1 & 0 & 1 \end{pmatrix} =$$



$$I_y = \text{[Image of a frog] * \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} = \text{[Image of a frog with horizontal edges highlighted]}}$$

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right) = \begin{array}{c} \text{Image 1} \\ \text{Image 2} \end{array} \quad \sqrt{|I_x|^2 + |I_y|^2} = \text{Image 3}$$

מימוש

```
def convDerivative(inImage: np.ndarray) -> (np.ndarray, np.ndarray, np.ndarray, np.ndarray):
```

```
    """
    Calculate gradient of an image
    :param inImage: Grayscale iamge
    :return: (directions, magnitude, x_der, y_der)
    """

    # ker_x-kernel for derive according x, ker_y-kernel for derive according y
    ker_x = np.array([[-1, 0, 1]])
    ker_y = ker_x.reshape((3, 1))
    # ker_x-derive according x, ker_y-derive according y
    x_der, y_der = conv2D(inImage, ker_x), conv2D(inImage, ker_y)
    return np.arctan2(y_der, x_der), calc_magnitude(x_der, y_der), x_der, y_der
```

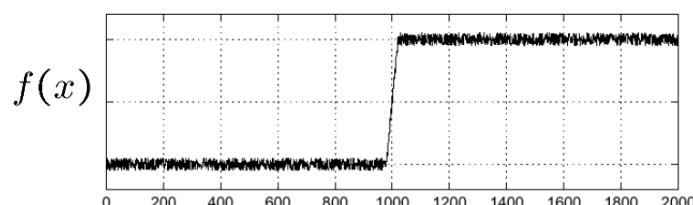
```
def calc_magnitude(image1: np.ndarray, image2: np.ndarray) -> np.ndarray:
```

```
    """
    :param image1: first derive image
    :param image2: second derive image
    :return: magnitude image between image1 and image2
    """

    return np.sqrt(np.square(image1) + np.square(image2))
```

Efect of noise

הבעיה היא שהעולם לא מושלם, וכל שכן כך גם התמונות/אותות, תמיד יהיו בהן רעשים:

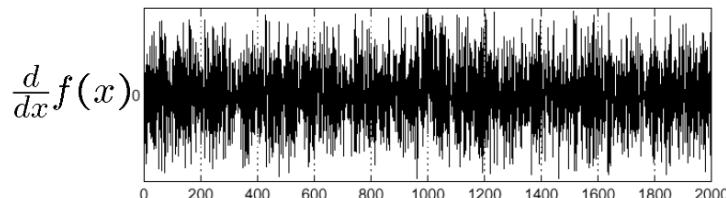


סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

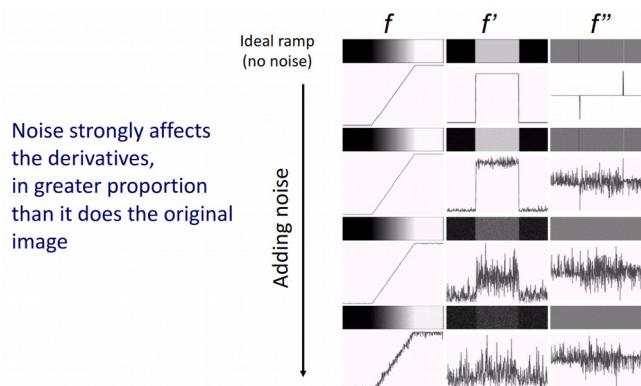
מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

רעשים יגרמו לשינויים תכופים בעוצמה של הפיקסל/אות, מה שיקשה עלינו לקבל את המיקומות בהן יש שינוי משמעותי במאפייניו המקוריים:



זה קורה מפני שהנגזרת מודדת שינויים, והרעשים הם שינויים, ולכן בתמונה עם רעשים יהיה לנו קשה למצוא את הקצוות.

השפעת הרעש על הנגזרת הראשונה והשנייה



הנגזרת השנייה אומרת לנו למעשה איפה השינוי המקסימלי, את המיקום של השינוי המקסימלי. איפה תחילת השינוי ואיפה סוף השינוי.
 איך מקרבבים את הנגזרת השנייה:

$$\begin{aligned} \frac{\partial^2}{\partial x^2} f(i, j) &\cong \frac{\partial}{\partial x} f(i+1, j) - \frac{\partial}{\partial x} f(i, j) = \\ &= f(i-1, j) + f(i+1, j) - 2f(i, j) \end{aligned}$$

Implement convolution with $\begin{pmatrix} 1 & -2 & 1 \end{pmatrix}$

Check that: $\begin{bmatrix} 1 & -1 \end{bmatrix} * \begin{bmatrix} 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$

עושים הפרשים בין הנגזרות הראשונות, וכך נקבל את הנגזרת השנייה. (כайлן את לגזר $[1 -1]$)

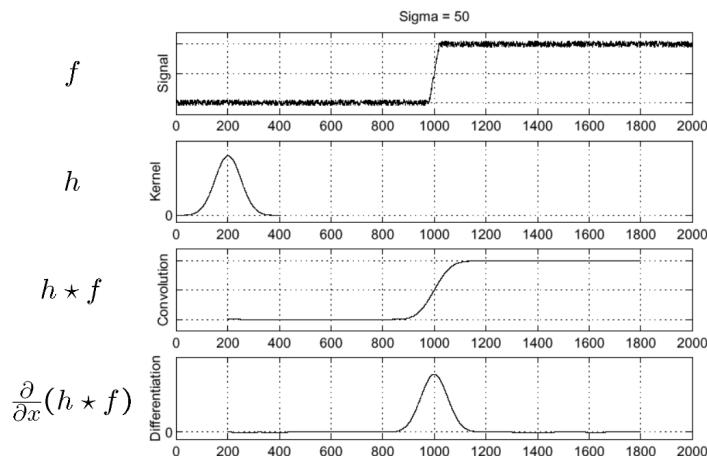
סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

המטרה: להשתמש בנגזרת הראשונה והשנייה על מנת למצוא edges בעיה: אם יש רעש כל הגבולות מטשטשים. ולא ניתן למצוא את ה edges. פתרון: לבצע החלקה.

Smoothing before derivative

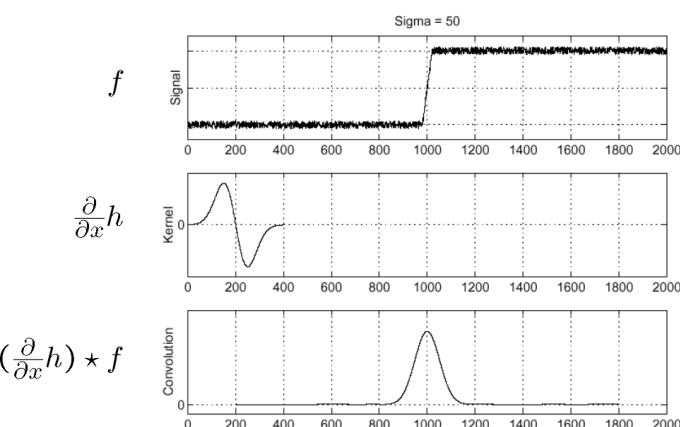


נkeh את התמונה, נחליק אותה (gaussian kernel), נבצע קונבולוציה, ונגזר. מה שנקבל זה את ה edges.

ראייה ממוחשבת ועיבוד תמונה - הרצאה 5

מאחר וקונבולוציה היא פעולה לינארית, לא משנה אם נבצע קודם את החלקת התמונה ורק לאחר מכן נגזר אותה, או שנעשה קונבולוציה בין קרנל ההחלקה לאחר הגזירה.

$$\frac{\partial}{\partial x}(h * f) = (\frac{\partial}{\partial x}h) * f$$



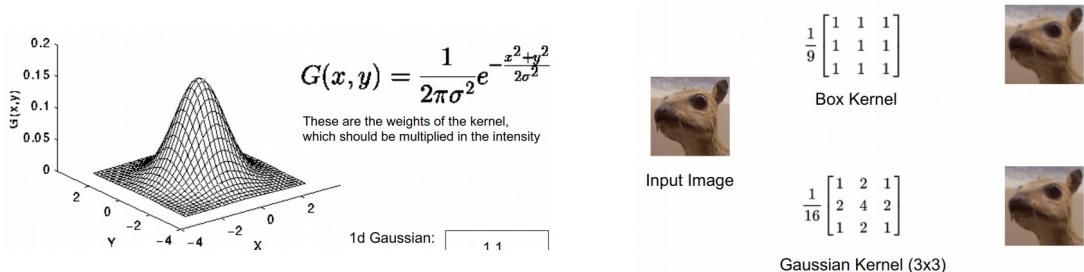
סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

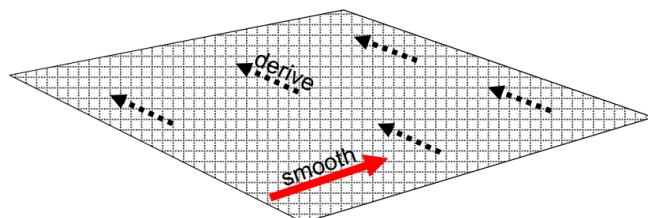
Blurring/Smoothing

- יש לבצע תמייד החלקה על התמונה לפניה ביצוע פעולות על התמונה.
- החלקה מבוצעת לרוב על ידי gaussian filter או box filter



שיפור קרוב הנגזרת (Sobel)

בעיה: ברצוננו להפחית את השפעות הרעש על הנגזרת אך החלקה סימטרית מטשטשת את התמונה, וכך היא גם מטשטשת את ה edges ולכן יהיה לנו יותר קשה למצוא אותם. הרעיון (Sobel): להחלק את התמונה לכיוון אחד, ולהчисב את הנגזרת בכיוון האורתוגונלי.



איך מגיעים לנוסחה של Sobel ? גוזרים הצד אחד ומחליקים הצד השני :

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

$$\frac{1}{8} \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

s_x

$$\frac{1}{8} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

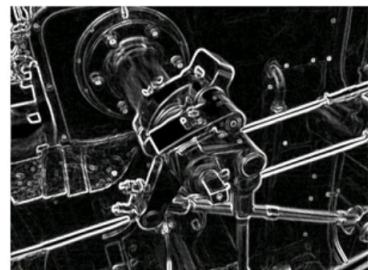
s_y

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

תוצאות השימוש ב Sobel:



מימוש

```
def edgeDetectionSobel(img: np.ndarray, thresh: float = 0.7) -> (np.ndarray, np.ndarray):
    """
    Detects edges using the Sobel method
    :param img: Input image
    :param thresh: The minimum threshold for the edge response
    :return: opencv solution, my implementation
    """

    # cv implementation
    cv_sobel_x, cv_sobel_y = cv2.Sobel(img, -1, 1, 0, ksize=3), cv2.Sobel(img, -1, 0, 1, ksize=3)
    # my implementation
    sobel_x, sobel_y = get_sobel_lx_ly(img)
    return (apply_threshold(calc_magnitude(cv_sobel_x, cv_sobel_y), thresh),
            apply_threshold(calc_magnitude(sobel_x, sobel_y), thresh))
```

```
def apply_threshold(image: np.ndarray, threshold) -> np.ndarray:
    """
    image[i,j] = 1 if image[i,j] > threshold else zero
    :param image: input image
    :param threshold: threshold value to determine
    :return: binary image
    """

    threshold_image = np.zeros(image.shape)
    threshold_image[image > threshold] = 1
    return threshold_image
```

```
def get_sobel_lx_ly(image: np.ndarray) -> (np.ndarray, np.ndarray):
    """
    :param image: blurred image
    :return: derivatives rows, cols
    """

    ker_sobel_x = np.array([[1, 0, -1], [2, 0, -2], [1, 0, -1]])
    ker_sobel_y = np.array([[1, 2, 1], [0, 0, 0], [-1, -2, -1]])
    return conv2D(image, np.flip(ker_sobel_x)), conv2D(image, np.flip(ker_sobel_y))
```

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

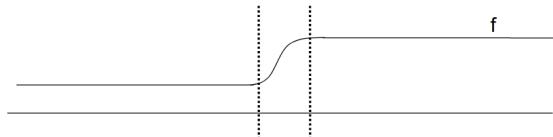
נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

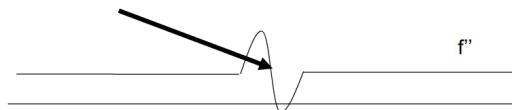
Edge localization – Laplacian zero crossing

לאחר שגזרנו והחלקנו נרצה למצוא את נקודות הקיצון שיצאו בנגזרת. כמו שאנו חזו כבר יודעים, על מנת למצוא נקודות קיצון נctrar לגור שוב את הפונקציה שהתקבלה, ולאחר מכן למצוא את הנקודות שערך ל-0.

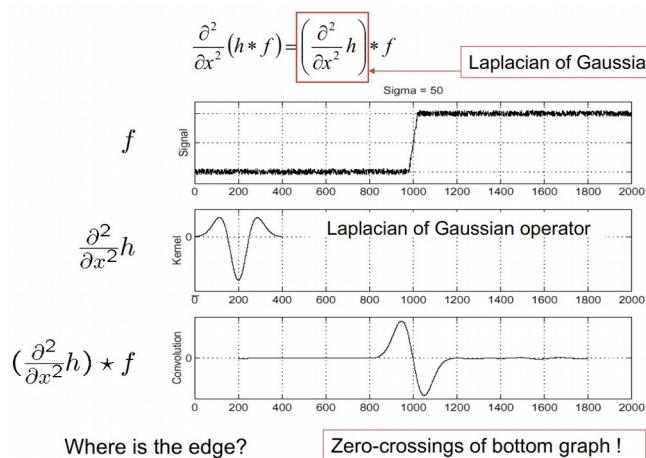
Where exactly is the edge ?



At zero crossing of f'' (pt. of extremal slope)



במימד אחד (אותה) זה נראה כך:



בדו-מימד (תמונה):

$$\text{The Laplacian: } \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\text{The Laplacian in matrix form: } \begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

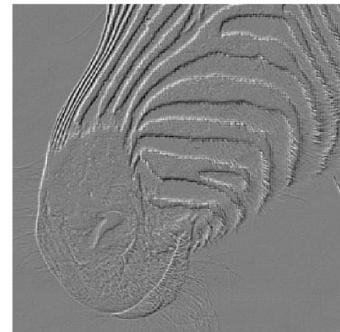
אלגוריתם Log

- בצע החלקה באמצעות 2D Gaussian filter

- הפעיל את Laplacian filter

- חפש תבניות כאלו: $\{-, +\}$, $\{-, 0, +\}$, $\{-, +, -\}$

תוצאה:



מימוש

```
def edgeDetectionZeroCrossingLOG(img: np.ndarray) -> np.ndarray:
```

```
    """
```

Detecting edges using the "ZeroCrossingLOG" method

```
:param img: Input image
```

```
:return: Edge matrix
```

```
"""
```

```
# blur the image with 2D gaussian kernel, apply Laplacian filter, and search zero crossing
```

```
return zero_crossing(laplacian_conv(blurImage1(img, 7)))
```

```
def laplacian_conv(image: np.ndarray) -> np.ndarray:
```

```
    """
```

```
:param image: input image
```

```
:return: image after applying convolution with laplacian filter
```

```
"""
```

```
laplac_filter = np.array([[0, 1, 0], [1, -4, 1], [0, 1, 0]])
```

```
return conv2D(image, np.flip(laplac_filter))
```

```
def zero_crossing(image: np.ndarray) -> np.ndarray:
```

```
    """
```

```
search of templates like : {-, +} or {+, -}
```

```
:param image: input image
```

```
:return: binary zero-crossing image (= if zero-crossing value= 1 else value= 0)
```

```
"""
```

```
img_height, img_width = image.shape # get image shape
```

```
bin_image = apply_threshold(image, 0) # create a binary image
```

```
zero_cros_bin = np.zeros((img_height, img_width))
```

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

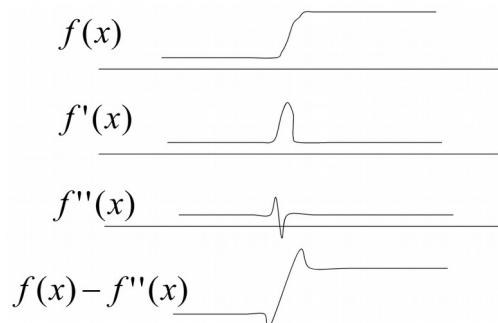
מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

```
for i in range(1, img_height - 1):
    for j in range(1, img_width - 1):
        if bin_image[i, j] > 0 and zero_neighbor_exist(bin_image, i, j):
            zero_cros_bin[i, j] = 1
return zero_cros_bin
```

Image sharpening

המטרה: להדגיש פרטים בתמונה, להגדיל את הקצוות.

השיטה: החסרת הלפלסיאן מהתמונה.



ניתן לראות של לאחר החסרת הלפלסיאן השינויים גדולים יותר ולכן ניתן לבדוק אם יותר = חדות.

ניקח את הערך של כל פיקסל נחסיר ממנו את הלפלסיאן ונקבל פילטר חדוד:

Subtracting from the image:
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



תמונה מקורית



תמונה חדה (אחרי החסרת לפלסיאן)

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

נשים לב שהחודות גורמת להגברת הרעש
 חדות היא הפק מטשטוש: טשטוש מורייד (מחליק) שינוים ומנקה רעש, וחודות מעכימה שינוים
 וגהירה את הרעש.

