סיכום עיבוד תמונה

19 במרץ 2023

1 הרצאה 1 ־ 23.10 :

- את האור המוחזר נסמן ב r.
את האור שיוצא מהאובייקט נסמן ב L.
את האור הנקלט נסמן ב I.
היחס: מתקיים היחס $I = L \cdot r$, כלומר האור הנקלט שווה למכפלת האור המוחזר והאור שיוצר מהאובייקט.

- **הראיה האנושית:** יש לנו 10^8 קנים ־ אחראים לראיית שחור לבן. 10^7 מדוכים ־ שאחרים על ראיית הצבע. ו10^4 עצבים ־ כל עצב אחרים על 10,000 קנים. החיישנים קולטים אורכי גל בין התדרים 350־780 ננו מטר.

- כאשר נאיר על תא בעין האנושית, התא המואר מגיב יותר. בעוד התאים שלידו מגיבים חלש יותר.

- **השלבים ביצירת תמונה:**
הטלה פרספקטיבית ־ מעבר מתלת לדו מימד, נסמן את הקאורדינטות של העולם ב(X, Y, Z) ונמיר לקאורדינטות של התמונה ־ (x, z).
פירוק התמונה לפיקסלים.
התאמת צבע לכל פיקסל.

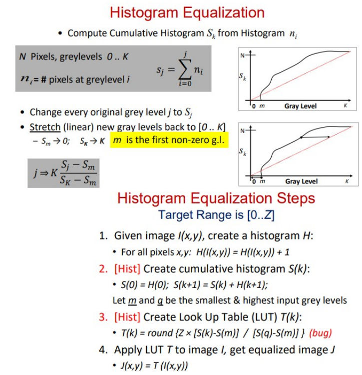
- **אורך המוקד:** נסמן ב f, והוא מסמן את המרחק בין הנקודה דרכה עוברות הקרניים ועד הקיר עליו מוטלת התמונה. הגדלת אורך המוקד תגדיל את התמונה, ולהיפך. נמיר מדו לתלת מימד כך

$$x = \frac{f}{Z}X \quad y = \frac{f}{Z}Y$$

- **היסטוגרמה:** מייצגת את דרגת האפור בתמונה ־ נספור כמה דרגות אפור יש לנו בכל פיקסל ונסדר על סקאלה. כדי לייצג היסטוגרמה של RGB נוכל לייצג ב 3 היסטוגרות ־ אחת לכל צבע. בנוסף ניתן לייצג גם בתלת מימד.

- **ייצירת ניגודיות לתמונה (שיווי היסטוגרמה):** כשיש לנו תמונה עם טווח אפור מצומצם ונרצה ניגודיות גדולה יותר, נוכל להשתמש בהיסטוגרמה כדי לייצר תמונה צבעונית יותר.

אלגוריתם: ניצור היסטוגרמה של התמונה והיסטוגרמה של כל דרגות האפור. נבצע אינטגרל ונעשה העתקה.



- טרנזפורמציה מונוטונית:

2 תרגול 1 - 26.10

- **מספר הביטים פר פיקסל (bpp):**

בתמונות שחור לבן (גווני אפור) - 8 ביטים פר פיקסל. כל ביט בטווח 0-255 או $float$ בטווח 0-1 (ע"י חלוקת המטריצה ב 255).

בתמונות צבעוניות - 24 ביטים פר פיקסל, 1 לכל צבע מה- RGB, כל ביט בטווח 0-255.

- **ייצוג הצבעים:** צבע כהה - יהיה קרוב ל 0, צבע בהיר - יהיה קרוב ל 255.

- **תמונות בינאריות:** תמונות שבהן יש רק שחור ולבן ללא דרגות אפור, נשתמש בזה כשנרצה להדגיש אזור מסויים בתמונה ולעשות עליו פעולות. הפעולה נקראת מסכה בינארית.

- **YIQ ו RGB :** ב YIQ יש הפרדה בין הצל לבהירות. לעומת זאת ב RGB הכל מעורבב. מיפוי מ YIQ ל RGB נעשה ע"י הכפלה במטריצה קבועה.

הערוץ Y מייצג את גווני האפור בתמונה (גווני האור והצל), לכן ניתן לבודד אותו לעשות עליו מניפולציות ואחכ לצרף את ה Y החדש ל IQ המקוריים, שמייצגים את הצבע.

השמשו ב YIQ להקרנה צבעונית על טלויזיה ששידרה שחור לבן.

2.1 טרנספורמציות:

- **סימונים:** s - הפיקסל החדש שיתקבל. r - הערך הנוכחי של הפיקסל.

- $Look\ up\ table$: טבלה שממפה פיקסלים למספרי פיקסלים חדשים.

את כל הפיקסלים באותו צבע נמיר לצבע החדש.

לדוגמה - ניגודיות: $s = 1 - r$

- **טרנסורמציית לוג:** אם נרצה למתוח את טווח הערכים לדוגמה כשיש לנו פיקסלים כהים עם טווח צמוד, ובהירים עם טווח צמוד. נפעיל לוג וכך נפריד את הפיקסלים הכהים ונמרח אותם על טווח גדול יותר, ונצמצם את הבהירים לטווח צמוד יותר.

  נבחר סקלר c, נבחר אותו כך כדי להבטיח שהמספר האחרון שלנו יהיה בטווח 255:

$$c = 255/\log(1 + \text{ max inp val})$$

לאחר מכן

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

- **טרנספורמציית חזקה \ טרנספורמציית גמא:** נבחר קבוע c, ו γ ונמיר כך:

$$s = c \cdot r^{\gamma}$$

ההמרה תתבצע לפי ערך ה γ, וכל γ תבצע פעולה אחרת (הפרדת או צמצום הפיקסלים הכהים).

