(1) 21.1.08
עיבוד
תמונה

אישרנו בפעם הראשונה מוכנו לכתוב אי-מייל למתרגל שאולי הוא ישים את החומר איפשהו.

בעיבוד תמונה מתחילים של תמונות או של וידאו. מצלמים את העולם ואז מתחילים של התמונות ומנסים להגיד עליהן משהו. אז במידה מסויימת זה הפוך מצילום. התהליכים שמעורבים בזה הם הפוכים.

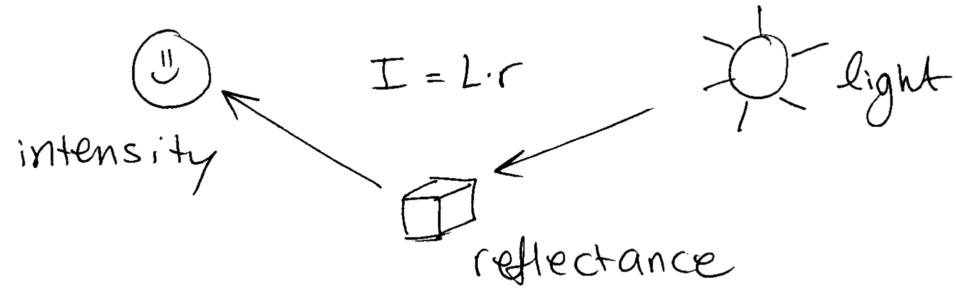
דוגמאות:

1. זיהוי. יש את התמונה המפורסמת מנשיונל ג'יאוגרפיק. צילמו שתי תמונות בהפרש של 16 שנה ורצו לבדוק אם אותה אישה מופיעה בתמונה. ע"י ניתוח של הקשתית הסתבר שכן.
2. ייצוג תלת ממד
3. מצלמות לעזור להחליט מי מנצח במרוץ (-במצגת יש תמונות מתחרות שח"ה)
4. מצלמות אבטחה - אפשר לקחת את הפעולות של 24 שעות ולהדגיש אך את שינוי.
5. clustering - להוציא תמונות של מכוניות מתמונה או להוציא תמונות של אנשים מתמונה.
6. Mobileye - הכרת תמונות במצגת.

אדמיניסטרציה:

- 4 תרגילים תיאורטיים
- 4 תרגילים מעשיים
- 3 בחינות במשך הסמסטר (פעם ב-4 שבועות)

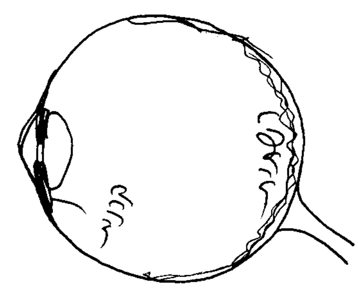
מה זה תמונה? יש אור שפוגע בסצנה ומוחזר ממנה לתוך מכשיר שמצלם.



זה מאוד פשטני. ומעשה זה תלוי בעוד דברים - למשל בזווית הצילום, בסוג המשטח, אורך הגל.

יש משטחים שלא משנה מאיזה כיוון מסתכלים עליהם הם מחזירים הכל - זה נקרא משטח למברט. מצד שני, מראה מחזירה רק בכיוונים מסויימים.

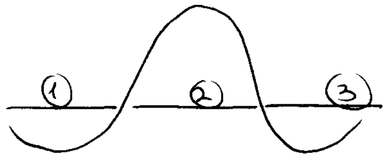
העין האנושית:



האור נכנס לעין ומגיע לרשתית. הרשתית מכילה שני סוגי תאים - קנים ומדוכים. הקנים אחראים על הראיה בשחור-לבן והמדוכים אחראים על ראיית הצבע. יש הרבה יותר קנים ממדוכים, אך את רוב הרזולוציה אנחנו מקבלים מהמידע השחור-לבן. כל החיווט של התאים נמצא לפניהם והאור עובר קודם דרך החיווטים. אגב, יש בערך 10^8 תאים ורק 10^4 עצבים שמחברים ביניהם. אם היה יותר פשוט לא היה לנו מקום מאוד...

העין האנושית רגישה רק לתחום מאוד קטן מכל הספקטרום האלקטרומגנטי 350-480 ננו-מטר. למה זה רק? אנחנו זוכרים שהאטמוספרה שלנו היא שקופה לאור הנראה. האור הנראה זה התחום הכי רחב שיש בו שקיפות של האטמוספרה. כנראה שהאבולוציה יצרה את העין שלנו ככה.

התאים בעין מגיבים באופן שונה לאור והתגובה גם תלויה באור שמגיע לתאים הסמוכים. איידגים לזה בגרף:



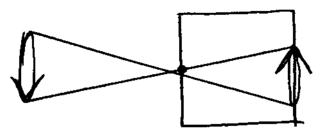
אם פוגע אור רק ב-2 היא מגיבה חזק, אבל אם פוגע גם ב-2 וגם ב-1 הם מגיבים פחות חזק ואם מוסיפים גם את 3 אז עוד פחות חזק.

במדע יש המון אשליות אופטיות תמונות .

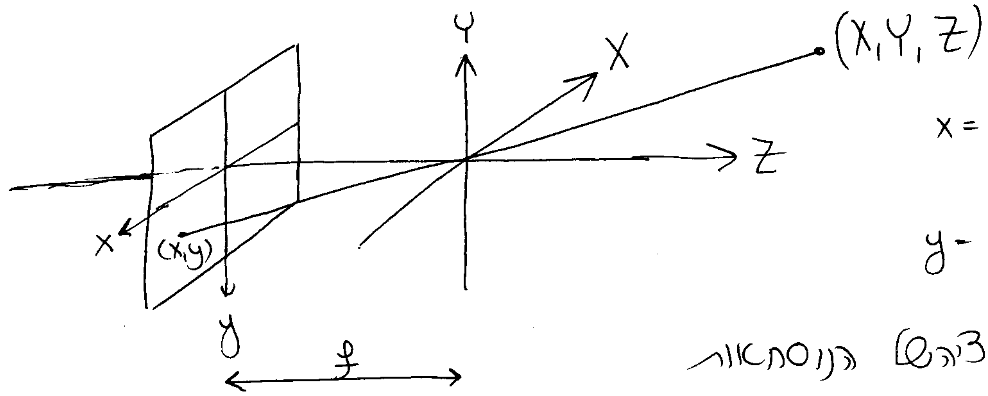
<u>יצירה של תמונה</u>

במצלמה לוקחים את העולם התלת ממדי לעולם דו ממדי. אח"כ את התמונה הופכים לפיקסלים ואז יש את אורכי הגל הרצפים הופכים למספרים .

מצלמה - מצלמה זה למעשה חור   Camera obscura



בד"כ מישור התמונה נמצא מאחורי החריר אבל אפשר גם לדמיין את מישור התמונה לפני החריר ואז מתקבלת אותה תמונה אבל ללא השיקוף .



$$x = \frac{f}{Z} X$$

$$y = \frac{f}{Z} Y$$

מה האינטואיציה של הנוסחאות

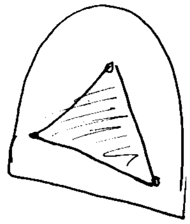
האם ? ככל שמתרחק יותר התמונה קטנה יותר. ככל ש- f גדול יותר התמונה גדולה יותר. f נקרא מרחק המוקד.

ידוע מצלמה חדשה עם מיליון מגה - פיקסל . האם כדאי לקנות ? ובכן, זה תלוי במטרה שלנו ובצרכים . אם צריך רק לזהות אם יש בן אדם או לא תספיק רזולוציה הרבה יותר נמוכה מאשר אם צריך לעשות ניתוח של הקרניות .
בד"כ משתמשים ב- 256 רמות אפור . בדפוס משתמשים ביותר כדי שלא יקרה מצב שמישהו יראה הבדל שלא ראו משהו בתמונה. זה כמובן כפולטה את עצמם .

תמונה במחשב זו פשוט מטריצה של פיקסלים שבכל פיקסל רשום מה הצבע שלו. מרחב הצבע מורכב מכחול, ירוק ואדום. בעזרתם אפשר להציג המון צבעים אחרים .

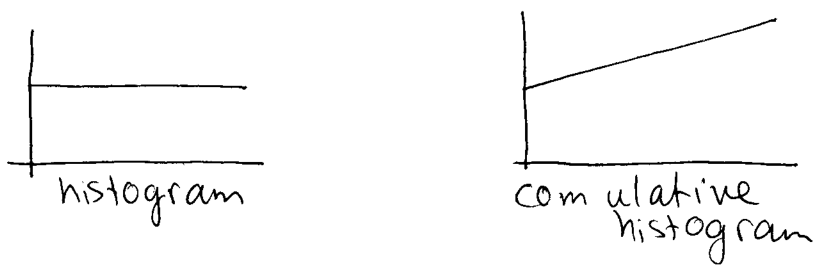
כל תמונה אפשר להציג ב- RGB או ב- YIQ . RGB פשוט אומר כמה

מה צבע יש בכל פיקסל? YIQ - זו איזה פונקציה של הצבעים. ראשית Y זה ה-intensity, כלומר מה רואים בתמונה בשחור לבן. וה-I וה-Q זה מה שאחראי על הצבע. בטלוויזיה משדרים ב-YIQ והטלוויזיה הופכת את זה חזרה ל-RGB ומציגה אותו. את ה-Y משדרים ברזולוציה גבוהה כי יש בעין הרבה קנים. אבל המדוכים פחות רגישים ולכן את ה-I וה-Q שאחראים על הצבע אפשר לשדר ברזולוציה פחות גבוהה. כך חוסכים בשידור והעין בכלל לא שמה לב להבדל.

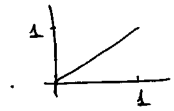


אפשר לסדר את הצבעים בתוך מבנה כזה של שלושה צבעים שבוחרים ניתן להציג באמצעותם את כל הצבעים שבמשולש שנוצר. האדום, הכחול והירוק של הטלוויזיה לא יכולים לייצר את כל הצבעים. למעשה, בגלל שהמבנה הזה לא משולש אז שלושה צבעים לא יכולים לייצר את כולם.

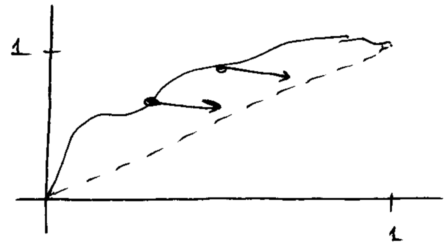
היסטוגרמה - אומרת לנו כמה פיקסלים יש בתמונה מכל דרגת אפור.
נניח שההיסטוגרמה שטוחה - אז לכל צבע יש אותו מספר פיקסלים. אז ההיסטוגרמה המצטברת - היא לינארית.



אם מנרמלים את הדרגות מ-0 ל-1 ואת ההיסטוגרמה מ-0 ל-1 אז ההיסטוגרמה המצטברת נראית כך:



אנחנו רוצים לקחת תמונה ולהפוך את ההיסטוגרמה שלה לשטוחה. אז את המצטברת אנחנו רוצים להפוך ללינארית.



נ"ו אנחנו רוצים לשנות את הצבעים של התמונה כך שההיסטוגרמה המצטברת תהיה לינארית. יש אילו כמה דרישות: שאם היו שני פיקסלים באותו צבע הם ישארו באותו צבע וגם שאם היו שני פיקסלים - אחד כהה מהשני אז בסוף עדיין הראשון יהיה כהה מהשני. אז מה שצריך לעשות זה לשנות את הצבע של כל פיקסל לפי מספר הפיקסלים שיש לפני אותו צבע. אח"כ אפשר לנרמל חזרה ל-0 עד 255. דוגמה במצגת. בתהליך הזה יתכן אובדן מידע - יכול להיות שיהיו פיקסלים שיקבלו את אותו הצבע. הקונטרסט בין הפיקסלים שמופיעים מעט קטן ואילו הקונטרסט בין פיקסלים שמופיעים הרבה גדל.

אחת הבעיות בשוויון היסטוגרמה היא שבאזורים שונים בתמונה יש סטטיסטיקה שונות. אז אפשר לעשות את זה עם חלון על כל פיקסל בתמונה, כל פעם מחשבים רק את ההיסטוגרמה של החלון אבל משנים רק את הערך של הפיקסל המרכזי. כמובן צריך לחשוב מה עושים עם השוליים. אפשר למשל להשתמש בפונקציה שהתקבלה מהחלון הכי קרוב.

קוונטיזציה

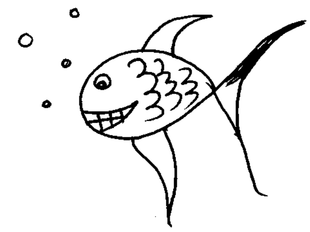
- אם יש 24 ביטים לפיקסל יש 256^3 צבעים.
- אם יש 8 ביטים לפיקסל אז יש רק 256 צבעים. איך מחלקים את זה בין הצבעים? יש כמה אופציות
  - 3-3-2 ביטים ל- R-G-B בהתאמה.
  - טבלת המרת צבעים - לוקחים את 256 הצבעים הכי חשובים בתמונה. לכל אחד נותנים מספר ומשתמשים ב- look up table כדי להמיר. כמובן נשאלת השאלה איך בוחרים את הצבעים האלה...

שגיאת קוונטיזציה - נניח שהיה צבע (r,g,b) לפיקסל p. וקודדנו אותו בטבלה k. אז השגיאה של הפיקסל היא

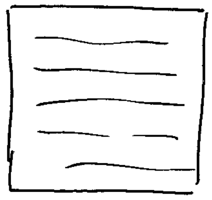
$$E_p^2 = (r-R_k)^2 + (g-G_k)^2 + (b-B_k)^2$$

אם השגיאה הזו היא אפס אז הצבע לא השתנה. את בחירת הצבעים רוצים לעשות כך שהשגיאה סה"כ $E^2 = \sum_p E_p^2$ תהיה מינימלית. בהינתן טבלה לכל (r,g,b) מקודדים אותו בעזרת הכניסה שממזערת את השגיאה.

8ס
עיבוד תמונה

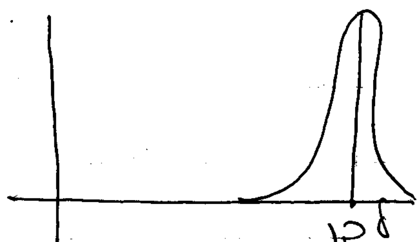


שאלה: נניח שיש דף עם טקסט.



המצב הוא לא אידיאלי ובדר"כ יש גם קצת מריחות והצבעים לא מופרדים באופן מושלם, אך באופן כללי ההיסטוגרמה נראית משהו כזה:

מה אם נעשה שיווי היסטוגרמה? - ייעלמו דרגות האפור שאין מהן אף פיקסל והוא ימתח את מה שיש על פני כל הדרגות. למשל, אם ישאר רק את הדף הלבן ללא טקסט ההיסטוגרמה שלו כזאת:



השיווי ייקח את הדרגה הכי נמוכה של הלבן ויהפוך אותה לשחור ובמצום הדף כבר לא ייראה אחיד - נראה יותר הבדלים בין דרגות הלבן. דבר דומה קורה גם עם הטקסט - ההבדלים בתוך הלבן ובתוך האותיות ייראו יותר, אבל ההבדלים בין האותיות לדף ייעלמו. אז זה רע - זה בכלל לא מה שרצינו..

המסקנה: אמנם אמרנו ששיווי היסטוגרמה גורם לתמונה להיראות טוב יותר, חדה יותר, אבל למעשה צריך לחשוב טוב טוב מה אנחנו רוצים ואם שיווי היסטוגרמה משיג את זה.

<u>קוונטיזציה של צבעים</u> זה פחות רלוונטי למסכים כי כיום מסכים יכולים להציג מלא צבעים, אבל זה מאוד רלוונטי להדפסה למשל.

משיעור הקודם, אמרנו שיש טבלת המרת צבעים ושאלנו את

צ"ל מהי הדרך הטובה ביותר להשתמש בטבלה.

בשביל המטרה הזאת הגדרתי את שגיאת הקוונטיזציה:

$$E_p^2(R_k,G_k,B_k) = (r-R_k)^2 + (g-G_k)^2 + (b-B_k)^2 \quad (r,g,b)$$

זה סכום הריבועים בין הצבע הרגיל לצבע המקורב אותו. כן כמובן הגדרה שנוחה לשימוש מתמטי אבל היא לא מתאימה לראיה האנושית.