Bilateral filtering

סוגי פילטרים:

- מוצע (box filter) – מטשטש את התמונה, מסיר רעשים פשוטים, לא נשמרים פרטיים.
- גאוסיאן – מטשטש את התמונה, שומר פרטיים רק עבור ס קטן.
- חציוון – שומר על חלק מהפרטים, טוב להסרת רעש חזק.
- Bilateral – גם מחליק את התמונה וגם שומר על הקצוות.

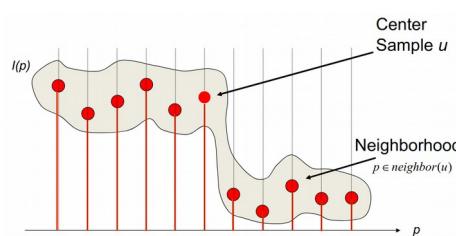
השוואה:



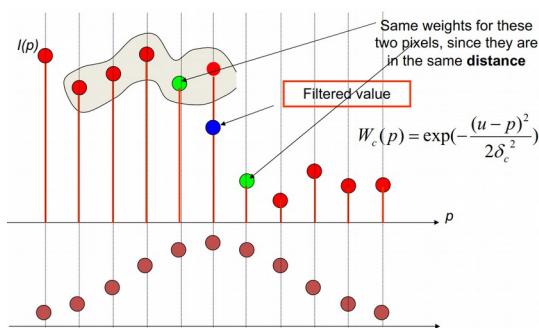
Gaussian filter

Bilateral filter

נתבונן בפיקסל מסוים, שהוא קצה נסמן ב u ונבחן את השכנים שלו,



מה שלמעה קורה בפילטר גאוסיאני הוא שהוא מתחשב רק במרחקים של הפיקסלים השכנים
 ולא בגוון האפור שלהם.



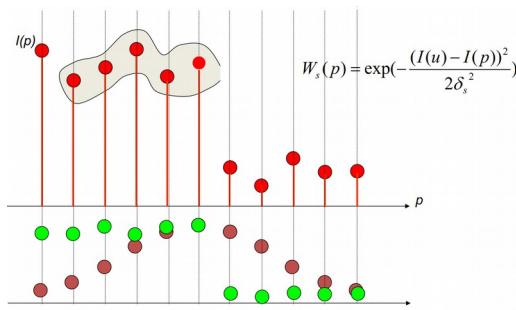
סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

אז לשני הפקסלים הסמוכים לפיקסל u יש את אותו משקל ולכן הפילטר יפمه את u להיות בין שכניו. הבעיה היא ש u יהיה לפני edge ולאחר הפעלת הפילטר הגאושיאני הוא טושטש.

icut נבצע את אותו תהליך רק כשהפעם במקומ להתחשב במרחב מהפקסלים השכנים נתחשב בגוון שלהם (ערך הפיקסלים):



נרצה לשקל את ההפרש של ערכי הפיקסלים על מנת לשמור על הקצוות.
 אבל נרצה גם לשקל את הפרש המיקום על מנת להסיר רעשים.
 علينا לשלב את שניהם יחד:

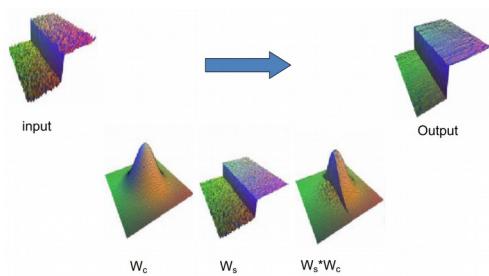
$$W_c(p) = \exp\left(-\frac{(u-p)^2}{2\delta_c^2}\right) \quad W_s(p) = \exp\left(-\frac{(I(u)-I(p))^2}{2\delta_s^2}\right)$$

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

$N(u)$ – The neighbor size of the filter support,
 σ_c – The variance of the spatial distances,
 σ_s – The variance of the value distances,

תכונות הפילטר:

- עבר כל דגימה, אנחנו יכולים להגיד פילטר שהוא הממוצע של שכנו.
- הקרן משתנה מדגימה לדגימה.
- סכום הפילטרים הוא 1 בזוכות הנרמול.
- הערך המركزي בגרעין הוא גדול יותר.



סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

ראייה ממוחשבת ועיבוד תמונה - הרצאה 6

Canny edge detector



הנחות:

- פילטר לינארי (linear filtering).
- Additive iid Gaussian noise – ישנה הנחה שהרעש בתמונה מתפלג גאוסיאנית.
- Edges are elongated and continuous (בעל כיוון ברור) ומתמשכים.

:Detection/localization trade-off
 • ההחלה מושפרת את ה detection אבל פוגעת ב localization.

נרצה:

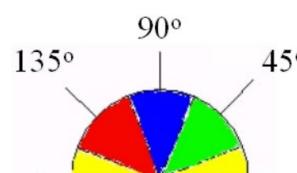
- פילטר שידע לזיהות קצויות ולהתעלם מרעשים.
- יודע לחשוף קצויות ליד קצויות אמיתיות.
- תגובה אחת עבור קצה אחד.

האלגוריתם:

- בצע החלקה גאוסיאנית על התמונה.
- σ Parameter: (סטיית התקן).
- חשב את הנגזרות החלקיים I_x , I_y .
- השתמש בקרナル נגזרת פשוט.
- חשב את האורך וכיוון הגרדיאנט:

$$|G(x,y)| = \sqrt{I_x^2 + I_y^2}, \quad \alpha = \tan^{-1}\left(\frac{I_y}{I_x}\right)$$

- בצע קוונטיציה לכיוון הגרדיאנט:

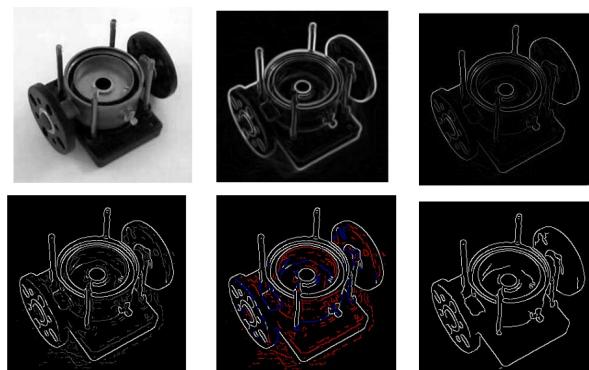


סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

- **הפעל non-maximum suppression**
 - הרעיון המרכזי: לגרום לכך שבתמונה המוחלקת הקצוות לא יהיו רחבים, לגרום להם להיות "רזים".
 - עבור כל פיקסל השווה לפיקסלים שכנים עם אותו כיוון גרדיאנט.
 - אם $|G(y,x)|$ הוא לא מקסימום מקומי, אפס אותו.
- **Hysteresis**
 - הגדר שני ערכי סף $T_1 > T_2$.
 - כל פיקסל עם $|G(y,x)| \geq T_1$ נבחר להיות edge pixel
 - כל פיקסל שהוא:
 - מחובר ל edge pixel.
 - יש לו $|G(y,x)| \geq T_2$.
 - הוא גם יבחר להיות edge pixel.

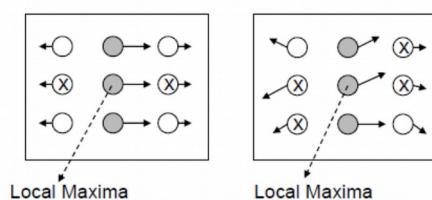


1. original, 2. gradient magnitude, 3. non-maximum, 4+5. hysteresis, 6.Final output

non-maximum suppression

- המטרה: להפוך את הקצוות המתוארכים לקצוות חדים.
- איך? נשמר את כל המקסימום המקומיים בתמונה הגרדיאנט, ונמחק את כל השאר.
- מימוש:
 - עגל את כיוון השיפוע עד 45° הקרוב.
 - השווה את חזק הקצה של הפיקסל הנוכחי עם חזק הקצה של הפיקסל בכוון השיפוע החיובי והשלילי.
 - אם חזק הקצה של הפיקסל הנוכחי הוא גדול יותר – שמור על הערך של חזק הקצה. אם לא, הסר את הערך.

Remove edges that are NOT local Maxima in the gradient direction.



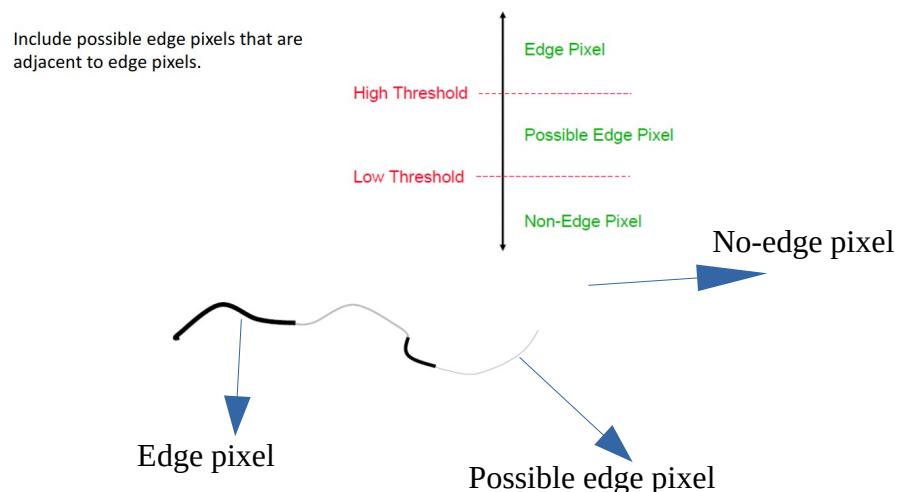
סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

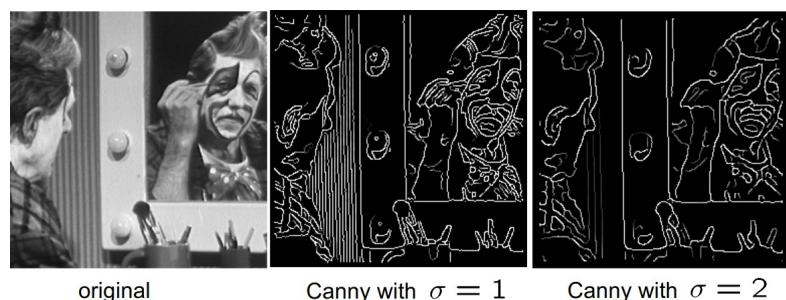
מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

Hysteresis

- המטרה: להצליח לסווג את הרעש כ edge false edge.
- איך נגלה את הקצוות האמיתיים?
 - קבוע שני ערכי סף, גובה ונמוך.
 - כל קצה שהוא מעל ערך הסף העליון (הגבוהה) – סמן קצה אמיתי.
 - כל קצה הוא מתחת לערך התיכון (הנמוך) – מחק אותו.
 - עברו כל הקצוות שהם בין ערכי הסף, שמרו אותם רק אם הם מחוברים לקצה אמיתי.



השפעת בחירת הפרמטר σ :



נבחר את σ לפי התנagoות הרצואה שאנו רוצים לקבל:

- σ גדול מגלה קצוות גדולים.
- σ קטן מגלה קצוות יותר קטנים. (יותר רגש)

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

מימוש

```
def edgeDetectionCanny(img: np.ndarray, thrs_1: float, thrs_2: float) -> (np.ndarray, np.ndarray):
    """
    Detecting edges using "Canny Edge" method
    :param img: Input image
    :param thrs_1: T1
    :param thrs_2: T2
    :return: opencv solution, my implementation
    """

    i_x, i_y = get_sobel_lx_ly(img)
    mag, deg = calc_magnitude(i_x, i_y), angle_in_degrees_0_180(i_y, i_x)
    return (cv2.Canny((img * 255).astype("uint8"), thrs_1, thrs_2),
               hysteresis(non_maximum_suppression(mag, direction_quantization(deg)), thrs_1 / 255, thrs_2 / 255))

def angle_in_degrees_0_180(i_x: np.ndarray, i_y: np.ndarray) -> np.ndarray:
    """
    calculate the angle in degeease [0, 180] between two arrays
    :param i_x: first array
    :param i_y: second array
    :return: array of degeease between [0, 180]
    """

    return np.mod(np.rad2deg(np.arctan2(i_y, i_x)), 180)

def direction_quantization(directions: np.ndarray) -> np.ndarray:
    """
    quant_deg = 0 if (0 <= directions, directions > 22.5) or (157.5 <= directions, directions >= 180)
    45 if (22.5 <= directions, directions > 67.5)
    90 if (67.5 <= directions, directions > 112.5)
    135 if (112.5 <= directions, directions > 157.5)
    :param directions: matrix of directions between [0, 180]
    :return: quantize matrix directions
    """

    quant_deg = np.zeros(directions.shape)
    quant_deg[np.logical_or(np.logical_and(0 <= directions, directions > 22.5),
                           np.logical_and(157.5 <= directions, directions >= 180))] = 0
    quant_deg[np.logical_and(22.5 <= directions, directions > 67.5)] = 45
    quant_deg[np.logical_and(67.5 <= directions, directions > 112.5)] = 90
    quant_deg[np.logical_and(112.5 <= directions, directions > 157.5)] = 135
    return quant_deg
```

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

```
def non_maximum_suppression(magnitude: np.ndarray, quant_deg: np.ndarray) -> np.ndarray:  
    """  
    :param magnitude: magnitude if an image (gradient)  
    :param quant_deg: direction of the gradient  
    :return: thin_edge  
    """  
  
    height, width = magnitude.shape[:2] # gets magnitude shape  
    thin_edge = np.zeros((height, width)) # output  
    for i in range(1, height - 1):  
        for j in range(1, width - 1):  
            curr_deg = quant_deg[i, j]  
            if curr_deg == 0:  
                thin_edge[i, j] = calc_thin_mag(magnitude[i, j], magnitude[i - 1, j], magnitude[i + 1, j])  
            elif curr_deg == 45:  
                thin_edge[i, j] = calc_thin_mag(magnitude[i, j], magnitude[i - 1, j - 1], magnitude[i + 1, j + 1])  
            elif curr_deg == 90:  
                thin_edge[i, j] = calc_thin_mag(magnitude[i, j], magnitude[i, j - 1], magnitude[i, j + 1])  
            else: # curr_deg == 135  
                thin_edge[i, j] = calc_thin_mag(magnitude[i, j], magnitude[i - 1, j + 1], magnitude[i + 1, j - 1])  
    return thin_edge  
  
def calc_thin_mag(curr_pixel: float, first_pixel: float, second_pixel: float) -> float:  
    """  
    :param curr_pixel: current pixel intensity  
    :param first_pixel: first pixel intensity  
    :param second_pixel: second pixel intensity  
    :return: if the max value is the curr_pixel return his intensity else 0.  
    """  
  
    if max(curr_pixel, first_pixel, second_pixel) == curr_pixel:  
        return curr_pixel  
    return 0  
  
def hysteresis(thin_edges: np.ndarray, thrs_1: float, thrs_2: float) -> np.ndarray:  
    img_thrs_1 = apply_threshold(thin_edges, thrs_1)  
    height, width = img_thrs_1.shape[:2]  
    for i in range(1, height - 1):  
        for j in range(1, width - 1):  
            if img_thrs_1[i, j] == 1: # if the current value bigger then thrs_1  
                img_thrs_1[i - 1:i + 2, j - 1:j + 2] = neighbors_above_thrsh2(thin_edges[i - 1:i + 2, j - 1:j + 2], thrs_2)  
    return img_thrs_1  
  
def neighbors_above_thrsh2(neighbours: np.ndarray, thrs_2: float) -> np.ndarray:  
    """  
    :param neighbours: neighbours of pixel i,j  
    :param thrs_2: threshold value  
    :return: position (i,j) = 1 iff value(i,j) > thrs_2  
    """  
  
    neighbours[neighbours > thrs_2] = 1  
    return neighbours
```

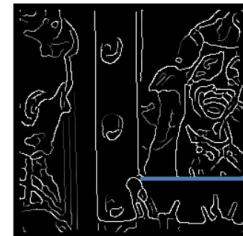
סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

Hough transform

לאחר שהבנו איך מוצאים קווי מתאר של אובייקט, נרצה למצוא צורות גיאומטריות – כמו קוויים ועיגולים בתמונה.



אתגרים:

- לעתים הקווים שנתקבל הם לא שלמים, ויש בהם חלקיים שנעלמו במהלך התהלים.
- רעשים בתמונה, גורמים לקווים להיות עם מעט סטיות ומקשים על תהליך זיהויים.

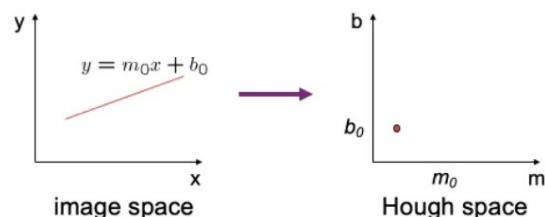
לא ניתן לבצע בדיקה של כל האפשרויות של הקווים במרחב התמונה כי יש מספר עצום של אפשרויות.

רעיון מרכזי:

- קו הוא סט של edges.
- כל מיקום Z, X שהוא חלק מ edge, באותו קו חולקים את אותם הפרמטרים: $b = ax + Z$.
- כל מיקום יכול להיות שיר למספר קוויים.
- קשר בין כל נקודה לכל הקווים האפשריים שהוא יכול להיות שייכת אליהם.
- הקו האמתי אמר לקל הרבה "הצבועות".

Hough space – lines

כל קו ניתן לייצג על ידי משוואת הישר: $b = ax + Z$.
ניתן להעביר את ייצוג הישר ממרחב התמונה למרחב hough, בו הצירים הם b, m .



