**סיכום החומר לבחינה – ראייה ממוחשבת**

* בתמונה דיגיטלית:

1. **דוגמים** (**Sample**) את הרצף לפיקסלים
2. **מכמתים** (**Quantize**) כל דגימה לעוצמה דיסקרטית בעלת מספר שלם

* אדם יכול לחוש בצבע בין 400 ל 700 נ"מ, כאשר מקסימום הרגישות שלו היא באיזור 600.
* הצבע הכחול נמצא באיזור 400-500, הירוק 500-600, והאדום באיזור ה 600-700.
* כשמתעסקים עם צבע עדיף להתעסק רק עם ספקטרום בעל פיזור נורמאלי, כך ש:

1. ממוצע עקומת הצבע מגדיר את הגוון
2. ככל שהשונות קטנה הצבע רווי יותר
3. ככל שהשטח תחת העקומה גדול יותר הצבע בהיר יותר

* ניתן ליצור צבעים חדשים בשתי שיטות:

1. **שיטת ההוספה** – הבסיס הוא שחור ואליו מוסיפים/ממזגים כמה צבעים, כמו ערבוב של אדום וירוק שמוציא צהוב. לדוגמא: מקרן.
2. **שיטת החיסור** – הבסיס הוא לבן וממנו מחסרים עוד ועוד צבעים. לדוגמא: הדפסה על נייר.

* בקוביית RGB גווני האפור הם על האלכסון בין השחור ללבן.
* RGB טוב למכשירי תצוגה, אבל לא לתיאור אנושי
* מודל ה HSV – גוון (הצבע הדומיננטי), רוויה (כמה הצבע טהור), בהירות. כאשר המודל מתואר כקונוס שעומד על הראש. הגוון הוא הזווית (מתחיל מאדום), הרוויה היא המרחק מקצות הקונוס, והבהירות היא הגובה בקונוס. טוב לתיאור אנושי.
* **בדיאגרמה הכרומטית (C.I.E)** אין בהכרח משמעות למרחק בין 2 נקודות צבע היות והמרחב הוא לא אחיד. ניתן לשפר זאת ע"י אליפסת **MacAdam** שמציירת אליפסות בתוך המרחב הזה כך שכל נקודה בתוך האליפסה היא לא מובחנת ע"י עין אנושית, אבל מחוצה לה כן.
* סוגי רעש:

1. Salt & paper – מופעים אקראיים של שחור ולבן על התמונה

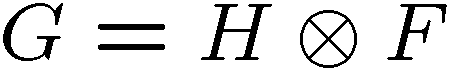
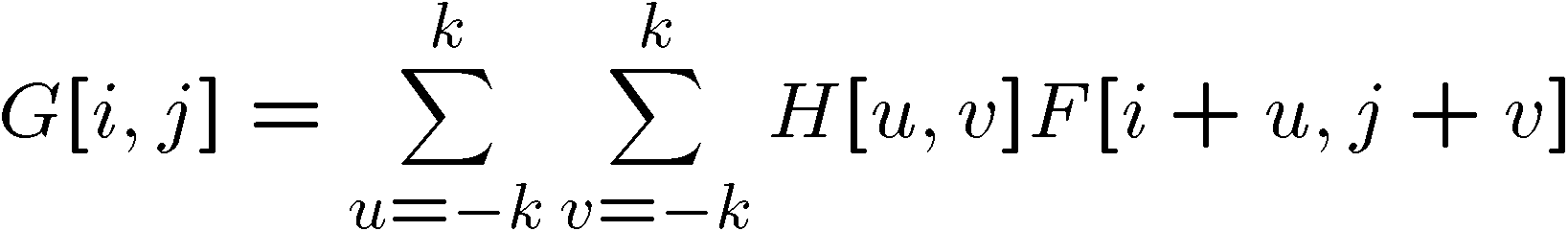
2. Impulse – מופעים אקראיים של פיקסלים לבנים

3. גאוסיאנית - שינויי עוצמה בצורת התפלגות גאוסיאנית נורמלית. ככל שהסיגמה גדולה יותר הרעש גדול יותר.

* שיטות לטיפול ברעש:

1. החלפת כל פיקסל בממוצע שכניו - תחת ההנחה של דמיון לפיקסלים השכנים, וכן שהרעש הוא בלתי תלוי מפיקסל לפיקסל.
2. החלפת כל פיקסל בממוצע שכניו עם משקולות שונות לכל שכן,

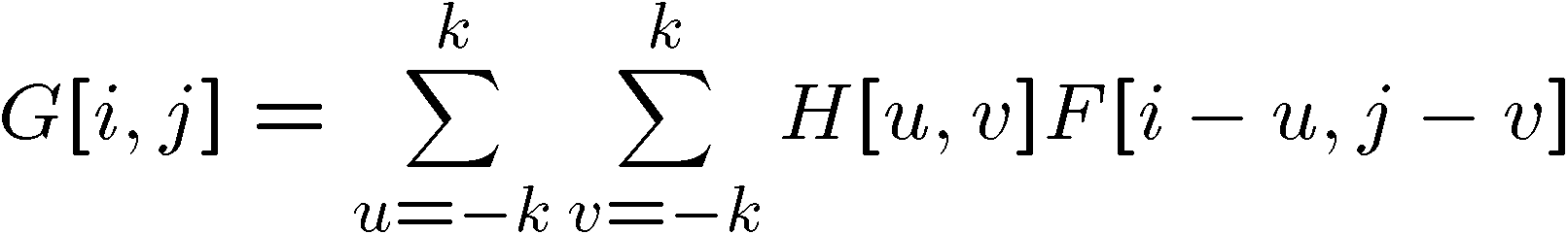
למשל: [1, 4, 6, 4, 1] / 16. חשוב שסכום המשקולות חלקי המכנה יהיה 1.

* **פילטור תמונה**: החלפת כל פיקסל בקומבינציה לינארית של שכניו.
* Cross-correlation: 
* 
* כאשר H נקראת הקרנל או המסיכה של המשקולות של הצירוף הלינארי והיא יכולה להיות אחידה או לא אחידה.
* בפילטר לינארי לכל נקודה בחישוב הערך החדש משתמשים באותו סט משקולות
* פילטר נראה כמו האפקט שהוא מחפש
* **box filter** הוא פילטר מיצוע שבו כל המשקולות שוות למשל: 3/[1,1,1], והוא גורם להחלקת התמונה.
* **פילטר גאוסיאני** הוא פילטר בעל משקולות לא אחידות שבו השכנים הקרובים מקבלים יותר משקל. למשל: 4/[1,2,1].
* בפילטר גאוסיאני יש שני פרמטרים שניתן לשחק איתם:

1. **גודל הפילטר** שמושג ע"י קיטום ההר הגאוסיאני האינסופי באיזה גובה מסוים כרצוננו
2. **השונות** של הפילטר σ שמשפיעה על מידת ההחלקה. ככל שהסיגמא גדולה יותר ההחלקה גדולה יותר, מטושטשת יותר.