לדוגמה: נשתמש כדי להנמיך את הבהירות בתמונה.

2.2 שיווי היסטוגרמות:

- **שיווי היסטוגרמות:** נספור את כל הפקסלים שדרגת האפור שלהם שווה לדרגה ה i עבור כל דרגות האפור, ונסמן ב n_i. נסכום את כל ה n_i ונגדיר את הסכום להיות Y_i. לאחר מכן נבצע: (כאשר N הוא מספר הפיקסלים בתמונה, ו k הוא הפיקסל הגבוה ביותר)

$$s_k = \frac{255}{N} y_k = \frac{255}{N} \sum_{i=1}^{k} n_i$$

ונשתמש בהיסטוגרמה להיות ה $look\ up\ table$.

אלגוריתם: (Z הוא הערך המקסימלי = 255)

Histogram Equalization: Algorithm

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

cumulative histogram *C(k)*
Let *m* be first grey level for which C*(m)*≠*0*
T(k) = round{ [C(k)-C(m)] / [C(255)-C(m)] × 255}

- **תכונות של שיווי היסטוגרמה:**

  1: נגיע לקירוב של פונקציית הזהות.

2: מתקיימת מונוטוניות, הטווח ישתנה אבל הסדר לא ישתנה. אם תא i היה בהיר יותר מתא j הוא ישאר כך גם אחרי השיווי.

3: כמות ערכי הפיקסלים השונים לא תגדל, אלא תקטן או תישאר שווה, משום שניתן רק לאחד ולא להפריד.

4: לא נעשה שיווי על מסמכים כי זה מיותר.

2.3 קוונטיזציה $Quantizarion$ - צמצום גוונים:

- **המטרה:** להשאיר את התמונה קרובה לתמונה המקורית, אך לצמצם את מספר הצבעים בתמונה.

- **אפשרות ראשונה:** נחלק את ההיסטוגרמה לשלש, ונבחר כל שליש להיות הצבע האמצעי שבאותו השליש.
חסרון: זה לא מתייחס לתמונה, יהיו תמונות שיעלמו לאחר התהליך.

- **אפשרות נוספת:** נרצה למפות את התמונה לקבוצות רלוונטיות, ולצמצם את השגיאה.
נעשה זאת ע"י נרמול השגיאה, והתייחסות לחשיבות של כל גוון:

$$\min \sum_{i=0}^{3} \sum_{z_i}^{z_{i+1}} (q_i - z)^2 \, p(z)$$

עבור:

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

כאשר: q_i- הוא הצבע החדש שאותו שליש של התמונה יקבל. ו z_i - הוא המיקום בו נחלק את התמונה לשלישים.
$p(z)$ - ההסתברות לקבל את הגוון הזה (ההיסטוגרמה המנורמלת עם ערכים בטווח [0,1]). לאחר מכן נגזור לפי z ופעם נוספת לפי q ונמצא נקודות קיצון.
המימוש: נתחיל מבחירה ראשונית של חלוקה - z, לאחר מכן נבחר את הנקודה q, ונתקן שוב את z ונחזור לתקן q וכן הלאה, עד שנתכנס למינימום.

3 טרנספורמציית פוריה:

- **רעיון:** ייצוג של אותות ותמונות, במקום ייצוג נקודתי של פיקסלים אנו נייצג בעזרת סכום של גלי סינוס.

- **גלים - פונקציה מחזורית:** לכל גל יש מספר מאפיינים המאפיינים אותו -
אורך הגל: מה אורך המחזור של הגל.
תדר: נמוך או גבוה - כמה גלים נמצאים במרחק של יחידת זמן אחת (כמה הפונקציה דחוסה או רחבה).
גובה - $amplitude$: גובה הגל.
פאזה: האם הוא מתחיל מ - 0 או לא (איפה אנו נמצאים לאורך המחזור - כמה הסיגנל הוסת מה - 0).

- **ההנחה של פוריה:** כל פונקציה **מחזורית** יכולה להירשם כסכום משוקלל של סינוס וקוסינוס בתדרים ומשקולות שונים.

- **נשים לב:** מכיוון שפוריה מניח מחזוריות של הפונקציה, כשנרצה לעבוד עם פוריה נצטרך לשכל את התמונה כדי שנקבל פונקציה מחזורית מלמעלה מלמטה ומהצדדים.

- **מספרים מרוכבים:** מספר המורב מחלק ממשי ומדומה, כאשר a ממשי ו b מדומה. ניתן לכתוב אותו גם בעזרת רדיוס R כך:

$$a + bi = R \cdot e^{i\alpha}$$
$$e^{i\alpha} = \cos(\alpha) + i\sin(\alpha)$$

נגדיר את R **להיות:** $R = \sqrt{(a^2 + b^2)}$ **ואת הזווית (פאזה):** $\alpha = \tan^{-1}\left(\frac{b}{a}\right)$

מוטיבציה: במכפלת מספרים מרוכבים נקבל - $R_1 e^{i\alpha_1} \cdot R_2 e^{i\alpha_2} = R_1 R_2 e^{i(\alpha_1 + \alpha_2)}$

- **גל סינוס של** $sin(x)$**:** אורך הגל הוא 2π. התדר הוא: $\frac{1}{2\pi}$.
הגדלת התדר: נחליף ל $sin(2x)$ כך התדר יגדל, אך אורך הגל יקטן ל π.
עבור $sin(ax)$**:** אורך הגל - $\frac{2\pi}{a}$. תדר הגל - $\frac{a}{2\pi}$

- **נוסחה למציאת תדר:** עבור אורך גל L. התדר יהיה $\frac{1}{L:}$.