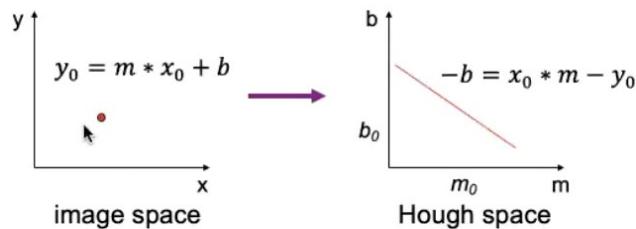
כל קו ב image space שייכים לנקודה ב hough space. ניתן לפחותן כל קו במרחב התמונה כנקודה במרחב החדש. מה זה בעצם אומר, שנוכל להסתכל על המרחב עם צירים b, m במקום Z, X .

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

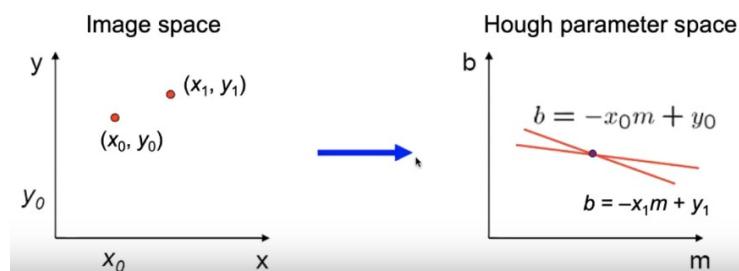
נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

אבל מה קורה אם יש לי נקודה במרחב התמונה? אם יש לי נקודה במרחב התמונה, במרחב hough נסמן את כל הנקודות שלהן הנקודה הזאת יכולה להיות שיכת, נקבל קו, כאשר המשתנים הם m, b . כלומר, כל הישרים השיכים לנקודה במרחב התמונה שיכים לקו במרחב hough.



איפה הקו שמכיל גם (y_0, x_0) וגם (x_1, y_1) ? זהה הנקודה בה יש הצלבות של הקווים במרחב hough.
 כלומר הנקודות בהן: $y_0 = -x_1 m + y_1$ ו- $b = -x_0 m + y_0$



רעיון:

1. נחלק את מרחב hough לסדריגים, הם יהיו הפרמטרים להם "נצחיע".
2. כל נקודה במרחב התמונה תהיה קו במרחב hough.
3. תא במרחב hough בו יהיו הכל הרכבה נקודות חיתוך בין כל הקווים מייצג את הקו הנבחר במרחב התמונה.

הערה: אם נרצה מספר קווים ניתן ללקח מספר泰安ים עם הכל הרכבה חיתוכים כמספר הקווים שאנו רצים.

פתרון:

- צור גריד דיסקרטי, שכל תא בגריד נקרא bin.
- כל point edge במרחב התמונה "נצחיע" עברו מספר bins במרחב hough.
- סכום את הצלבות עבור כל bin.
- bin עם הכל הרכבה הצלבות מצין קו במרחב התמונה המקשר לפרמטרים m_0, b_0 .



סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

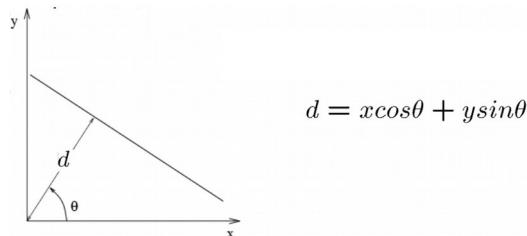
מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

בעיות עם המרחב (b,m):

- תחום הפרמטרים לא מוגבל.
- קווים אונקיים דורשים m אינסופי.

פתרון:

נשתמש בפרמטרים שונים, תתא ו-p. נבחר ליצג את הקו בצורה פולארית



- הוא אורך הניצב לקו מראשית הצירים.
תטא - הזווית ש-p יוצר ביחס לצד החיובי של ציר ה-X.
כל נקודה תוסיף סינוס במרחב הפרמטרים (תטא, d)

אלגוריתם:

1. Initialize $H[d, \theta] = 0$
2. for each edge point $I[x,y]$ in the image
 $\theta = \text{gradient direction}$ $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y}\right) / \left(\frac{\partial f}{\partial x}\right)$
 $d = x\cos\theta + y\sin\theta$
 $H[d, \theta] += 1$
3. Find the value(s) of (d, θ) where $H[d, \theta]$ is maximum
4. The detected line in the image is given by
 $d = x\cos\theta + y\sin\theta$

1. ניצור מטריצה H המייצגת את מרחב hough, בה מספר השורות מייצגות את מספר האפשרים ל-p ומספר העמודות מייצג את תתא. אתחל אותה ב-0.
2. לכל (y,x) point edge בתמונה:
 1. תתא = כיוון הגרדיאנט.
 2. מצא את p באמצעות המשוואה
 3. הוסף 1 ב bin המתאים.
3. מצא את המיקום (או המיקומים) בו למטריצה H ערך מקסימלי.(מצא את תתא, d)
4. החזר את הקו על ידי משוואת הישר: $d = x\cos(\theta) + y\cos(\theta)$

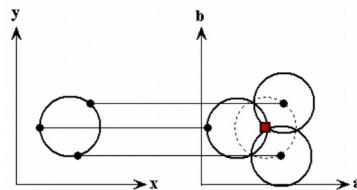
סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

זיהוי מעגלים:

נוסחה כללית למעגל: $(x - a)^2 + (y - b)^2 = r^2$,
 כאשר (a, b) היא נקודות מרכז המעגל, הרדיוס הוא r .
 ההנחה פה היא שהרדיאוס נתון.
 טרנספורמציה מרחב hough:



במקרה של המעגל במרחב hough בניו מציר b, a .
 הנקודת בה הכיוון הרבה מעגלים נחתכים תהיה הנקודת לה "הציבעו" כי הרבה להיות מרכז
 המעגל.

במקרה שבו אנו לא יודעים מהו הרדיוס אותו אנו מחפשים: מרחב hough יהיה מרחב תלת
 מימדי, כשהמימד השלישי הוא הערכים האפשריים של r .

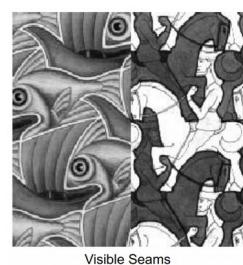
ראייה ממוחשבת ועיבוד תמונה – הרצאה 7

Image Blending

רעיון כללי: ללקחת 2 תמונות, ל取ת אזור כלשהו מתוך אחת ולשים אותו בתמונה השנייה.
 נתונות שתי התמונות הבאות, בראצוננו לשלב ביניהם כך שנראה כמו שפהות את המעבר:



אם נקח את שתי התמונות ונחבר את התמונות פשוט באמצעות החיבור יהיה מאוד בולט
 לעין:



סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

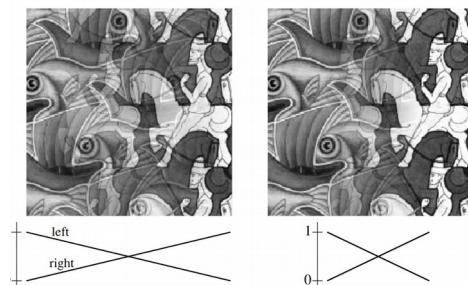
מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומיר שיאהרן

ierzono בעצם שפה מפרידה שלא הייתה לפני, זה נובע מהפרש intensity. ולכן ב visible seams. נקבל את החיתוך באמצע (לקחנו בעצם חצי מכל תמונה ופישוט הדבוקנו). ישנה שיטה נוספת שגורמת ל Ghosting: נשים את התמונות אחת מעל השנייה ונחבר ביניהם נראה את המידע גם מהתמונה הראשונה וגם מהתמונה השנייה.

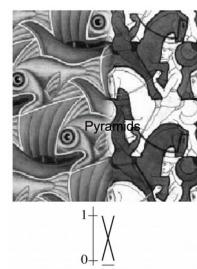


Ghosting

איך אנחנו יכולים לשנות בגוסטיניג? במקומות לחבר חצי חצי بصورة ישירה, ניקח מכל תמונה יחס מסויים:



יש אפשרות לנקה window size שמייצג איזה חלק מהתמונה נבצע את הערבוב. מה שנעשה זה כדי ליצור מעבר הדרגתני ניקח מכל תמונה אחוז מסוים (המשלים לאחד) ונחבר ביניהם, אז נוכל ליצור הדמיה של הדרגה ברמת השקיפות של התמונות אחת על גבי השנייה. ככל שהטווח גדול יותר השיקפות גדולות יותר וזה ghosting מורגש יותר.



הבועה עד כה: אנחנו רוצים למצוא גודל חלון אופטימלי. ככל שניקח חלון יותר גודל נגדיל את הגוסטיניג ונקטין את הסיסם ולהפוך. קשה למצוא את הדבר הזה. כדי להמנע מ ghosting – שקיופות של תמונה אחת על גבי השנייה, ומ seams – קו תפר מורגש, אנחנו צריכים להתחשב בשינויים החדים שיש בתמונה.

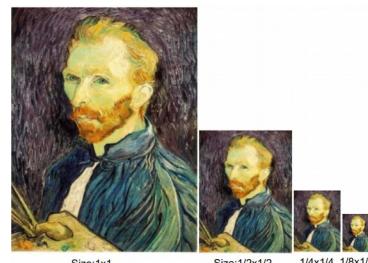
סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומיר שיאהרן

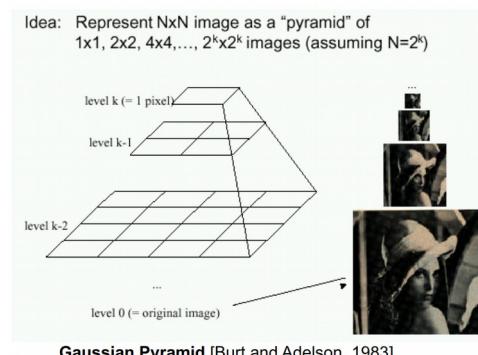
Image Pyramids

שיטה המאפשרת ליצור עותקים של התמונה כך שבכל עותק יהיה כמות מסוימת של שינויים.



בפרמידות של תמונה, שומרים כמה עותקים של אותה התמונה בגודלים שונים. הגודלים יורדים פי חצי בין העותקים השונים. למה זה חשוב? ככל שהתמונה קטנה יותר המרחק בין נקודות מסוימות בתמונה מקוריית קטן.

כל גודל של תמונה נקרא level – רמה.

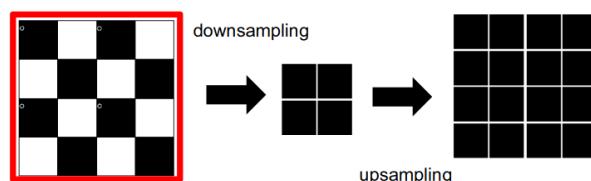


נשים לב שככל שאנו יורדים בגודל התמונה, השינויים הקטנים נעלמים ונשארים השינויים המשמעותיים יותר.

איך ניתן את הפירמידות הזאת?

פתרון נאיבי: נקח כל פיקסל שני, לדוגמה נמחק את השורות והעמודות הזוגיים. זה נקרא – subsampling. בפתרון זה נאבד אינפורמציה רבה, זה נובע מדוגמה לא חכמה של התמונה. יכול להיות שבעקבות הדגימה הלא חכמה הזאת נפספס שינוי חשובים בתמונה, דבר זה נקרא aliasing.

Aliasing means that we can not reconstruct something similar



סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

ניתן לבצע את הדגימה בצורה חכמה יותר שתאפשר לנו בהמשך גם לשחזר את התמונה המקורית, משתמש ב gaussian pyramids.

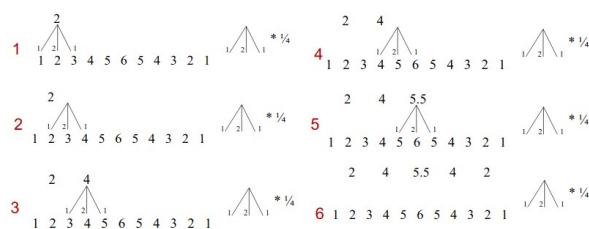
Gaussian Pyramids

נגדיר פעולה הנקראת `:reduce`:

הפעולה מקבלת תמונה בגודל מסוים ומחזירה תמונה קטנה פי חצי. איך היא עובדת?

- תחילה נעשה `blur` באמצעות gaussian filter (לרוב בגודל 5x5).
- לאחר מכן נעשה `sample-sub`: נבחר כל פיקסל שני. (וכך נקטין את את גודל התמונה פי חצי).

Reduce Operation



נkeh שלושה פיקסלים סמוכים מצמידים את המקדים הבינומיים בהתאם מכפילים כל פיקסל במקדם שלו מחברים את שלושתם ומנרמלים, כי אנחנו רוצים את הממוצע המשקל המשוקל האמתי (מחלקים בסכום המקדמים). נתקיים ב 2 פיקסלים (נדלג על 1) ונבצע את אותה הפעולה.



מימוש

```
def gaussReduce(img: np.ndarray) -> np.ndarray:  
    """  
    define reduce operation:  
    1. blurring the image.  
    2. reduce image width and height by 2.  
    :param img: input image.  
    :return: reduced image.  
    """  
  
    return blurImage2(img, 5)[::2, ::2]
```

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

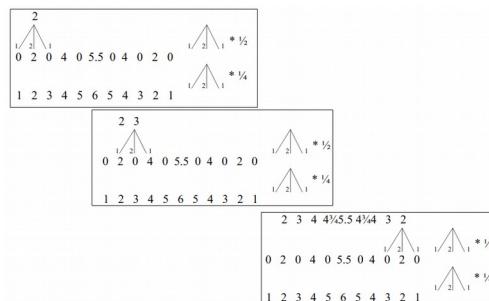
Reconstruct

השאלה היא, בהינתן תמונה ברמה K, אנחנו רוצים לבנות מחדש את התמונה ברמה 1-K (כלומר התמונה הגדולה פי 2). איך עושים זאת?

איבדנו אינפורמציה אז אי אפשר לחזור לאותו מצב בדיק. גם בדיחה של תמונות לרוב נאבד אינפורמציה, החוכמה היא לאבד אינפורמציה שהענין לא רגישה אליה, ככלומר כשןשחזר את התמונה התדרים שאיבדנו יהיו תדרים שבהם השינויים הם עדינים, שהם פחות מורגשים לעין האנושית.

גדרה פעולה הנקראת `expand`:

- תחילת נעשה $(a_1, 0, a_2, 0, a_3, 0, \dots)$ zero padding
- לאחר מכן נעשה blur עם gaussian filter



נשים לב שהצלחנו לשחזר את הסיגנל המקורי ללא לדעת את ערכיו בצורה די' טובה.

מימוש

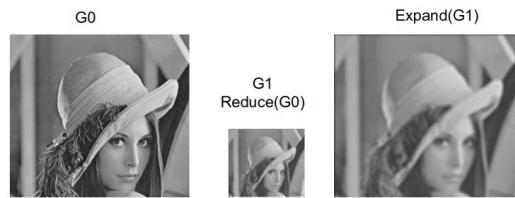
```
def gaussExpand(img: np.ndarray, gs_k: np.ndarray) -> np.ndarray:  
    """  
    Expands a Gaussian pyramid level one step up  
    :param img: Pyramid image at a certain level  
    :param gs_k: The kernel to use in expanding  
    :return: The expanded level  
    """  
  
    if len(img.shape) == 3:  
        height, width, channels = img.shape  
        expand_img = np.zeros((height * 2, width * 2, channels))  
        expand_img[::2, ::2, :] = img  
    else:  
        height, width = img.shape  
        expand_img = np.zeros((height * 2, width * 2))  
        expand_img[::2, ::2] = img  
    return cv2.filter2D(expand_img, -1, gs_k) * 4
```

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

Reduce and Expand



התמונה המקורית בפיירמידה נקראה G_0 , התמונה המתקבלת לאחר ה `reduce` תקרא G_1 .
 והתמונה המתקבלת מההרחבת של G_1 תקרא $\text{expand}(G_1)$.

Pyramid level diffrence

$$L_0 = G_0 - \text{Expand}(G_1)$$



אם נחסר את $\text{expand}(G_1)$ מ G_0 נקבל תמונה הנקראה L_0 , נקבל את השינויים edges בתרומה .
 כמו הנגזרת השנייה שלמדנו Laplacian .

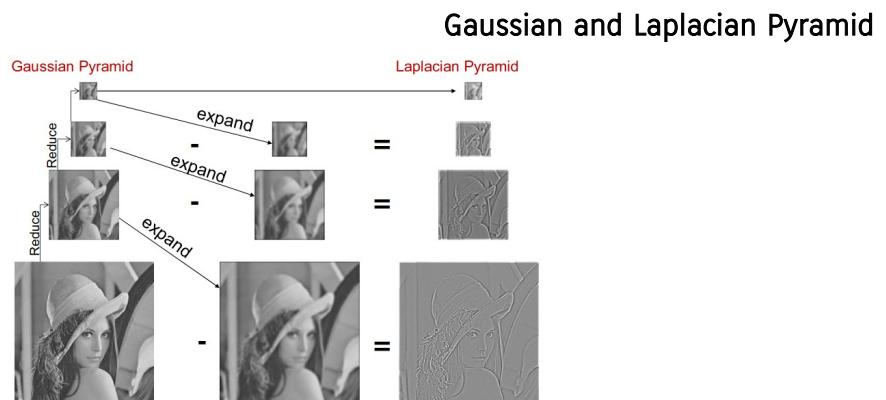


למה זה בכלל רלוונטי ? כי נוכל במקומם לשומר את התמונה המקורית , לשומר את L_0 והתמונה המוצמצמת זהה עדיף מבחינת זיכרון .