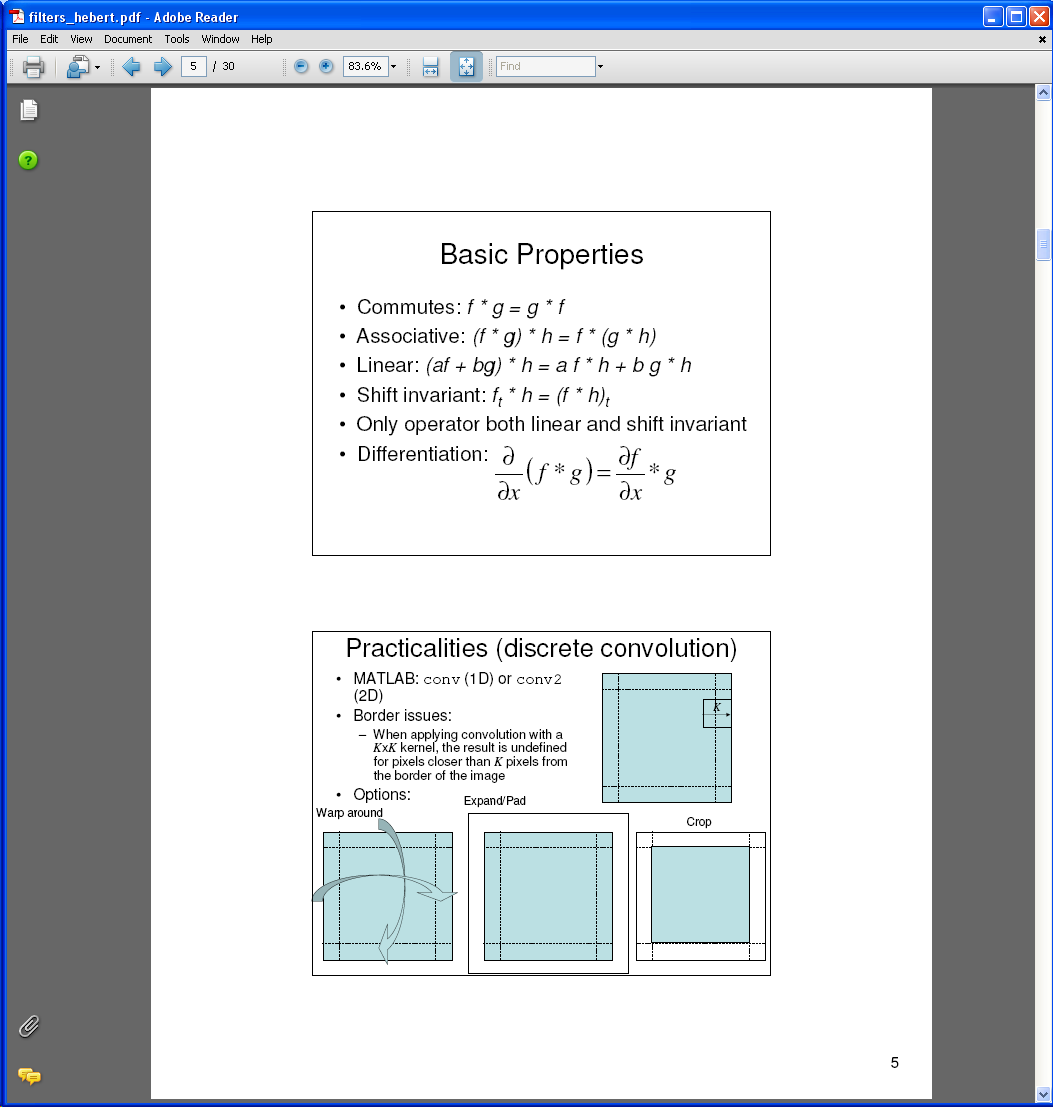
* שיטות ביחס לגבולות התמונה בפילטור:

1. full – תמונת הפלט היא סכום התמונה המקורית והפילטר – בגבולות מרכז הפילטר מחוץ לתמונה כך שקצה הפילטר כולל את גבולות התמונה, ולכן נותנים ערכים גם לפיקסלים מחוץ לתמונה.
2. same – גודל הפלט הוא גודל התמונה המקורית – בגבולות מרכז הפילטר מונח על גבול התמונה.
3. valid – גודל הפלט הוא גודל התמונה פחות הפילטר – מרכז הפילטר בתוך התמונה כך שקצה הפילטר מונח על גבול התמונה.

* אם הפילטר חורג מגבולות התמונה ניתן לשים ערכי תמונה של 0, או להניח שהתמונה מחזורית, או להעתיק את ערכי הגבול או לשקף את הערכים ליד הגבול כמראה.
* בקורלציה עם אימפולס מתקבל הפילטר הפוך במרכז האימפולס.
* ניתן לחדד תמונה בעזרת פילטר שמדגיש את הפיקסל שמופחת ממנו פילטר המיצוע
* **קונבולוציה**: Edittex,
* 
* קונבולוציה מתקבלת ע"י היפוך הפילטר בשני המימדים: מעלה-מטה,ימין-שמאל, ואחר כך ביצוע קורלציה.
* אם הפילטר הוא סימטרי אין הבדל בין קורלציה וקונבולוציה
* **shift invariant** - אי תלות במיקום השכונה, אלא בערכי הפיקסלים בשכונה בלבד.
* **לינאריות**:

1. Superposition: h \* (f1 + f2) = (h \* f1) + (h \* f2)
2. Scaling: h \* (k f) = k (h \* f)

* תכונות קונבולוציה:

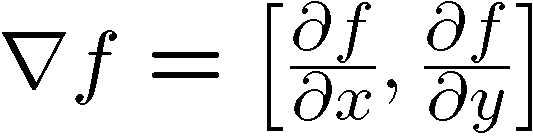
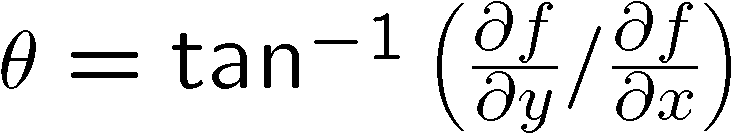
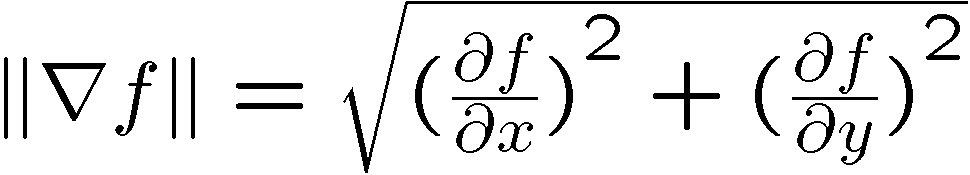
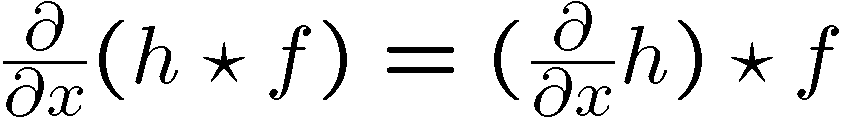
1. shift invariant
2. לינאריות
3. קומוטטיביות - f \* g = g \* f
4. אסוציאטיביות - (f \* g) \* h = f \* (g \* h)
5. זהות - f \* e = f כאשר e הוא אימפולס.
6. Differentiation - 

* **פילטר ספרבילי** – פילטר שניתן לפרקו לשני מרכיבים שורה ועמודה ולעשותו בצורה שקולה ע"י 2 צעדים: פילטור השורות ע"י מרכיב השורה ולאחר מכן פילטור העמודות ע"י מרכבי העמודה. אם הפילטר מוצג כנוסחא והוא שקול למכפלת 2 פונקציות אשר האחת היא פונקציה של x בלבד והשניה של y בלבד אזי הפילטר ספרבילי.
* החיסכון החישובי של פילטר ספרבילי:

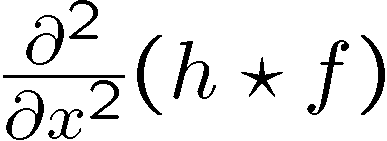
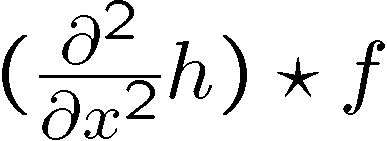
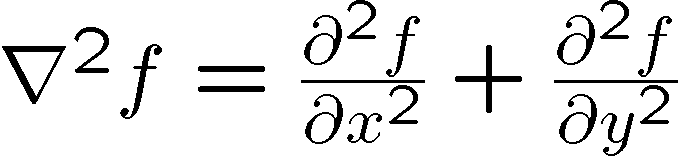
עבור תמונה בגודל n\*n, ומסיכה בגודל k\*k:

סיבוכיות f\* (g \* h) היא O(n2k2)

ואילו סיבוכיות (f\*g) \* h היא O(n2k)

* פילטר גאוסיאן ופילטר בוקס הם ספרביליים
* **פילטר חציון** – לוקח את הערך החציוני של השכונה. יתרונו הוא שהוא אינו ממציא ערכים חדשים, והוא טוב לרעשי מלח-פלפל ורעשי אימפולס. כמו כן הוא שומר על הקצוות ולא מחליק אותם כמו פילטר מיצוע.
* הגורמים ל edge – גבולות אוביקט, צללים, צורה, טקסטורה
* edge הוא מקום של שינוי מהיר בערכי העוצמות וניתן לזיהוי ע"י הנגזרת הראשונה בערכי אקסטרים – מינימום או מקסימום לוקאליים.
* בפועל נגזרת היא: f(x+1,y)-f(x,y) בגזירה לפי x, f(x,y+1)-f(x,y) בגזירה לפי y, וניתן לממש ע"י קונבולוציה עם מסכה של [-1 1]
* **גראדיאנט** של תמונה: והוא מצביע על כיוון השינוי החזק ביותר בעוצמה.
* **כיוון הגראדיאנט** ניתן ע"י 
* **עוצמת הגראדיאנט** ניתנת ע"י 
* לפני שמפעילים גראדיאנט למציאת edge חייבים להחליק את התמונה, אחרת תיווצר תמונת גראדיאנט רועשת, ולא נוכל לאתר את ה edge.
* תיאורית הנגזרת של הקונבולוציה: 

ולכן אם h קטנה יעיל יותר לגזור אותה קודם ואז לעשות קונבולוציה עם f.

* **Laplacian of Gaussian – LOG** - , הנגזרת השניה של גאוסיאן. נבצע , ואז נחפש חציית 0 בשביל למצוא edge במקום ערכי אקסטרים.
* **אופקטור לפלסיאני** - 
* מציאת blob- ים ע"י LOG – בהפעלת LOG על blob שחור עם רקע לבן מתקבלים ערכי מקסימום. Blob מאופיין ע"י ערך אקסטרים לוקאלי במרחב ובסקלה עבור LOG או DOG.
* תמונת הנגזרת עוזרת לחידוד התמונה
* תכונות מסיכהלהחלקה**:** ערכים חיוביים, סכום המסכה 1, עוצמת ההחלקה פרופורציונאלית לגודל המסיכה, מסיר רכיבים בעלי תדר גבוה – low-pass
* תכונות מסיכה לנגזרת: סימנים הפוכים ע"מ לגלות איזורים בעלי קונטרסט גבוה, סכום 0 כך שלא תהיה תגובה באיזורים עם עוצמה קבועה, ערכים גבוהים בנקודות בעלי קונטרסט גבוה.
* צעדים לגילוי edge:

1. החלקת התמונה
2. פילטור התמונה
3. מציאת ערכי אקסטרים לוקאליים, וניפוי ערכים שנחשיב כרעש שלא חוצים סף מסוים.

* בפילטור גאוסיאני ככל שהסיגמא גדולה יותר נמצא edge-ים בסקלה גבוהה יותר.
* **Thresholding** – כל הערכים שמתחת לסף יהיו 0, וכל אלו שמעל או שווים לסף יהיו 1.
* **Canny edge detector**:

1. פילטור התמונה עם נגזרת של גאוסיאן
2. מציאת עוצמה וכיוון של של הגראדיאנט
3. **Non-maximum suppression** **–** "דיכוי" ערכים לא מקסימליים בסביבה לוקאלית לאורך כיוון הגראדינט, ובכך נוצרים קווים בעלי רוחב פיקסל אחד. זה מצריך בדיקה מול פיקסלים פיקטיביים שעברו אינטרפולציה בגלל שכיוון הגראדינט לא בהכרח בזוויות שמתלכדות עם פיקסלים.
4. **Hysteresis –** הגדרת 2 ספים גבוה ונמוך, כשהגבוה יוצר את העקומה הכללית ברמה גבוהה, והנמוך על מנת לקשר בין עקומות.

* **פירמידות גאוסיאניות** – יצוג מרובה סקלות של התמונה.

בניתהפירמידהבכל רמה (**reduce**) מתבצעת ע"י :

1. פילטר **קירוב** לתמונת הפלט (ממוצע, גאוס)

2. **דגימה** שבה לוקחים כל פיקסל שני בשורות ובעמודות וכך נוצרת תמונה קטנה פי 4 מהמקורית, לדוגמא מ 16\*16 ל 8\*8.

* **פירמידות לפלסיאניות** – .

בניתהפירמידהבכל רמה מתבצעת ע"י :

1. Gi+1=Reduce(Gi)
2. **Expand**(G**i+1)** Approx(Gi) =
3. **דגימה (up-sampling)** ל "ניפוח" התמונה ע"י:
4. **פילטר אינטרפולציה** (ממוצע, גאוס) לכל הפיקסלים שקיבלו ערך 0 לעיל ולזכור להכפיל ב 4 ללא תלות בגודל הפילטר, כיוון שעל כל פיקסל "אמיתי" מתחשבים ב 3 פיקסלים פיקטיביים.
5. Li = **I – expand(reduce (I))=** Gi*-*expand(G**i+1)**

* תכונות פירמידה לפלסיאנית:

1. דרך טובה לאיתור סף כי בהחלקה בreduce איבדנו את המעברים החדים בתמונה שהם הסף, ולכן ההבדלים הגדולים ביותר בין התמונה לקירוב שלה יהיו בסף. לכן ערך גבוה ברמה הנמוכה (הרמות הראשוניות – תמונה גדולה יותר) הוא סף בתמונה המקורית.
2. ערך גבוה ברמה הגבוהה (הרמות המתקדמות יותר – תמונות קטנות יותר) הוא שינוי הדרגתי בתמונה המקורית, כי זהו סף בתמונה המוקטנת המתאים למספר רב של פיקסלים בתמונה המקורית והוא "נמרח" על כולם ולכן זהו שינוי הדרגתי.
3. רמות שונות כוללות תדרים שונים
4. ניתן לשחזר את התמונה המקורית בעזרת פירמידה לפלסיאנית עם קודקוד גאוסיאני, כיוון ש Gi **=** Li*+*expand(G**i+1).** ה L ידוע בכל רמה וה G**i+1** הוא קלט בהתחלה ובהמשך הוא מחושב ע"י הנוסחא. כך עד שמגיעים לתמונה המקורית G0.
5. טובות לניתוח רב רזולוציות, דחיסת תמונות, ניקוי תמונות ושילוב תמונות.