- **ביטוי חלופי:** אנו לא רוצים להתעסק עם 2π ואורכי גל ותדר. לכן נסתכל על - $sin(2\pi\omega x)$. עבורו **אורך הגל** - $\frac{1}{\omega}$ **והתדר** - ω.

- **כפל בקבוע** A **ושינוי** $amplitude$**:** אם נכפול $A \cdot sin(2\pi\omega x)$ נקבל $amplitude$ ששווה ל A - גבוה יותר.

- **שינוי פאזה:** אם נרצה לשנות את הפאזה נחסר את φ בתוך הסינוס כך - $\sin(2\pi\omega x - \varphi)$. באופן זה הסינוס יתחיל את המחזור ב - $\frac{\varphi}{2\pi\omega}$.

3.1 טרנספורמציית פוריה דיסקרטית:

- **הרעיון:** יש לנו סדרת ערכים (ממשיים במקרה של תמונה) ונמיר אותם לערכים מרוכבים. כלומר נרצה לייצג את הפונקציה שלנו באמצעות סכום של סינוסים וקוסינוסים בתדרים שונים, כאשר כל מספר מייצג כמה לקחנו מהתדר ה i כדי להרכיב את הפונקציה.

$$f(x)|(f(0), f(1), \ldots, f(N-1)) \Rightarrow F(u)|(F(0), F(1), \ldots, F(N-1))$$

למעבר הזה מ $f(u)$ ל $F(x)$ נקרא טרנספורמציית פוריה.

- **נוסחה - טרנספורמציית פוריה:** עבור טרנספורמציה חד מימדית - עבור כל אחד מהתדרים נחשב את הערך הזה

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) e^{\frac{-2\pi iux}{N}}$$

כך שעבור $u = 0$ (תדר ה־ 0) - מייצג את ממוצע התמונה, הגדלת תדר ה 0 תגדיל את הממוצע ותבהיר את התמונה. $u = 1$ - ייצג כמה צד שמאל יהיה גבוה מצד ימין וכן הלאה.

- **נוסחה ־ טרנספורמציית פוריה הפוכה:**

$$f(x) = \frac{1}{1}\sum_{u=0}^{N-1} F(u)e^{\frac{2\pi iux}{N}}$$

- **סיבוכיות חישוב** ־ $O(N^2)$. ניתן לעשות אותו באופן מהיר יותר ב $O(N \cdot log(N))$.
- **פוריה וקטור בסיס ־ נוסחת אויילר:** עבור כל תדר $0 \leq u \leq N-1$ יש פונקציות בסיס עבור כל $0 \leq x \leq N-1$:

$$e^{\frac{2\pi iux}{N}} = cos\left(\frac{2\pi iux}{N}\right) + i \cdot sin\left(\frac{2\pi iux}{N}\right)$$

- **הקשר בין תדר ומספר הדגימות:** התדר הגבוה ביותר עבור N דגימות הוא $\frac{N}{2}$. ואורך הגל הקטן ביותר הוא 2 .
- **תכונות של טרנספורמציית פוריה:**

$$1: F(u) = F(u+N)$$
$$2: F(u) = F^*(-u) = F^*(N-u)$$
$$3: (a+bi)^* = (a-bi)$$
$$4: |F(u)| = |F(-u)$$
$$5: \Phi(f(x)+g(x)) = \Phi(f(x)) + \Phi(g(x))$$
$$6: \Phi(a \cdot f(x)) = a \cdot \Phi(f(x))$$
$$7: \boldsymbol{if} - f(x) \underset{\text{Fourier}}{\longrightarrow} F(u) \ \ \boldsymbol{than} - f(ax) \underset{\text{Fourier}}{\longrightarrow} \frac{1}{|a|} \cdot F\left(\frac{u}{a}\right)$$

1: עבור מחזור באורך N הטרנספורמציה גם מחזורית באותו N.

2: הטרנספורמצייה שווה לצמוד שלה ־ סימטריה.

- **ייתרונות הסימטריה:** נוכל לחשב את החצי הראשון $F(0), F(1), \ldots, F\left(\frac{N}{2}-1\right)$ ואת החצי השני לחשב בעזרת הסימטריה כי $F(N-u) = F^*(u)$.

  לכן גם המימד לא גדל עם המעבר ממשי למרוכב (נשארנו עם N מספרים ב"ת).
- **כיצד נסתכל על טרנסורמציית פוריה:** מכיוון שהיא מחזורית נעדיף להסתכל עליה כאשר 0 נמצא באמצע והטווח הוא $[-N/2, N/2]$.

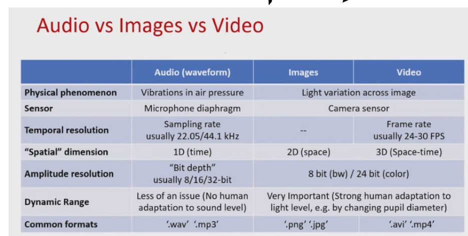
3.2 גלי קול:

- **איכות הקול:** תלויה בתדר הדגימה ־ כמות הדגימות שאנו דוגמים בשניה.

- **מספר הדגימות:** צריך להיות **לפחות כפול** מהתדר המקסימלי (אם נרצה תדר של $20KH$ נצטרך $40KH$ דגימות בשניה).

- **פורמט הקול:** מיוצג בעזרת מספרים בטווח $[0, 1]$.