1) בהינתן טבלת צבעים כזו (LUT = Lookup Table) נציג לצבע (r,g,b) באינדקס k שעבורו E_p^2 מינימלי.
זה גם ממזער את השגיאה הגלובלית $E^2=\sum_p E_p^2$.

2) בהינתן אינדקס k לכל פיקסל $p(i,j)$ בתמונה. כדי לשפר את הטבלה - כלומר לשפר את השגיאה E^2 אפשר להחליף את הכניסות בטבלה לממוצע הפיקסלים שמצביעים לכל שורה.
משנים את הטבלה כך ש- $(R_k,G_k,B_k)=\sum_{\text{פיקסלים עם אינדקס } k} \frac{p(i,j)}{\#k}$

אבל עכשיו האינדקסים לא מתאימים לצבעים כך שהשגיאה מינימלית. אז צריך לשנות את האינדקסים. ואז שוב אפשר לשפר בעזרת הממוצעים. ואז כל התהליך הזה חוזר על עצמו שוב ושוב? האם הוא נעצר? כן! כי כל איטרציה מורידה את השגיאה והשגיאה חסומה מלמטה ע"י 0!

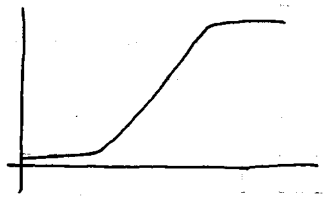
ההיסטוגרמה

רק איך בוחרים את הטבלה הראשונה?
אם יש N פיקסלים ו-k תאים ב-LUT
נרצה שבכל תא יהיו $\frac{N}{k}$ פיקסלים.
אז יש שתי אופציות - או שנותנים לכל קבוצת הפיקסלים בגודל $\frac{N}{k}$ את אותו אינדקס או שמתחילים [illegible] בשלב ב לאינדקס k נותנים את הערך צבע שהוא ממוצע הצבעים של הפיקסלים המתאימים.

בד"כ התהליך הזה מתכנס תוך 3-4 איטרציות.
(בלמידה יש תהליך דומה שנקרא EM).

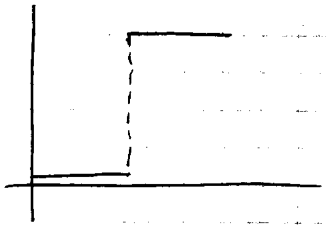
בעזרת LUT אפשר לעשות מיני דברים:

- מתיחה - נשתמש ב- LUT שנראה ככה:

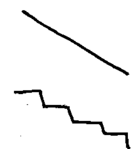


זה מגדיל את הקונטרסט
בתחום של האפור שבאמצע

- מתיחה אונטימטיבית - נשתמש ב- LUT שנראה ככה:

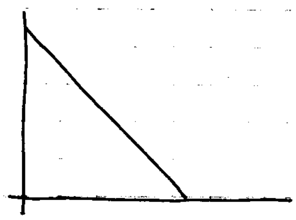


זה יכול לצאת קצת פחות יפה:
כי אם היה קו אלכסוני



יכול לצאת להיות ככה
וזה לא יפה...

- היפוך - הופכים את הלבן לשחור ולהפך. וזה LUT הנראה ככה:



אגב, שינוי היסטוגרמה זה סוג של LUT.
כל האופרטורים שראינו עד כאן פועלים על פיקסלים בודדים.
אבל יש גם אופרטורים שפועלים על סביבות של פיקסלים.

קונבולוציה חד-ממדית

$$h(i) = (f*g)(i) = \sum_{k=1}^{n} f(k)g(i-k)$$

למשל
$$f = (0\ 0\ 1\ 0\ 0) \qquad g = (0\ 0\ 1\ -1\ 0)$$

$$\Rightarrow h(6) = \sum_{k=1}^{5} f(k)g(6-k) = f(3)\cdot g(3) = 1$$

$\downarrow$

$f(i)=0 \quad \forall i\neq 3$

$$h(7) = \sum_{k=1}^{5} f(k)g(7-k) = f(3)\cdot g(4) = -1$$

$\downarrow$

מתחילים על הסדרה כסדרה ציקלית ולכן
$g(6) = g(1)$

זה במקרה הדיסקרטי אבל גם אפשר להגדיר באופן דומה למקרה הרציף:

$$(f*g)(x) = \int_{-\infty}^{\infty} f(a)g(x-a)\,da$$

אלא שכאן גם הפונקציות מוגדרות רק על הקו מ- $\mathbb{R}$ לא מניחים שהן מחזוריות אלא מניחים שמשני הצדדים הן שוות לאפס.

בקונבולוציה פשוטה, לפי ההגדרה כפי שראינו סיבוכיות החישוב היא $O(n^2)$.

למה בהגדרה של קונבולוציה מופיע $g(i-k)$ ולא $g(i+k)$? כלומר
למה אנחנו "משקפים" אותה?

נניח ש- $g=(\overset{-2}{0}\,\overset{-1}{0}\,\overset{0}{1}\,\overset{1}{0}\,\overset{2}{0})$ ו- $f=(1\,2\,3\,4\,5)$.
אז הקונבולוציה היא $(1\,2\,3\,4\,5)$.
אם ההגדרה הייתה ללא שיקוף היה יוצא $(5\,4\,3\,2\,1)$. זה משהו
שבטבע קורה.

<u>תכונות של קונבולוציה:</u> (לא להשתמש בנכונותן)

1) קומוטטיביות $f*g=g*f$
2) אסוציאטיביות $(f*g)*h=f*(g*h)$
3) דיסטריביוטיביות $f*(g+h)=f*g+f*h$ (הסכום הוא איבר-איבר)

<u>קונבולוציה דו-ממדית</u>

$$h=f*g \quad h(i,j)=\sum_{k=1}^{n}\sum_{l=1}^{m} f(k,l)\,g(i-k,j-l)$$

כאן אם מדובר בפיקסלים אז יוצא שהאופרטור של כל פיקסל תלוי בסביבה שלו, סוכם הפיקסלים שמסביב. ולמעשה g זה אילוצים משקלות שנותנים לפיקסלים. ככה עושים החלקה של התמונה. אם $g=\frac{1}{9}I_3$ אז כל פיקסל ב-f יוחלף בממוצע הפיקסלים שסביבו.
למעשה g היא בגודל של f (התמונה) ואנחנו ממלאים את השאר עם אפסים ולא בצורה ציקלית.
צריך לשים לב שכדי שלא נצא מתחום הערכים של התמונה הסכום של כל איברי g צריך להיות 1 וגם כל האיברים חיוביים. אז אחרי שעושים החלקה ממוצע רמות האפור בתמונה לא משתנה.
ההחלקה הזו טובה לטשטוש רעשים מהצורה $\text{קלט}=\text{קלט}_{\text{אמיתי}}+\eta$ כאשר $\eta\sim N(0,\sigma)$. מיצוע פיקסלים מטשטש את התמונה אבל מאפס רעש שהתוחלת שלו היא אפס. חשוב מאוד לדעת כאן את ה- tradeoff שבין הטשטוש להוצאת הרעש.

6

עד כאן החלפנו כל פיקסל בממוצע הפיקסלים שמסביב.
אפשר גם להחליף כל פיקסל בחציון. למה זה טוב? החציון ממזער את סכום הערכים המוחלטים.

$$\arg\min_{\bar{x}} \sum_{i=1}^{n} (\bar{x}-x_i)^2 = \frac{\sum x_i}{N} = \text{mean}(x_i)$$

$$\arg\min_{\bar{x}} \sum_{i=1}^{n} |\bar{x}-x_i| = \text{median}(x_i) = \left(\begin{array}{c} \bar{x} \text{ רק ש-} \\ \#\{x_i \mid x_i > \bar{x}\} = \#\{x_i \mid x_i < \bar{x}\} \end{array} \right)$$

החלפת פיקסל בחציון טוב לרעש מסוג salt & pepper - פשוט נקודות לבנות ושחורות שלא קשורות לכלום.
באופן כללי חציון בד"כ מנקה רעשים יותר טוב כי הוא לא מטשטש.
במצגת יש דוגמא למה שקורה עם תמונה בשימוש בחציון ובממוצע.

הדבר ההפוך מטשטוש זה חידוד. אז אם טשטוש זה אינטגרל אז חידוד זה גזירה.

מה ההגדרה של נגזרת?

$$\frac{d}{dx} f(x) = \lim_{h\to 0} \frac{f(x)-f(x-h)}{h}$$

אבל בתמונות אנחנו בבעיה כי זה דיסקרטי. ה-h הכי קטן שיש הוא 1 אז

$$\frac{\partial}{\partial x} f(i,j) \cong f(i,j) - f(i-1,j)$$

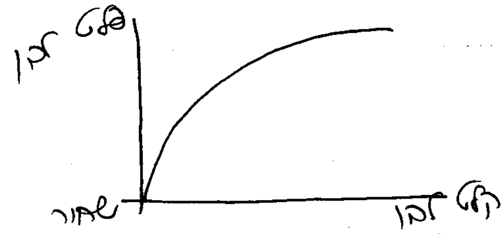
את זה אפשר להשיג בעזרת קונבולוציה עם $(1,-1)$ מרופד באפסים.
נא להשתכנע: קונבולוציה עם הדבר הזה נותנת תמונה עם ממוצע 0.
בדומה לזאת, נגזרת לפי y אפשר להשיג בעזרת קונבולוציה עם $\binom{1}{-1}$

איך משלבים בין הדברים? כלומר, איך מזהים שפה כלשהי - אופקית או אנכית? אי אפשר לעשות את זה עם קונבולוציה.

1.08
עיבוד תמונה

המבחן הראשון יתקיים ב- 18/6 ___

דיברנו על LUT - טבלת המרה של צבעים. יש לזה אפליקציה נוספת Gamma Correction. ומשתמש בו בצורה שנראית כך:



הפרמטר הוא γ. אם $\gamma=1$ זה קו-ישר שלא עושה שום דבר לתמונה. מה שמצויר בתמונה הופך את השחור לבהיר יותר. אזור קטן של שחור ממופה לאזור גדול יותר. כלומר אומרים שזה פותח את השחור. באזורים הבהירים זה דווקא סוגר - מעביר אזורים גדולים לקטנים. לפעמים זה יכול להיות טוב אם יש לנו טקסט ולא רואים שם את ההבדלים טוב כ"כ. בניגוד לשיווי היסטוגרמה, תיקון γ בכלל לא תלוי בתמונה!

תיאורטית כשמדברים על קונבולוציה בין מטריצות יש שתי מטריצות P,A מאותו מימד. אבל בתמונות P תמונה ו-A זה ה mask והם לא חייבים להיות מאותו גודל. בד"כ לוקחים מטריצות A בגודל אי-זוגי ולאמצע שלה מתייחסים בקואורדינטה (0,0). ומדמיינים ש-A מרופדת באפסים עד הצדדים (נשים בניגוד לתיאוריה שם מניחים שהמטריצה היא מחזורית).

דוגמאות:

אינדקס 0

$$(1\ 2\ 3\ 4\ 5)*(0\ 0\ 1\ 0\ 0) = (1\ 2\ 3\ 4\ 5)$$

$$(1\ 2\ 3\ 4\ 5)*(0\ 0\ 0\ 1\ 0) = (5\ 1\ 2\ 3\ 4)$$

$$(1\ 2\ 3\ 4\ 5)*\frac{1}{3}(1\ 1\ 1) = ---$$

כל פיקסל יהיה ממוצע שלו ושל השכנים שלו.

ומה יצא ב $(1\ 2\ 3\ 4\ 5)*\frac{1}{2}(1,1)$? כאן תלוי איפה אנחנו מגדירים

את הגאפס? נניח שבחרנו כך שכל פיקסל יהיה ממוצע שלו של השכן השמאלי שלו. אז מעבר לזה שטשטשנו קצת את התמונה, התמונה גם תזוז בפיקסל אחד ימינה. כי למשל,

$$(0\,0\,1\,0\,0) * \tfrac{1}{2}(1,1) = (0\,0\,\tfrac{1}{2}\,\tfrac{1}{2}\,0)$$

במקום שיהיה קו אחד באמצע יש קו מטושטש וימני יותר.
והסיבה הזאת בד"כ מעדיפים לעשות קונבולוציה עם משהו בגודל אי-זוגי.

קונבולוציה היא אופרטור לינארי:

$$A * \alpha B = \alpha (A * B)$$
$$A * (B + C) = A*B + A*C$$

אז בתור אופרטור לינארי אפשר לייצג אותה כמכפלה במטריצה.
נדון קודם במקרה החד-ממדי. נניח שרוצים לחשב את $p * a$
כאשר p ; a באורך N. אז

$$p * a = \underbrace{\begin{pmatrix} a_0 & \cdots & a_{N-1} \\ a_{N-1} & a_0 \cdots & a_{N-2} \\ & \vdots & \\ a_1 & a_2 \cdots & a_0 \end{pmatrix}}_{\text{מטריצה מעגלית}} \begin{pmatrix} p_0 \\ \vdots \\ p_{N-1} \end{pmatrix}$$

כעת, במקרה הדו ממדי נניח ש- A, P בגודל $N \times N$.
ראשית נהפוך את המטריצות לוקטורים ואז המטריצה תהיה בגודל $N^2 \times N^2$ ומורכבת מבלוקים שכל אחד מהם מעגלי.

<u>החלקה - טשטוש</u>

- סכום הגורמים ב-mask צריך להיות 1 כדי לא לשנות את טווח הערכים בתמונה.
- החלקה מורידה רעש לבן - רעש בלתי תלוי ואדיטיבי עם תוחלת אפס.
- דוגמא: כשמדפיסים תמונות משתמשים בדיו - יש נייר לבן ושמים עליו כתם שחור. אז זה בינארי - או לבן או שחור. אי אפשר לשים כתם אפור. אבל כדי שבכל זאת נוכל לראות אפור שמים את כתמי השחור במרווחים גדולים יותר. ואז אם מסתכלים מרחוק אז לעין זה נראה אפור.

אם סורקים תמונה רואים את זה בד"כ ברזולוציה הרבה יותר גבוהה מרזולוציית ההדפסה ואז יוצא בסריקה שיש נקודות נקודות. בד"כ מטפלים בזה בעזרת טשטוש.

יש שתי אפשרויות:

- AM (זה מה שקורה בעיתונים) - הסריג שעליו שמים את הנקודות קבוע אבל גודל הנקודה משתנה
- FM (כמו במדפסת InkJet) - הנקודות בגודל קבוע אבל הצפיפות שלהן משתנה.

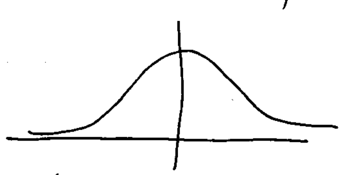
<u>פילטר חציון</u>

חציון של אוכלוסיה זה אותו מספר כך שיש בדיוק חצי מעליו וחצי מתחתיו. בניגוד לממוצע אם יש ערך אחד שמאוד רחוק מהשאר זה לא משפיע עליו הרבה.

<u>החלקה</u>

- הטשטוש הקטן ביותר שאפשר לעשות זה עם $\frac{1}{2}(1,1)$ - זה ממוצע בין פיקסל לשכן שלו (ראשית הציר תלוי במה אנחנו קובעים מהו אינדקס 0)
- עוד קצת החלקה זה עם $\frac{1}{2}(1,1)*\frac{1}{2}(1,1) = \frac{1}{4}(1,2,1)$
  כפי שה-0 יהיה באמצע דרך שה-0 של אחד מהטשטושים יהיה ממקום אחר.
- ועוד קצת זה עם $\frac{1}{4}(1\ 2\ 1) * \frac{1}{2}(1\ 1) = \frac{1}{8}(1\ 3\ 3\ 1)$

באופן כללי הנקודות מגיעות ממשולש פסקל. באופן כללי ככל שעושים יותר טשטושים זה הולך ונהיה דומה לגאוסיאן



וזה נחשב החלקה טובה (נסביר את זה בעתיד).