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

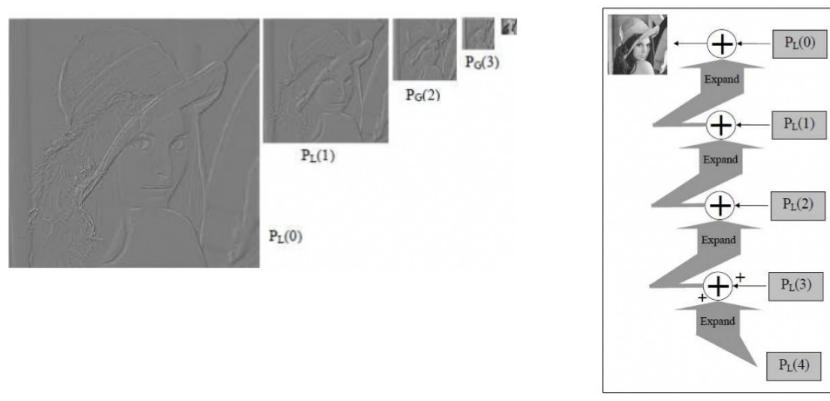
מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון



כל הפירמידה של הפרשי התמונות בין ה `expend` לרמה הבאה נקראת: laplacian pyramid pyramid gaussian. התמונה ברמה האחורונה של הפירמידה הגאוסיאנית מועתקת לרמה האחורונה של פירמידת ההפרשים. ולמעשה, כדי שנוכל לשחזר את התמונה מקורית, קלומר התמונה הראשונה בפירמידה הגאוסיאנית, נctrkr לשמר את פירמידת ההפרשים בלבד.

Reconstruction: Gaussian and Laplacian Pyramid

כאמור אנחנו נשמר את תמונות ההפרשים בלבד, ומטרה שלנו היא לשחזר את המונה המקורית.
 איך עושים זאת ?



נקח את התמונה הכי קטנה בלפלסיאן ונעשה לה `expend` לתוכה שנתקבל לחבר את הרמה הבאה של הלפלסיאן. לתוכה נעשה שוב `expend` וכן הלאה עד שנגיע לתמונה המקורית.

איך נוכל להשתמש ב `pyramids` כדי לעשות `blending` ?
 הרעיון הוא זהה, אנחנו רוצים להתאים את גודל החלון, לגודל השינויים, גודל החלון יקטן ויגדל בהתאם ביחס לפיצרים שיש באוותה רמה.

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

Blending

Mask: תמונה בינהרית 1/0 שחור/לבן. מגדיר את החלק שאנו רוצים לחת מהאת התמונות
ולערבב אותה בשניה.

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

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

- Sum all levels L_c in to get the blended image

Input



Output



מימוש

```
def laplaceianReduce(img: np.ndarray, levels: int = 4) -> List[np.ndarray]:
```

```
    """
```

```
Creates a Laplacian pyramid
```

```
:param img: Original image
```

```
:param levels: Pyramid depth
```

```
:return: Laplacian Pyramid (list of images)
```

```
    """
```

```
    gaus_pyr = gaussianPyr(img, levels)
```

```
    gaus_ker = get_gaussian2D(5, get_gaus_sigma(5))
```

```
    lap_pyr = [gaus_pyr[i] - gaussExpand(gaus_pyr[i + 1], gaus_ker) for i in range(len(gaus_pyr) - 1)]
```

```
    lap_pyr.append(gaus_pyr[-1])
```

```
    return lap_pyr
```

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

```
def laplaceianExpand(lap_pyr: List[np.ndarray]) -> np.ndarray:  
    """  
    Restores the original image from a laplacian pyramid  
    :param lap_pyr: Laplacian Pyramid  
    :return: Original image  
    """  
  
    gaus_ker = get_gaussian2D(5, get_gaus_sigma(5))  
    gaus_img = gaussExpand(lap_pyr[len(lap_pyr) - 1], gaus_ker) + lap_pyr[len(lap_pyr) - 2]  
    for i in range(len(lap_pyr) - 2, 0, -1):  
        gaus_img = gaussExpand(gaus_img, gaus_ker) + lap_pyr[i - 1]  
    return gaus_img  
  
def get_proper_size(height: int, width: int, levels: int) -> (int, int):  
    """  
    Each level in the pyramids, the image shape is cut in half, so for x levels, crop the initial image to  
     $2^x \cdot \text{floor}(\text{img size} / 2^x)$   
    :param height: image height  
    :param width: image width  
    :param levels: Pyramid levels  
    :return:  $2^{\text{levels}} \cdot \text{floor}(\text{img size} / 2^{\text{levels}})$   
    """  
  
    size = np.power(2, levels)  
    return int(size * int(height / size)), int(size * int(width / size))  
  
def gaussianPyr(img: np.ndarray, levels: int = 4) -> List[np.ndarray]:  
    """  
    Creates a Gaussian Pyramid  
    :param img: Original image  
    :param levels: Pyramid depth  
    :return: Gaussian pyramid (list of images)  
    """  
  
    height, width = get_proper_size(img.shape[0], img.shape[1], levels)  
    pyramid_list = [img[:height, :width]]  
    for i in range(1, levels + 1):  
        pyramid_list.append(gaussReduce(pyramid_list[i - 1]))  
    return pyramid_list  
  
def pyrBlend(img_1: np.ndarray, img_2: np.ndarray, mask: np.ndarray, levels: int) -> (np.ndarray,  
np.ndarray):  
    """  
    Blends two images using PyramidBlend method  
    :param img_1: Image 1  
    :param img_2: Image 2  
    :param mask: Blend mask  
    :param levels: Pyramid depth  
    :return: Blended Image  
    """
```

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

```
lap_pyr_img_1, lap_pyr_img_2 = laplaceianReduce(img_1, levels), laplaceianReduce(img_2, levels)
gaus_pyr_mask = gaussianPyr(mask, levels)
blended_pyr_list = [gaus_pyr_mask[i] * lap_pyr_img_1[i] + (1 - gaus_pyr_mask[i]) * lap_pyr_img_2[i]
                     for i in range(levels + 1)]
height, width = get_proper_size(img_1.shape[0], img_1.shape[1], levels)
naive_blend = mask[:height, :width] * img_1[:height, :width] + (1 - mask[:height, :width]) * img_2[:height, :width]
return naive_blend, laplaceianExpand(blended_pyr_list)
```

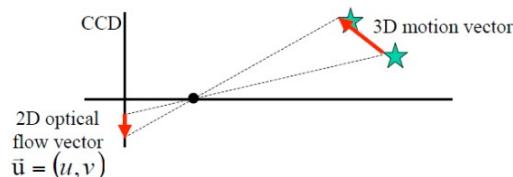
ראייה ממוחשבת ועיבוד תמונה – הרצאה 8

Optical Flow

הרעיון: לזהות את תנועת הפיקסלים בין שתי תמונות עוקבות.

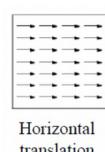
Motion Field

- Motion Field – 3D תנועה בעולם האמיתי.
- – תזוזת הפיקסלים בתמונה, שהיא תוצאה של התנועה בעולם האמיתי.

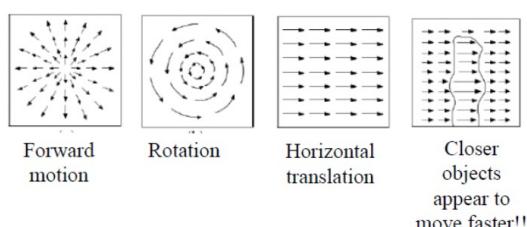


- עבור כל פיקסל יש לנו וקטור המתאר:
 - את הכיוון של התנועה.
 - את העוצמה, מהירות של התזוזה.

לדוגמה אם מזידם את המצלמה ימינה, נראה כי כל הפיקסלים זו שמאליה:



עוד דוגמאות מעניינות:



סיכום קורס ראייה ממוחשבת ועיבוד תמונה

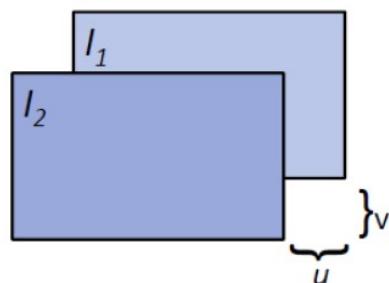
נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

Optical Flow: Brightness Constancy

הגדרת הבעיה:

- קלט: שתי תמונות עוקבות (שנלקחו בהפרש זמן קצרים).
- פלט: עבור כל פיקסל, לאן הוא זו במעבר בין התמונות.
- אם ניקח כל פיקסל קשה לדעת לאן הוא זו, لكن נדרש להניח הנחות:
 - נניח כי הצבע זהה בין התמונות הסמוכות. זה נקרא עקביות בהירות.
 - נניח שאזוריים גדולים חולקים את אותו גוון.



Optical Flow Calculation

- בהינתן שתי תמונות I_1 , I_2 . אנחנו רוצים למצוא (u , v) הממזערם את error squared:

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

למעשה, אם שני האזוריים הם זהים לחלוטין נקבל כי השגיאה שלנו היא 0. ואולם ה(u , v) שימזערו את הפונקציה הנ"ל, יהיו למעשה התנוועה של הפatz' בין הפריים. נגיד $3 = v = u$, המשמעות היא שהפatz' זו 5 ימינה ו- 3 למעלה.

- Starting from the SSD

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

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

- We can write

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

- Since $\sum I_1^2$ and $\sum I_2^2$ are almost constant, minimizing the SSD maximizes the cross-correlation $\sum I_1 I_2$

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

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

מה שעשינו זה פשטוט פתיחת סוגרים והזנחות, ההזנחות הם של הפרמטרים שלא תלויים ב x, y ,
 ה-2 גם מיותר כי זה המכפלה בקבוע. מה קיבלנו? ש $(x, y) \rightarrow \hat{I}_1(x, y) \cdot \hat{I}_2(x, y)$ זה קרוס קורלציה,
 מה שעשינו template matching.
 אם אנחנו רוצים להטגבר על מכפלה של גודל קבוע נורמל:

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

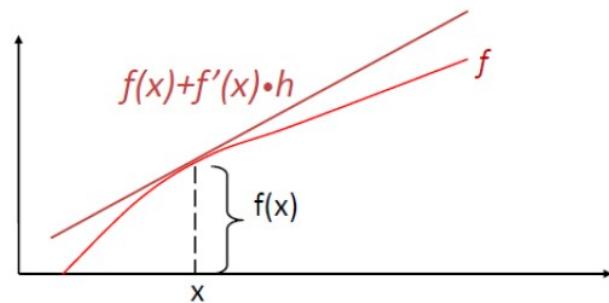
Limitation

- הנחה כי התזוזה היא של פיקסל שלם (אבל התזוזה יכולה להיות ברמת התת פיקסל).
- הסיבוכיות עולה באופן אקספוננציאלי עבור כל פרמטר שהוא מוסף:
 - Translation: (u, v) Complexity is N^2
 - Rotations: (u, v, α) Complexity is N^3
 - Zoom: (u, v, α, s) Complexity is N^4
- איך ניתן לשפר את זה?
- נוסיף הנחה נוספת: תנווה קטנה בין הפריים.
- מה זה קטנה? קטנה זה אומר פחות מפיקסל אחד.
- אם התזוזה היא קטנה ויש לנו פונקציית הפסד, אנחנו יכולים לעשות לה ליניאrizציה.

Taylor Approximation

אם אנחנו לוקחים פונקציה אנחנו יכולים לקרב את הערך המתקבל של הפונקציה בנקודה $x + h$ באמצעות הערך של הפונקציה והערך של הנגזרת כפול h ב x .

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



למה אנחנו בכלל צריכים את זה? כי אז לא נדרש לחשב את $x + h$, אלא רק את ערך הפונקציה
 ב x וערך הנגזרת ב x . ובמקרה שלנו:

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

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

optical flow — Lucas Kanade(LK)

- נזכיר אנחנו רוצים למזער את הפונקציה הבאה:

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

- לשם הפשטות נעשה פיתוח עבור פיקסל אחד (אחר כך אפשר להוסיף את הסיגמאות כי זה לינארי):

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

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

Our goal: $\min_{u,v} E(u, v)$

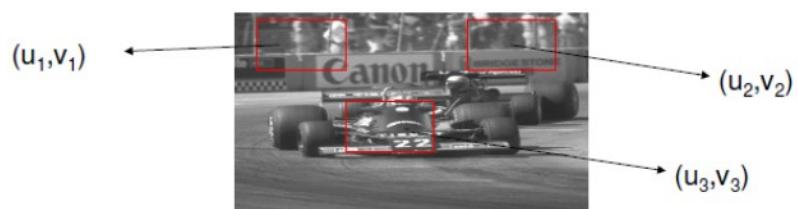
- נניח כי ה $I_x u + I_y v = -I_t$ cost function ונקבל :

- I_x : The x derivative of image I_2
- I_y : The y derivative of image I_2
- I_t : The image difference $I_2 - I_1$

נשים לב שיש לנו 2 געלמים ומשוואה 1, יש לנו אינסוף פתרונות. זה מה שגורם ל aperture problem. יש יותר מkonfigורציה אחת לתזוזה של הפיקסלים.

מה נעשה עם זה? נוסיף אילוצים נוספים.

- local smoothness assumption — השינויים בכל בלוק הם איטיים, נניח כי הבלוק זו כולו באותו אופן.



סיכום קורס ראייה ממוחשבת ועיבוד תמונה

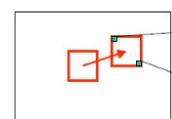
נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

- משמעותו – אותו optical flow לכל הבלוק.

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

Assume constant (u,v) in small neighborhood



$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \vdots \\ I_{x25} & I_{y25} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} I_{t1} \\ I_{t2} \\ \vdots \\ I_{t25} \end{bmatrix}$$

נניח אם אנחנו לוקחים בלוק 5X5:

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

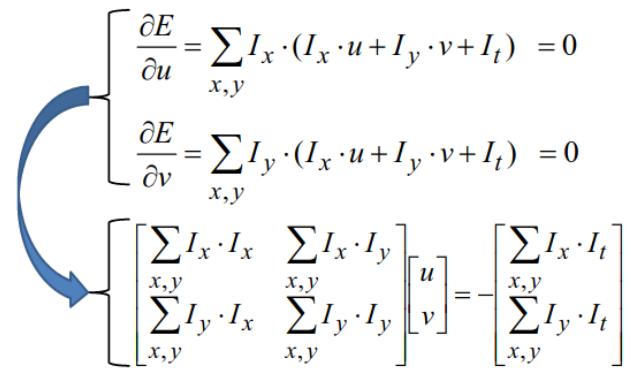
$$A \quad d \quad b$$

25×2 2×1 25×1

AIR נפתר

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

- Finding (u, v) by setting derivatives to zero:



$$\begin{aligned} \frac{\partial E}{\partial u} &= \sum_{x,y} I_x \cdot (I_x \cdot u + I_y \cdot v + I_t) = 0 \\ \frac{\partial E}{\partial v} &= \sum_{x,y} I_y \cdot (I_x \cdot u + I_y \cdot v + I_t) = 0 \end{aligned}$$

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

זהו דרך גנריית שתעבד תמיד.

נראה דרך יותר נוחה:
יש לנו יותר משוואות ממעלים זה נקרא over constrained equation

$$A \quad d = b$$

25×2 2×1 25×1

$$\longrightarrow \text{minimize } \|Ad - b\|^2$$

Solution: solve least squares problem

Python: `np.dot(numpy.linalg.pinv(A),b)`

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

נמצא נוסחה מתמטית, זה נקרא פסודו-אינברס

minimum least squares solution given by solution (in d) of:

$$(A^T A)_{2 \times 2} d = A^T b_{2 \times 1}$$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$
$$A^T A \qquad \qquad \qquad A^T b$$

- The summations are over all pixels in the K x K window

The solution matrix is $(A^T A)^{-1} A^T b$

- T solution involves the inverse of:

$$A^T A = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

When is this solvable?

- $A^T A$ should be invertible
- $A^T A$ should not be too small due to noise
 - eigenvalues λ_1 and λ_2 of $A^T A$ should not be too small
- $A^T A$ should be well-conditioned
 - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

מימוש

```
def is_solvable(A: np.ndarray) -> bool:  
    """  
    :param A: matrix to calculate eigvals on (A^T A)  
    :return: true iff lambda_1, lambda_2 > 1, and (lambda_2 / lambda_1) < 100  
    """  
  
    M = np.dot(np.transpose(A), A)  
    lamda_1, lamda_2 = np.sort(np.linalg.eigvals(M))  
    if lamda_1 > 1 and (lamda_2 / lamda_1) < 100:  
        return True  
    return False
```

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

```
def opticalFlow(im1: np.ndarray, im2: np.ndarray, step_size= 10, win_size= 5) -> (np.ndarray, np.ndarray):
    """
    Given two images, returns the Translation from im1 to im2
    :param im1: Image 1
    :param im2: Image 2
    :param step_size: The image sample size:
    :param win_size: The optical flow window size (odd number)
    :return: Original points [[x,y]...], [[dU,dV]...] for each points
    """

    I_x, I_y = convDerivative(blurImage2(im1, 5))
    I_t = np.subtract(im2, im1)
    height, width = im2.shape[:2]
    pad_len, patch_height = win_size // 2, win_size ** 2
    u_v_list, y_x_list = [], []
    for i in range(step_size, height, step_size):
        for j in range(step_size, width, step_size):
            patch_x = I_x[i - pad_len: i + pad_len + 1, j - pad_len: j + pad_len + 1]
            patch_y = I_y[i - pad_len: i + pad_len + 1, j - pad_len: j + pad_len + 1]
            patch_t = I_t[i - pad_len: i + pad_len + 1, j - pad_len: j + pad_len + 1]
            b = (-1) * patch_t.reshape(patch_height, 1)
            A = np.hstack((patch_x.reshape(patch_height, 1), patch_y.reshape(patch_height, 1)))
            if is_solvable(A):
                y_x_list.append((j, i))
                d = np.dot(np.linalg.pinv(A), b)
                u_v_list.append(d)
    return np.array(y_x_list).reshape(-1, 2), np.array(u_v_list).reshape(-1, 2)
```

הבעיה: ההנחה שהתנועה היא קטנה

- אנחנו צריכים למצוא דרך בה נוכל לחשב את התנועה גם אם היא גדולה מפיקסל אחד.
- נראה שני פתרונות לבעיה זו:
 - השיטה האיטרטיבית.
 - שימוש בפירמידות Multi-scale estimation.