- **תמונות לעומת קול ווידאו:**



Audio vs Images vs Video

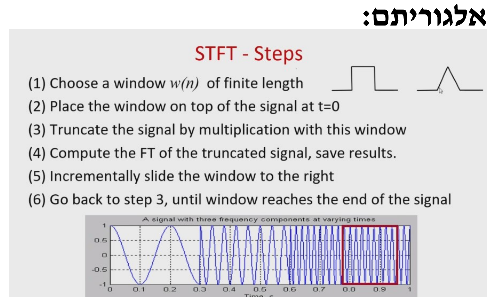
	Audio (waveform)	Images	Video
Physical phenomenon	Vibrations in air pressure	Light variation across image	
Sensor	Microphone diaphragm	Camera sensor	
Temporal resolution	Sampling rate usually 22.05/44.1 kHz	--	Frame rate usually 24-30 FPS
“Spatial” dimension	1D (time)	2D (space)	3D (Space-time)
Amplitude resolution	“Bit depth” usually 8/16/32-bit	8 bit (bw) / 24 bit (color)	
Dynamic Range	Less of an issue (No human adaptation to sound level)	Very Important (Strong human adaptation to light level, e.g. by changing pupil diameter)	
Common formats	‘.wav’ ‘.mp3’	‘.png’ ‘.jpg’	‘.avi’ ‘.mp4’

3.2.1 סיגנל סטציונרי לעומת לא סטציונרי:

- **סיגנל סטציונרי:** גל כך שהתדרים בכל נקודה הם אותם תדרים (מספר תדרים משולבים יחד).

- **סיגנל לא סטציונרי:** גל שבכל נקודת זמן נמצא בתדר אחר, התדר משתנה לאורך הזמן (מספר תדרים אחד אחרי השני). במצב זה לא נוכל לדעת לאחר טרנספורמציה איפה נמצא כל תדר, כי ההתמרה מחזירה לנו את מרחב התדר אך מאבדת את מרחב הזמן, כי היא מניחה שאותו התדר היה כל הזמן.
הפתרון: נבצע אנליזה מקומית של טרנספורמציה.

- **התמרת פוריה בגל קול לא סטציונרי -** $STFT$: נחלק את הגל למרחבים לפי חלונות (מרובעים או משולשים) של התדרים השונים של הגל ונבצע התמרה על כל חלון בנפרד.
אלגוריתם:



כעת נוסחת ההתמרה תראה כך:

$$\text{STFT (for every } n\text{)} \quad F(n,u) = \sum_{m=-\infty}^{\infty} f(m)w(n-m)e^{\frac{-2\pi i u n}{N}}$$
$$\text{where } n \text{ is time}$$

וההתמרה ההפוכה תראה כך:

$$\text{ISTFT (Inverse)} \quad f(n) = K \sum_{p=-\infty}^{\infty} \sum_{u=0}^{N-1} F(pL,u)e^{\frac{2\pi i u n}{N}}$$
$$\text{Skipping } L$$

For a carefully selected window w, skip L, and normalization K

L אומר לנו בכמה לקדם את החלון בכל פעם - נרצה שיהיו חלקים חופפים בין שני חלונות עוקבים.

- **ספקטרוגרמה - $Spectogram$:** מייצגת את הגלים לפי צבעים, כאשר ציר ה Y מייצג את התדר (נמוך = כחול, גבוה = אדום), וציר ה X מייצג את הזמן.

כשנרצה שהיא תהיה עשירה יותר נבצע פעולת Log כך: $\log(|F(u)|+1)$. פעולה זו תצמצם את המרחק בין התדרים השונים וכך תציג את כולם.

לאחר מכן נמרח את התדרים מחדש על כל הטווח המקורי.

3.2.2 בחירת גודל החלון:

- **כיצד נבחר את גודל החלון:** אם נבחר חלון גדול אנחנו נקבל הרבה רזולוציה בתדר אך אין לנו רזולוציה בזמן. מצד שני עם חלון קטן אנו נאבד את הרזולוציה בתדרים אך יהיה לנו ערך בכל נקודת זמן.

  נרצה למצוא את החלון שיאזן בין שניהם. נעבוד לפי הטבלה הבאה:

Sampling KHz	Window length		Hop length	
8	200	25 ms	80	10 ms
22.05	551	25 ms	220	10 ms
44.1	1102	25 ms	441	10 ms

  העמודה השלישית מייצגת את רוחב החלון בזמן.

- **חפיפה בין החלונות:** נשאף שהחלונות יחפפו ביניהם משום שכך אנו נוכל לוודא להכנסנו את כל האינפורמציה ולא נאבד מידע.

- **צורת החלון:** נעדיף לקחת חלון גאוסיאני מאשר חלון סימטרי. כך החלק הרלוונטי לחלון הספציפי יהיה באמצע הגאוסיאן ויקבל את המשקל הגבוה ביותר.

3.2.3 מדוע עדיף לעבוד עם פוריה:

- **כיצד נבצע $Fast\ Forward$:** בד"כ כשנעלה את מהירות הקלטה, התדר גם אמור לעלות ולהישמע צורם יותר. כיצד ניתן היום להריץ את ההקלטה מבלי לקבל תדר צורם יותר?

- **דרך ראשונה:** עבור קצב דגימה של $8KH$, נגדיר שקצה הדגימה כרגע הוא כפול 2 - $16KH$ וכך זמן ההקלטה יקטן בחצי, אך התדר יהיה גבוה והקול ישמע צורף.

- **דרך שניה:** לא נשנה את התדר ונשאיר אותו $8KH$, אך נזרוק כל נקודת דגימה זוגית ונשמור את האי זוגיות. אך גם כאן הקול ישמע צורמני. (איכול הקול באפשרות זו גרועה יותר).