לגאוסיאן יש נוסחה מסובכת אז בד"כ משתמשים בקירוב של מקדמים בינומיים וזה טוב כי אפשר לחשב את זה בצורה מאוד יעילה.

<u>נגזרת של גאוסה</u>

נגזרת מגבירה רעשים. אז אם מגבירים את הרעשים צריך להחליק.
מה עושים קודם? זה לא משנה כי קונבולוציה זה קומוטטיבי!
את ההחלקה עושים בכיוון ניצב לכיוון הנגזרת. כי אם נחליק
באותו כיוון אז בדיוק נפספס את מה שרצינו להשיג עם הנגזרת -
למצוא קצוות.

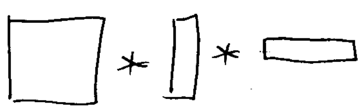
מסכות פופולריות הן

$$\frac{\partial f}{\partial x} = \frac{1}{8}\begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \qquad \frac{\partial f}{\partial y} = \frac{1}{8}\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

מאיפה הגיע המקדם $\frac{1}{8}$? $\frac{1}{4}$ הגיע מההחלקה? מאיפה
עוד $\frac{1}{2}$? אמרנו שנגזרת זה $(1\ -1)$ אבל פה אנחנו
משתמשים ב $(1\ 0\ -1)$ שזה לעשות פעמיים הנגזרת...

מבחינת האסוציאטיביות יש שתי דרכים לחשב את $Q = P*A*B$
או $Q = P*(A*B)$ או $Q = (P*A)*B$.

מה יותר יעיל?

- למשל נסתכל על $(p_1 \cdots p_N)*(a\ b)*(c\ d)$.
  כדי לחשב את $(p_1 \cdots p_N)*(a\ b)$ צריך $2N$ כפלים
  ואז עם $(c\ d)$ זה יוצא עוד $2(N+1)$. סה"כ $4N+2$
  אם מחשבים קודם את $(a\ b)*(c\ d)$ זה 4 כפלים
  וזה יוצא בגודל 3 אז נוספים $3N$ כפלים. סה"כ $3N+4$
  אז זה עדיף.
- ומה עם המקרה הדו-ממדי 
  כאן דווקא עדיף דווקא לחשב. כל כיוון בנפרד.

<u>קירוב נגזרות</u> נניח שרוצים לגזור פעמיים. זה אומר שצריך
פעמיים להפעיל את $(1\ -1)$. וזה אומר שצריך להפעיל פעם
אחת את $(1\ -2\ 1)$.

למה שנרצה בכלל לעשות נגזרת? נגזרת ראשונה מוצאת מעברים
בין גוף לרקע אבל גם מגבירה רעשים. אז נגזרת שנייה מגבירה

נעשים בריבוע. אז זה גרוע. אבל למה זה טוב?

לפלסיאן זה $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$. בקונבולוציה זה למעשה

$$(1\ -2\ 1) + \begin{pmatrix}1\\-2\\1\end{pmatrix} = \begin{pmatrix}0&1&0\\1&-4&1\\0&1&0\end{pmatrix}$$

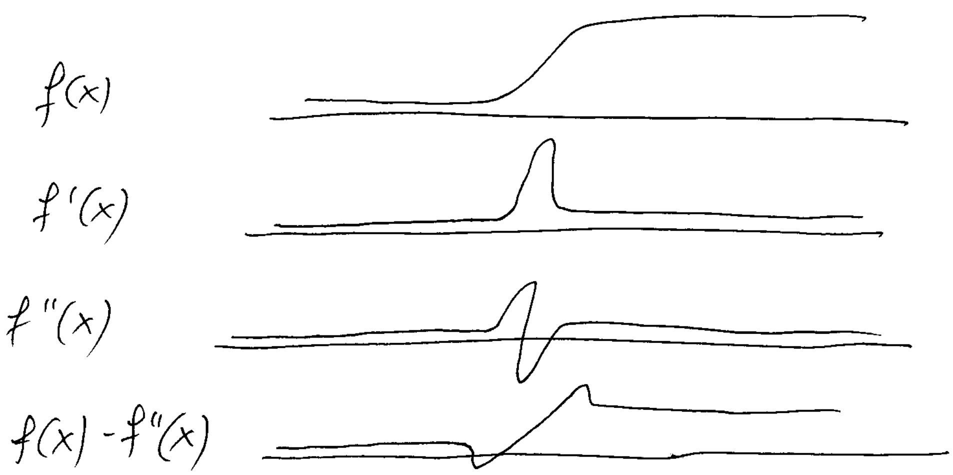
אם התמונה אחידה התוצאה תהיה פשוט אפס.

אם דרגת האפור משתנה לינארית אז גם התוצאה תהיה אפס.

איזו קונפיגורציה של פיקסלים תיתן לפלסיאן גדול? ברגע שיש הפרש גדול בין פיקסל לשכניו. הלפלסיאן מגיב חזק מאוד לנקודות בודדות.

מה קורה אם מחסירים מהתמונה את הלפלסיאן שלה. בקונבולוציה זה

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



זה נותן איזושהי "מסגרת" לתמונה. זה מגדיל את הקונטרסט

וזה נקרא פסי מך. אז זה נותן מין חידוד.

(למעשה - בטרנספורם פורייה זה למעשה high-pass filter).

<u>גילוי שפות</u> יש כל מיני שפות - מטושטשות, חדות, בין מרקמים או בין גופים, צמתים בין גופים.

איך מוצאים מיקום מדויק של השפה? בד"כ אומרים שהשפה היא במקום שבו הנגזרת השניה מתאפסת.

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

אם מסתכלים על הערך המוחלט של הגרדיאנט רואים
את קווי המתאר של התמונה.

(10) 11.02.08

עיבוד תמונה

פירמידות

פירמידה זה למעשה אוסף עותקים הולכים וקטנים של תמונה מקורית $N \times N$ כלשהי. בכל שלב מקטינים את המימדים פי שניים (נניח בד"כ ש-N חזקה של 2). ככה אפשר לעשות $\log N$ עותקים מוקטנים.

סה"כ יש $N^2 + \left(\frac{N}{2}\right)^2 + \left(\frac{N}{4}\right)^2 + \dots + 1 \le$

$$\le N^2 \sum_{i=1}^{\infty} \frac{1}{4^{i-1}} = N^2 \frac{1}{1-\frac{1}{4}} = \frac{4}{3} N^2$$

(באופן כללי $\sum_{n=0}^{\infty} q^n = \frac{1}{1-q}$ עבור $|q| < 1$).

למה שנרצה להפוך תמונה לפירמידה?

- שידור הדרגתי - אפשר לקבל חלק מהפרטים במהלך השידור ולא צריך לחכות לסופו כדי לקבל מושג כללי על התמונה.
- חישובים הדרגתיים -
  - חיפוש ברזולוציה משתנה (כמו מה שקורה ב- google maps למשל)
- preview - למשל אם צריך להפעיל איזה פילטר מסובך אפשר קודם להפעיל אותו על גרסה מוקטנת ולקבל תחושה של הפעולה שלו
- חיפוש בין תמונות - קשה לעבור על הרבה תמונות ברזולוציה מלאה אז משתמשים על thumbnails.

איך בונים פירמידה?

מה שצריך לעשות זה להקטין את התמונה פי שניים בכל מימד. הדרך המיידית לעשות את זה היא לבחור כל פיקסל שני. אבל זה בעצם לא מומלץ כי למשל אם התמונה מורכבת מפסים שחורים ולבנים לסירוגין אז אחרי דגימה כזאת נקבל או רק לבן או רק שחור. מה שצריך לעשות הוא לסנן לפי משפט הדגימה. לעשות Low Pass Filter שמאפס חצי מהתדרים - אולי הטבעי ביותר: נעשה FFT על כל התמונה ואז IFFT על החלק $N/2 \times N/2$ המרכזי של התוצאה. או לחילופין אפשר לעשות IFFT על כל התמונה אחרי שאיפסנו את התדרים הגבוהים ואז דוגמים כל פיקסל שני.

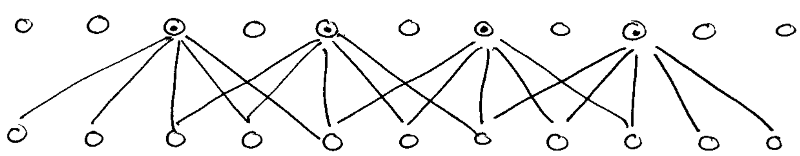
– שאלה למחשבה: איך תיראה התמונה אם במקום לקחת מהתמונה של הגדר את האמצע ניקח ממנה כל פיקסל שני?

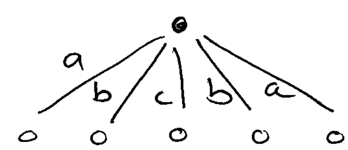
– רמז: לקחת מהטרנספורם כל תדר שני זה מכפלה עם מסרק. כלומר, מה קורה במרחב התמונה?

בד״כ לא משתמשים ב-FFT כדי לעשות את זה כי חבל להתעסק עם מרוכבים אז עושים קונבולוציה עם מסכת טשטוש גאוסיאני (למשל $\frac{1}{16}\begin{pmatrix}1&2&1\\2&4&2\\1&2&1\end{pmatrix}$) שמורכבת ממקדמי פסקל או זוגיים. הבעיה עם זה היא שקשה לדעת מראש מה הגודל של המסכה שמספיק אז זה מבוסס על ניסוי וטעייה. עוד משהו שצריך לשים לב אליו - הפילטר כאן הוא לא מושלם אלא גאוסיאני. זה נותן בד״כ תוצאות יפות יותר.

הערה של יעילות: אנחנו במילא הרי הולכים לזרוק $\frac{3}{4}$ מהתמונה אז למה לחשב את הפילטר לכל הפיקסלים? נחשב את הממוצע המשוקלל רק עבור הפיקסלים שאנחנו הולכים לשמור.

למה הפילטר נראה כמו שהוא נראה? נסתכל על קונבולוציה של שורה.





קל להשתכנע בניפנופי ידיים שכדאי שהמשקלות יהיו סימטריים ושה- $a<b<c$.

כדי לשמור את ממוצע דרגת האפור נרצה ש- $\boxed{2a+2b+c=1}$

נדרוש גם ש- $0\le a\le b\le c$.

נרצה גם שכל הפיקסלים יתרמו לתוצאה בצורה שווה. פיקסל שנמצא בשריג הדגימה תורם c לפיקסל שמעליו ו-a לכל אחד מהפיקסלים שבצדדים. הפיקסלים שלא בשריג תורמים b לכל אחד מהפיקסלים שלצידם. אז כדי שהתרומה תהיה זהה נרצה ש- $\boxed{c+2a=2b}$.

המקדמים הבינומיים מקיימים את התנאים האלה.

יש שפין אומרים שעשור קונבולוציה עם (1 2 1) מטשטש יותר מדי.
אז למה שלא נטשטש ע"י (1 1). כבר דיברנו על הבעיה שזה מזיז קצת את התמונה אבל זה לא כזה נורא. הבעיה האמיתית היא שזה קונבולוציה עם ⊓ שהטרנספורם פורייה שלו הוא sinc וזאת פונקציה בעייתית כי יש לה אפסים. אנחנו אף פעם לא רוצים להכניס sinc לתמונה שלנו!

איך מגדילים תמונה פי שניים?

1) מרפדים את התמונה באפסים:

$$\begin{matrix} a & b & c \\ d & e & f \\ g & e & h \end{matrix} \rightarrow \begin{matrix} a & o & b & o & c & o \\ o & o & o & o & o & o \\ d & o & e & o & f & o \\ o & o & o & o & o & o \\ g & o & e & o & h & o \\ o & o & o & o & o & o \end{matrix}$$

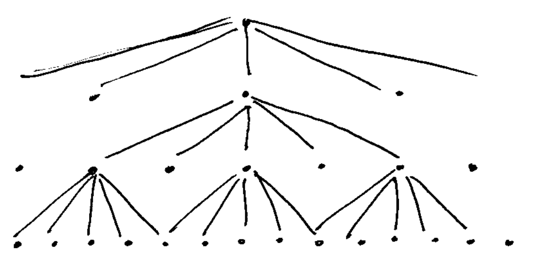
2) מטשטשים: $(a\,o\,b\,o\,c\,o) * \frac{1}{2}(1\;2\;1) = (a, \frac{a+b}{2}, b, \frac{b+c}{2}, \ldots)$

נשים לב שכאן הסכום של המסכה הוא 2 ולא 1 וזה כי ע"י הריפוד הקטנו את רמת האפור פי 2 אז אנחנו צריכים להחזיר אותה למעלה.

למטריצות שנוצרות ככה נקראות <u>פירמידה גאוסיאן</u>. אפשר להסתכל על זה ככה:

$$G_n = \text{Reduce}(G_{n-1})$$
$$\vdots$$
$$G_2 = \text{Reduce}(G_1)$$
$$G_1 = \text{Reduce}(G_0)$$

התמונה המקורית $G_0 =$



פירמידת <u>הלפלסיאן</u> מוגדרת ע"י ההפרשים

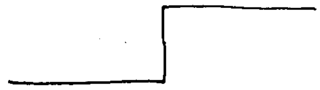
$$L_n = G_n$$
$$\vdots$$
$$L_k = G_k - \text{Expand}(G_{k+1})$$
$$\vdots$$
$$L_2 = G_2 - \text{Expand}(G_3)$$
$$L_1 = G_1 - \text{Expand}(G_2)$$
$$L_0 = G_0 - \text{Expand}(G_1)$$

אם נמחקה לנו ה פירמידה הגאוסיאן אפשר לשחזר אותה מהלפלסיאן:

$$G_k = \text{Expand}(G_{k+1}) + L_k \longrightarrow \cdots \longrightarrow G_0 = \text{Expand}(G_1) + L_0$$

פירמידת הלפלסיאן היא למעשה מה שהלך לאיבוד כשהקטנו את התמונה והגדלנו אותה חזרה. היא שומרת בעצמה את האינפורמציה שהלכה לאיבוד בין השלבים של הפירמידה.

למשל, אם התמונה היא



אחרי טשטוש והרחבה לה יצא משהו כזה:

ואז אחרי שלב אחד של מטריצת הלפלסיאן זה יוצא

שזה דומה ללפלסיאן של תמונה זאת. הפירמידה הזו נקראת פירמידת לפלסיאן.

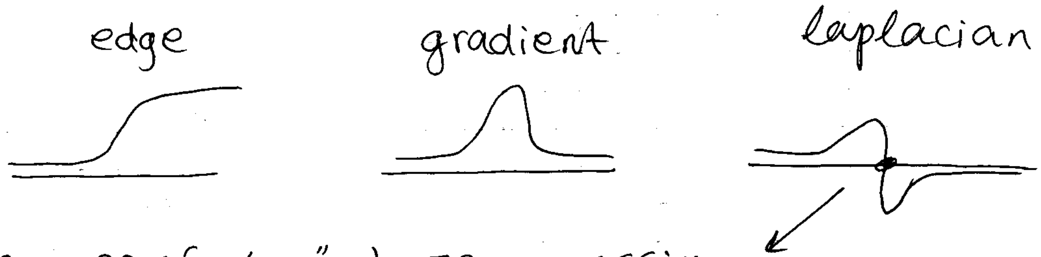
בשיעור הבא נראה שימושים של פירמידת הלפלסיאן (מיזוג תמונות ללא קו תפר, כיווץ).

(12) 18.02.08

עיבוד תמונה

מציאת שפה - יש שני מאפיינים של שפה:

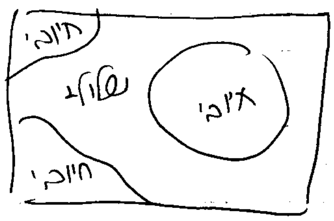
- גרדיאנט גבוה
- הנגזרת השניה "מתאפסת".



zero crossing (ובד"כ זה לא בדיוק 0 אלא קרוב לאפס. אז למעשה מה שצריך לחפש הינו מעבר מערך חיובי לערך שלילי)

ה- zero crossing נמצא בנקודת השיא של הגרדיאנט.

גרף של zero cossings יכול להיראות ככה:



חל מאשר מקצות הצורות חיבות להיות סגורות.

בד"כ משתמשים ב- zero crossings כדי למצוא קצוות.