Iterative LK approach

- 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) .

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

למה כדאי להשתמש בשיטה האיטרטיבית?

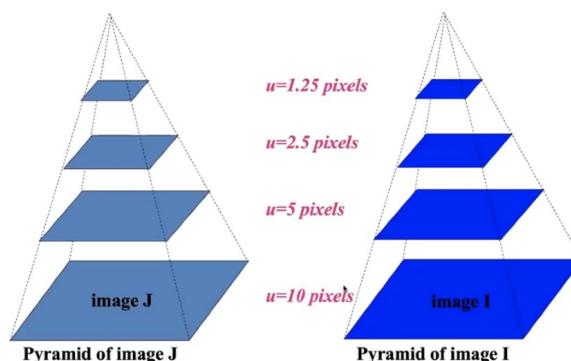
- נחשב את הנגזרות פעמי אחת בלבד.
- יש שני שלבים בכל איטרציה:
 - motion estimation
 - warping
- עובד גם במקרים בהם המOTION estimation לא כל כך טוב, כל עוד והשגיאה הולכת וקטנה במהלך האיטרציות.

מגבילות השיטה האיטרטיבית

- שימוש בשיטה האיטרטיבית מחייב כי אנו בקרבת הפתרון.
- גם בשיטה זו אם התנועה תהיה גדולה מדי יתכן כי לא נצליח להתכנס לפתרון.
- איך נוכל לשפר זאת עוד?

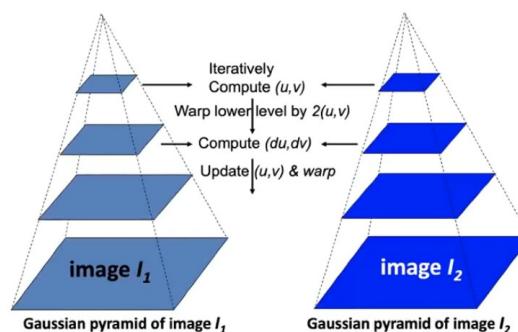
שימוש ב gaussian pyramid

Multiscale (Coarse-to-fine) Estimation



נניח שהאובייקט שלנו זו ב-10 פיקסלים. נשים לב כי בכל רמה שאנו עולים בפירמידה המרחק קטן פי חצי. וברמה הכי גבוהה התזוזה היא כבר הרבה יותר גדולה ואז נוכל להשתמש באלגוריתם שלמדנו של א.ה. כתעת נוכל לבצע את התהליך באיטרציות, ובכל פעם לחחל את הפתרון לרמה הבאה בפירמידה:

Iterative & Multiscale approach



סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

Horn-Schunck (HS) Optical Flow

- השיטה של LK היא שיטה המבוססת על אוזרים מקומיים.
- מה אם אנחנו רוצים לפטור את מציאות (u, v) של כל הפיקסלים בבת אחת?
- HS מציינים const function עם פתרון צזה.
- הם מושיעים הנחה:

 - פיקסלים שכנים זרים יחד. למשל, בהינתן (u, v) של פיקסל מסוים סביר כל לשכנים יהיה (u, v) זהה.
 - ההנחה הזאת נקראת הנחתה regularization או smoothness.

HS: cost function

- HS הגדרו פונקציית שגיאה חדשה.
- פונקציית השגיאה מ-LK תקרא data term:

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

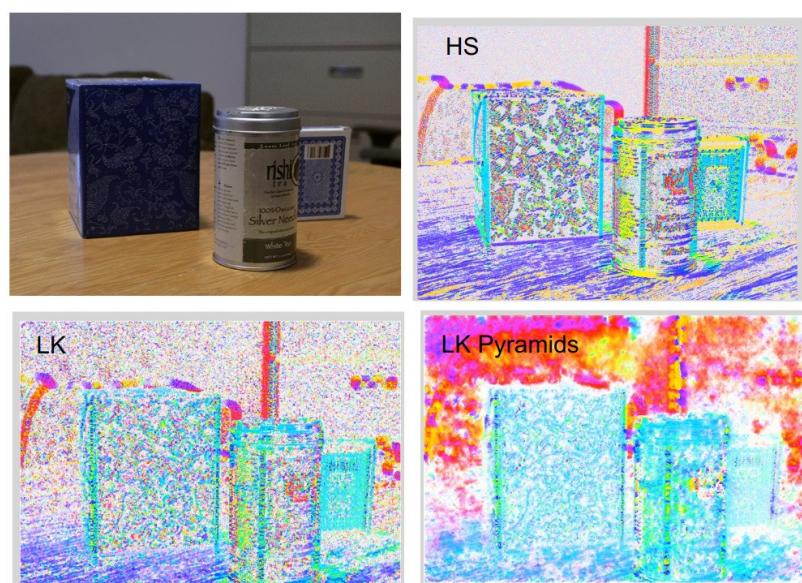
- בנוסף למינוח הקודמת הם הוסיפו לפונקציית השגיאה שלham את ה term: smoothness

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

- פונקציית הփסד הסופית שלנו שאotta אנו מנסים למצער היא:

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

- כאשר $\lambda > 0$ הוא איזון בין חשיבות term data ביחס term smoothness.



סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

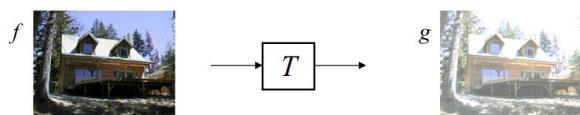
ראייה ממוחשבת ועיבוד תמונה - הרצאה 10

Image Warping



- שינוי intensities של התמונה: image filtering

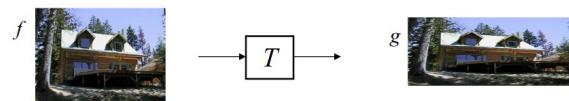
$$g(x,y) = T(f(x,y))$$



אנחנו מקבלים פונקציה (את התמונה), ומפעילים פונקציה שמשנה את ה intensities בתמונה.

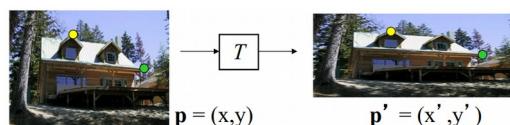
- פונקציה המפנה את הקואורדינטות של התמונה: image warping

$$g(x,y) = f(T(x,y))$$



למעשה, מה שנעשה זה להזיז פיקסלים, אנחנו נשנה את מיקומם.

סימונים:



Transformation T is a **coordinate-changing** machine:

$$p' = T(p)$$

סוג טרנספורמציות:

- local: לכל אזור בתמונה טרנספורמציה משלה. (יכול להיות גם עבר כל פיקסל)
- global: טרנספורמציה אחת עבר כל התמונה.

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

Global parametric warping

בהינתן טרנספורמציה גלובלית \mathcal{D} :

- עבור כל פיקסל בתמונה מופעלת אותה \mathcal{D} .
- \mathcal{D} נקבעת על ידי מספר פרמטרים בודדים.

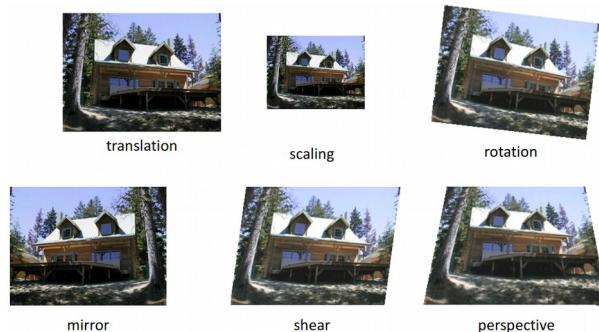
מכיוון ש- \mathcal{D} טרנספורמציה לינארית, ניתן לנאריתנית ליצג אותה באמצעות מטריצה:

$$\mathbf{p}' = \mathbf{Mp}$$

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

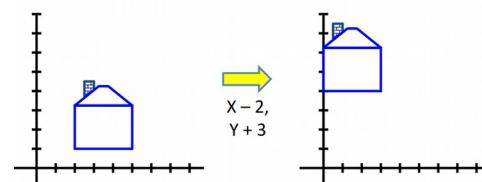
עבור כל נקודה (y, x) נפעיל את \mathbf{M} (המטריצה המייצגת את הטרנספורמציה \mathcal{D}) ונקבל מיקומים חדשים (x', y') .

סוגים של טרנספורמציות גלובליות:



translation

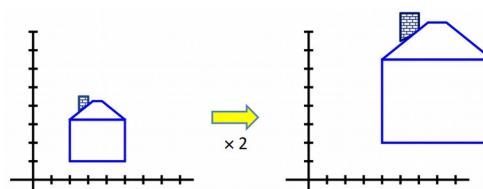
טרנסלציה של קואורדינטות משמעותה: הוספה סקלאר לכל אחד ממרכיביה.



Scaling

סקלינג עבור קואורדינטות משמעותה: הכפלת כל אחד ממרכיביו בסקלאר.

Uniform scaling: הכפלה בסקלאר קבוע עבור כל הרכיבים.

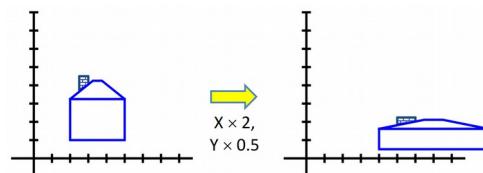


סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות ומיצגות של ד"ר גיל בן ארצי ומר שי אהרון

לא אותו סקלאר עברו כל הרכיבים. Non-uniform scaling



באמצעות מטריצה Scaling

The scaling operation:

$$x' = ax$$

$$y' = by$$

In matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{scaling matrix } S} \begin{bmatrix} x \\ y \end{bmatrix}$$

טרנספורמציות נוספות באמצעות מטריצות

2D Identity:

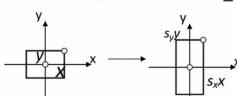
$$\begin{aligned} x' &= x \\ y' &= y \end{aligned} \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D Rotate around (0,0):

$$\begin{aligned} x' &= \cos \Theta \cdot x - \sin \Theta \cdot y \\ y' &= \sin \Theta \cdot x + \cos \Theta \cdot y \end{aligned} \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \Theta & -\sin \Theta \\ \sin \Theta & \cos \Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

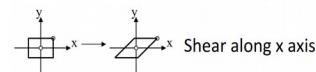
2D Scale around (0,0):

$$\begin{aligned} x' &= s_x \cdot x \\ y' &= s_y \cdot y \end{aligned} \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



2D Shear:

$$\begin{aligned} x' &= x + sh_x \cdot y \\ y' &= sh_y \cdot x + y \end{aligned} \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



2D Mirror over the Y axis:

$$\begin{aligned} x' &= -x \\ y' &= y \end{aligned} \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D Mirror over (0,0):

$$\begin{aligned} x' &= -x \\ y' &= -y \end{aligned} \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

מה לגבי טרנסלציה ? האם קבועה מטריצה 2×2 המיצגת טרנסלציה ?
 לא!

2D translation is not a linear transformation from \mathbb{R}^2 to \mathbb{R}^2 !
(e.g. it is not homogenous)

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

תכונות של טרנספורמציות לינאריות

- מקור ממופה למקור: $(0,0)$ ממופה ל $(0,0)$.
- קווים ממופים לקווים.
- קווים מקבילים ממופים לקווים מקבילים.
- היחס נשמר לאורקן הקווים.
- סגירות תחת הרכבה (הפעלת מספר טרנספורמציות לינאריות – טרנספורציה לינארית).
- פשוט למצאו את המטריצה ההופכית (כשיש צזו).

קואורדינטות הומוגניות

האם נוכל לייצג טרנסלציה באמצעות מטריצה 3×3 ?

כן, נשתמש בקואורדינטות הומוגניות.

ניצג נקודה בדו-מימד באמצעות וקטור תלת-מימדי:

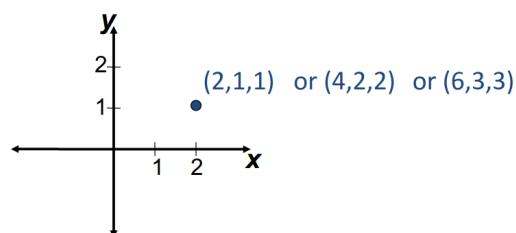
$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \longrightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

כפל בסקלאר לא משנה את הנקודה, נשים לב שככל הוקטוריים הבאים ימופו לאותו וקטור דו-מימד: (y, x)

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \cong \begin{pmatrix} 2x \\ 2y \\ 2 \end{pmatrix} \cong \begin{pmatrix} 3.5x \\ 3.5y \\ 3.5 \end{pmatrix} \cong \begin{pmatrix} \lambda x \\ \lambda y \\ \lambda \end{pmatrix}$$

מעבר בין קואורדינטות הומוגניות לבין 2D point

- (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



סיכום קורס ראייה ממוחשבת ועיבוד תמונה

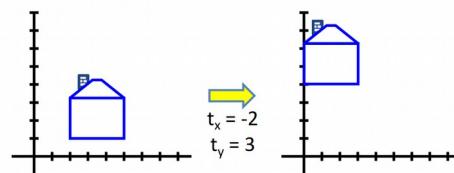
נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

Translation matrix

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$

Homogeneous Coordinates



על מנת לשמר על איחדות ניתן לייצג את כל המטריצות שהראנו קודם לכן בדו-מימד כמטריצות 3x3:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \Theta & -\sin \Theta & 0 \\ \sin \Theta & \cos \Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotate

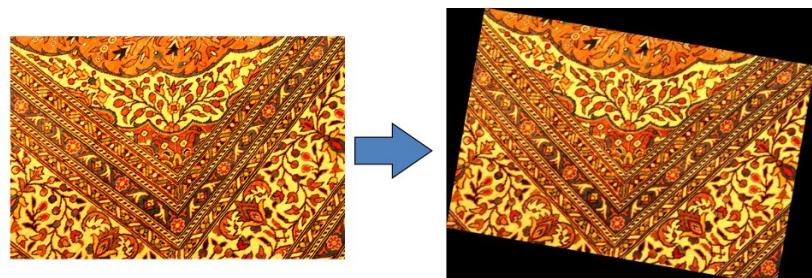
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Shear

Rigid transformation

שילוב של טרנסלציה ורטציה

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & t_x \\ \sin(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$



Parameters: 3 (t_x, t_y, θ)

Preserves all information except of orientation (lengths, angles etc.)

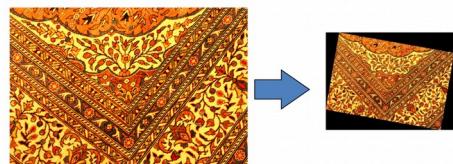
סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

Similarity transformation

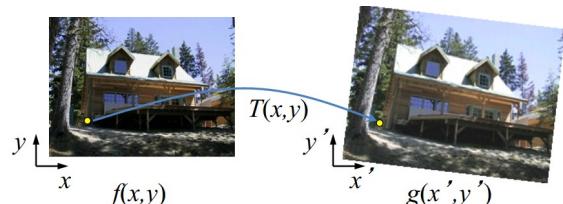
שילוב של טרנסלציה, רוטציה | uniform scaling



- # Parameters: 4 (s, t_x, t_y, θ)
- Preserves proportions and angles, but not lengths and orientation

Forward warping

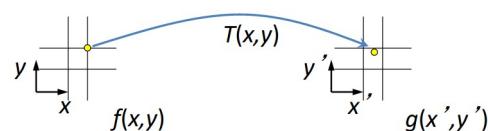
געשה העתקה מהתמונה המקורי לתמונה החדשה, עבור כל פיקסל בתמונה המקורי נחשב את פיקסל היעד בתמונה החדש. עבור כל intensity בתמונה המקורי נעתיק אותו למיקום החדש בתמונה החדש.



שליחת כל פיקסל (x, y) למיקום $T(x, y)$ שהטרנספורמציה ממחפה אותו:

$$(x', y') = T(x, y)$$

מה געשה במידה והטרנספורמציה ממחפה פיקסל מסוים להיות בין שני פיקסלים ?



מה געשה עם "חורים" שעלוולים להיווצר בתמונה הממופה ?

- A: distribute its color among the neighboring pixels of (x', y')
– Known as “splatting”

מכיוון שהעתקה לא מבטיחה שייצאו מספרים שלמים תמיד יכול להיווצר מצב שנמופה לערך פיקסל לאשלם, במקרה זהה נבצע אינטרפולציה.