* בעיות בהתאמת תמונות:

1. שינויים גיאומטריים
2. שינויים פוטומטריים
3. רעש
4. חדות
5. הסתרות

* שיטה להתאמת תמונות:

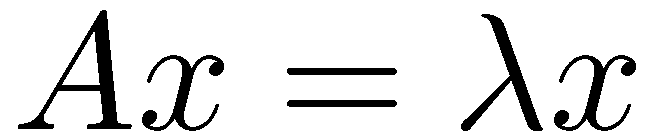
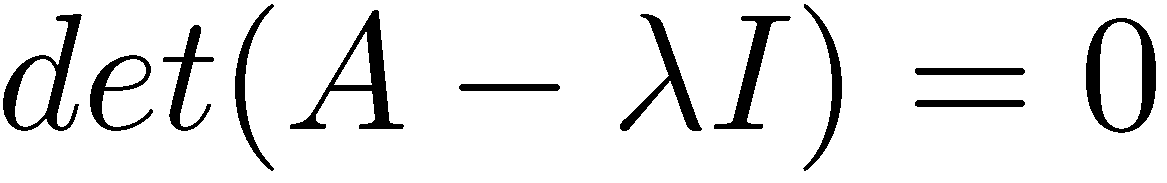
1. מציאת נקודות עניין מיוחדות בשתי התמונות, בפרט **פינות** אשר:
2. הן חזרתיות ומובחנות
3. לגראדיאנט בפינות יש 2 או יותר כיוונים דומיננטיים
4. יצוגם באמצעות invariant descriptors
5. מציאת התאמות בין הנקודות
6. ישר/התאם את התמונה ע"י זוגות ההתאמות הללו

* מציאת פינה:

1. ניקח את תמונת הגראדיאנטים ונתאים להם אליפסה שמרכזה בראשית בצורה הבאה:

כאשר M היא **מטריצת השונות המשותפת/מטריצת תפזורת**. M מגדירה את האליפסה, ערכיה העצמיים הם ארכי צירי האליפסה, תחת ההנחה שממוצע ההתפלגות האמיתית הוא הראשית.

1. פינה היא אליפסה הדומה לעיגול גדול במרחב הגראדיאנטים, התפלגות גאוסיאנית עם שונות גבוהה בשני צירים במרחב הגראדיאנטים.

* **וקטורים עצמיים** של מטריצה A הם וקטורים x המקיימים: 
* כאשר l הוא ה**ערך העצמי** המתאים ל x ואותם ניתן למצוא ע"י: 
* נמצא את הוקטורים העצמיים ע"י:
* ערכיה העצמיים של M:

1. l2 >> l1 או l1 >> l2 -> **edge** אופקי או אנכי בהתאמה
2. l1 and l2 קטנים -> איזור שטוח
3. l1 and l2 גדולים וגם l1 ~ l2-> **פינה**

* **שיטת האריס** (יעילה יותר בלי חישוב הערכים עצמיים במפורש ולכן בלי חישוב שורש)**:**

1. כאשר trace של מטריצה הוא סכום איברי האלכסון.
2. *α*: constant (0.04 to 0.06)
3. אם R<0 -> **edge**
4. אם *|R|* small -> איזור שטוח
5. אם R>0 -> **פינה**

* **אלגוריתם האריס למציאת פינות:**

1. עבור כל פיקסל נחשב באיזור סביבו את ה M ונחשב את ה R התואם שלו
2. מציאת פיקסלים עם R גבוה מסף מסוים
3. לקיחת פיקסלים עם R שהוא מקסימום לוקאלי

* תכונות מזהה פינות של האריס:

1. Rotation invariance - האליפסה מסתובבת אבל צורתה נשארת
2. - Not invariant to image scaleכי מה שנראה כפינה בחלון אחד אם מגדילים אותו כבר לא נראה כפינה אלא סתם edge.

* טיפול בבעית ה scale - מציאת עיגולים בגדלים שונים סביב הנקודה בכל אחת מהתמונות כך שיראו אותו הדבר.
* מציאת scale אוטומטי:

1. שימוש בפונקציה שמוגדרת על איזור בתמונה ש:

א. נותנת אותם ערכים עבור איזורים תואמים בתמונות שונות ללא תלות ב scale.

ב. הפונקציה נותנת ערך מקסימום באיזה מקום לפי גודל האיזור ואינה שטוחה.

ג. פונקציות טובות הן פונקציות שמגיבות לקונטראסט.

1. עבור כל נקודה בכל תמונה (באופן בלתי תלוי) ניקח את גודל האיזור שהניב בפונקציה ערך מקסימום לוקאלי.
2. כשנרצה לבצע התאמה בין שתי תמונות עבור כל נקודת ענין ניקח את הסקאלה התואמת לפי האלגוריתם הנ"ל ומובטח לנו שאנו מסתכלים על אותו איזור. ובשביל הנוחות ננרמל איזור אחד שיהיה באותו גודל כמו בתמונה השניה.

* ***Scale* אופייני** – *scale* שמניב פיק בהפעלת לפלס עליו
* **Laplacian-of-Gaussian (LoG)** – הפעלת פילטר גאוס ואחריו פילטר לפלס על התמונה בסיגמות שונות, כאשר הסיגמא מייצגת את ה scale. בוחרים את הסיגמא שנותן עבורה ה LOG בנקודת העניין מקסימום לוקאלי בתור הסקאלה האופיינית.
* **לפלס** **על גאוס** -
* ניתן לצרף את האריס ללפלס ע"י מציאת הפינות במרחב התמונה, ועל כל פינה מחפשים עבורה את הסקאלה שנותנת מקסימום עבור פילטר לפלס.
* **difference of Gaussians (DoG)** – קירוב יעיל ללפלס. מפחיתים 2 סיגמות/סקאלות סמוכות של תמונת גאוס ומחפשים עבור הנקודה מקסימום לוקאלי.
* שימוש בעוצמות בתוך חלון מסוים כמתאר נקודת ענין הוא לא טוב משום שתזוזה קטנה מתמונה לתמונה משפיעה המון.
* **Scale Invariant Feature Transform – SIFT**:

1. חלוקת החלון ל 4\*4 תתי חלון
2. עבור כל תת חלון נחשב את ההיסטוגרמה של כיווני הגראדיאנטים ברזולוציה של 45 מעלות, כלומר 8 זוויות יחוס.
3. סך הכל נקבל 4\*4\*8 = 128 מימדים עבור כל חלון, כאשר המתאר חסין לסקאלות שונות

* מציאת זווית אופיינית לחלון של נקודת ענין:

1. צור אליפסת second moment שמתארת את החלון
2. נרמל אותה לצורת עיגול
3. צור היסטוגרמה מכיווני הגראדיאנטים בחלון
4. מצא את הפיק של ההיסטוגרמה המוחלקת
5. הפיק הזה הוא הזווית האופיינית שאליה יש לסובב את העיגול

* תכונות SIFT:

1. רובסטי
2. יכול להתמודד עם 60 מעלות סיבוב
3. יכול להתמודד עם שינויי תאורה
4. מהיר ויעיל ל realtime
5. לכל נקודת עניין בתמונה יש:
6. 128 מימדים של כיווני גראדיאנטים
7. סקלה אופיינית
8. זווית אופיינית
9. 2 ערכים שמתארים את הנקודה: x,y

* על מנת למצוא התאמות בין 2 מתארים נקודות מגדירים פונקציית מרחק ומחפשים עבור כל המתארים את המתאר הקרוב ביותר.
* **SSD** – פונקציית מרחק בין 2 מתארים המחשבת את סכום ריבועי ההפרשים בין עבור כל מימד. היא לא טובה כל כך כי לעיתים יש כמה התאמות שמקבלות ציונים גבוהים
* **שיטת ) :ratio distance**לחישוב טוב יותר של ציון המרחק)

ratio distance = SSD(f1, f2) / SSD(f1, f2’), כאשר f2 הוא ההתאמה הטובה ביותר ו f2’ הוא ההתאמה השניה הטובה ביותר. אם הערכים קטנים (בהנחה ש SSD נותנת ציונים גבוהים להתאמה) יש ערפול, אבל אם הערכים גבוהים יש בטחון בהתאמה.