פילטר high-pass אמנם מגביר את הקצוות אבל גם מגביר את הרעשים ומלא דברים אחרים, אז זה לא כדאי כ"כ.

נניח שיש תמונה ...1010101010101010

ודגמנו כל פיקסל שישי וקיבלנו 10101010.

אז קיבלנו את אותם פיקסלים אבל במציאות זו תוצאה גרועה כי אם למשל במקור הפיקסלים מייצגים גדר - ס"מ עמוד ס"מ רווח.

הקטנו פי 3 ושוב קיבלנו את אותו הדבר. אז היינו מסיקים מזה שבמציאות 3 עמוד הינו 3 ס"מ והרווחים הם גם 3 ס"מ.

בהקטנה אנחנו רוצים לראות שאיבדנו מידע!

נניח שיש תמונה g, פילטר h ועושים $g*h = i$.
איך פותרים את $g = i * ?$.

למשל $g = (1\,0\,1\,0\,1\,0\,1\,0)$

$h = (\frac{1}{2}\,\frac{1}{2})$

$\Rightarrow \quad i = (\frac{1}{2}\,\frac{1}{2}\,\frac{1}{2}\,\frac{1}{2}\,\frac{1}{2} \dots)$

במקרה הזה אי אפשר להפוך! אז יש כמה מיני תנאים ש-h צריכה לעמוד בהם. אבל נניח שזה אפשרי.

נעבור למישור התדר ונקבל $G \cdot H = I$

$I \cdot \mathcal{F}(?) = G$

$\Rightarrow \mathcal{F}(?) = \frac{G}{I}$

וכאן גם רואים למה זה לא תמיד אפשרי. אם יש ב-I אפסים אי אפשר למצוא את הפונקציה ההפוכה.

נניח שרוצים לפתור $h * ? = Id = (0 \dots 1 \dots 0)$ אז $\mathcal{F}(?) = \frac{1}{H}$

ושוב, זה עובד רק אם ב-H אין אפסים.

יש לנו תמונה עם אותיות ורעש. רוצים לראות את האותיות טוב יותר. (להגיע לתמונה בינארית)

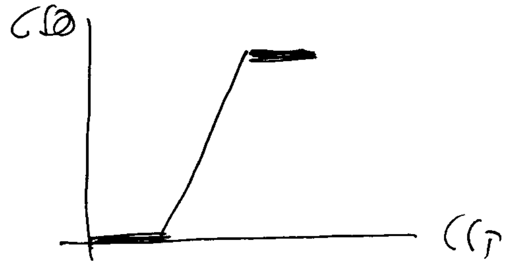
מה עדיף?

א) low pass ואז סף בתזיון או

ב) סף בתזיון ואז low-pass

טוב, זאת שאלה לא טובה כ"כ כי היא לא מוגדרת. אבל נראה בכל אופן שאפשרות השניה לא טובה. קודם כל כי בסוף יוצא משהו לא בינארי. אבל חוץ מזה אם עושים ישר איזה סף בינארי זה יכול למחוק חלק מהאותיות וזה לא טוב.

רצוי להגדיל contrast משתמשים מד"כ בפונקציה כזאת:



שיקוף של תמונה סביב ציר זו פעולה לינארית אבל אי אפשר לבצע את זה בעזרת קונבולוציה !

בהצלחה במבחן

(14) 25.02.08

עיבוד תמונה

המבחן הבא: יום שני 17/3 על החומר עד 10/3 כולל.

המבחן האחרון: 4/4 - הודעה על המיקום תבוא בהמשך

דחיסה

למה צריך דחיסה?

- תמונה צבעונית רגילה בגודל 1000x1000 תופסת בערך 3MB.
- סרט באורך שעה עם 30 frames לשנייה באורך שעה תופס 360GB.

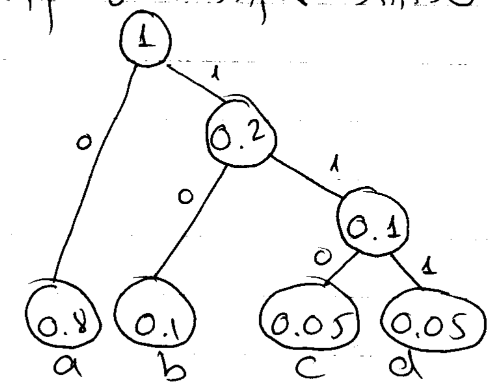
אי אפשר להתמודד עם דברים כאלה! DVD מחזיק בתוכו 5GB! צריך רק להתגבר גם על התקשורת וגם על האחסון!

למה אפשר לדחוס?

- יש אזורים דומים אחד לשני אז אולי אפשר לייצג אותם בפחות מקום
- יש שינויים שהעין האנושית לא מבחין בהם.

קוד הופמן ולמפל-זיו הן שיטות דחיסה שהן לא lossy - אחרי הדחיסה אפשר לשחזר את התמונה במלואה ושום מידע לא נאבד.

קוד הופמן מנצל את זה שהתפלגות האותיות לא אחידה - את האותיות השכיחות ביותר מקודדים ע"י קוד קצר, את האותיות הכי פחות שכיחות מקודדים ע"י קוד ארוך יותר ובסה"כ אורך הקובץ יורד.



דוגמה:

אות	שכיחות	קוד
a	0.8	0
b	0.1	10
c	0.05	110
d	0.05	111

רואים שהקידוד באורך משתנה. אם היינו עושים קידוד באורך קבוע לכל אות היה קוד באורך 2. ואילו כאן יוצא שבממוצע הקידוד הוא באורך 1.3 (ממוצע משוקלל של האורכים עם השכיחות).

דחיסת הופמן לא נותנת כלום אם לכל האותיות אותה התפלגות.

נניח שבתמונה שלנו לפיקסל אפור i יש הסתברות p_i. נגדיר את האנטרופיה של התמונה - $-\sum_{i=1}^{N} p_i \log p_i$. האנטרופיה הכי גבוהה כאשר $p_i=\frac{1}{N}$ והכי נמוכה כאשר $p_k=1$ ו- $p_j=0$ לכל $j\neq k$. בסה"כ המצב הוא איפשהו באמצע. לפי האנטרופיה של התמונה אפשר לנחש כמה ניתן לדחוס אותה.

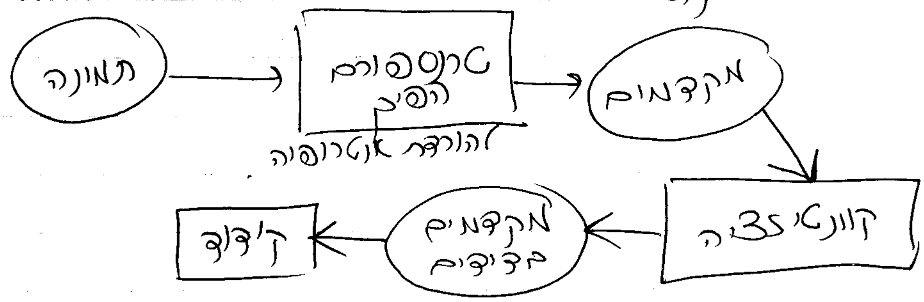
האם אפשר לקחת תמונה ולהעביר אותה לתמונה עם אנטרופיה נמוכה כך שהמעבר יהיה הפיך? אם נעשה קונבולוציה עם (1-1) נקבל את הנגזרת. אם ההנחה היא שהתמונה רציפה, רוב הפיקסלים יתאפסו וזה מוריד מאוד את האנטרופיה. אם נשמור תנאי התחלה אפשר לשחזר את התמונה במלואה.
משהו דומה לזה שימש בפירמידת הלפלסיאן - שומרים את ההפרשים, ואותם אפשר לדחוס היטב. אח"כ אפשר לשחזר את זה בצורה מלאה.
הבעיה בקוד הופמן היא שצריך לעשות את זה on-the-fly. הכיוון הזה דורש הרבה פעולות מקדימות. מצד שני, אם בקוד נפל ביט אחד או השתנה אז מתקבלות 2-3 אותיות שגויות אבל אז הוא מסתנכרן עם עצמו. זה בניגוד לקוד באורך קבוע אשר שאם נפל ביט אחד זה דופק את כל האותיות שאח"כ.

למפל-זיו הציעו שיטה שמגיעה לאנטרופיה בלי שצריך כמו להופמן היא מסתכלת על סדרות מאורכים שונים ולא על אותיות בודדות. השיטה הזאת נותנת תוצאות יותר טובות מהופמן וזה מה שנפוץ היום. חוץ מזה אפשר גם לעשות אותה on-the-fly.
איך השיטה עובדת? יש טבלה של מחרוזות רק שהרישא של כל מחרוזת בטבלה נמצאת לפניה.
איך בונים את הטבלה?
(1) מתחילים מסימנים בודדים
(2) קוראים סימן k ושומרים אותו ב-w. $w \leftarrow k$
(3) קוראים עוד סימן k. אם זה סוף הקובץ רושמים ב-output $code(w)$
אם wk (השרשור) בטבלה אז $w \leftarrow wk$ וחוזרים ל-(3)
אם wk לא בטבלה:
(א) כותבים $code(w)$ (w כן נמצא בטבלה ולכן יש לו קוד)
(ב) מכניסים את wk לטבלה
(ג) $w \leftarrow k$ (מחליפים את w לגמרי) וחוזרים ל-(3).
למשל, נניח שהקובץ הוא abcbcd
הפלט יוצא: 1, a, ---

1	a
2	b
3	c
4	d
5	ab
6	bc
⋮	

עד עכשיו לא דנו את העובדה שיש שינויים שהעין לא מבחינה בהם. למשל אם יש לנו תמונה שכולה נראית רעש לבן אזי דחיסה אנטרופית היא בלתי אפשרית. אבל אם נראה שתי תמונות של רעש לבן הסטטיסטיקה שונה מאוד מאוד אבל נגיד שמדובר באותה התמונה. מבחינת העין רעש לבן זה רעש לבן וזהו.

באופן כללי, מערכות דחיסה נראות ככה:



איבוד האינפורמציה מתרחש רק בקוונטיזציה! אם לא עושים קוונטיזציה אז הדחיסה היא ללא איבוד. בוודאי אם מרשים איבוד אינפורמציה הדחיסה עובדת הרבה יותר טוב.

<u>טרנספורמציות בדחיסה:</u>

- Run Length - למשל אפשר להשתמש בזה במקרה שיש רק שני צבעים ורצפים ארוכים של אותו צבע. כל שורה מייצגים ע"י סדרה $(n_1, n_2, n_3, \ldots)$ כאשר n_{2i} מייצג את אורך הרצף השחור ו-n_{2i+1} מייצג את אורך הרצף הלבן. אפשר להרחיב את זה ליותר צבעים אבל ככל שיש יותר צבעים הסיכוי לרצפים ארוכים של אותו צבע יורד ואז זה כבר לא יעיל...
- DCT (Discrete Cosine Transform) - דומה מאוד לפורייה, אבל שולחים רק את המקדמים הממשיים. JPEG משתמש בטרנספורם הזה. (גם Wavelets (JPEG2000)).

<u>איך עובד JPEG?</u>

(1) שינוי צבע ל- (Y, C_b, C_r)
Y - שחור/לבן; C_b, C_r - רכיבי צבע

(2) חלוקה לבלוקים בגודל 8x8 (כדי שיהיה קל לממש את זה בחומרה)

(3) מבצעים DCT על כל חלון:

$$F(u) = \sum_{x=0}^{N-1} f(x) \cos\left(\frac{\pi}{N} u \left(x + \frac{1}{2}\right)\right)$$

$$f(x) = \frac{1}{2} F(0) + \sum_{u=1}^{N-1} F(u) \cos\left(\frac{\pi}{N} u \left(x + \frac{1}{2}\right)\right)$$

בניגוד לפורייה, המטריצה כאן היא רפלקציה - בפורייה מגיעים למחזור שלם ב-N ואילו ב-DCT מגיעים למחזור שלם ב-2N.

(4) למקדמים עושים קוונטיזציה (למשל מחלקים ב-8 וזורקים את השארית)

(5) דוחסים את המקדמים ב-הופמן ומשדרים.

אם עושים דחיסה מאוד חזקה מתחילים לראות בתוצאה את הקוביות.
וזה ממש גרוע...

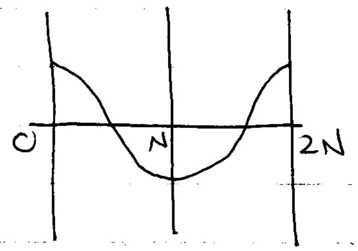
איך בודקים איכות של דחיסה? JND - Just Noticable Difference
נותנים לאנשים להסתכל על התמונות ובודקים כמה אנשים רואים שיש הבדל.
אין מנגנון אוטומטי שמחשב את זה.

16 3/12/08
עיבוד
תמונה

למה טרנספורם קוסינוס?

1) לא ממשי, אז זה מקל קצת על החישובים

2) סימטריה - מחזור של 2N במקום N בפורייה.



אחרי הטרנספורם $f(X+N) = f(N-X)$ ואילו

בטרנספורם פורייה התמונה מקיימת $f(X+N) = f(X)$

קוסינוס

b	d
p	q

פורייה

b	b
b	b

למה זה טוב לדחיסה מה שקורה בקוסינוס? בגלל הקפיצות הגבוליות בתמונה אחרי פורייה יש שם תדרים גבוהים ותדרים גבוהים זה משהו שקשה לדחוס. אחרי קוסינוס המעברים חלקים יותר, הטרנספורם לא מכיל תדרים גבוהים ולכן קל יותר לדחוס את זה.

JPEG 2000 - היתרון הגדול שלו מול JPEG הוא שאין חלוקה לריבועים. אז אם עושים דחיסה מאוד מאוד חזקה JPEG נראה כמו לגו - רואים את הריבועים, ואילו ב-JPEG-2000 לא רואים דבר כזה והתמונה עדיין נראית בסדר. חוץ מזה יש עוד מיני תכונות כמו מימוש נוח בחומרה, אפשרות להעביר חלקים ועוד.

טרנספורמציות גאומטריות

קואורדינטות הומוגניות: במקום להסתכל על התמונה כמערכת קואורדינטות (X,Y) נעבוד בקואורדינטות $(X,Y,1)$. נעשה 6 מיני פעולות על הקואורדינטה הזאת ונקבל (X',Y',Z'). בסוף נחזיר את זה ל- $(\frac{X'}{Z'}, \frac{Y'}{Z'})$. עושים את זה רק שנוכל לעבוד במרחבים (תמיד נשווה ל-1). בקיצור המעבר בין מרחב רגיל ל- מרחב הומוגני הוא:

מרחב רגיל		מרחב הומוגני
(X,Y)	$\longmapsto$	$(X,Y,1)$
$(\frac{X}{Z}, \frac{Y}{Z})$	$\longleftarrow$	(X,Y,Z)

איזה טרנספורמציות אפשר לעשות?

① הזזה - $\vec{x}' = \vec{x} + \vec{t}$ בקואורדינטות הומוגניות אפשר להציג את זה באופן הבא:

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

למה זה טוב? כי זו דרך נוחה להרכיב טרנספורמציות. לכן נרצה להציג הכל בעזרת כפל מטריצות ולא בעזרת חיבור או דברים אחרים.

② סיבוב - $\vec{x}' = \underbrace{\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}}_{R} \vec{x}$ מטריצת הסיבוב R

מקיימת $RR^t = I$, $|R| = 1$

③ סיבוב והזזה אפשר לעשות ביחד בקואורדינטות הומוגניות!

$$\vec{x}' = \begin{pmatrix} \cos\theta & -\sin\theta & a \\ \sin\theta & \cos\theta & b \\ 0 & 0 & 1 \end{pmatrix} \vec{x}$$

תרגיל:
מה קורה כאן קודם - סיבוב או הזזה?

④ הזזה + סיבוב + מתיחה

$$\vec{x}' = \begin{pmatrix} p & -q & a \\ q & p & b \\ 0 & 0 & 1 \end{pmatrix} \vec{x}$$

כדי למצוא את ה- p, q, a, b שנותנים מספיק נקודות צריך לפתור משוואות.

הזזה משמרת אורכים וכיוונים (אופקי נשאר אופקי ואנכי נשאר אנכי)
הזזה + סיבוב משמר אורכים אבל לא כיוונים אלא זווית
הזזה + סיבוב + מתיחה משמר רק זווית.