- **דרך שלישית ונכונה:** נבנה ספקטוגרמה, ונכווץ אותה בציר הזמן - x **ולא את הסיגנל**, כך התדר (ציר ה y לא ישתנה) ורק ציר הזמן יקטן והמהירות תגדל.

- **$Fast\ Forward$ עם ספקטוגרמה:** לאחר הכיווץ, אנו נצטרך לתקן את הפאזה, משום שהזזת החלון לא תהיה באותו הקצב.

3.3 התמרת פורייה לתמונות:

תמונות מיוצגות בעזרת מערך דו מימדי כאשר התמונות הרגילות מיוצגות בבסיס הטבעי - כל פיקסל אומר מה הערך באותו הפיקסל.
בבסיס פורייה יהיו לנו n^2 מספרים שמיוצגים בעזרת מקדמי פורייה. כלורמ נעביר מייצוג פיקסלי לייצוג פורייה.

- **בסיס פורייה:** עבור כל תמונה בגודל n יש בסיס נתון שמרכיב את כל התמונות בעולם שהן בגודל n. אנו רק מחפשים את המשקולות המתאימים.

- **טרנספורמציית פורייה דו ממדית:**

$$F(u,v) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)e^{\frac{-2\pi i(ux+vy)}{N}}$$

- **הטרנספורמציה ההפוכה לדו מימד:**

$$f(x,y) = \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F(u,v)e^{\frac{2\pi i(ux+vy)}{N}}$$

- **הפעלת טרנספורמציית פורייה פעמיים:** יחזיר לנו את השיקוף של התמונה - $x, y \Rightarrow -x, -y$

- **מה מסמל** $F(u,v)$**:** (u,v) - מסמנים לנו על איזה פונקציות בסיס אנחנו מדברים (פונקציות הבסיס שמרכיבות את התמונה). ו F אומר לנו באיזה מקדמים להכפיל את פונקציית הבסיס (α, β וכו מהדוגמה של הקול).
כלומר - הבסיס עבור תמונה בגודל n כבר נתון לנו, אנו מחפשים את המקדמים בהם נכפול את הבסיס הנתון בכדי להגיע לתמונה הספציפית.

- **פורייה ספקטרום:**
פורייה זהו מספר מרוכב:

$$F(u) = R(u) + i \cdot l(u)$$

והספקטרום $(Magnitude)$ **הוא הערך המוחלט של פורייה:**

$$|F(u)| = \sqrt{R^2(u) + I^2(u)}$$

הפאזה היא:

$$\theta(u) = \tan^{-1}\left(\frac{I(u)}{R(u)}\right)$$

ייצוג נוסף לפורייה בעזרת הספקטרום והפאזה:

$$F(u) = |F(u)|\exp(i\theta)$$

- **הצגה:** נחשב את $log\left(|F(u)+1|\right)$, לאחר מכן נמתח על כל הטווח $[0,255]$ ואח"כ נעביר את הנקודה $(0,0)$ להיות באמצע.

- **שאלה**: מדוע ל $|F(0,0)|$ יש את הספטרום הגבוה ביותר?

תשובה: משום שהוא לא מוכפל בסינוס והקוסינוס. כלומר זה סכום כל התדרים בצורה חיובית בלבד ללא אף גורם שלילי.

הוא יהיה הכי גבוה רק אם כל הפיקסלים בתמונה חיוביים.

- **תכונות של טרנספורם פורייה:**

הזזה ציקלית של התמונה - משפיע על הפאזה:

$$f\left(x-x_0,y-y_0\right)\Leftrightarrow F(u,v)e^{\frac{2\pi i(ux_0+vy_0)}{N}}$$

הזזה ציקלית של פורייה - שינוי פאזה של התמונה יהיו מספרים מרוכבים:

$$F\left(u-u_0,v-y_0\right)\Leftrightarrow f(x,y)e^{\frac{-2\pi i(ux_0+vy_0)}{N}}$$

לכן: נוכל להזיז את התמונה ולקבל פורייה ולחזור. אך לא ניתן להזיז את פורייה ולחזור, כי אז לא נקבל את התכונות שיחזירו אותנו לתמונה.

- **דיקומפוזיציה - הפרדת של טרנספורם פורייה בין x ל y:** ניתן לחלק לטרנספורם על העמודות ועל השורות בנפרד, כך נפעיל פורייה חד מימד פעמיים על דו מימד.

Decomposition Equation

$$F(u,v)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)e^{\frac{-2\pi i(ux+vy)}{N}}$$

$$e^{\frac{-2\pi i(ux+vy)}{N}}=e^{\frac{-2\pi iux}{N}}\ e^{\frac{-2\pi ivy}{N}}$$

$$\boxed{F(u,v)=\frac{1}{N}\sum_{x=0}^{N-1}\left(e^{\frac{-2\pi iux}{N}}\cdot\sum_{y=0}^{N-1}f(x,y)e^{\frac{-2\pi ivy}{N}}\right)=}$$

$$F(x,v)=\sum_{y=0}^{N-1}f(x,y)e^{\frac{-2\pi ivy}{N}}$$

$$\boxed{F(u,v)=\frac{1}{N}\sum_{x=0}^{N-1}\left(e^{\frac{-2\pi iux}{N}}\cdot F(x,v)\right)}$$

המימוש: נעשה טרנספורם על העמודות, ואח"כ נקח את התוצאה ונעשה טרנספורם על השורות של התוצאה.

מה נעשה עם חלוקה ב N: צריך לבדוק שאנו מחלקים וכופלים בקבוע הנכון.