* True positives – כמות ההתאמות שהיו נכונות
* False positives – כמות ההתאמות שהתגלו כשגויות
* True positive rate – כמות ההתאמות הנכונות מתוך כלל התואמים
* False positive rate – כמות ההתאמות שהתגלו כשגויות מתוך כלל הלא התואמים
* **עקומת ה ROC** – עקומה המייצרת עבור ספים שונים את היחס בין True positive rate מול False positive rate, כאשר מנסים למקסם את השטח מתחת לעקומה ((**AUC.** טוב להשוואה בין שיטות שונות.
* **שיטת least squares ורטיקלית למציאת קו**:

עבור n נקודות מצא m,b הממזערים את והישר הוא *y=mx+b .* המשוואה שיש לפתור היא *XB=Y* ופתרונההוא B=X\Y. {לשים לב שזו לא חלוקה רגילה '/'}

* הבעיה עם least squares ורטיקלית היא שהיא לא חסינה לסיבוב וגם שהיא כושלת בקווים אנכיים
* **שיטת total least squares למציאת קו**:
* עבור n נקודות מצא a,b,d הממזערים את והישר הוא *ax+by=d*.

הפתרון הוא למצוא את הוקטור העצמי של UTU

המתאים לערך העצמי הקטן ביותר, בכפוף לזה

ש *a*2*+b*2*=*1ו *N=*(*a, b*)הוא הנורמל לקו*.* כמו כן *d=*

* ניתן לראות את total least squaresכמזעור של שגיאות רעש גאוסיאניות שהן בכיוון אנכי לקו.

(*x*, *y*) נקודה כל שהיא, (*u*, *v*) נקודה הקרובה ביותר על הקו, רעש.

* הבעיה בשיטת הריבועים הפחותים היא ש outliers משבשים את הקו שמתאים לרוב הנקודות.
* שיטת ריבועים פחותים והנורמל לא יתנו אותה תוצאה בהכרח, למשל אם הקו קרוב ל 90 מעלות. סטייה קלה מהקו בריבועים פחותים היא גדולה מאוד לעומת זו שבשיטת הנורמל.
* **RANSAC**:

שיטה טובה להתאמת מודל בנוכחות outliers.

האלגוריתם להתאמת קווים:

א. חזור **N** פעמים:

1. בחר **s** נקודות באקראי
2. התאם להם קו
3. מצא את כל הנקודות הפנימיות ב data שהשגיאה/מרחק שלהן מהקו קטן מ **t**
4. אם יש **d** נקודות פנימיות או יותר קבל את הקו, ועדן אותו בעזרת כל הנקודות הפנימיות.

ב. בחר את המודל הטוב ביותר מבין כל הנסיונות

* הערות לפרמטרים לעיל:

1. s בד"כ הוא המספר המינימלי הנחוץ להתאמה למודל
2. t יהיה כזה שיש סבירות גבוהה להמצאות נקודה פנימית בתוכו למשל 95 אחוז
3. N יהיה כזה שבהסתברות גבוהה למשל 99 אחוז, לפחות פעם אחת נגריל נקודות ללא outliers, וככל שאחוז ה outliers גדל יש לבחור N גדול יותר.
4. D צריך לתאום לאחוז הנקודות הפנימיות הצפוי

* יתרונות ה RANSAC: פשוט, כללי, עובד טוב פרקטית ומתאים לבעיות רבות
* חסרונות ה RANSAC: פרמטרים רבים לכוונן, לעיתים נחוצות איטרציות רבות, נכשל כשיש מעט inlier, לעיתים דומה לביצועי brute-force
* **Voting schemes** – כשיש יותר ממודל אחד ניתן לכל feature להצביע על המודל שהם תואמים לו, בהנחה ש feature רעש לא יצביעו בעקביות על מודל מסוים, וכן שישנם מספיק feature-ים טובים שיצביעו על מודל מסוים
* **Hough transform לקווים עבור מרחב (m,b):**

1. חלק את המרחב הפרמטרי (m,b) לתאים דיסקרטיים ואפס אותם
2. כל נקודה בתמונה תיתן הצבעה לתאים הפרמטריים שאליהם היא יכולה להיות שייכת
3. מצא את התאים עם הכי הרבה הצבעות

* קו במרחב התמונה מתאים לנקודה במרחב הפרמטרי (m,b), נקודה במרחב התמונה מתאימה לקו במרחב הפרמטרי (m,b), ולכן אם יש 2 נקודות ששיכות לאותו ישר הן יצרו קווים נחתכים במרחב הפרמטרי ובפועל התא עם הפרמטרים של הקו יקבל ניקוד כפול.
* הבעיות עם המרחב הפרמטרי (m,b) הן שהפרמטרים הם אינסופיים וכן שצריך m אינסופי לייצג קו אנכי
* **Hough transform לקווים עבור מרחב פולרי (,):**

1. את הקו נייצג באמצעות כאשר היא הזווית של הנורמל לקו אשר עובר דרך הראשית, ו הוא אורך הנורמל מהראשית לקו.
2. לכל נקודה (x,y):

אפשרות אחת: עבור על כל הזוויות הדיסקרטיות מ 0 עד 180 חשב לכל אחת מהן את ה על פי הנוסחא שלעיל ותן הצבעה לתא התואם במרחב הפולרי.

אפשרות אחרת: לקחת רק את הזווית של הגראדיאנט בנקודה ורק עליה לחשב את ה .

1. בחר את התא/ים שהם מקסימום לוקאלי.

* בשיטת Hough ככל שיש יותר רעש על הקו מספר ההצבעות בתא הנכון יורד.
* ככל שיש יותר רעש אחיד נראה יותר נקודות בתא עם ההצבעות הרבות ביותר
* טיפול יעיל ברעשים עבור Hough:

1. בחירת קוונטיזציה תואמת לתאים – לא גס מדי שלא לאחד קווים שונים, ולא עדין מדי שלא לפספס קווים
2. החלקת המערך והצבעה לתאים שכנים
3. התחשבות מראש רק בנקודות בעלות עניין עם עוצמת גראדיאנט גבוהה

* **Hough transform למעגלים**:

1. כל נקודה (x,y) על מעגל מקיימת: x = a + Rcosθ , y = b + Rsinθ
2. עבור R ידוע כל נקודה במרחב התמונה שקולה לעיגול במרחב הפרמטרים (a,b). ניתן הצבעה למעגל הזה במרחב הפרמטרים
3. 3 נקודות יציירו 3 מעגלים במרחב הפרמטרים (a,b) שהמפגש ביניהם יתן את הפרמטרים a,b הרצויים, כלומר במיקום הנכון במרחב הפרמטרים הם יקבלו 3 הצבעות.
4. אם R לא ידוע נרחיב את מרחב הפרמטרים להיות תלת מימדי ולכלול את ה R כפרמטר.

* **אלגוריתם HOUGH מוכלל למציאת צורה**:

1. המטרה היא למצוא צורה בהינתן נקודות הגבול שלה ונקודה יחוס פנימית בתוכה a.
2. עבור כל נקודת גבול p נחשב את וקטור ה displacement r=a-p, שהוא הוקטור בין p ל a, כאשר נשייך אותו לזווית θ שהיא זווית הגראדיאנט עבור p.
3. בהינתן צורה לאבחון, עבור כל נקודת גבול p בעלת זווית גראדיאנט θ:
4. מצא את כל וקטורי r המשוייכים לזווית הנ"ל
5. עבור כל אחד מהם הצבע במרחב hough (מרחב תמונה) במקום ה p+r
6. המיקום במרחב hough שיתן את השיא היא נקודת היחוס
7. אם השיא חוצה איזה סף אזי זוהי הצורה המבוקשת

* הערות לאלגוריתם HOUGH מוכלל למציאת צורה:

1. ההנחה היא שיש רק הזזה, ואין שינוי באוריינטציה או ב scsaling
2. טוב להסתרות, לכמה מופעים של המודל וכן לרעש כי נקודות רועשות לא יתרמו בצורה עקבית לתא מסוים במרחב hough
3. החסרונות הם שזמן החיפוש עולה אקספוננציאלית ככל שמספר הפרמטרים גדל, קשה לבחור גודל גריד מספיק טוב, צורות אחרות מהצורה המבוקשת גם יכולות לתת פיקים.