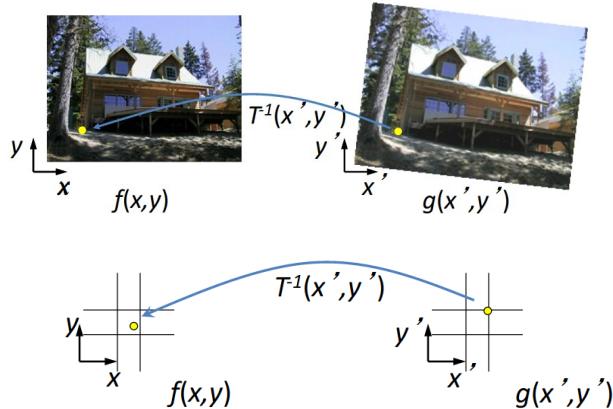
סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

Backward (inverse) warping

הפעם נעשה את זה הפוך ממקודם. מה זה אומר? נעבור על כל הפיקסלים בתמונה היעד ונמצא את המיקום שלו בתמונה המקורי באמצעות הטרנספורמציה ההפוכה.



Get each pixel $g(x',y')$ from its corresponding location $(x,y) = T^{-1}(x',y')$ in the first image

Q: what if pixel comes from “between” two pixels?

A: *Interpolate* the color value from its neighbors

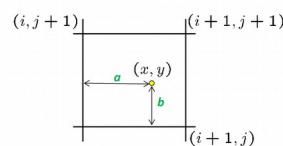
- nearest neighbor, bilinear, bicubic, Gaussian

גם בשיטה הזאת עלולה להיות אותה הבעיה כמו בשיטה הקודמת. נמצא פיקסל מסוים בתמונה היעד ונמצא את הפיקסל המתאים לו באמצעות הטרנספורמציה. יתכן כי קיבל ערך שהוא לא מספרשלם. גם כאן נפתרו זאת באמצעות אינטראפולציה.

bilinear interpolation

הרעיון הוא לחשב את ערך האינטנסיטי לפי השכנים שלן. הערך יקבע לפי המרחק היחסי מהשכנים.

Sampling at $f(x,y)$:



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

Foward VS. Inverse warping

איזו שיטה יותר טובה ?

בדרך כלל משתמש ב inverse warping, השיטה הזאת בדרך כלל מביאה תוצאות יותר טובות, ו金陵 יותר להימנע מ"חורים" בתמונה היעד.

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

Camera Model

אובייקטים עשויים להראות שונה בתמונות שלנו מאשר באמת נראהים ב"עולם האמיתי". זה עשוי לקרות מכיוון שבתמונות דו-ממדיות אנחנו מפסידים את הממד השלישי, ממד העומק.

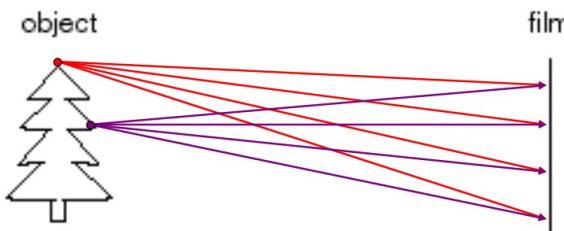


Image information

איך מיצרים תמונה ?

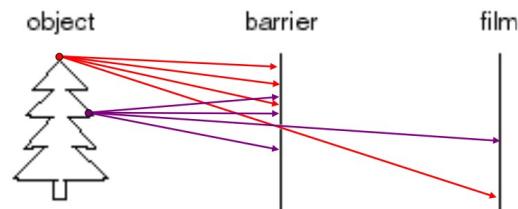
נניח ואנחנו רוצים לבנות מצלמה. איך علينا לעשות זאת ?

ניסיון ראשון: נשים פשוט את הפילם אל מול האובייקט.



לא נקבל תמונה הגיונית מאחור וקרניים מואתו מקום באובייקט מגיעות למקומות שונים בפイルם, מה שגורת לטשטוש.

ניסיון שני: נוסיף מחסום לפני הפילם, שיחסום את רוב הקרןאים.



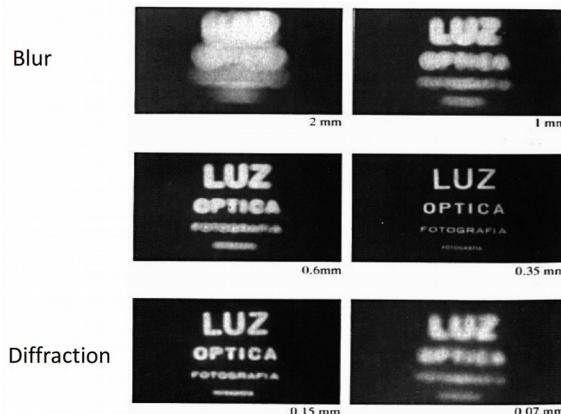
הוספה המחסום תגרום להפחחת הטשטוש.
 הפתח במחסום נקרא צמצם (aperture)

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

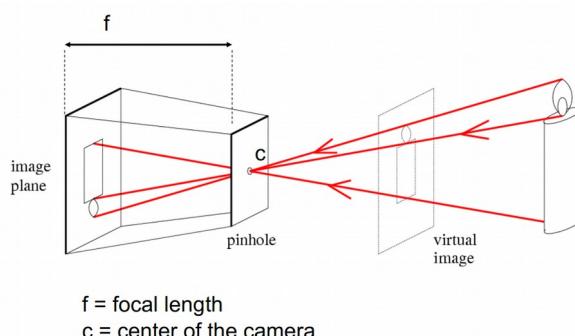
מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

השפעת הצמצם



במידה והפתח של הצמצם גדול מדי קיבל תמונה מטושטשת, ואם הוא קטן מדי אז יחדרו מעט קרני אור אל הפילם ושוב מקבל תמונה מטושטשת.

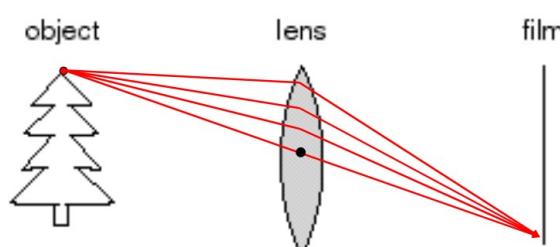
Pinhole camera – no lens



f = focal length
 c = center of the camera

מצלמות עם עדשות = פוקוס

בנוסף העדשות למצלמות ממוקדות את קרני האור לפילם: העדשות יגרמו לזה שקרנים שונים שייצאו מאותה נקודה יצטלבו באותה נקודה על הפילם.



התוצאה: מקבל תמונה חדה יותר.

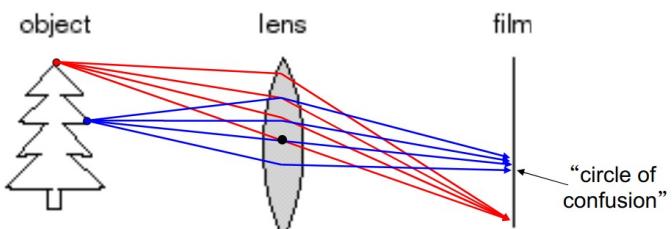
סיכון קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

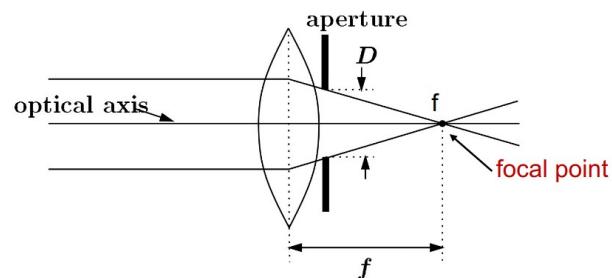
מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

מגבלות העדשה

העדשה ממקדת את קרני האור בהגעתם לפילם. אך, נקודות הנמצאות המרחקים השונים יהיו מוחז לミוקוד.



Lens concept



: focal point

• הנקודה בה קרניים מקבילות מצטלבות.

• המיקום בו הקרניים המקבילות מצטלבות עם הציר האופטי.

• F: המרחק של ה focal point מהעדשה.

: Aperture

• D: קוטר החור המגביל את טווח הקרניים.

• יכול להיות משני הצדדים.

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

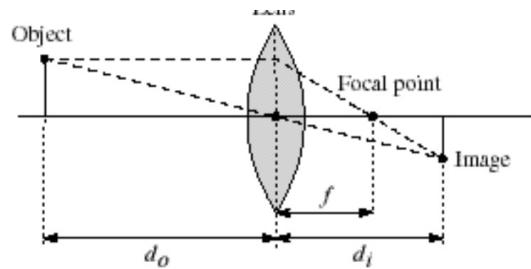
נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

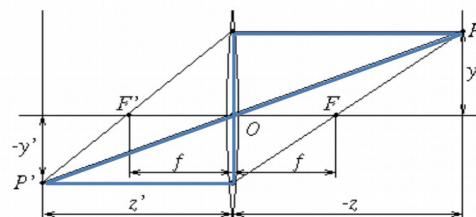
ראייה ממוחשבת ועיבוד תמונה - הרצאה 11

Focus

פוקוס: קרניים שונות מאותה נקודה נפגשים באותו מרחק.

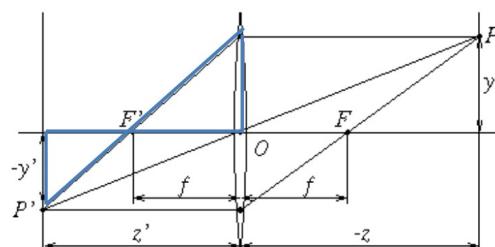


איך אנחנו יכולים למצוא את הפוקוס ?
 מה יחס בין הפוקל פוינט לפוקוס ?
 מה לגבי עובי העדשה ? נניח שיש לנו עדשה דקה .



$$\frac{y}{-y'} = \frac{-z}{z'}$$

$$P' = (-y', z'), P = (y, -z)$$



$$\frac{-f}{z'-f} = \frac{y}{-y'} = \frac{-z}{z'}$$

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

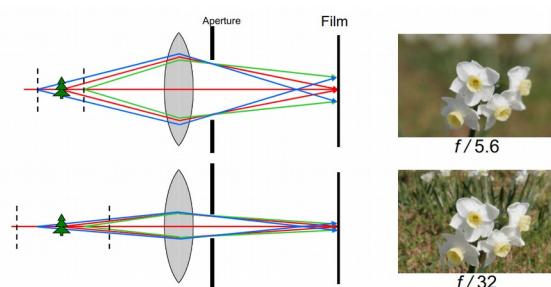
נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

Depth of field

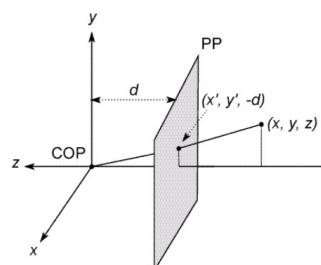
שינוי גודל הצמצם משפיע על האזוריים שיהיו בפוקוס בתמונה.

- צמצם קטן יותר מגדיל את הטווח בו האובייקט נמצא בפוקוס.



Perspective projection

נרצה שתהיה לנו היכולת לחשב עברו נקודה בעולם لأن היא ממופת בתמונה.
 מודל הקואורדינטות: משתמש במודל של hole-hole-pin כמודל קירוב.
 נשים את ה (optical center) (center of projection) בראשית הצירים.
 נשים את מרחב התמונה (projection plane) לפני ה COP. למה?
 על מנת להימנע מקואורדינטות שליליות.



:Projection equations

- חשב את המפגש בין ה PP לבין הקמן (z, y, x) ל COP.
- ניתן לחשב זאת באמצעות משולשים דומים:

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

- אנחנו יכולים לקבל את המיקום על התמונה (x', y') על ידי הצלמת מהרכיב השלישי (משור התמונה הוא דו-מימדי):

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

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

קואורדינטות הומוגניות

איך מmirים מרחב התמונה מהעולם האמיתי לקואורדינטות הומוגניות:

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous image coordinates

$$(x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

homogeneous scene coordinates

איך מmirים מקואורדינטות הומוגניות של מרחב התמונה והעולם האמיתי ללא הומוגניות:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$
$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

Perspective projection

ההקRNA היא טרנספורמציה מנקודת תלת-ממדית לנקודת דו-ממדית על ידי הכפלת מטריצה ושימוש בקואורדינטות הומוגניות:

$$\begin{bmatrix} -d & 0 & 0 & 0 \\ 0 & -d & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & & & \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} -dx \\ -dy \\ z \\ 1 \end{bmatrix} \Rightarrow \left(-d \frac{x}{z}, -d \frac{y}{z} \right)$$

divide by third coordinate

זה מכונה הטלה פרספקטיבית, ומטריצת היא מטריצת ההקRNA שלנו.

דוגמה:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \\ 1 & & & \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/2 & 0 \\ 1 & & & \end{bmatrix} \begin{bmatrix} 10 \\ 6 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 10 \\ 6 \\ -2 \\ 1 \end{bmatrix}$$

$\Rightarrow x' = -5, y' = -3$

2. Object point at (25, 15, 10)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \\ 1 & & & \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/2 & 0 \\ 1 & & & \end{bmatrix} \begin{bmatrix} 25 \\ 15 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 25 \\ 15 \\ -5 \\ 1 \end{bmatrix}$$

$\Rightarrow x' = -5, y' = -3$

בדוגמה ניתן לראות שתי נקודות המופיעות אותה נקודה במישור התמונה.
 איך דבר זה יכול לקרות?

זה יכול לקרות אם שתי הנקודות ישבות על אותה קרן. כל הנקודות שישובות על אותה קרן
 במרחב, ימופו אותה נקודה במישור התמונה.

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

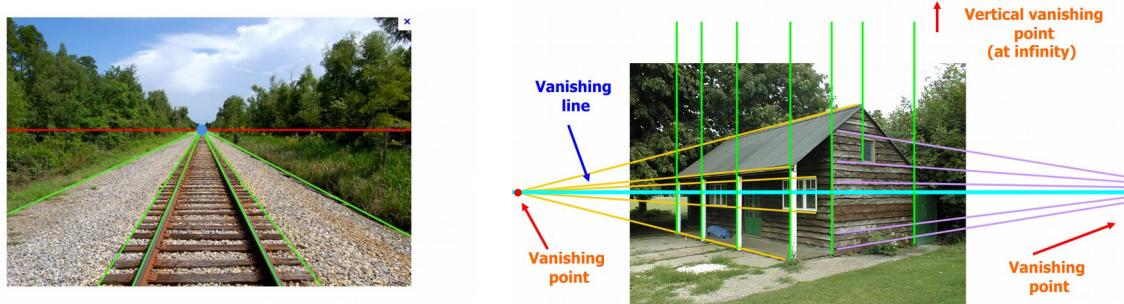
תכונה נוספת של הטלה היא שהיא לא משפעת מ scaling. כלומר, הכפלת בסקלאר לא משנה את נקודת המיפוי במרחב התמונה.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z/d \\ 1 \end{bmatrix} \Rightarrow (-d\frac{x}{z}, -d\frac{y}{z})$$

$$\begin{bmatrix} -d & 0 & 0 & 0 \\ 0 & -d & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} -dx \\ -dy \\ z \\ 1 \end{bmatrix} \Rightarrow (-d\frac{x}{z}, -d\frac{y}{z})$$

Vanishing Point

אוסף של קוויים מוגבלים בעולם, נפגשים באותה נקודה בתמונה. הם נפגשים בנקודה הנקראת vanishing point שנמצאת על הקו שנראה jako vanishing line. זה קורה בגל שבתמונה אין לנו את הממד של העומק שיש בעולם האמיתי, הקווים נפגשים באופק.

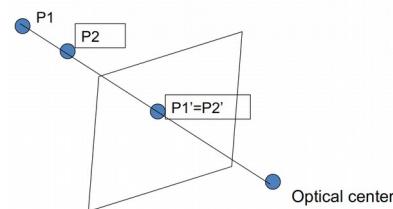


Parallel lines in the world intersect in the image at a "vanishing point"

Stereo

מדו צרי מס' נקודות מבט

מבנה ועומק הם מטבע הדברים מעורפלים כאשר יש לנו נקודות מבט אחת בלבד.



כמו שראינו גם קודם לכן, כדי לנו נקודות ייחוס אחת, הנקודות הנמצאות על אותה קרן ממופות לאותה נקודה במרחב התמונה. וכך אנו למעשה משבדים את הממד השלישי, העומק.

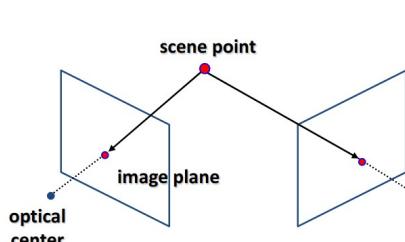
סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

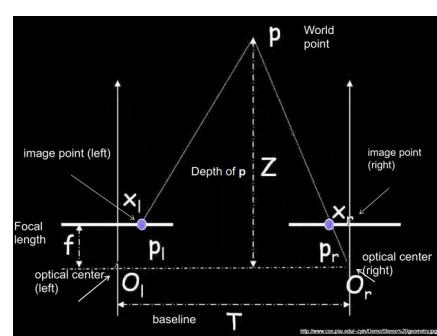
מבוסס על הרצאות ומיצגות של ד"ר גיל בן ארצי ומר שי אהרון

באמצעות סטריאו אנו יכולים לפתור את הבעיה הזאת, ולהצליח לזהות את העומק של האובייקטים בעולם האמיתי.