הפכת פעולות גאומטריות בעקבות התמונה, לכן אמורים להפעיל כמה פעולות עדיף לא לעשות את זה בשרשרת, אלא קודם להכפיל את כל המטריצות הרלוונטיות ואז להפעיל על התמונה.

⑤ טרנספורמציות אפיניות:
כאן נשמרים רק קווים מקבילים

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

⑥ פרספקטיבה:

$$\begin{pmatrix} \tilde{x}' \\ \tilde{y}' \\ \tilde{z}' \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\Rightarrow \quad x' = \frac{ax+by+c}{gx+hy+1} \qquad y' = \frac{dx+ey+f}{gx+hy+1}$$

כאן נשמרים קווים ישרים.

* באתר יש קובץ בן כ-80 עמודים שמומלץ מאוד לקרוא.

סיכום:

טרנספורמציה	פרמטרים	תכונות נשמרות
הזזה	2	מרחקים, כיוונים
+ סיבוב	3	מרחקים
+ מתיחה	4	זוויות
אפיני	6	מקבילות
פרוייקטיבי	8	קווים ישרים

נניח שיש לנו תמונה ואנחנו רוצים לעשות לה משהו. איך מבצעים טרנספורמציות גיאומטריות (Texture Mapping, Warping)

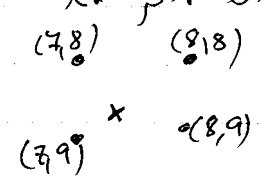
יש לנו טרנספורמציה A:

$$\boxed{f} \xRightarrow{A} \boxed{g}$$

איך מחשבים את g? יש כמה גישות

- forward projection: נעבור על (x,y) ב-f, נחשב (x',y'), נכניס את הערך של $f(x,y)$ ב-$g(x',y')$.
הבעיה הגדולה כאן היא קוונטיזציה של הקואורדינטות. הכי קל לראות את זה בהגדלה - פשוט נשארים הרבה מקומות ריקים!
מהשיטה הזאת מנסים להימנע לחלוטין.
- back projection: נעבור על (x',y') ב-g, נחשב $\binom{x}{y}=A^{-1}\binom{x'}{y'}$, נכניס את הערך של $f(x,y)$ לתוך $g(x',y')$.
לא כל טרנספורמציה היא הפיכה. אך אם היא לא הפיכה אי אפשר להשתמש בשיטה הזו.

גם פה יש בעיה בקוונטיזציה אבל היא פחות חמורה. יכול להיות ש-$A^{-1}\binom{x'}{y'}$ לא ערך שלם. $f(7.25,8.85)=?$ אם יש ערך לא שלם זה אומר שנופלים בין ארבעה פיקסלים.

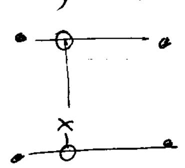


אז צריך לעשות אינטרפולציה.

- סדר "0" - הפיקסל הקרוב. במקרה זה $(7,9)$
- סדר "1" - אינטרפולציה בי-לינארית - נחשב את $f(7.25,9)$
ע"י אינטרפולציה לינארית של $f(7,9)$ ו-$f(8,9)$:

$$f(7+a,9)=f(7,9)+a\left(f(8,9)-f(7,9)\right)$$

נחשב את $f(7.25,8)$ באופן דומה ואז נעשה אינטרפולציה לינארית בכיוון y



בגלל האינטרפולציות האלה טרנספורמציות גיאומטריות מזיקות
לתמונה! ה back projection מטשטש את התמונה.

הנושא הבא: בהינתן שתי תמונות איך מוצאים את הטרנספורמציה
ביניהן?

(18) 10.03.08
עיבוד
תמונה

התאמת תמונה

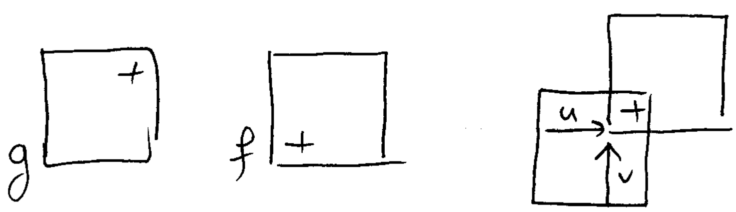
מטרה: בהינתן נתונים לבדוק כמה התמונות זהות.

מה השימושים -

① איפה בתוך תמונה נמצא קטע נתון?
כמובן מה שמחפשים לא נראה <u>בדיוק</u> כמו בתמונה כי אחרת זה לא מעניין.

② איפה בתוך אוסף תמונות נמצא קטע? (Flickr - אתר עם מלא מלא תמונות). Flickr זה אתר מאוד יעיל שמעשה שינה את עולם עיבוד התמונות. למשל, נניח שיש לנו תמונה עם חור. הולכים ל-Flickr ובוחנים מליון תמונות. מוצאים את התמונה שהכי דומה לתמונה שלנו ומשלימים.

③ מה ההזזה בין שתי תמונות?



השימוש הכי נפוץ של הזזה הוא בפנורמה. אם יודע ששתי תמונות הן הזזה אחת של השניה אז לא באמת צריך לשמור את שתיהן באופן מלא. צריך לשמור אחת מהן ואת גודל ההזזה.

בעיות -

- רעש פוטומטרי - כל צבע בתמונה משתנה - בגלל סוג מצלמה, רעש בסנסורים ...
- רעש גאומטרי - אולי לא כל הפיקסלים זזו בדיוק באותו היסט. בכל התמונות יש עיוותים, למשל קווים ישרים במצלמה נראים עקומים

איך בכלל מודדים תאימות בין תמונות?

① סכום ריבועי הפרשים. $\sum_x \sum_y (f(x,y)-g(x,y))^2$

אם התמונות זהות השגיאה היא 0.

אבל אם למשל יש לנו תמונה של שריג, אם אנחנו מזיזים בפיקסל אחד השגיאה תצא עצומה אבל התמונות הרי דומות.

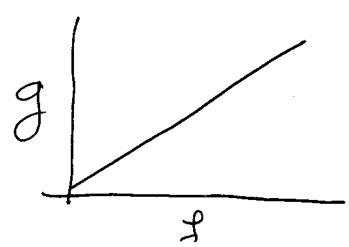
אז נניח שלא נותנים לנו תמונות כאלה בעייתיות. בכל זאת יש כאן בעיה גדולה: ככל שההפרש גדול יותר אנחנו מעלים אותו בריבוע ונותנים לו עוד יותר חשיבות. אז אם יש לנו שתי תמונות, אפילו אם מצאנו את ההזזה הנכונה אם יש רעש עדיין תצא שגיאה גדולה.

② ערכים מוחלטים של הפרשים $\sum_{x,y} |f(x,y)-g(x,y)|$

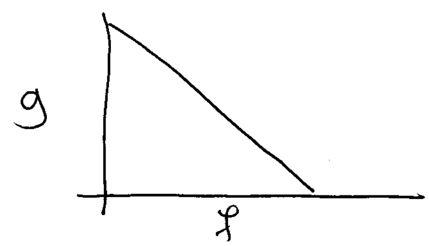
עם זה עובדים פחות כי אי אפשר לגזור את זה ולמצוא את המינימום. אבל בכל זאת אפשר לפתור את זה עם תכנון לינארי.

③ היסטוגרמה משותפת - זו מטריצה שגודלה לפי מספר דרגות האפור. עוברים על כל הפיקסלים (x,y) ובמקום $(f(x,y), g(x,y))$ במטריצה מוסיפים 1.

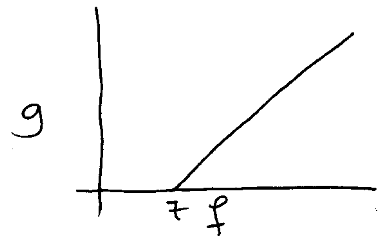
אם התמונות זהות ~~התמו~~ המטריצה תיראה רק:



אם התמונות הן נגטיב אחת של השניה אז:



אם $g = f+7$ אז



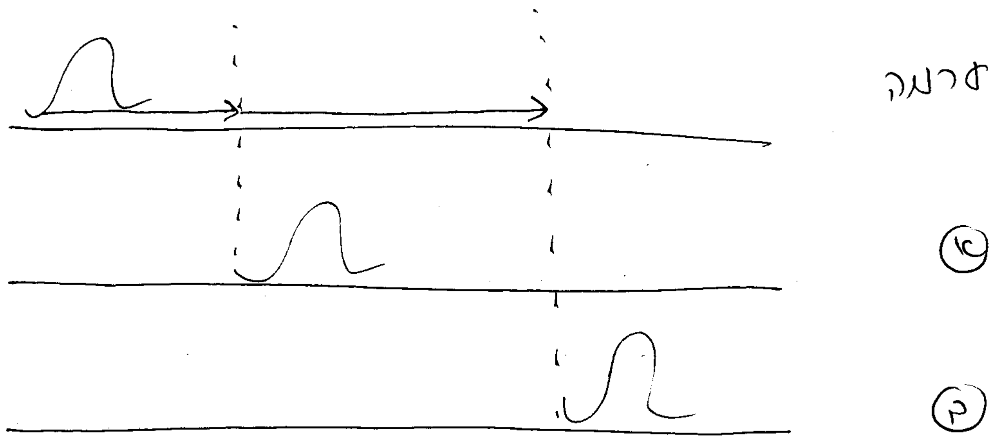
בכל אופן האנטרופיה של ההיסטוגרמה הזאת היא מדד לדמיון בין התמונות. אנטרופיה נמוכה = תמונות דומות.

השיטה הזאת אינווריאנטית לפונקציות של דרגות אפור (LUT).

אם אין קשר בין f ל-g להיסטוגרמה המשותפת תהיה אנטרופיה גבוהה.

(4) Earth Moving Distance - (EMD)

נניח שאנחנו צריכים להזיז ערמת עפר:



מה עדיף - ל-(א) או ל-(ב).

לפי סכום ריבועי ההפרשים / ערכים מוחלטים של הפרשים המרחק בין הערמה ל-(א) והמרחק בין הערמה ל-(ב) הם זהים! אבל הרי ברור ש-(א) זה יותר קרוב!

ב- EMD מחשבים את סכום המרחקים המינימלי שצריך להזיז כל פיקסל.

השימוש של זה הוא בעיקר במרחק בין היסטוגרמות ומרחק בין צורות בינאריות.

נשים לב כדי שבכלל אפשר יהיה לדבר על EMD צריך לעשות נרמול לרמת האפור של התמונה.

חישוב הזזה ע"י מזעור מרחק ריבועי

$$E^2(u,v) = \sum_{x,y} \left(f(x,y) - g(x-u, y-v)\right)^2$$

אנחנו רוצים למצוא את u,v שממזערים את $E^2(u,v)$

נניח ש- g הזזה של f. האם זאת הנחה סבירה?

במשטח למברטי זה מסדר כי זווית ההסתכלות לא משנה את רמת האפור ואילו לגופים מבריקים זה לא טוב. ברוב המקרים זו הנחה סבירה.

$$E^2 = \sum f^2 - 2\sum fg + \sum g^2$$

כדי למזער את E^2 מספיק למקסם את $\sum fg$ משום ש- $\sum f^2$ ו $\sum g^2$ לא תלויים ב- u,v.

$C(u,v) = \sum f(x,y)g(x-u,y-v)$ נקרא <u>הקורלציה</u> של f ו- g.

כדי שהקורלציה לא תהיה תלויה מהוספה והחסרה של כמות אפור צריך להחסיר מכל תמונה את הממוצע שלה:

$$\sum [f(x,y) - \bar{f}][g(x-u, y-v) - \bar{g}]$$

עוד אנחנו רוצים שזה יהיה אינווריאנטי למכפלה בקבוע אז נחלק בשונות:

$$\frac{\sum_{x,y} [f(x,y) - \bar{f}][g(x-u, y-v) - \bar{g}]}{\sqrt{\sum_{x,y}(f(x,y) - \bar{f})^2 \cdot \sum_{x,y}(g(x,y) - \bar{g})^2}} = N_c(u,v)$$

$\downarrow$

Normalized Correlation

כמו שאמרנו בהתחלה, זה בטוח שמה שאנחנו מחפשים הוא בגודל הנכון, הסיבוב הנכון וכן הלאה. אז צריך לחפש לפי כל אחד מהפרמטרים ואז זמן החיפוש עולה אקספוננציאלית עם מספר הפרמטרים. זו בעיה!

דוגמה:

תמונה 1000x1000 וההצבה בתחום 100x100.
בשביל לחשב $N_c(u,v)$ ל- u,v ספציפיים צריך 10^6 חישובים.
יש 10^4 זוגות (u,v). אז בסה"כ יש 10^{10} פעולות
כדי רק לבדוק את ההצבה. זה לא משהו שאפשר לעשות בזמן
אמת. ואם עוד נוסיף עוד פרמטרים כמו מתיחה, סיבוב
או פרספקטיבה אז זה כבר ממש בלתי אפשרי.

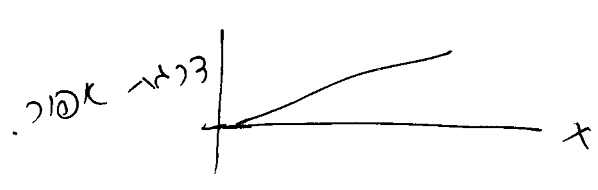
פירמידה

נניח שהקטנו את התמונה פי 2. התמונה 500x500 והתחום
50x50. אז מספר הפעולות ירד פי 2^4.
אחרי עוד הקטנה מספר הפעולות ירד עוד פי 2^4, כלומר סה"כ
פי 2^8. ואם עושים 3 רמות לפירמידה אז מקטינים את
כמות החישובים פי $2^{12} = 4096$.
אז אפשר קודם לחפש ברמה הכי גבוהה וזה מצמצם את מרחב
החיפוש וככה לרדת ברמות.
השיטה הנ"ל עובדת רק בתמונות שיש בהן רק תדרים גבוהים-
למשל אם יש לנו תמונה של רעש לבן. למזלנו הרבה לא
מתעסקים בתמונות כאלה.

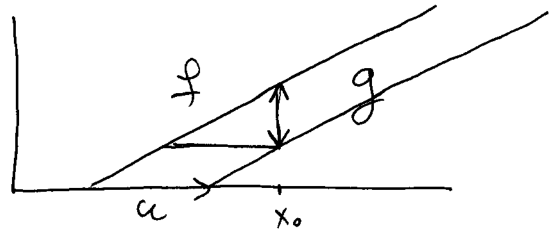
הצבות תת פיקסליות - איך מוצאים הצבה של פיקסל חצי?
אנחנו מחפשים את ההצבה בפיקסלים שלמים. יש לנו
את פונקציית הקורלציה $N_c(u,v)$. אפשר להתאים
אותה בערך פרבולואיד ואז אפשר אנליטית למצוא
את המקסימום שלו וזה ייתן את ההצבה תת-פיקסלית.

נגזרות בזמן ומרחב (Lucas Kanade)

נניח שיש תמונה חד ממדית בצורת רמפה



אם נזיז אותה כ- u נקבל



$$f(x) = g(x+u)$$

$$\frac{f(x_0) - g(x_0)}{u} = \frac{\partial g}{\partial x} = f_x$$

$$\Longrightarrow \quad u = \frac{f_t}{f_x}$$

$$f_t = f(x_0) - g(x_0)$$

ככל ש- f_t גדול יותר ההזזה גדולה יותר. ככל ש- f_x גדול יותר ההזזה קטנה יותר.

בפו ממש זה קצת יותר בעייתי. נמשיך ביום חמישי.