* **אלגוריתם K-means**:

1. בהנתן מספר הקבוצות K עם data מסוימת, המטרה היא למצוא K מרכזים שמשוייכים ל K קבוצות, כך שהסכום הכללי עבור כל הקבוצות של {סכום ריבועי המרחקים האוקלידיים בין כל חבר בקבוצה למרכז שלו} הוא מינימלי.



1. הצעדים:
2. בחר אקראית K מרכזים
3. בצע איטרטיבית עד להתכנסות – אין שינוי בשיוך נקודות לקבוצות
4. שייך כל נקודה למרכז הקרוב ביותר
5. חשב את המרכז כממוצע של כל הנקודות שמשויכות אליו

* **איתור פנים** – איפה יש פנים בתמונה, **זיהוי פנים** – מי הבן אדם בתמונה
* הקושי באיתור פנים נובע מתאורה משתנית, זווית הפנים והסתרות
* **זיהוי עור באמצעות צבע**:

1. ההנחה היא שצבע עור נמצא בטווחים מסוימים
2. על סמך הניסיון נבנו כללים וטווחים שבעזרתם נוכל לומר שפיקסל R,G,B הוא עור אם הוא עונה על הכללים הנ"ל

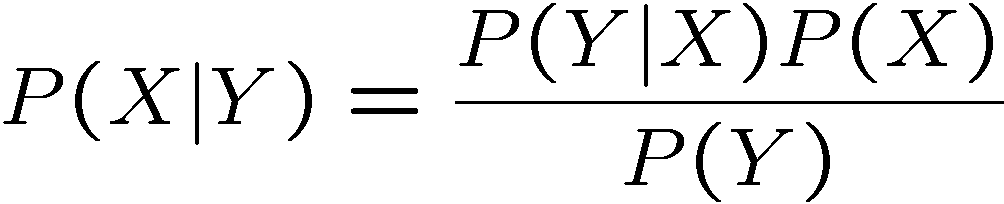
* **PDF** – פונקצית הסתברות הצפיפות שבהנתן X משתנה מקרי נותנת את ההסתברות לקבל את הערך הנ"ל. כשהיא רציפה השטח תחתיה הוא 1, וכשהיא בדידה סכום ערכיה עולה ל 1.
* **Bayesian estimation להסתברות לעור**:

1. מצא את העקומות P(skin|R) (**posterior**) ואת P(~skin|R)=1- P(skin|R)
2. נקודות החיתוך ביניהן R1,R2 מציינות את הטווח שבו ההסתברות שערכי R הם עור גדולים מההסתברות שהם לא עור.
3. **Likelihood** - בצורה בדידה נחשב הסתברות לצבע בהינתן פיקסל עור ע"י היסטוגרמה על הפיקסלים בתמונות האימון:



1. עבור מרחב רב מימדי נחשב ע"י התפלגות גאוסיאנית רב מימדית:

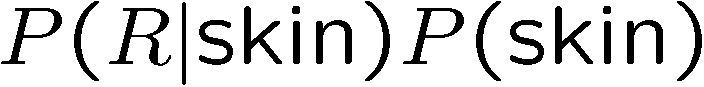
כאשר הוא ממוצע רב מימדי, ו מטריצת שונות משותפת.

1. **כלל בייס**: 
2. 

כאשר **normalization** **term** -

Edittex

ו **prior** הוא P(skin) הנלמד מתוך ה domain למשל: הסבירות לעור עולה אם יש אדם בתמונה, או במרכז פורטרט סיכוי גבוה למציאת עור, או מתוך הפרופורציה של פיקסלי העור מתוך תמונות האימון.

1. **Maximum A Posteriori (MAP)** – להביא למינימום את האפשרות לטעות בזיהוי של ה posterior.
2. **posterior (unnormalized)** – החיתוך שבין העקומה  לבין העקומה 

* **הגישה ה"הוליסטית" ליצוג פנים**:

1. המטרה היא למצוא תת-מרחב לינארי נמוך מימדים שמתאר היטב את השונות של תמונות פרצופים.
2. תכונות מטריצת Covariance:
3. סימטרית
4. האלכסון מכיל את השונות של כל פרמטר
5. כל איבר אחר שאינו על האלכסון מכיל את ה co-variance בין 2 הפרמטרים, או כמה הם קשורים. 0 מסמן שאין קשר ביניהם.
6. משפט: לכל מטריצה A ממשית וסימטרית קיימת מטריצה ממשית אורתוגונלית (WT=W-1) W כך ש: B = WTAW כש B היא מטריצה אלכסונית.
7. כל eigenvalue מייצג את סך השונות במימד שלו
8. **eigenfaces** – תת המרחב של הפרצופים/וקטורים (נמצא בעזרת PCA) המייצגים את השונות הפרצופית המקסימלית, ובעזרתם נוכל לתאר כל פרצוף כצירוף לינארי שלהם.
9. **face space –** תת-המרחב שאליו מטילים את תמונות הפרצופים המקוריות

* PCA Summary:

1. Given X=[x1|…|xN] |d x N|, d – dimensions, N samples
2. Calc Cov(X) |d x d|, where Cov(i,j) = , µ is the sample mean. We can normalize by dividing with N.
3. Find **d** eigenvalues so that: |Cov(X)- *λ*I| = 0.

Recall

1. For each eigenvalue *λ* find eigenvector v so: Cov(X)\*v= *λ*\*v
2. Construct W matrix |d x m| by selecting the first m eigenvectors after they were sorted by descending order of appropriate eigenvalues. m << d.
3. Data projection: y|m| = WT|m x d| \* (x - µ) |d|
4. Data reconstruction: x|d| = (W|d x m| \*y|m|) + µ|d|

* זיהוי פנים בעזרת PCA:

1. בהינתן תמונות פנים חשב את "מרחב הפנים" בעזרת PCA
2. בהגיע תמונה X, הטל אותה למרחב הפנים ושחזר אותה למרחב התמונות. אם המרחק בין התמונה המקורית למשוחזרת גדול מסף מסוים סווג את התמונה כ"לא פנים".
3. אחרת, נחשב את המרחק בין ההטלה של X לכל אחת מההטלות של פרצופי האימון. ההטלה הקרובה ביותר היא סיווג הפרצוף, בכפוף למרחק מתחת לסף מסוים. אחרת נסווג כפנים לא מוכרות.

* הערות ל Eigenfaces:

1. מערכת אוטומטית, קלה למימוש, מצוינת כ baseline
2. מערכת גרועה לבדה, לא טובה כשאין alignment או רקעים שונים
3. ההנחה של התפלגות גאוסיאנית לא תמיד נכונה
4. המימד של מקסימום שונות לא תמיד טוב לסיווג

* **FLD/LDA**:

1. המטרה: למצוא הטלה למימד נמוך כך שתמקסם את הפיזור הכולל בין הקבוצות, ותמזער את הפיזור הכולל בתוך הקבוצות.
2. בהינתן N דגימות ו C קבוצות:
3. מטריצת הפיזור בתוך הקבוצה:
4. מטריצת הפיזור הכולל בתוך הקבוצות:
5. מטריצת הפיזור הכולל בין הקבוצות:
6. הפיזור הכולל:
7. מחפשים הטלה: ולפיכך הפיזורים הם: ו
8. ההטלה היא איפה:

כי היא מבקשת למקסם את הפיזור בין קבוצתי בהטלות ולמזער את הפיזור התוך קבוצתי בהטלות.