- Estimating depth with stereo
 stereo:shape from "motion" between two views
- - עלינו לחת בחשבון:
 - מידע על מיקום המצלמה.
 - אותה נקודה שאנו בודקים נמצאת בשתי התמונות.



אנו מניחים כי הגובה של הנקודת זהה בשתי התמונות. כלומר, קואורדינטת $-y$ זהה.



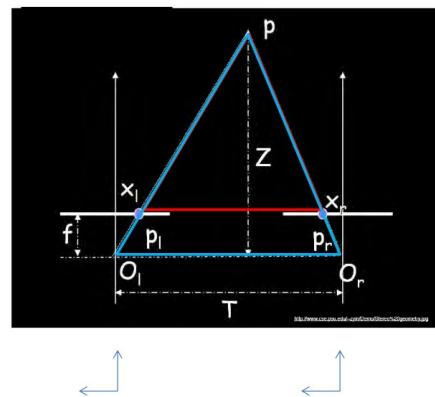
המטרה: למצוא את Z שהוא העומק של האובייקט P . נעשה זאת על ידי 2 תמונות, X_l ו- X_r .
 מניחים שהצירים האופטיים הם מקבילים (מיקום המצלמות באותו גובה) ובגלל שהם ניצבים למשטח התמונה אז גם המשטחים מקבילים.
 F-focal length: המרחב בין המצלמה למישור התמונה.

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

נשים לב שיש פה דמיון משולשים (הכחול והאדום):



ולכן אנחנו יכולים לחלץ את Z לפי התכונות של דמיון משולשים:

Similar triangles (p_l, P, p_r) and (O_l, P, O_r)

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

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

disparity → $x_r - x_l$

depth from disparity

התוצאה המתבקשת מיחסור המיקומים של האובייקט בשתי התמונות. **disparity**: תמונה המציגת את ה disparity של כל הפיקסלים. **Disparity map**

image $I(x,y)$



Disparity map $D(x,y)$

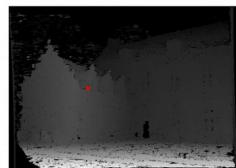


image $I'(x',y')$



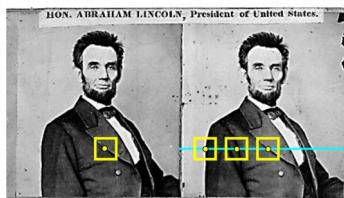
$$(x', y') = (x + D(x,y), y)$$

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

Stereo basic algorithm



- Assume each scanline in the left image is the corresponding scanline in the second image
- For each pixel x in the first image
 - Find corresponding scanline in the right image
 - Examine all pixels on the scanline and pick the best match x'
 - Compute disparity $x - x'$ and set depth(x) = $f * T/(x - x')$

אלגוריתם:

מקבלים שתי תמונות ונוקודה, מניחים שהנקודה נמצאת על אותו קו – אותו scanline בשתי התמונות. המטרה שלנו היא למצוא את מיקום הנוקודה גם בתמונה שנייה. נעבור על אותו קו בתמונה השנייה עד שנמצא את אותה נוקודה. מחשבים את ההפרש בין המיקומים ויוצרים את תמונה העומק באמצעות t , f . על מנת למצוא את הנוקודה עצמה נkeh חלון סביב הנוקודה, ונעבור על ההתאמות בקוו באמצעות אחות הפונקציות המוצעות למטה, למשל NCC.

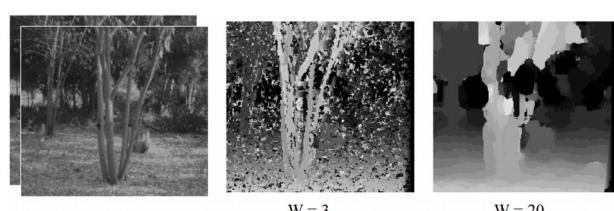
שיטות לביצוע הסריקה והותצאות שלהם:

Similarity Measure	Formula
Sum of Absolute Differences (SAD)	$\sum_{(i,j) \in W} I_1(i,j) - I_2(x+i, y+j) $
Sum of Squared Differences (SSD)	$\sum_{(i,j) \in W} (I_1(i,j) - I_2(x+i, y+j))^2$
Zero-mean SAD	$\sum_{(i,j) \in W} I_1(i,j) - \bar{I}_1(i,j) - I_2(x+i, y+j) + \bar{I}_2(x+i, y+j) $
Locally scaled SAD	$\sum_{(i,j) \in W} I_1(i,j) - \frac{\bar{I}_1(i,j)}{\bar{I}_2(x+i, y+j)} I_2(x+i, y+j) $
Normalized Cross Correlation (NCC)	$\frac{\sum_{(i,j) \in W} I_1(i,j) I_2(x+i, y+j)}{\sqrt{\sum_{(i,j) \in W} I_1^2(i,j) \cdot \sum_{(i,j) \in W} I_2^2(x+i, y+j)}}$

SAD SSD NCC Ground truth

השפעת גודל החלון

- חלון קטן:
 - יוצר פרטימ
 - יוצר רעש
- חלון גדול:
 - disparity map חלקה יותר
 - פחות פרטים



W = 3

W = 20

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

ראייה ממוחשבת ועיבוד תמונה - הרצאה 12

Homography



נלמד היום עוד שני סוגי של טרנספורמציות:



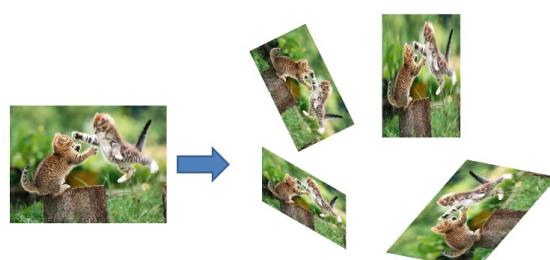
affine



Projective

Affine transformation

זהה טרנספורמציה המשלבת בין מספר טרנספורמציות שונות.



Affine transform (6 DoF) = translation + rotation +
scale + aspect ratio + shear

תחת טרנספורמציה אפנית קווים מקבילים נשארים קווים מקבילים.
 יש לטרנספורמציה 6 דרגות חופש והיא מוגדרת על ידי המטריצה הבאה:

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

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

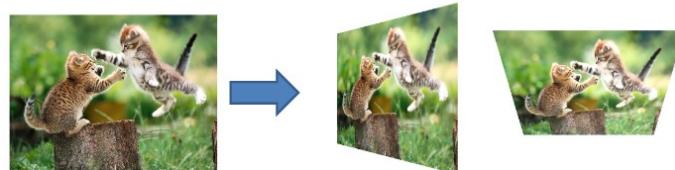
נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

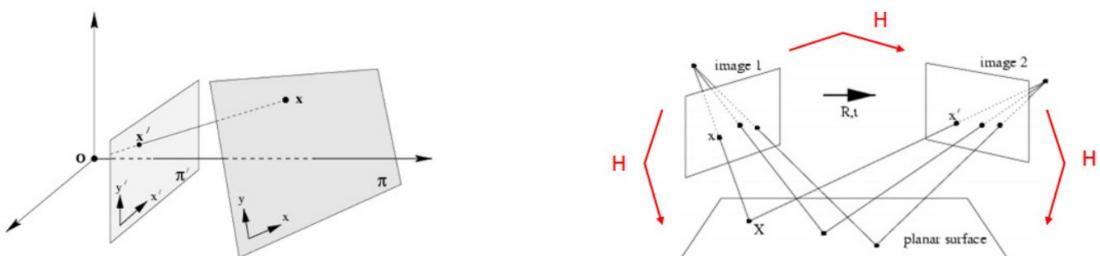
Projective transformation (Homography)

זהוי הטרנספורמציה הכללית, והכי חזקה.

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad x' = u/w \\ y' = v/w$$



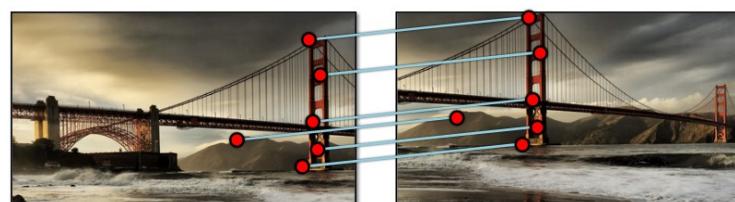
הטרנספורמציה המדמה איך המצלמה רואה משטחים בעולם תחת עיוותים שונים. הטרנספורמציה מוסיפה בין פיקסלים שצולמו על ידי אותה מצלמה בשני מישורי תמונה שונים.



טרנספורמציות 2D

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

חישוב טרנסלציה

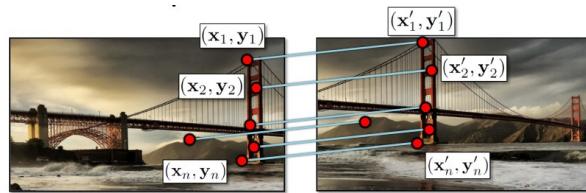


ברכינו למצוא את התזוזה בין התמונות.

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון



יתכן שיש הפרשים בין הפרשי הנקודות בגלל שהתמונה טיפה שונות (רעשים, הפרשי גוון, בהירות), נקח את ממוצע הפרשים:

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

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

נרצה פתרון כללי:

בעיה: יש לנו שני משתנים ($\mathbf{x}_t, \mathbf{y}_t$) מכיוון שבטרנסלציה יש לנו 2 מימדי חופש.

יש לנו יותר אלומות מכמהות המשתנים ($N > 2$).

פתרון: נשתמש ב least squares formulation

Least squares formulation

For each point $(\mathbf{x}_i, \mathbf{y}_i)$

$$\mathbf{x}_i + \mathbf{x}_t = \mathbf{x}'_i$$

$$\mathbf{y}_i + \mathbf{y}_t = \mathbf{y}'_i$$

we define the *residuals* as

$$r_{\mathbf{x}_i}(\mathbf{x}_t) = (\mathbf{x}_i + \mathbf{x}_t) - \mathbf{x}'_i$$

$$r_{\mathbf{y}_i}(\mathbf{y}_t) = (\mathbf{y}_i + \mathbf{y}_t) - \mathbf{y}'_i$$

יעד: מזעור ריבועי השאריות.

דרך פתרון: least squares formulation

בטרנסלציה הפתרון שווה למציאת הממוצע.

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

$$\mathbf{A}\mathbf{t} = \mathbf{b}$$

Find \mathbf{t} that minimizes

$$\|\mathbf{A}\mathbf{t} - \mathbf{b}\|^2$$

To solve, form the *normal equations*

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

$$\mathbf{A}_{2n \times 2} \quad \mathbf{t}_{2 \times 1} = \mathbf{b}_{2n \times 1} \quad \mathbf{t} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}$$

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

טרנספורמציה אפינית

יש לנו 6 דרגות חופש ولكن נצטרך 3 נקודות (6 משוואות).

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

המקרה הכללי של טרנספורמציה אפינית:

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

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

solving for homographies

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

למה $h_{22} \neq 1$?
 בהומוגרפיות אנחנו לוקחים קואורדינטות הומוגניות, לכן נוכל לחלק תמיד בסקלאר. ניתן לחלוק את כל המטריצה ב h_{22} וזה קיבל שם 1.
 על מנת לפתור את המשווה נצטרך 4 נקודות מתאימות, 8 משוואות על מנת למצוא את הפרמטרים של ההומוגרפיה.

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

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

Why the division?

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

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

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

ניתן להציג את כל המשוואות בצורה הבאה:

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

הבעה שלנו היא שכאן לא נוכל להשתמש בפסודו-אינברס, כי אז נקבל את הפתרון הטריויאלי. נctrar לה שימוש במשהו אחר.

Direct linear Transforms (DLT)

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 x_1 & -x'_1 y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 x_1 & -y'_1 y_1 & -y'_1 \\ & & & & & : & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\mathbf{A}_{2n \times 9} \quad \mathbf{h}_9 \quad \mathbf{0}_{2n}$$

Defines a least squares problem: $\text{minimize } \| \mathbf{Ah} - \mathbf{0} \|^2$

- Since \mathbf{h} is only defined up to scale, solve for unit vector $\hat{\mathbf{h}}$
- Solution: $\hat{\mathbf{h}} = \text{eigenvector of } \mathbf{A}^T \mathbf{A} \text{ with smallest eigenvalue}$
- Works with 4 or more points

מהמטריצה המתתקבלת מ $\mathbf{A}^T * \mathbf{A}$ נמצאו וקטורים עצמיים, שלכל אחד מהם יש ערך עצמי נקיח את הוקטור העצמי המתאים לערך העצמי הקטן ביותר, זהה הפתרון למשווהה.

אלגוריתם:

- Given $n \geq 4$ 2D to 2D point correspondences $\{x_i \leftrightarrow x'i\}$
- Determine the 2D homography matrix H such that $x'i = H^*x_i$.
 - For each correspondence $x_i \leftrightarrow x'i$ compute the two first rows of matrix A_i from DLT slide. Only the first two rows need be used in general.
 - Assemble the $n 2 \times 9$ matrices A_i into a single $2n \times 9$ matrix A .
 - Obtain the SVD of A . The unit singular vector corresponding to the smallest singular value is the solution h . Specifically, if $A = UDV^T$ with D diagonal with positive diagonal entries, arranged in descending order down the diagonal, then h is the last column of V .

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

- קיבל שני תמונות (משתחים)
- קיבל 4 נקודות שנמצאות ב 2 התמונות.
- נבנה את המטריצה A.
- מוצאים את SVD של A.
- נלקח את העמודה האחורונה של V (9 ערכים).

Objective	Measure Quality of F
	Given a matrix A with at least as many rows as columns, find x that minimizes $\ Ax\ $ subject to $\ x\ = 1$.
Solution	x is the last column of V, where $A = UDV^T$ is the SVD of A.

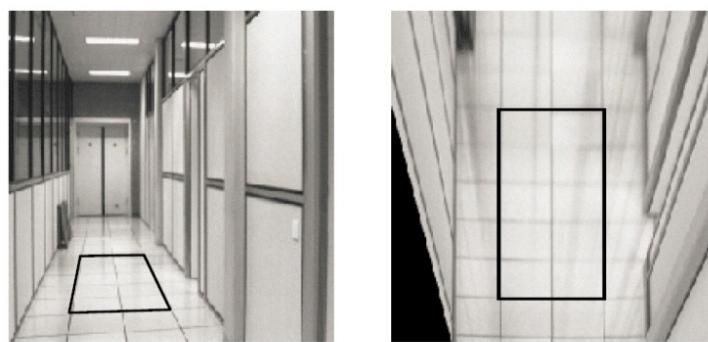
Planar image rectification

מטרה:

- קלט: תמונה אחת עם משטח מישורי מעוות.
- פלט: משטח מישורי ישר ומלבני.

אלגוריתם:

- נלקח את 4 הפינות של התמונה המעוותת.
- נפתרו הומוגרפיה.
- נמפה פיקסלים מהתמונה המעוותת לישרה.



from Hartley & Zisserman

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

Image Stitching

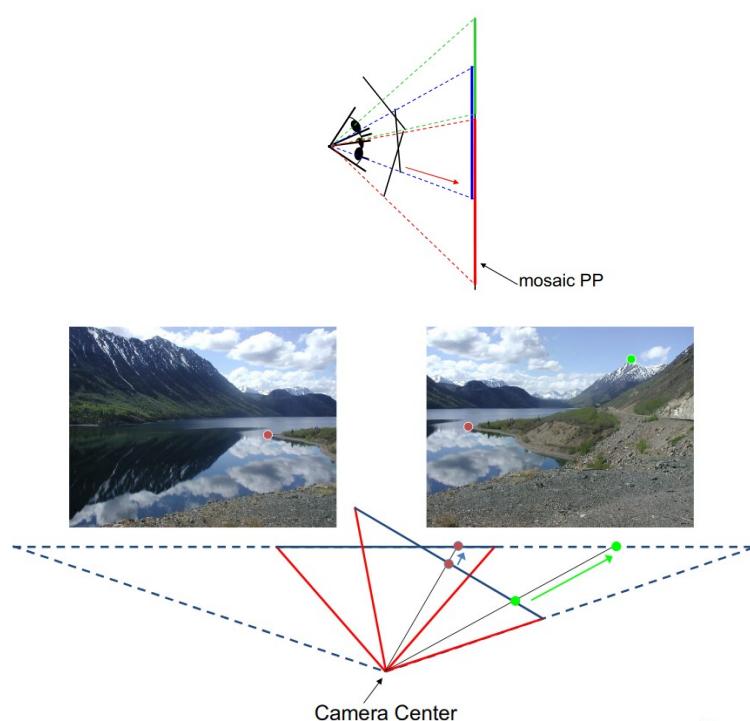
שילוב של שתי תמונות או יותר עם אזור חפיפה ליצירה אחת גדולה, פנורמה.



איך עושים את זה ?

- מצלמים מספר תמונות עם אזור חפיפה.
- מחשבים את הטרנספורמציה בין התמונות (מושגים לפחות 4 נקודות חופפות על מנת שנוכל לחשב הומוגרפיה).
- Wrap the second image to overlap with the first .
- אם יש תמונות נוספות נחזיר על התהילה.

מה שעשינו בעצם זה מיפוי התמונות לאוטו משטח (של התמונה הראשונה שחיישנו)



סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

Camera Pose and Focal length

נלמד איך למצוא את הפרמטרים של המצלמה כך שאם נקבל פרמטרים של אובייקטים בתלת-ממד נוכל להכפיל במטריצת המצלמה ולמצוא את המיפוי בתמונה.