<u>המבחן</u>: החומר הוא כולל את יום חמישי גם (כי התבטל שיעור)

שיעור חזרה: יום חמישי 2-4

21

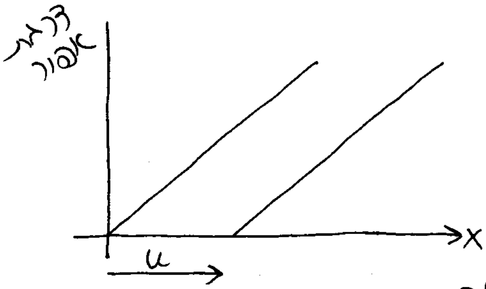
13.03.08

עיבוד תמונה

מציאת הזזה בין תמונות

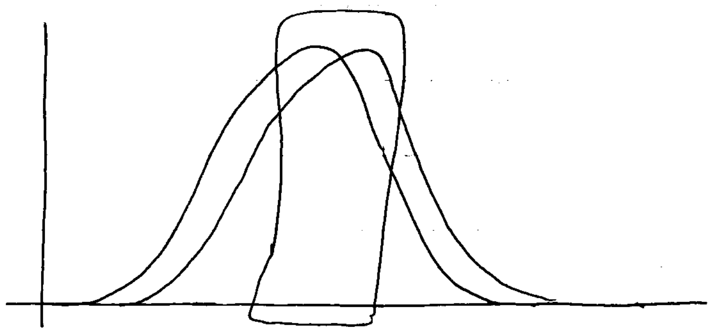
נגזרות במרחב וזמן Spacio-Temporal Derivatives

נניח שיש תמונה חד ממדית - כלומר שורה אחת ונניח שדרגות האפור עולות בצורה לינארית. אז את ההזזה אפשר למצוא באופן הבא:



$$u = \frac{f_x(x_0)}{f_t(x_0)}$$

למעשה הנוסחה הזאת עובדת בכל אזור שבו התמונה נראית לינארית. למשל:



רק באזור הזה השיטה לא תעבוד.

באופן כללי החלקה של תמונה מגדילה את הסיכוי שהשיטה תעבוד כי טשטוש מגדיל את האזורים הלינאריים ואז אם נבחר פיקסל באקראי הסיכוי גדל שהסביבה שלו לינארית.

נעבור לתמונות דו-ממדיות

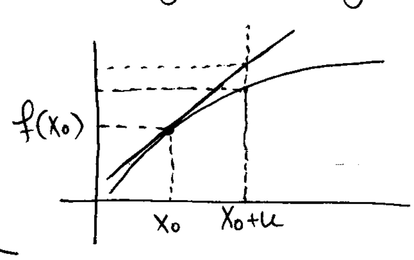
בשלב הראשון נניח שפרט להזזה ב- (u,v) הצבעים בתמונה לא השתנו. כלומר

$$f(x,y) = g(x-u, y-v) \qquad (1)$$

נעשה פיתוח טיילור ונקבל תחת הנחת לינאריות או הזזה קטנה

$$(2) \quad g(x-u, y-v) \approx g(x,y) - u \cdot g_x(x,y) - v \cdot g_y(x,y) + \dots$$

תזכורת:



$$f(x_0+u) \approx f(x_0) + u f_x(x_0)$$

נציב את (2) ב-(1) ונקבל

$$f(x,y) \approx g(x,y) - u g_x(x,y) - v g_y(x,y)$$

$$\Rightarrow \underbrace{f(x,y) - g(x,y)} + u g_x(x,y) + v g_y(x,y) = 0$$

זאת למעשה הנגזרת בזמן -
איך פיקסל מסוים משתנה
עם הזמן...

אז קיבלנו שלכל (x,y) שמקיימים את ההנחות הנ"ל

$$\boxed{u g_x + v g_y + g_t = 0}$$

הנוסחה הזאת נקראת Optical flow constraint equation

בעזרת המשוואה הזאת לכל פיקסל מתקבלת משוואה. אנחנו רוצים למצוא (u,v) - הזוג שממזערת את הסטיה מהאילוץ הלזה:

$$E^2(u,v) = \sum_{x,y} (u g_x + v g_x + g_t)^2$$

זאת השגיאה שאנחנו ממזערים משום שסכום ריבועים זה משהו שאנחנו יכולים להתמודד איתו מתמטית:

$$0 = \frac{\partial E^2}{\partial u} = 2 \sum_{x,y} g_x (u g_x + v g_y + g_t) =$$

$$= 2 \sum_{x,y} (u g_x^2 + v g_x g_y + g_x g_t)$$

$$\Rightarrow u \sum_{x,y} g_x^2 + v \sum_{x,y} g_x g_y + \sum_{x,y} g_x g_t = 0$$

באותו אופן נטפל ב-v:

$$0 = \frac{\partial E^2}{\partial v} = u \sum_{x,y} g_x g_y + v \sum_{x,y} g_y^2 + \sum_{x,y} g_t g_y$$

קיבלנו שתי משוואות בשני נעלמים:

$$\begin{pmatrix} \sum_{x,y} g_x^2 & \sum_{x,y} g_x g_y \\ \sum_{x,y} g_x g_y & \sum_{x,y} g_y^2 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} -\sum_{x,y} g_t g_x \\ -\sum_{x,y} g_t g_y \end{pmatrix}$$

נפתור את המשוואות ונקבל את הפתרון. מי שמפחד מהפיקסלים שבקצוות יכול לא להכניס אותם לסכימה..

התהליך:

(1) מחשבים נגזרות g_t, g_y, g_x

(2) מחשבים את ששת הסכומים

(3) פותרים את המשוואות - אלא שרוב הסיכויים הם שבמציאות למשוואות האלה אין פתרון ולכן פועלים בדרך הבאה:

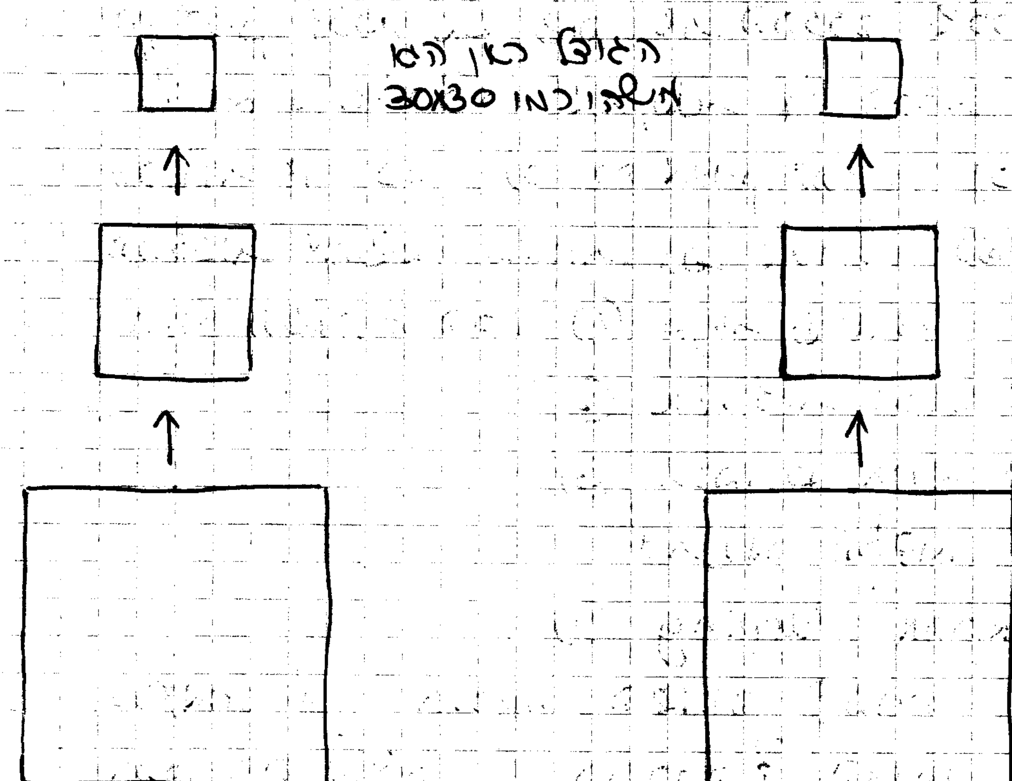
(4) פתרון בפירמידות + איטרציות.

תרגיל: לפתח 3 משוואות ב-3 נעלמים עבור

$$u(x,y) = u_0 + ax$$
$$v(x,y) = v_0 + ay$$

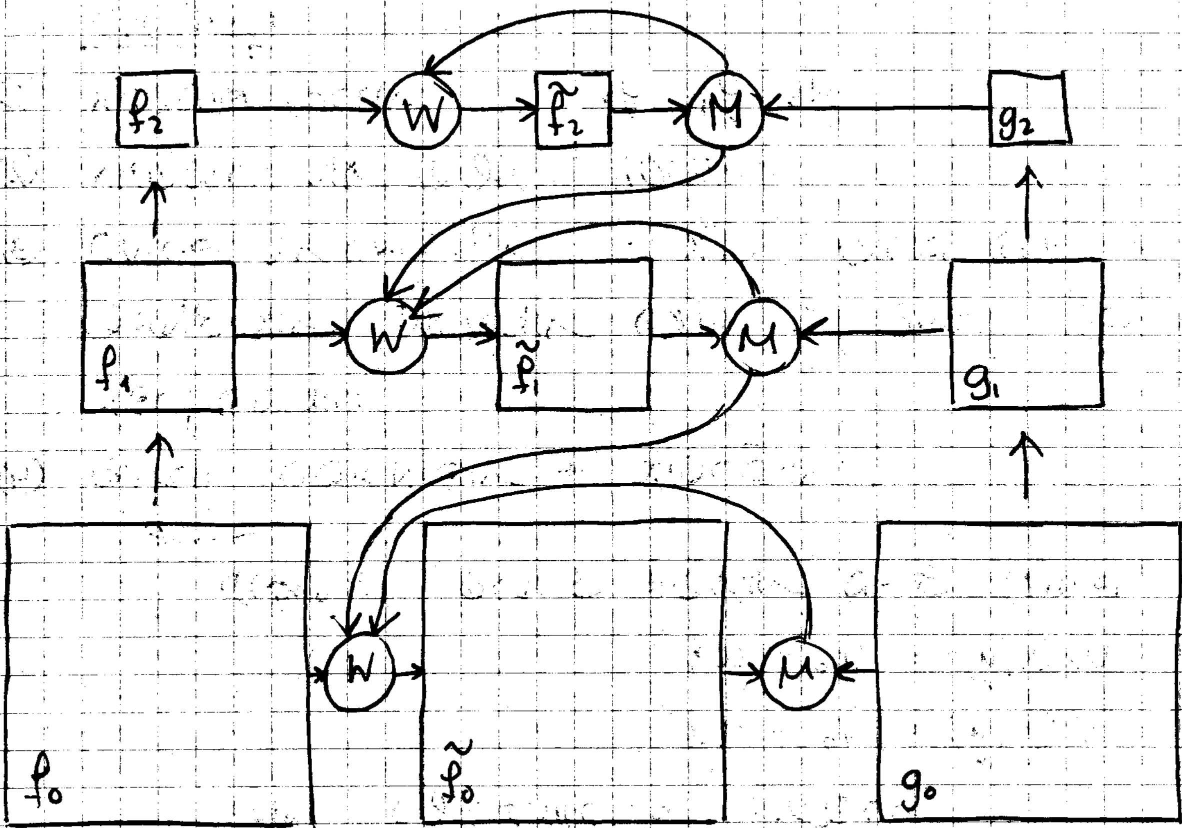
כלומר כשהתמונה גם מתנפחת

נסביר קודם באופן ידני ראשית, נבנה פירמידות של התמונות:



מתחילים מהתמונות הקטנות. יש שני תהליכים שמשתמשים בהם: M - חישוב התנועה ע"י פתרון המשוואות

W - הזזת התמונה ובדיקת הפתרון



בשלב הראשון $\tilde{f}_2 = f_2$. מחשבים את התנועה. אם הנגזרת בזמן הגיעה 0 אז יופי. אחרת מחשבים את התנועה שוב. ככה עושים שוב ושוב עד שנמאס לנו. רק צריך לזכור בכל שלב את ההזזה שצריך לבצע: $\tilde{f}_2$ בכל איטרציה מתקבלת ע"י הזזת f_2 בהזזה הכוללת עד כה (כי זה מדויק יותר). נזכור שאת ההזזות מחשבים בעזרת כפל מטריצות כפי שלמדנו.

סה"כ השלבים הם:
1. בניית פירמידה
2. איטרציות ברמה העליונה
3. הרכבת כל תנועה באיטרציה לתנועה הקודמת
4. Warping מהתמונה המקורית

כשנמאס לנו מהרמה העליונה עוברים לרמה הבאה. מהם הפרמטרים של ההזזה? במעבר מרמה i לרמה $i+1$ צריך להגדיל את ההזזה כל ממד פי 2! ככה ממשיכים לעשות איטרציות בכל הרמות עד שאנחנו מרוצים.

(23)

מה המחיר של האלגוריתם?

הדבר הכי יקר הוא בניית הפירמידות אבל זה קורה רק פעם אחת.

הדבר הכי יקר הבא הוא ה- Warping בגלל שמדובר שם באינטרפולציה אבל סה"כ כל שעושים ברמה מספר האיטרציות יורד אז זה פחות נורא.

הדבר הבא שצריך לשים לב אליו הוא שבכל כמה לראות צריך לחשב מחדש את הנגזרות. אבל בגלל שאת התמונה g לא משנים לא צריך לחשב את g_x ואת g_y בכל איטרציה, אלא רק את g_t (בגלל ש- f_i דווקא כן משתנה אז הנגזרת בזמן משתנה בכל איטרציה) אז את g_x, g_y צריך לחשב רק פעם אחת לכל רמה.

מה עושים אם יש גם סיבוב בתמונה?

$$x' = x_0 + u(x,y) = u_0 + \cos\alpha\, x - \sin\alpha\, y$$
$$y' = y_0 + v(x,y) = v_0 + \sin\alpha\cdot x + \cos\alpha\, y$$

אם ננסה לפתור את המשוואות (ע"י גזירה לפי α גם) זה יוצא נורא מסובך. אז עושים את הקירוב הבא:
לצורך חישוב התנועה M תחת ההנחה שהסיבוב קטן משתמשים בקירוב הבא:

$$\cos\alpha \approx 1$$
$$\sin\alpha \approx \alpha$$

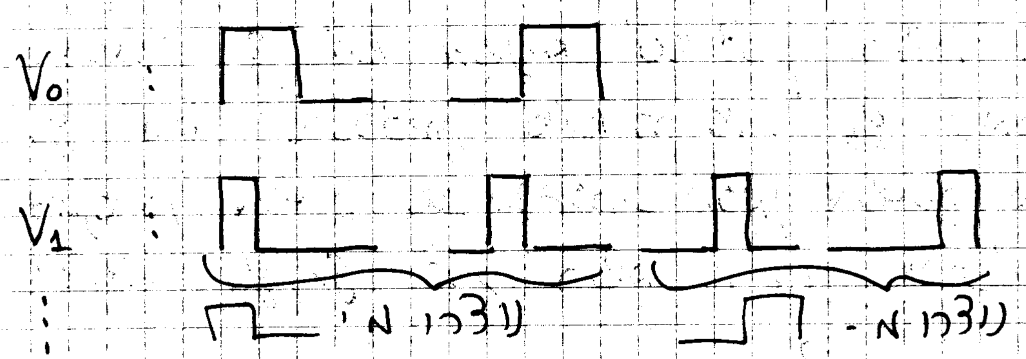
אז
$$\boxed{\begin{aligned} u(x,y) &= u_0 - \alpha y \\ v(x,y) &= v_0 + \alpha x \end{aligned}} \Leftarrow \begin{cases} x' = u_0 + x - \alpha y \\ y' = v_0 + \alpha x + y \end{cases}$$

אז בונים 3 משוואות ב-3 נעלמים לפי המשוואות החדשות שמצאנו. אבל זה רק קירוב לפיתרון וב- Warping משתמשים בנוסחה המדוייקת עם $\sin$ ו- $\cos$.
שוב, זה עובד בהנחה של זוויות קטנות!

חזרה למבחן

מה הקטע עם ה- wavelets?

אנחנו מנסים למצוא איזשהו ייצוג של תמונה ברמות
רזולוציה שונות. לכל רמת רזולוציה יש בסיס.
נדבר עכשיו על תמונות חד ממדיות
שמשתמש בבסיס Haar אפשר לדבר על הבסיסים
המתאימים:



כך יש היררכיה של רזולוציות כאשר הבסיסים
למעשה משכפלים את עצמם ומוסיפים את מה שחסר.
לכל i נסמן $W_i = V_i \setminus V_{i-1}$ זאת השארית שצריך
להוסיף ל- V_{i+1} כדי לפרש את הרזולוציה הגבוהה
יותר. אז W_i נותן איזשהו ייצוג למה שצרכנו.