- **תכונות נוספות - מחזוריות וסימטריה:**

$$F(u,v) = F(u+N,v) = F(u,v+N) = F(u+N,v+N)$$

תכונות של הצמוד F^*:

$$F(u,v) = F^*(-u,-v) \quad ,(a+bi)^* = (a-bi)$$
$$|F(u,v)| = |F(-u,-v)|$$

- **תכונות נוספות - לינאריות:**

$$\Phi\left(f_1(x,y) + f_2(x,y)\right) = \Phi\left(f_1(x,y)\right) + \Phi\left(f_2(x,y)\right)$$
$$\Phi(a \cdot f(x,y)) = a \cdot \Phi(f(x,y))$$
$$\Phi(f(ax,by)) = \frac{1}{|ab|} F(u/a, v/b)$$

בנוסחה האחרונה אם $a > 0$ התדר יגדל ב a.

- **שינוי בתדר ביחס לכפל בקבוע:** עבור פונקציה $f(x)$ -

אם נכפול ל $2x$: התדר יגדל הפונקציה תהיה צרה יותר (במקום x דגימות יש לנו $2x$ ולכן נשיג את אותה כמות בפחות שטח). אבל הטרנספורם פורייה יימרח.

אם נחלק ל $\frac{x}{2}$: הפונקציה תמרח, והטרנספורם יהיה צר יותר עם יותר תדרים נמוכים.

מסקנה: טרנספורם פוריה מגיב **ביחס הפוך** להגדלת או כיווץ התמונה (יחס הפוך בין מרחב הזמן והתדר).

3.3.1 נגזרות של תמונה בעזרת פורייה:

כעת, כשיש לנו ייצוג של פורייה נוכל לגזור את התמונה:

$$f'(x) = \left(\sum_u F(u) e^{\frac{2\pi u x}{N}}\right)' = \sum_u F(u) \left(e^{\frac{2\pi u x}{N}}\right) = \frac{2\pi i}{N} \sum_u u F(u) e^{\frac{2\pi u x}{N}}$$

למעשה: טרנספורם פוריה של הנגזרת של התמונה = טרנספורם המקורי כפול u.

- **מוטיבציה:** נגזרת תגדיר לנו כיצד התמונה משתנה - היכן יש שינויי צבע.

- **מה הנגזרת עושה לתמונה:** מחזקת את התדרים הגבוהים ומחלישה את התדרים הנמוכים, ומוחקת את תדר 0.

- **חישוב נגזרת דו מימדית:**

1: נחשב את הטרנספורם הדו מימדי $F(u,v)$.

2: עבור נגזרת לפי x נכפול את התוצאה ב u (האינדקס של התדר על ציר ה x).

עבור נגזרת לפי y- נכפול את התוצאה ב v (האינדקס של התדר על ציר ה y).

3: נחשב את ההופכית, ונכפול את ההופכית ב $\frac{2\pi i}{N}$.

$$\frac{\partial f(x,y)}{\partial x} = \frac{2\pi i}{N} \cdot \Phi^{-1}(u \cdot \Phi(f(x,y)))$$

- **רעש בתמונה:** הינם פיקסלים מסויימים ששונים מהתדר המקורי, ללא שום קשר לסצינה, לדומה - גרגר אבק שהופיע על המצלמה.
- **הגדלת הרעש:** כשאנו עושים נגזרת לתמונה בנוסף להגברת התדרים הגבוהים, הרעש מתגבר מאד גם כן. משום שרוב הרעש נמצא בתדרים הגבוהים.
- **ממוצע התמונה לאחר הנגזרת:** שווה ל 0, משום שתדר האפס נמחק.

3.3.2 טרנספורם פורייה של פונקציות מוכרות:

- **הפונקציה** $rect$: יכולה לשמש כחלון ריבועי

$$rect(x) = \begin{cases} 1 & \text{if } |x| < \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

והטרנספורם פורייה שלה הוא:

$$F(\omega) = \frac{\sin(\pi\omega)}{\pi\omega} = sinc(\pi\omega)$$

ולהיפך - טרנספורם פורייה של $sinc$ יחזיר לנו $rect$.

- **טרנספורם פורייה של חלון גאוסיאני:** הוא גם גאוסיאן. אם נפעיל אותו על תמונה ונוציא את התדרים הגבוהים נקבל תמונה מטושטשת.
- **טרנספורם פוריה של פונקציה קבועה:** בנקודה 0 - יהיה שווה לסכום כל האיברים בתמונה, ו 0 בשאר הסקאלה.

3.3.3 כיווץ תמונה ו $Aliasing$:

- **תדר גבוה ונמוך בתמונה:**
תדר נמוך - $low\ pass$: אם נוציא מהתמונה את התדרים הגבוהים נשאר עם תמונה מטושטשת.
תדר גבוה - $high\ pass$: אם נישאר עם התדר הגבוה בלבד, נקבל את השפות - המעברים בין שחור ללבן בתמונה, ששם נמצאים התדרים הגבוהים.
- **החשיבות של פאזה:** הפאזה משפיעה על תוכן התמונה יותר מאשר $Magnitude$.
- $Aliasing$: תופעה שקורית כאשר אנו דוגמים גל עם תדר גבוה בנקודות רחוקות. לכן כשנבוא לשחזר נחשוב בטעות כי התדר נמוך יותר.
כיצד נימנע מ $Aliasing$: נדגום כל פעם חצי אורך גל עבור כל גל.
- $Sampling$ **בכיווץ תמונה:** בכל פעם שנרצה להקטין תמונה - לא נדגום כל פיקסל שני. אלא נעשה $low\ pass$ (תדרים נמוכים בלבד) - טשטוש התמונה ואז נדגום. אחרת - נקבל $aliasing$.
- **כיווץ תמונה נכון:** אם נרצה להקטין תמונה מ N נקודות ל $\frac{N}{2}$, נעשה טרנספורם פורייה, נמחק את התדרים הגבוהים באופן הבא - נשחזר ע"י ביצוע טרנספורם הפוך על הטווח $\left[-\frac{N}{4}, \frac{N}{4}\right]$.