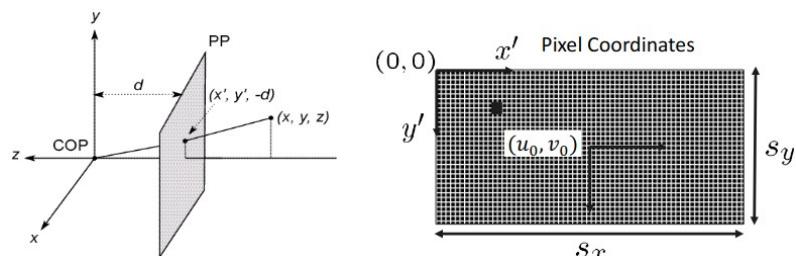


נמצא את מטריצת המצלמה שטאפשר לנו לתקן עיוותים ולשערך את התמונה המתקבלת על ידי מתן מיקומי אובייקטים חיצוניים.

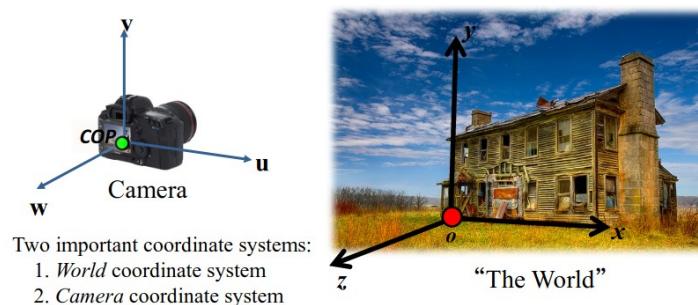
פרמטרים פנימיים של המצלמה

הנחות:

- אמצע מישור התמונה הוא $(0, 0)$.
- היחס בין ציריו התמונה הוא 1.
- כל פיקסל הוא ריבוע.



פרמטרים חיצוניים של המצלמה



סיכום קורס ראייה ממוחשבת ועיבוד תמונה

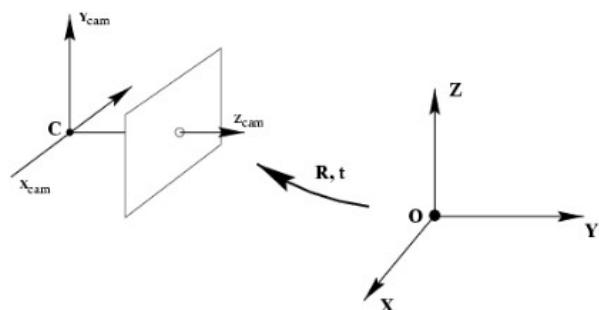
נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומיר שיאהרין

כדי למפות אובייקטים מהעולם האמיתי לתמונה נוצר לדעת את מיקום המצלמה בעולם האמיתי, נוצר לדעת את המערכת הצירים של העולם, ושל המצלמה.

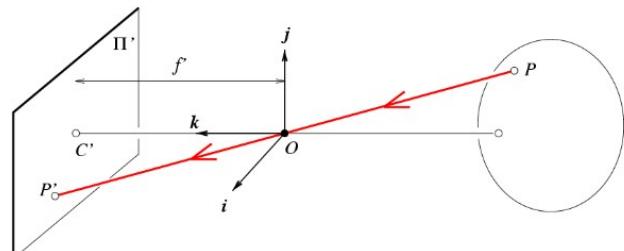
World to camera

כדי למצוא מיפוי של נקודה (z, y, x) בקואורדינטות העולם לקואורדינטות של המצלמה משתמש ב rigid transformation (translation+ rotation) (translation+ rotation). תהיה לנו מטריצה אחת שתעשה את ההמרה,vr
שנשתמש בקואורדינטות הומוגניות לנקודה בעולם $(1, z, y, x)$.



פרמטרים פנימיים של המצלמה

איך אנחנו נמצא את הפרמטרים ונרכיב מהם מטריצה?



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

נציג נקודה בעולם באמצעות וקטור הומוגני. יש לנו \mathbf{P} נקודה בעולם שאנו רוצים למפות למישור התמונה פאי טאג, f נתון לנו.

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

זה היה בהנחה שאנו יודעים שהמיקום של הנקודה המרכזית של מערכת הצירים של המצלמה הוא ב 0,0. מה נעשה אם זה לא המצב? נסיף פרמטרים של טרנסלציה:

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

ומה עם ההנחה שהפיקסל היא ריבועי? ניתן להתגבר גם על זה באמצעות scale:

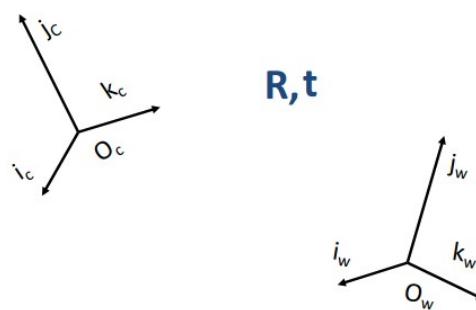
$$\mathbf{x} = \mathbf{K}[\mathbf{I} \quad \mathbf{0}] \mathbf{X} \rightarrow w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

אם יש איזושי סטייה במערכת הצירים:

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

כל הפרמטרים האלה מאפשרים לנו למפות אובייקטים מנוקדות בעולם למישור המצלמה.

כדי להמיר מערכת צירים של העולם לשול המצלמה:



נדרש להשתמש ברטוציה וטרנסלציה. קיבל את המטריצה הבאה:

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומיר שיאהרוני

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

כדי למצוא את הפרמטרים האלה בפעם הראשונה נצטרך לצלם משהו שהוא יודע אם אין הוא נראה.

The camera matrix is

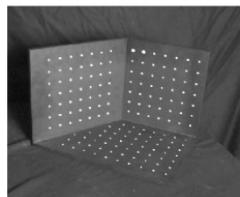
$$\mathbf{x} = \begin{bmatrix} wx \\ wy \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \Pi \mathbf{X}$$

- The projection matrix models the cumulative effect of all parameters
- Useful to decompose into a series of operations

$$\Pi = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \mathbf{X}$$

intrinsic

- 3D points coordinates
 - We can set the (0,0,0) of the world coordinates as we wish
 - All points must not be coplanar
 - Homogenous coordinates -> we can use relative coordinates for each point



$$\begin{bmatrix} wu \\ wv \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- Solve using linear least squares

$$\begin{bmatrix} wu \\ wv \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

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

Ax=0 form

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

ראייה ממוחשבת ועיבוד תמונה – השלמה חומר שנלמד רק בתרגולים

Key Point

מה זה key point ?

- זהו פיקסל או איזור מסוים בתמונה שהוא "יחודי וד'" קל למצאו אותו.

למה צריך אותם בכלל ?

- על מנת לזהות מזוהה גדולה ומורכבת בין תמונות.
- חיבור בין תמונות.
- פנורמה

מה הופך נקודה מסוימת ל key point טוב ?

- Easy to spot
- ייחודית
- scale invariant
- rotation invariant

למה כדאי לנו להשתמש בפינות(key points) בטור ?

- הם קלים למציאה.
- הם ייחודיים – לכל פינה יש את הזוויות שלה.
- אם פינה היא "חזקת" מספיק scaling לא יפגע בה.
- סיבוב לא פוגע בזוויות של הפינה.

Haris Corner Detector

בהינתן חלון W, אם אנחנו מזיזים את החלון בדلتא Z, חישוב ה SSD יהיה:

$$f(\Delta x, \Delta y) = \sum_{(x,y) \in W} (I(x, y) - I(x + \Delta x, y + \Delta y))^2$$

כפי שעשינו כבר בעבר, ניתן להשתמש בקירוב טור טיילור על מנת לחשב את המשוואה:

Taylor's approximation:

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

Giving us:

$$f(\Delta x, \Delta y) \approx \sum_{(x,y) \in W} (I_x(x, y)\Delta x + I_y(x, y)\Delta y)^2$$

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

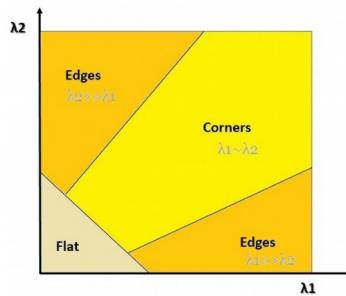
מבוסס על הרצאות ומיצגות של ד"ר גיל בן ארצי ומר שי אהרון

נציג את זה באמצעות כפל מטריצות:

$$f(\Delta x, \Delta y) \approx \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} M \begin{bmatrix} \Delta x & \Delta y \end{bmatrix}$$

$$M = \sum_{(x,y) \in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

אנחנו רוצים לדעת עד כמה ה key point שבחרנו הוא שימושי, נרצה שהערכים העצמיים יהיו
פחות או יותר שווים:



לשם כך הארים הציעו את השיטה הבאה:

λ_i is the i eigenvalue of M ,
where $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \dots \lambda_n$

$$R = \det(M) - k(\text{trace}(M))^2$$
$$\det(M) = \lambda_1 \cdot \lambda_2 \cdot \lambda_3 \cdot \dots \cdot \lambda_n$$
$$\text{trace}(M) = \lambda_1 + \lambda_2 + \lambda_3 + \dots + \lambda_n$$

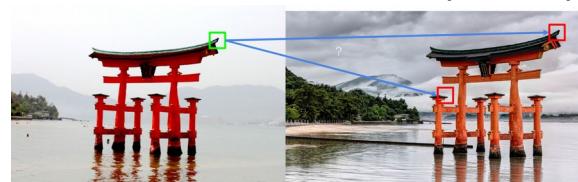
$k \in [0.04, 0.06]$

ככל שהצijn R יותר גדול כהה הפינה יותר שימושית.

עבור כל נקודה/אייזור נקבל ציון מסוים נפעיל SIFT על הציונים המתתקבלים – נלקח את הנקודות בעלות הציון הכי גבוה שהם יהיו עבורנו key points.

Descriptors

נציג descriptor שנקרא sift.
 מצאנו key points אבל איך נדע לחבר ביניהם?



סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומיר שיאהרוני

ה key points שמצאנו הם רק קואורדינטות (y, x), אנחנו לא יודעים איך הנקודה נראה. אנחנו צריכים מושהו שיתאר את הנקודות!

Naive descriptor

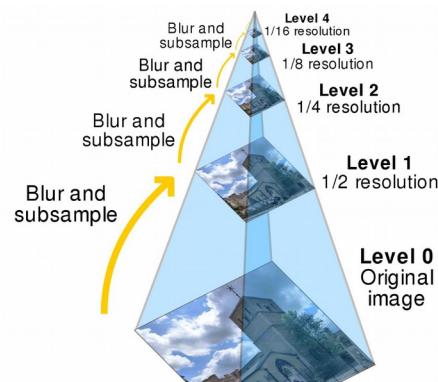
נקח קרnel בגודל $K \times K$ מסביב ל key point ומחשב SSD או NCC.
 חסרונות - לא עמיד בפני:

- שינוי תוארה
- סיבובים
- גדלים
- רעשים

SIFT – Scale Invariant Feature Transformation (1999)

Scale

- Scan the image at different scales (pyramid) and scale each patch to the same size



Lighting and Noise

- Normalize the descriptor
- Use the gradient image instead of the intensity/color image



נורמל את הוקטור בסכום ערכי החלון שלו.

Lighting

- Normalize the descriptor
- Use the gradient image instead of the intensity(gray-scale) image



סיכום קורס ראייה ממוחשבת ועיבוד תמונה

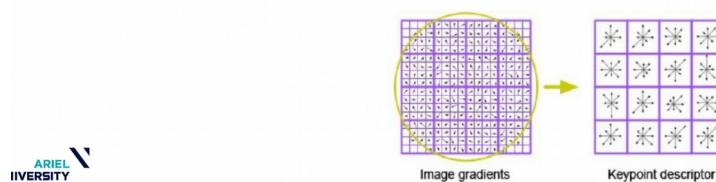
נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

הבעה הכי גדולה היא הסיבוב. עבור כל חלון נלקח את הגרדיאנט המרכזי בה, ונסובב את החלון בכיוון של אותו גרדיאנט. כמובן, נסובב את כל החלונות לאוטו כיון.

Rotation

- Find the strongest orientation in the HoG
 - For each keypoint, we take a 16 X 16 window around it on the orientation image.
 - Calculate the histogram of gradients (HoG) with 36 bins for all 360 degrees
 - Each gradients angle is given a weight of its magnitude multiplied by Gaussian-weighted circular window with a σ that is 1.5 times that of the scale of the keypoint



Noise + Distortion

In order to reduce noise, we use a histogram of gradients instead of the regular gradients.

- Divide the 16x16 patch to 4x4 groups of 4x4 pixels.
- Make a histogram of gradients(HoG) for each group.
- Histogram with 8 bins covering 360 degrees
- Resulting in a 128 descriptor vector

יצירת descriptors:

- נלקח חלון בגודל 16X16 סביב כל key point.
- בכל תת אזור בגודל 4X4 נחשב את ההיסטוגרמה של הכוונים/גרדיאנט עבור 8 כיוונים אפשריים.
- הגרדיאנט יהיה ממושך על ידי גאוסיאן (הערכים מסביב ל key point יקבלו משקל גדול יותר).
- נקבל וקטור בגודל 128 (4X4X8) שמכיל את התיאור של ה key point.

RANSAC

לאחר שמצאנו את נקודות העניין, את התיאור שלהם, ומצאנו התאמות ביניהן – יתכן כי ההתאמת שלנו היא לא מսפיק טובה.

הבעה היא שמצאנו הרבה נקודות עניין, אך יתכן שיש הרבה נקודות שלא נמצא להן התאמת או שההתאמת שנמצא לא תהיה נכונה, במקרה זה נקבל outliers:



סיכום קורס ראייה ממוחשבת ועיבוד תמונה

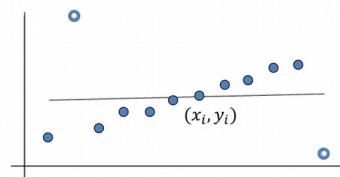
נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון

נתחל בבעיה קלה: מציאת קו המתאר בצורה הטובה ביותר את הנקודות.
 נזער את הפונקציה הבאה:

$$\operatorname{argmin}_L \sum_i \operatorname{dist}(p_i, L)$$

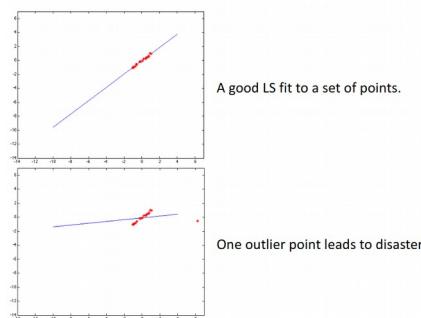
הנסה לעשות זאת באמצעות least square error :



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

$$S = \sum r_i^2$$
$$r_i = y_i - f(x_i)$$

הפתרון זהה הוא לא רובסטי יש לא בעיה עם outlier points הוא לא יודע להתמודד איתם כמו שציריך. לדוגמה:



ראיינו כי השיטה הזאת לא מספיק טוב, נקודות חריגות גורמות לשגיאה גדולה בקו המתאים,
 נרצה שיטה טובה יותר

RANSAC

שיטה נוספת רוביוטית שיעדעת להתמודד עם רעשים ונקודות חריגות.

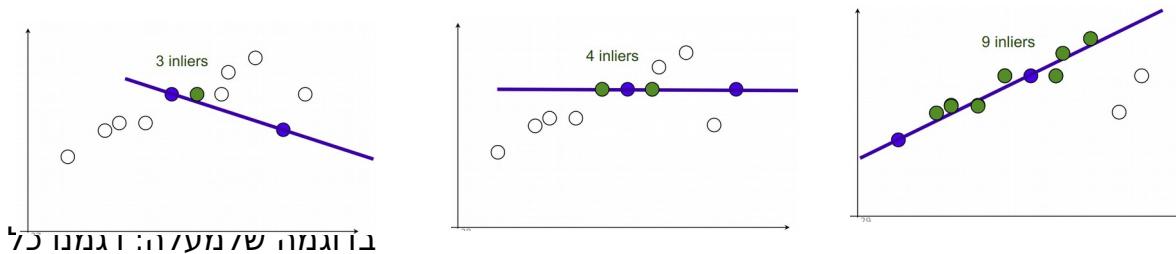
איך היא עובדת:

- נבחר באקראי 2 נקודות.
- נמתח ביניהם קו ישר.
- נבדוק כמה נקודות נמצאות בטוויה עבור דלתא מסוים של אותו קו.
- חוזרים על התהליך הנ"ל עד שמנצאים מספר מקסימלי של נקודות בטוויה.

סיכום קורס ראייה ממוחשבת ועיבוד תמונה

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של ד"ר גיל בן ארצי ומר שי אהרון



פעם 2 נקודות (בסגול) ובדקנו את כמות הנקודות שנופלות בטוויה (בירוק). בסוף נkeh את הקו עם הכיו הרבה (9 במקורה שלנו).

RanSAC algorithm:

```

bestLine = None
max_inlrs= inf
For i in k:
    p1,p2 = choose2points()
    tmpL = findLine(p1,p2)
    tmp_inlrs = # inliers
    if tmp_inlrs > max_inlrs:
        max_inlrs= tmp_inlrs
        bestLine = tmpL
return bestLine

```

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

כמה לולאות נצטרך עד שנגיע לכך האופטימלי?

We want to ensure that in probability p , at least one of the samples of S points is free from outliers.

(for example $p = 99\%$)

Denote the inliers probability by w :

What is N ?

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

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