מה השימוש של הדבר הזה?
נזכור מה עשינו בפירמידת הגאוסיאן - בכל שלב טשטשנו
תמונה ושמרנו את ההפרש מהרמה הקודמת.
יחד עם פירמידת הגאוסיאן יש לנו ייצוג של התדרים
הנמוכים ושל התדרים הגבוהים. וזה במחיר של ניפוח
של התמונה. בלפלסיאן קיבלנו אנטרופיה נמוכה אבל
במחיר של יותר מקום.
$\therefore$ Wavelets באים לפתור את בעיית המקום - אפשר
לשמור את תמונה ה- low pass ותמונה ה- high pass
אבל בלי להוסיף פיקסלים.

24

נסתכל בבסיס [1,1] , [1,-1]. אז

ה- low-pass מתקבל ע"י קונבולוציה עם $\frac{1}{2}[1,1]$

וה- high-pass מתקבל ע"י קונבולוציה עם $\frac{1}{2}[1,-1]$

למשל: $A = a_0 \; a_1 \; a_2 \; a_3 \; a_4 \; a_5 \; a_6 \; a_7$

low pass $\frac{a_0+a_1}{2} \quad \frac{a_2+a_3}{2} \quad \frac{a_4+a_5}{2} \quad \frac{a_6+a_7}{2}$

highpass $\frac{a_0-a_1}{2} \quad \frac{a_2-a_3}{2} \quad \frac{a_4-a_5}{2} \quad \frac{a_6-a_7}{2}$

קיבלנו סה"כ שמונה מספרים חדשים ומהם אפשר לשחזר באופן מלא את A המקורית. למשל

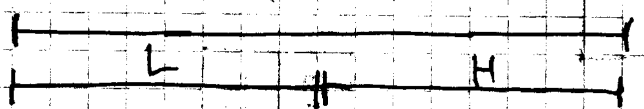
$\frac{a_0+a_1}{2} \quad - \quad \frac{a_0-a_1}{2}$

נותנות שתי משוואות בשני נעלמים ומהן אפשר לשחזר את a_0 ואת a_1.

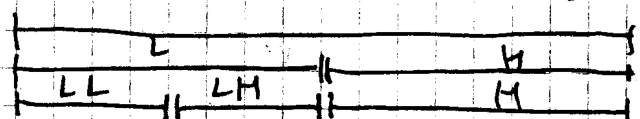
זה ה- wavelet הכי פרימיטיבי - הוא מאפשר לנו לקבל תמונת low-pass ותמונת high-pass וכל זה ללא הגדלת מספר הפיקסלים.

הבעיה היא שאלה ה- wavelets הכי גרועים ויש wavelets הרבה יותר טובים! הטיבה שלהם נמדדת ביחס לאנטרופיה שנוצרת.

איך מקטינים כאן את האנטרופיה? בשלב הראשון חילקנו את התמונה לשניים:



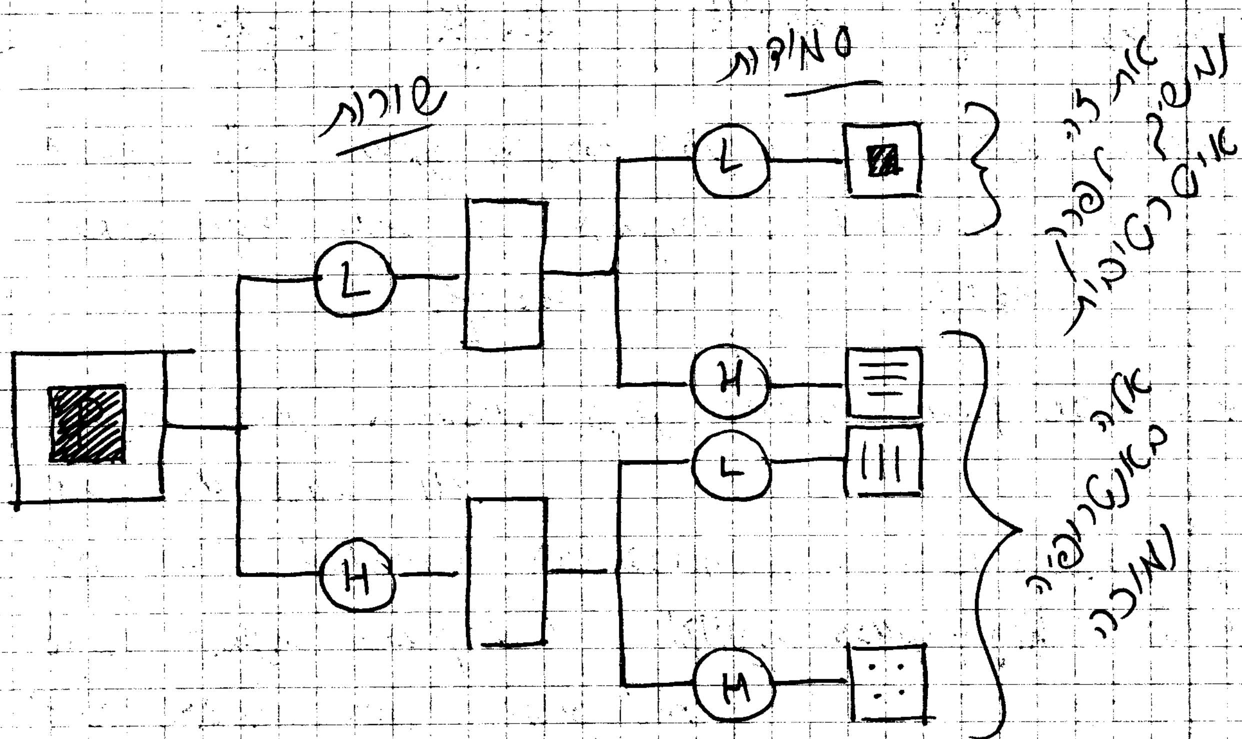
H זה רק חלק עם אנטרופיה נמוכה אז אנחנו עוזבים אותו ועובדים שוב על L:



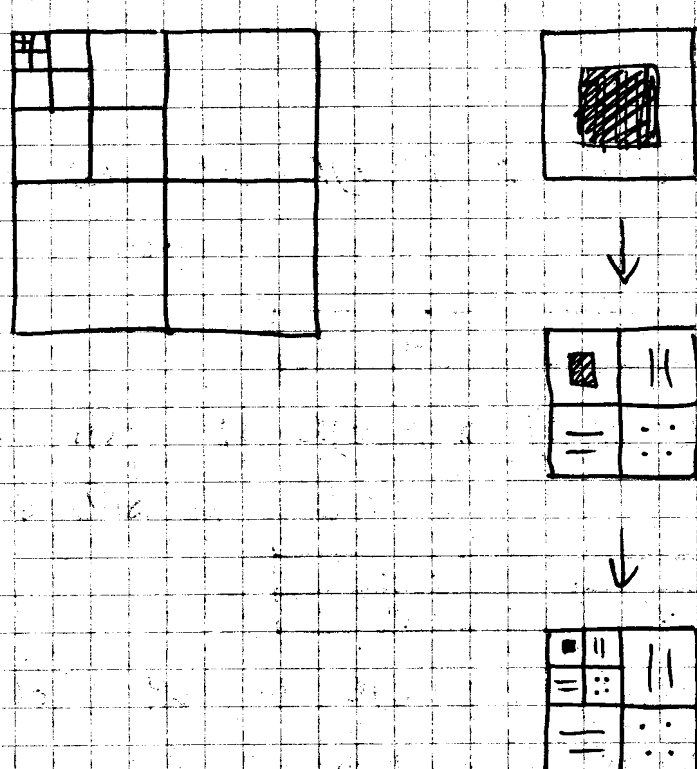
עכשיו ל- H ול- LH יש אנטרופיה נמוכה ונשארים לעבוד על LL. ככה עושים שוב ושוב עד שנמאס לנו.

כל זה תחת ההנחה שהתדרים הגבוהים הם עם
אנטרופיה נמוכה, כלומר המידע שהם מחזיקים הוא בעל
אנטרופיה נמוכה.
ה-wavelet-ים נמדדים באנטרופיה שמתקבלת
לתמונה אחרי השימוש בהם.

מה עושים עם תמונה דו-ממדית? פועלים על כל ממד בנפרד:



מסה"כ ברור גם שזה שומר על כמות הפיקסלים:



אז הקשר עם ה- wavelets הוא שצריך למצוא
את הזוג הטוב. אנחנו יודעים שקיים זוג כזה - [1 1]
ו-[1 -1] אבל רוצים למצוא משהו יותר טוב.
זו הדרך הכי טובה שמצאו להוריד את האנטרופיה בתמונות
בצורה כזאת (בניגוד לקביעת בטרנספורם קוסינוס) מבלי
להגדיל את מספר הפיקסלים.

<u>LZ vs. Huffman</u>: קוד האפמן לא משתמש בשום
אינפורמציה של השכנים ואם כבר מגיעים לתמונה שהיא כמעט
רעש לבן אז באמת האפמן כנראה יתן את התוצאה
הכי טובה. היתרון של למפל-זיו הוא שהם למעשה עובד
על סדרות ולכן בד"כ מקבל תוצאות טובות יותר.

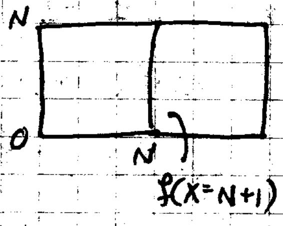
<u>DCT vs. Wavelets</u>: משתמשים היום ב- JPEG פשוט
כי זה נפוץ יותר. חוץ מזה יש איזה פטנט רשום של
JPEG2000 אז אנשים פוחדים להשתמש בזה. אבל מבחינה
תיאורטית אין ל- DCT שום יתרון ל- DCT

<u>מה לא צריך לדעת על wavelets?</u>
לא תהיה שאלה שדורשת לתכנת את ה- wavelets
האופטימליים למצב מסוים.

<u>מה כן צריך לדעת על wavelets?</u>
תכונות כלליות של הפונקציה כמו אורתוגונליות וכו'.
צריך לדעת שסכום האיברים של ה- low-pass הוא 1,
סכום האיברים של ה- high-pass הוא 0, במכפלה
הפנימית שלהם היא 0 מאוד חשוב שהם יהיו אורתוגונליים.
אחרת זה יגרום מיני בעיות בשכבת הפעולות.
נשים לב שתצוגה של מאל יכולה להשפיע בצורה ניכרת על
אופטימיות ה- wavelets ולכן לא משתמשים בהם בד"כ
ב- recognition.

DCT vs. DFT

• טרנספורם פורייה $f(x) = \sum_u F(u) e^{\frac{2\pi i u x}{N}}$



$f(N+1) = f(1)$

הרציפות כאן היא במודול N!
מבחינת דחיסה יש פה בעיה:
הסיכוי שהפיקסל הימני יהיה דומה לפיקסל השמאלי הוא נמוך. אבל במעבר בין השוליים של התמונה יש קפיצה והקפיצה הזו פוגעת בדחיסה, כי זו אנטרופיה גבוהה.

• טרנספורם קוסינוס $f(x) = \frac{1}{2}F(0) + \sum_u F(u)\cos\left(\frac{\pi}{N}(x+\frac{1}{2})u\right)$

(לא אולי $f(N)$. צריך לבדוק) $\Rightarrow f(N+1) \approx f(N-1)$

אז כאן הקפיצה פחות גדולה וזה טוב לדחיסה!

התאמת תמונות

מה קורה אם יש שינויים בגווני אפור? בד"כ ההבדלים הם בגודל מקבוע (מצלמה שונה וכו'). אז למעשה המשוואה היא

$$f(x,y) = a\cdot g(x-u, y-v)$$

ואת זה אפשר להמשיך לפתור באופן דומה למה שעשינו קודם. ודבר דומה קורה בתוספת קבוע:

$$f(x,y) = g(a-u, y-v) + a$$

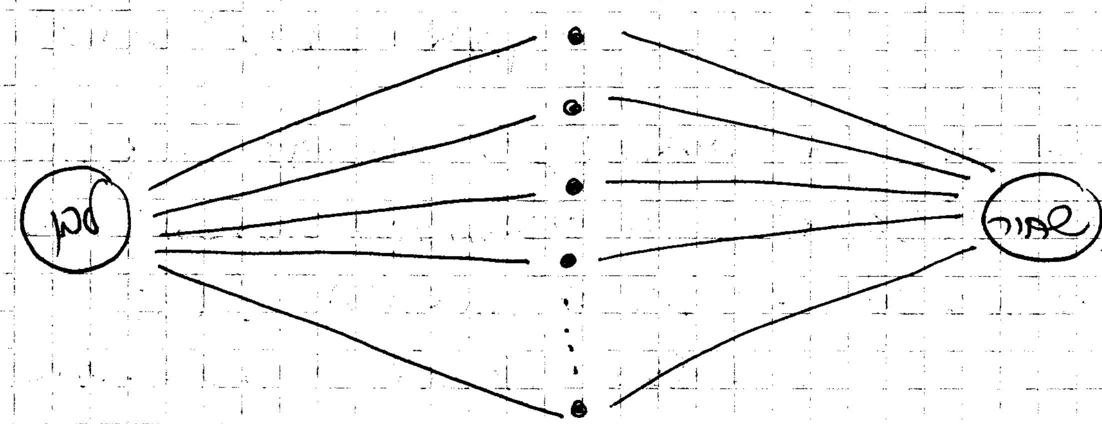
עוד דרך להתמודד עם זה במקרה של הבדל חיבורי אפשר לחשב את ההצלבה על הלפלסיאן ואז הקבוע לא משפיע. קונבולוציה היא אינווריאנטית להזזה. אז תמיד אפשר לעשות קונבולוציה ואח"כ לחשב את ההצלבה.

(26) 17/3/08
עיבוד תמונה

עיבוד תמונות ע"י פעולות גרפים

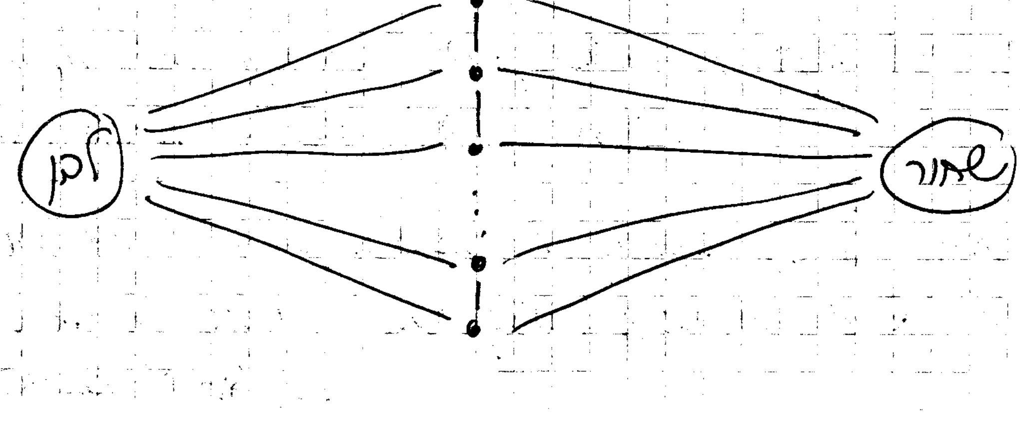
נניח שרוצים לקבוע סף לתמונה כדי להפוך אותה לשחור-לבן. נסתכל בגרף:

פיקסלים



חתך בגרף הזה יוגדר הפרדה לשחור ולבן. אפשר לתת לכל צלע משקל (הפרש דרגות אפור) - ש אז חתך מינימלי יוגדר הפרדה שבה דרגות נמוכות עוברות ללבן ודרגות גבוהות עוברות לשחור.

אם רוצים להתחשב גם במה שמדובר בתמונה ולא מספיק אוסף נקודות אקראי אז נראה שנרצה שתהיה הפרדה ב edges. אז אנחנו רוצים שההפרדה תקרה איפה שהגרדיאנט גבוה. אז אפשר להוסיף לגרף עוד צלעות בין כל שני פיקסלים סמוכים שהערך שלהן (גרדיאנט) - ש ואז שוב נחפש חתך מינימלי.