3.3.4 הגדלת תמונה בעזרת פורייה:

- **הגדלת תמונה:** נקח תמונה $N \times N$ נעשה לה טרנספורמציית פורייה, ונרפד אותה מסביב באפסים. לאחר מכן נבצע טרנספורם הפוך על התמונה המרופדת.

3.3.5 פילטרים:

כיצד נעשה פילטר לתמונה:

1: נחשב את התמרת פורייה.

2: ונכפיל בפונקציית פילטר, פונקציה שאומרת לנו איזה תדרים לשנות ואיך.

3: נעשה להתמרה החדשה $invers$ ונקבל את התמונה החדשה.

- $Low - pass\ Filter$**:** פילטר שמעביר רק את התדרים הנמוכים וזורק את התדרים הגבוהים.

- $Low - pass\ Filter$**:** פילטר שיוציא את התדרים הנמוכים וישאיר את הגבוהים. אנו נקבל תמונה רק עם הגבולות - המעברים בין שחור ללבן.

- $ideal\ band - pass\ Filter$**:** פילטר שבוחר את כל התדרים מטווח מסויים ומוציא אותם.

- **פילטר גאוסיאני לעומת אידיאלי:** פילטר גאוסיאני לא יכניס לנו לתמונה קווים מיותרים ורינגס שהורסים את התמונה כמו פילטר אידיאלי, מכיוון שהמעבר רך יותר. לכן נעדיף פילטר גאוסיאני.

4 קונבולוציה - $Convolution$:

- **הרעיון:** פעולה לינארית שנבצע על כל הפיקסלים של שני ווקטורים - לדוגמה ממוצע משוקלל. נקח את סביבת הפיקסל ונבצע עליה את הפעולה הלינארית, ונשים את המספר הזה להיות הפיקסל החדש.
אינטואציה: ניתן להסתכל על פעולת הקונבולוציה כאילו אחד מהווקטורים עומד במקום והשני נע מעליו, ואנו מבצעים כפל איברים וסכימה עבור כל המקומות החופפים.

- **טשטוש בעזרת ממוצע משוקלל:** נגדיר פונקציה f שנותנת משקל שווה לכל פיקסל בסביבת הפיקסל הנבחר, ומה שנקבל זאת תמונה מטושטשת כי אנו נמצע את המעברים החדים.

- **טשטוש בעזרת גאוסיאן:** ניתן לטשטש עם קרנל גאוסיאני - קרנל כזה יתן יותר משקל לפיקסל המקורי עליו אנו מסתכלים מאשר לשאר הסביבה שלו.

- **גודל הקרנל וטשטוש:** ככל שהקרנל גדול יותר הטשטוש יהיה חזק יותר.

- **הקלט והפלט:** קלט שני סיגנלים של שני ווקטורים חד מימדיים, נקבל כפלט ווקטור חד מימדי.
גודל הפלט: גודל סיגנל הפלט עשוי להיות גדול יותר מהסיגנל המקורי, משום שזה תלוי בקרנל, בגודל כל אחד מהאלמנטים, בקפיצות של g ועוד.

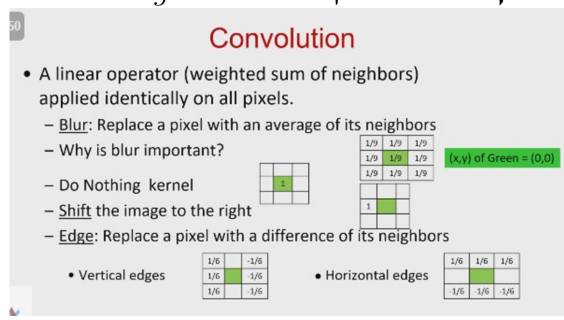
- **דוגמה עבור $2D$:** נחליף כל פיקסל בממוצע של 3×3 השכנים שלו.

$$h(x,y) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} f(i,j) \cdot g(x-i, y-j)$$

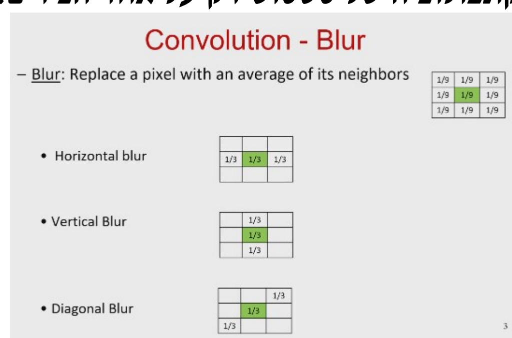
כאשר h זה הפיקסלים החדשים. f זה ה $kernel$ - משקל לכל פיקסל. g זה הפיקסל המקורי.

אינטואציה: נסתכל על המערך הדו מימדי g, ונעביר מעל כל שורה שלו את g לפי הסדר, ונחשב כמו בחד מימדי.

- **סוגי קונבולוציות:** קונבולוציות $edges$ מדמות נגזרת שתגיב לשינויים שקורים בתמונה.