1. הפתרון הוא: שזוהי בעיית eigenvalue כללית. ניקח את הוקטורים העצמיים שלהם משוייכים הערכים העצמיים הגדולים ביותר. עבור C קבוצות יהיו רק C-1 ערכים עצמיים שאינם 0.
2. עבור 2 קבוצות: 

* הערות ל LDA:

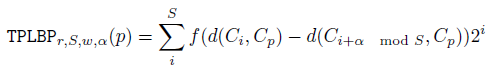
1. יעיל ופשוט למימוש, עובד טוב עם קבוצות לא מאוזנות
2. אין טיפול ישיר ביותר מ 2 קבוצות
3. מפריד לינארי
4. מניח שההתפלגות היא גאוסיאנית, אך יכול לעבוד גם כשזה מופר

* :**LBP**

1. מתאר חסין לתזוזות קטנות ושינויי תאורה
2. עבור כל נקודה ניקח patch סביבה בגודל 3\*3:
3. ערך הנקודה שהיא מרכז ה patch תהיה הסף
4. בצע thresholding בינארי ביחס לסף
5. יצג את ה patch בעזרת מספר בינארי בן 8 ספרות (כלומר מספר דצימלי בין 0 ל 255) הלוקח את ערכי ה thresholding מהפינה השמאלית עליונה עם כיוון השעון (ללא המרכז)
6. ניתן לשחק עם גודל ה patch, להוסיף חסינות לסיבוב ועוד

* **TPLBP**:

1. הרעיון הוא לקודד קידוד בינארי של דמיון בין patch של הנקודה ל patch- ים סביבו ברדיוס r. הוא נקרא three כיוון שבכל שורת השוואה יש 3 patch מושווים.
2. עבור כל פיקסל p נחשב:

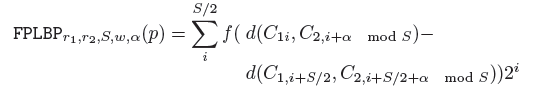


כאשר:

1. **w** – הוא גודל ה patch, **r** – הוא הרדיוס בין ה patch המרכזי לאלו שסביבו, **S** – מספר ה patch-ים סביב ה patch המרכזי, **α** – המרחק בין שני patch-ים סביב שמושווים.
2. **d** – היא פונקציית מרחק כל שהיא בין שני patch-ים למשל L2.
3. **f** – היא פונקציית סף בינארית כלשהיא כשהסף הוא קצת גדול מ 0 , למשל 0.01.

* **TPLBP**:

1. כמו קודם רק שבכל השוואה יש 4 patch-ים מושווים וישנם 2 רדיוסים r1,r2 של patch-ים סביב ה patch המרכזי.
2. עבור כל פיקסל p נחשב:



כאשר הפרמטרים כמו קודם ורק נוספו 2 רדיוסים: **r1 –** הרדיוס הקרוב, **r2 –** הרדיוס הרחוק.

* שלבים כללים בזיהוי פנים:

1. זיהוי הפנים בתמונה
2. ישור
3. יצוג במימד נמוך
4. לימוד
5. בהינתן תמונת מבחן - ציון לדמיון לפנים בכלל/פנים ספציפיות

* **תיאוריית הגבול המרכזי**: בהינתן התפלגות כל שהיא עם ממוצע ושונות, ככל שמספר הדגימות עולה ההתפלגות דומה יותר להתפלגות נורמלית
* **קללת הרב מימדיות**: הקושי לסווג קבוצות כשיש כמה וכמה מימדים
* **מודלים דיסקרימנטיביים לסיווג** – מודלים המנתחים את ההסתברות להימצאות האוביקט בהינתן התמונה, כאשר ישנו סף שממנו והלאה ההסתברות גבוהה שמדובר באוביקט המבוקש. יעיל עם תוצאות טובות. קשה לעיתים להסבירו, וזקוק לדוגמאות חיוביות ושליליות בשלב האימון. לומדים את ההבדלים בין המחלקות וכך מסווגים. כמו SVM, LDA
* **מודלים גנרטיביים לסיווג** – מודלים המנתחים את ההסתברות להמצאות ערכי פיקסלים שונים בהינתן האוביקט, וכשרוצים לסווג יש טווח של הסתברויות עבור כל מחלקה. ניתן להפעיל על מחלקה אחת, וכן ניתן לייצר מהם אובייקטים של מחלקות. לומדים את מבנה המחלקות וכך מסווגים. כמו בייס.
* **Underfitting –** מודל פשוט מדי בעל שגיאות רבות באימון ובבדיקה
* **Overfitting –** מודל מסובך מדי בעל תוצאות טובות (מדי) באימון שכוללות התאמת הרעש וטעויות רבות בבדיקה
* **תער אוקהם** – בהינתן 2 מודלים שמייצגים טוב את ה data בצורה טובה ודומה, העדף את הפשוט יותר.
* **Supervision** – עד כמה המשתמש מעורב בסיווג – מ"בכלל לא" ועד השגחה מלאה.
* ***Textons*** – אלמנטים בסיסיים שחוזרים על עצמם
* **אלגוריתם Bag-of-features לסיווג:**

1. חלץ features
2. צור מילון ויזואלי (**Codebook**) מתוך ה features השונים ע"י אישכול בשיטת K-means. כל מרכז קבוצה הוא **Codevector**. ניתן לעשות על קבוצת אימון נפרדת.
3. **קוונטיזיציה של ה features** - מפה את ה features בעזרת המילון הויזואלי לCodevector הקרוב ביותר
4. יצג את התמונה באמצעות היסטוגרמה של ה"מילים" הויזואליות שלה
5. סווג את התמונה על פי ההיסטוגרמה שלה

* הערות ל"מילון ויזואלי":

1. אם הוא קטן מדי הוא לא ייצג את כל ה patch-ים, אם הוא גדול מדי יהיו בעיות של התאמת יתר בגלל הרזולוציה הדקה, וכן בעיות קוונטיזציה
2. על מנת ליעל את ההתאמה של features ניתן להשתמש בעצים שבהם יש סיווגים ותתי סיווגים בענפים שונים

* **Spatial pyramid representation**:

יצוג תמונה המתחשב במרחביות. ישנן כמה רמות. ברמה האפס יש היסטוגרמה רגילה. ברמה אחריה התמונה מחולקת ל 4 ולכל חלק יש היסטוגרמה משלו, וכן הלאה בשאר הרמות.

* **מסווג kNN:**

1. לכל נקודת בדיקה מצא את K הנקודות הקרובות ביותר מקבוצת האימון
2. סווג את הנקודה לפי התווית של רוב הנקודות הקרובות
3. עובד טוב אם יש הרבה data ופונקציית המרחק היא טובה

* הערות ל kNN:

1. הסיווג הוא ישירות על ה data ללא צורך בלימוד או אימון
2. מטפל בטבעיות ביותר משתי קבוצות
3. אין פרמטרים ולכן אין חשש להתאמת יתר
4. עשוי לצרוך הרבה זמן עיבוד
5. ביצועים פחותים ממסווגים עם פרמטרים כיוון ש:
6. הדיסקריפטורים שמבדילים הכי טוב בין מחלקות הם אלו הנדירים, ובשל נדירותם בתהליך ה k-means הם נבלעים במחלקות גדולות יותר ולא מקבלים מחלקה משל עצמם, ולכן אנחנו מאבדים אינפורמציה שהכי עוזרת לסיווג.
7. ההשואה היא תמונה לתמונה, ולעיתים יש תמונה שכוללת רכיבים אחרים, או שיש שונות גדולה במחלקה ואז הסיווג נהיה גרוע. לכן מול כל מחלקה יש להשוות את המרחק בין כל רכיב בתמונה המבוקשת לרכיב הקרוב אליו ביותר מכל הרכיבים **בכל** תמונות המחלקה, ולקחת את המחלקה שעבורה סכום המרחקים עבור כל רכיבי התמונה המבוקשת הוא מינימלי.

ובנוסחא: , C היא מחלקה, di רכיב בתמונה המבוקשת מתוך N רכיבים.