בהמשך הקורס נשתמש הרבה ב- min-cut max-flow
צריך להכיר אותו בעיניים עצמאות ⇐ נא לחזור
עליו עד יום חמישי.
בתרגיל האחרון אנחנו נשתמש בזה בשביל לתפור
תמונות - כלומר בשביל ליצור תמונות פנורמה.

שאלות לקראת המבחן -

* בהתאמת תמונות - איך מחשבים את f_x ואת f_y
סביר להשתמש בקונבולוציה עם $\frac{1}{2}(1,0,-1)$ וכו'.
$\frac{1}{2}\begin{pmatrix}1\\0\\-1\end{pmatrix}$ בהתאמה. זה טוב כי זה סימטרי
סביב הפיקסל ולא יוצר הזזות.

* נניח $f(x,y) = dg(x-u, y-v) + b$
כאשר u,v,d,b קבועים. פה נרצה להגיע ל-4
משוואות ב-4 נעלמים.
רוצים למזער את $\sum(f(x,y) - dg(x-u,y-v) - b)^2$
נשתמש בקירוב לינארי ונרצה למזער את
$$\sum(f - d(g - g_x u - g_y v) + b)^2$$
זה דומה למה שעשינו קודם אלא שיש סמוך קטן -
בגלל הכפל בקבוע אין לנו את f_t במשוואות.

* Lucas Kanade זה בד"כ יותר טוב וזה גם נותן
תוצאה מיידית - כלומר כדי למצוא הזזות תת פיקסליות
מ-Lucas Kanade זה יוצא מייד ואילו כשמשתמשים
בקורלציה מנורמלת צריך אחרי זה לעשות עוד פעולות -
לעשות אינטרפולציה וכו'.

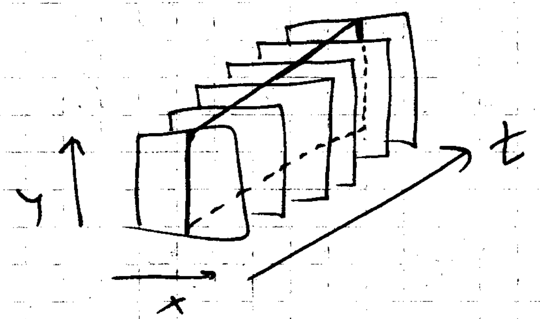
* עוד דרך למצוא תזוזה בין תמונות היא ע"י התאמת אזורים.
למשל אם מצאנו שתי נקודות מתאימות אפשר לחשב הזזה,
סיבוב והגדלה.

27 24/03/08
עיבוד תמונה

Video Mosaicing

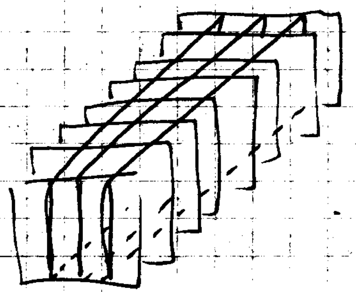
היום נראה מה אפשר לעשות ואנחנו אמורים להיות מסוגלים לחשוב לבד איך עושים את זה.

איך עושים פסיפס מסורתי? יש כמה תמונות ובין כל זוג תמונות עוקבות מחשבים את ההזזה. בוחרים מסגרת אחת ומכניסים לתוכה את כל התמונות. איך מדביקים את התמונות? בתרגיל ראינו איך משתמשים ב- min-cut כדי למצוא את המקום הכי טוב להדביק בו.
לפני שזה היה פופולרי היו לוקחים מכל תמונה פס מהמרכז ברוחב ההזזה ומדביקים אותם אחד לשני. כדי שלא יראו את ההדבקה משתמשים ב pyramid blending.
אפשר להסתכל על כל התמונות של כ- space-time-volume
יש תמונות שמשתנות
בזמן ומכל אחת לוקחים
את המרכז.



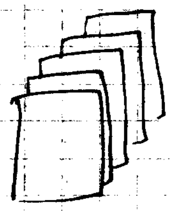
Stereo - ויש לנו שתי עיניים מרוחקות זו מזו ובגלל הריחוק כל אחת רואה את העולם מכיוון שונה. המוח יודע לקבל מזה תמונת עומק.
איך אפשר לייצר תמונות סטריאו כאלה? אפשר לייצר שתי פנורמות שאחת מורכבת מפסים שנלקחו מימין והשנייה מפסים שנלקחו משמאל. אם נשים את שתי הפנורמות בשני ערוצים שונים. נוריד מאחת ונוריד את האדום ומהשנייה את הכחול ונסתכל במשקפיים מתאימים נראה תמונה תלת-ממדית.

אפשר ליצור סדרה של תמונות שכל אחת מהן נלקחה
מחתך אחר ב- space-time volume
כך אפשר ליצור סרט
שבו כל frame זה
חתך אחר ואז יוצא סרט
שרואים את הנוף מזוויות שונות.



במצגת - pushbroom projection -
- two-slit camera
- dynamic mosaics

<u>סדר לא כרונולוגי</u> - נניח שיש לנו מצלמה סטטית וסרט
אפשר להעביר יריעה בסרט הזה
וככה ליצור סרט שבו המאורעות
המקוריים מתרחשים בזמנים שונים.
במצגת יש דוגמה מגניבה עם
אמפיתיאטרון שמתפוצץ.



<u>פסיפסים וחתכים בגרפים</u>

באופן כללי יש לנו תמונה וכל פיקסל אנחנו רוצים
להחליט מאיזה frame הוא בא, ב- min-cut
אנחנו פתרנו את הבעיה למקרה שיש שני frame-ים.
אנחנו רוצים לפתור את זה גם למקרה שיש יותר frame-ים.
זאת בעיית Graph-Cut. זו בעיה NP-שלמה.
אז מה שאפשר לעשות זה להסתכל על זה כ- min-cut
אבל תלת ממדי, כלומר מחפשים את היריעה ב-
space-time-volume. יש פה את היתרון שהכל מסתכל
סמולטנית בכל ה frame-ים וזה יוצר רציפות בסרט.

31.03.08

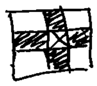
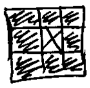
עיבוד תמונה

גיאומטריה דיגיטלית

נתבונן בתמונות בינריות:

פיקסלים עם ערך 1 הם הגוף ופיקסלים עם ערך 0 הם הרקע.

נגדיר שכנויות:

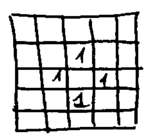
- שכנות 4 - שכנים בצלע -  - יש 4 כאלה
- שכנות 8 - שכנים בצלע או בקודקוד -  - יש 8 כאלה
- רכיב קשיר - גוף שבין כל שני פיקסלים בו קיים מסלול של פיקסלים בגוף שבו כל שני פיקסלים עוקבים במסלול הם שכנים.

איזה סוג של שכנים? יש רכיב קשיר 4 ורכיב קשיר 8 עבור שכנות 4 ושכנות 8 בהתאמה.

דוגמה:

0	1	0
1	0	1
0	1	0

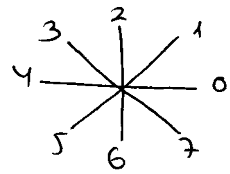
כאן, אם נסתכל על הגוף אז יש 4 רכיבי קשירות 4 ורכיב קשירות 8 אחד!
אם נתבונן ברקע אז יש 5 רכיבי קשירות 4 ורכיב קשירות 8 אחד!



בגוף - יש 4 רכיבי קשירות 4 ורכיב קשירות 8 אחד.
ברקע - יש 2 רכיבי קשירות 4 ורכיב קשירות 8 אחד!

בד"כ משתמשים בקשירויות שונות לגוף ולרקע!

- chain code - משמש לתיאור מסלול של פיקסלים. יש 8 כיוונים אפשריים לתזוזה:



אם משתמשים בקשירות 8 משתמשים בכל הכיוונים ואם משתמשים בקשירות 4 משתמשים רק בכיוונים הזוגיים.

דוגמה: את הגוף

(0,0) →

1 1
1 1 1 1
1 1 1 1

אפשר לייצג ע"י ה- chain code

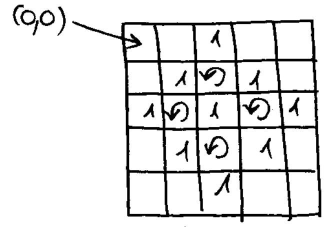
$\underbrace{(0,0)}_{\text{מסמן את ההתחלה}}$ 007644422

כאשר מתחילים ממקום (0,0).

אם יודעים שגוף הוא ללא חורים, נותנים לנו קוד שרשרת סגור ואנחנו יודעים באיזה כיוון של השרשרת נמצא הגוף אז יודעים בדיוק איפה הגוף.

המחיר הוא 3 ביטים לכל פיקסל במסלול.

אם בגוף יש חורים אפשר לייצג גם את החורים ע"י קוד שרשרת.



(2,0) 7 7 5 5 3 3 1 1

זה החור העליון ← (2,0) 5 7 1 3

• מרחקים - יש כל מיני מרחקים שאפשר לדבר עליהם. נניח שיש שני פיקסלים (x,y) ו- (u,v).

- מרחק אוקלידי $d_e = \sqrt{(x-u)^2 + (y-v)^2}$
- מרחק 4 - אורך המסלול הכי קצר במסלול של שכני 4: (city block) $d_4 = |x-u| + |y-v|$
- מרחק 8 - אורך המסלול הכי קצר במסלול של שכני 8: $d_8 = \max(|x-u|, |y-v|)$

ברור שמתקיים $d_8 \le d_e \le d_4$ אבל אפשר להוכיח שמתקיים גם $\sqrt{2}\, d_8 \ge d_e \ge d_4/\sqrt{2}$

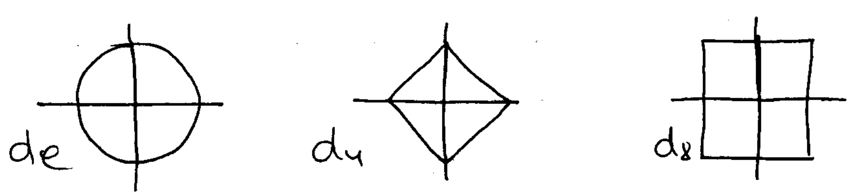
• מטריקה היא פונקציה d שמקיימת:

- $d(P,Q) \ge 0$ ו- $d(P,Q) = 0 \iff P = Q$
- $d(P,Q) = d(Q,P)$
- $d(P,Q) \le d(P,R) + d(R,Q)$

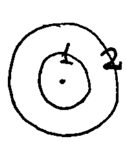
קל לראות ש- d_4 ו- d_8 הן אכן מטריקות (לאי שיוויון, הן מושרות מהנורמות $\|\cdot\|_1$ ו- $\|\cdot\|_\infty$ בהתאמה)

איך נראים עיגולי היחידה במטריקות האלה?

עיגול היחידה



מפת מרחקים



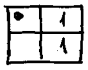
		2		
	2	1	2	
2	1	0	1	2
	2	1	2	
		2		

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

מורפולוגיה - Mathematical Morphology

נסמן את הראשית ב- •.

בגוף בינארי אפשר לייצג ע"י קבוצת הפיקסלים שערכם 1.

 $\Rightarrow \{(1,0),(1,1)\}$

• נניח ש-A קבוצה ו-x פיקסל. נסמן את ההזזה של A ב-x:

$$A_x = \{a+x : a\in A\}$$

למשל, $\{(1,0),(1,1)\}_{(-1,0)} = \{(0,0),(0,1)\}$

• שיקוף - $\check{B} = \{-b : b\in B\}$

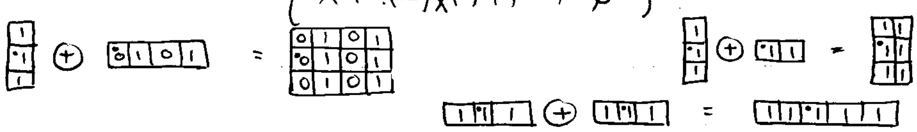
• השלמה - $A^c = \{b : b\notin A\}$

• הפרש - $A-B = A\setminus B = \{c : c\in A \wedge c\notin B\}$

• הרחבה (dilation) -

$$A\oplus B = \{a+b : a\in A,\ b\in B\} = \bigcup_{b\in B} A_b = \bigcup_{a\in A} B_a$$
$$= \{x : (\check{B})_x \cap A \neq \emptyset\}$$

למשל,



תכונות של ההרחבה:

$A\oplus B = B\oplus A$ קומוטטיביות

$(A\oplus B)\oplus C = A\oplus(B\oplus C)$ אסוציאטיביות

$A\subseteq B \Rightarrow D\oplus A \subseteq D\oplus B$ מונוטוניות

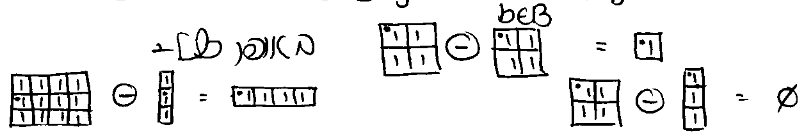
$(A\cup B)\oplus C = A\oplus C \cup B\oplus C$ דיסטריביוטיביות

$(A\cup B)\oplus C = A\oplus C \cap B\oplus C$

• כיווץ (erosion) -

$$A\ominus B = \{x : x+b\in A\ \forall b\in B\} = \bigcap_{b\in B} A_{-b}$$

למשל, $A\neq\emptyset$ $A\ominus A =$  באופן כללי



• פתיחה וסגירה:

$B\circ K = (B\ominus K)\oplus K$ פתיחה

$B\bullet K = (B\oplus K)\ominus K$ סגירה

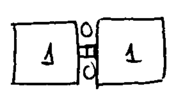
סגירה עם K סוגרת את החורים בגוף B בתלות בגודל של K.

בפתיחה הולכות לאיבוד נקודות בודדות מרקע.

זה משמש לניקוי מרעשים - אם יש גוף עם חורים או רקע עם נקודות...

למשל

$$\begin{matrix}1&0&1&0&1&0\\0&1&0&1&0&1\\1&0&1&0&1&0\end{matrix} \circ \begin{matrix}1&1&1\\1&1&1\\1&1&1\end{matrix} = 0 \qquad \begin{matrix}1&0&1&0&1&0\\0&1&0&1&0&1\\1&0&1&0&1&0\end{matrix} \bullet \begin{matrix}1&1&1\\1&1&1\\1&1&1\end{matrix} = 1$$



פתיחה תפריד את החיבור

סגירה תהפוך את שני הגופים לאחד.

נניח שרוצים להשאיר מהתמונה רק קווים אנכיים בגובה לפחות 4.

תמונה שיש בה רק קוים בגובה 4 = $\begin{bmatrix}1\\1\\1\\1\end{bmatrix} \circ$ תמונה

זה יתן להשאיר גם קוביות שגובהן לפחות 4. אם רוצים רק קווים צריך עוד לשחק עם זה.

פתיחה וסגירה הן פעולות אידמפוטנטיות - ההרכבה שלהן עם עצמן נותנת את אותו הדבר (כלומר לפתוח פעם אחת ולפתוח 70 פעם זה אותו דבר).

• שלד מרכזי - (Medial Axis)

רוצים לקחת גוף ולייצג אותו בצורה מינימליסטית ע"י ישרים. למשל בן אדם אפשר לייצג רק  ומלבן רק —. מתמטית:

$$S_0(A) = A - (A \circ K)$$

כאשר $K = \begin{smallmatrix}1&1&1\\1&1&1\\1&1&1\end{smallmatrix}$ עבור קשירות 8 או $K = \begin{smallmatrix}0&1&0\\1&1&1\\0&1&0\end{smallmatrix}$ עבור קשירות 8

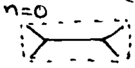
למשל, אם $A = \square$ אז בקשירות 8 $S_0(A) = \emptyset$

ובקשירות 4 $S_0(A) = \begin{smallmatrix}\square & \square\\ \square & \square\end{smallmatrix}$

כעת נגדיר $S_n(A) = (A \ominus_n K) - ((A \ominus_n K) \circ K)$

כאשר $\ominus_n$ מסמן כיווץ n פעמים.

ולבסוף $S(A) = \bigcup_{n=0}^{\infty} S_n(A)$

רק עבור קשירות 4 נקבל $S(\square) =$ 

ועבור קשירות 8 נקבל $S(\square) =$