* **מרחק L1**:
* **מרחק L2**:
* **מסווג SVM**:

1. המטרה היא למצוא hyperplane שממקסם את השוליים (**support vectors**) שבין קבוצה אחת (החיובית) לקבוצה השניה (השלילית) מחד ומאידך שההפרדה תיתן כמה שפחות שגיאות
2. ועבור ה support vectors
3. מרחק בין נקודה ל hyperplane:
4. Margin הוא 2 / ||**w**|| ואותו יש למקסם
5. הפתרון הוא בעזרת אופטימיזציה כך ש:
6. נמזער את
7. *yi*(**w**·***x****i*+*b*) ≥ 1
8. בעזרת אנליזה נומרית נמצא משקולות כך ש: , כאשר ה מייצגים את ה support vectors.
9. את ה b נחשב ע"י: *b* = *yi* – **w**·**x***i*
10. פונקציית הסיווג לכל x לבדיקה: , לפי התוצאה 1 או -1 נסווג את ה x.

* **Nonlinear SVMs:**

1. במקרים שבהם קשה להפריד בין הקבוצות ניתן למפות למימד גבוה יותר שבו ניתן יהיה להפריד ביניהן.

2. הדרך היא למצוא טרנספורמציה ***φ***(**x**) שמעבירה את x למימד אחד.

3. *The kernel trick* - במקום לחשב את ***φ***(**x**) ישירות נגדיר פונקציית קרנל: *K*(**x***i ,* **x***j*) ***= φ***(**x***i* )*·* ***φ***(**x***j*) (שמקיימת את תנאי מרסר) כך שפונקציית ההחלטה תהיה:

* הערות ל SVM:

1. על מנת לסווג יותר משתי קבוצות ניתן לעשות:

א. לבצע SVM על כל קבוצה מול השאר באימון, ובבדיקה לבדוק את כל אחד מה SVM-ים ולתת סיווג ל SVM עם ההסתברות הגבוהה ביותר.

ב. לבצע SVM על כל זוג בקבוצות באימון, ובבדיקה לבדוק את כל אחד מה SVM-ים ולתת סיווג ל SVM עם ההסתברות הגבוהה ביותר.

2. עובד טוב אפילו עם מעט דגימות, השימוש ב KERNEL יעיל וגמיש.

3. SVM ליותר משתי קבוצות מסובך

4. קבוצות לא מאוזנות עלולות להיות בעייתיות

5. זמן חישוב ארוך וצריכת זכרון – זמן הלימוד עלול להיות ארוך, וכן צריך לחשב מטריצה לכל זוג דגימות

* **Rectangle Features –** פונקציה המקבלת תת חלון ומחזירה מספר , כאשר ישנם 3 סוגים: Featuresבעלי2,3 או 4 מלבנים שצבועים שחור או לבן שמולבשים על התת חלון, כאשר הפונקציה מחזירה את סכום ערכי הפיקסלים במלבנים השחורים פחות סכום ערכי הפיקסלים במלבנים הלבנים.
* **Integral Image Representation:**

1. תמונה הזהה בגודלה לתמונה המקורית כאשר כל פיקסל בתמונה החדשה מכיל את סכום ערכי כל הפיקסלים מעליו ומשמאלו.

2. ניתן לחשב במעבר אחד על התמונה המקורית בצורה הבאה:

א. s(x,y) – סכום ערכי הפיקסלים המצטבר בשורה x, בעמודה y.

ב. s(x,-1)=0, ii(-1,y)=0

ג. s(x,y) = s(x,y-1) + i(x,y)

ד. ii(x,y) = ii(x-1,y) + s(x,y), כאשר ii היא התמונה המחושבת.

3. ניתן לחשב בעזרת הנ"ל סכום ערכי הפיקסלים שכל מלבן D בתמונה ב O(1):

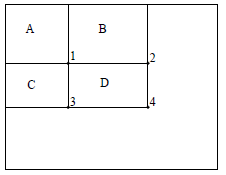
ii(1) = sum(A)

ii(2)= sum(A)+sum(B)

ii(3)= sum(A)+sum(C)

ii(4)=sum(A)+sum(B)+sum(C)+sum(D)

D = ii(4)-ii(2)-ii(3)+ii(1)

כאשר A,B,C,D הם כך: 

* **AdaBoost**:

1. שיטה ליצירת מסווג חזק ע"י שילוב מסווגים חלשים שהביצועים שלהם קצת יותר טובים מאקראי.

2. בהתחלה נותנים משקולות שווים לכל אחת מהדגימות.

3. בצע T איטרציות:

א. לכל אחד מה Features אמן מסווג שמשתמש רק ב Feature אחד, ומדוד את השגיאה הממושקלת שלו.

ב. בחר את המסווג שנותן את השגיאה הממושקלת הקטנה ביותר

ג. עדכן את המשקולות כך שהדוגמאות שכשלו בסיווג מקבלות משקל יותר

ד. המסווג הסופי הוא שילוב כל המסווגים שנבחרו בכל איטרציה כשהם משוקללים לפי השגיאות שלהם.

* אלגוריתם ויאולה וג'ונס:

1. המטרה היא בעזרת שיטת AdaBoost למצוא שילוב של Rectangle Features חלשים עם ספים (שמפרידים בין המחלקות של פנים ו"לא פנים") ומשקולות לפי השגיאות שלהם כך שיתנו מסווג אחד טוב.

2. בכל שלב באיטרציה:

A.עבור כל Rectangle Features (הממוקם במיקום מסוים בתמונה):

א. הפעל את ה Feature על כל הדוגמאות – ה"פנים" וה"לא פנים" וקבל מספר עבור כל דוגמא.

ב. מיין את הדוגמאות לפי המספרים שקיבלו

ג. מצא את הסף שמפריד בצורה הטובה ביותר בין המחלקות (בעל השגיאה המצטברת המינימלית) וזכור את השגיאה עבור ה Feature הזה

B. בחר את ה Feature בעל השגיאה הקטנה ביותר (זכור את הסף עבורו)

C. עדכן את המשקולות.

3. המסווג הסופי הוא שילוב כל ה Feature עם הספים שלהם עם המשקולות שלהם לפי השגיאות שנתנו.

* **Cascading classifiers –** על מנתלסווג בצורה יעילה ולא לטבוע ב"ים" ה Featureהאפשריים, נעשה מפל של מסווגים כאשר בתחילה נפעיל מסווגים פחות מדויקים אך מהירים במקומות שקל לקבוע שמדובר בדוגמאות השליליות, ורק אחר כך נפעיל את המסווגים ה"כבדים" יותר בזמן.
* הערות לשיטת ויאולה וג'ונס:

1. ניתן לישם על כל משימה של זיהוי אוביקט, אבל בעייתי באוביקטים שלא בצורת ריבוע, אוביקטים מעוותים, אוביקטים לא קשיחים, עם טקסטורה לא רגילה

2. מזהה פי 15 יותר מהר מהשיטות הטובות ביותר

3. מגיע לאחוזי דיוק גבוהים

4. גישה מבוססת Features טובה משום שאינה ניגשת ישירות לערכי פיקסלים ובכך לטפל טוב בשונות בין תמונה לתמונה

5. על מנת לזהות פנים בפרופיל יש לבצע אימון מיוחד עם תמונות פרופיל

6. ניתן להשתמש בקונטקסט גיאומטרי או של נקודת מבט כדי להגביל את החיפוש של ה sliding window

7. רגישות להסתרות חלקיות בשל חיפוש ההתאמה הגלובלית

8. דורש סט אימון גדול ומסומן של פנים.

* **HOG – Histogram of gradient**:

1. כל תמונה מחולקת לבלוקים של 8\*8

2. בכל בלוק מחשבים היסטוגרמת גרדיאנט

3. חישוב Features ברזולוציות שונות

* בלה בלה