- **קונבולוציה של טשטוש רק על אחד הצירים:**



- **רעש לבן:** רעש שבו כל פיקסל לא תלוי בפיקסל לידו, וממוצע פיקסלי הרעש שווה ל 0.
טרנספורם פורייה שלו - יוביל לערך שווה בכל התדרים. כי רעש לבן מכיל את כל התדרים באותה העוצמה. לעומת תמונה שיש בה יותר תדרים נמוכים מגבוהים.

- **מה יקרה כשנטשטש תמונה שהוספנו לה רעש לבן (חיבורי):** טשטוש יכול לבטל את הרעש כי הוא ממצע את הרעש עם התמונה. טשטוש פוגע בתדרים הגבוהים - לכן הוא יבטל את הרעש שהתדרים שלו גבוהים יותר מתדרי התמונה.

- **קונבולוציה חד מימדית:** עבור שתי פונקציות f, g באותו האורך

$$h = f * g; \quad h(x) = (f*g)(x) = \sum_{a=1}^{n} f(a) g(x-a)$$

- **ציקליות של קונבולוציה (לא עבור תמונות):** אם יש לנו סדרה בת n איברים, אזי אנו נסתכל עליה כסדרה מחזורית.
לדוגמה: עבור סדרה $[0...5]$ מתקיים כי $f(-3) = f(9)$ כי נבצע $mod(6)$ אורך הסדרה.

עבור תמונות:

אפשרות 1: נניח כי כל אינדקס מחוץ לתמונה הקונבולוציה שלו שווה 0.

אפשרות שניה לתמונות (הטובה יותר):

$$h = f * g; \quad h(x) = (f * g)(x) = \sum_{a=1}^{n} f(a)g(x-a)$$

- קונבולוציה של שתי פונקיציות $rect$ (מרובעות) תתן לנו פונקציה בצורת משולש.
- **זמן ריצה של קונבולוציה:** $O(N^2)$

4.0.1 קונבולוציה והתמרת פורייה:

- **קונבולוציה והתמרת פורייה:** התמרת פורייה (Φ) על קונבולוציה של שתי פונקציות f, g = להתמרה על כל אחת מהפונקציות וכפל איבר איבר.($*$ מסמן קונבולוציה, ו Φ, Φ^{-1} מסמנים התמרה והתמרה הפוכה).

$$f * g = \Phi^{-1}(F \cdot G) = \Phi^{-1}(\Phi(f) \cdot \Phi(g))$$

מסקנה: קונבולוציה במרחב התמונות (x, y) שווה לכפל נקודתי בתדר (מרחב פורייה f, g). לכן ניתן לעשות קונבולוציה בעזרת התמרת פורייה.

- **משפט הקונבולוציה:** קונבולוציה בזמן שווה למכפלה בתדר ולהיפך.

1: אם יש לנו קונבולוציה ועליה אנחנו עושים התמרת פורייה זה שקול לעשות פוריה לכל אחת מהן ומכפלת אלמנטים.

$$\Phi(f * g) = F \cdot G$$

2: אם נכפול $f \cdot g$ ונעשה פורייה על המכפלה, זה שקול ללעשות קונבולוציה בין שתי הפורייה.

$$\Phi(f \cdot g) = F * G$$

- **קונבולוציה בעזרת פורייה:** ניתן לעשת התמרת פורייה במקום לעשות קונבולוציה

$$f * g = \Phi^{-1}(F \cdot G)$$

כאשר F, G מסמל פורייה על f, g.

הערה: כדי שנהיה מסוגרים להכפיל מטריצות אנו צריכים לרפד את הקרנל G באפסים כדי שיהיה בגודל של F.

- **פילטרים:** כפל עם פילטר ופורייה שקול ללעשות קונבולוציה עם הפילטר במרחב הזמן.

- **תכונות של קונבולוצה:**

$$f * g = g * f$$
$$f * (g * h) = (f * g) * h$$
$$f * (g + h) = f * g + f * h$$

- קונבולוציה כמטריצה: ניתן להתייחס את פעולת הקונבולוציה בין שני וקטורים כאל כפל מטריצות FG, כך שהמטריצה G מכילה אפסים ומדמה את החפיפה בין שני הווקטורים.

- **לינאריות קונבולציה וכפל מטריצה:** בגלל שקונבולוציה היא פונקציה לינארית ניתן להגדיר אותה בעזרת כפל מטריצה.

- **שיטה נוספת לטיפול ברעש ־ חציון:** ניקח באותו האופן סביבה של כל פיקסל, אך כעת במקום לחשב ממוצע משוקלל נחשב את החציון של הפיקסל.

- **רעש מלח פלפל:** בשונה מרעש חיבורי, שאנו מחברים את הרעש לתמונה המקורית. ברעש מלח פלפל אנו מחליפים פיקסלים בתמונה עצמה לפקסלים לבנים ושחורים. כדי לטפל ברעש זה אנו נשתמש בחציון, משום שטשטוש רגיל בעזרת ממוצע לא יעזור .

4.1 אומד לנגזרת וקונבולוציה:

מכיוון שבתמונות אנו לא יכולים להסתכל על נגזרת באיזור הקרוב, האיזור הקרוב ביותר הוא הפיקסל הקרוב. לכן נסתכל על קונבולוציה עם הפיקסל הקרוב. עם קרנל $[1, -1]$.

An Approximation to Derivatives

$$\frac{\partial}{\partial x} f(i,j) = \lim_{h \to 0} \frac{f(i,j) - f(i-h,j)}{h} \qquad \cong f(i,j) - f(i-1,j)$$

convolution with $(1 \quad -1)$

$$\cong \frac{f(i+1,j) - f(i-1,j)}{2}$$

convolution with $\frac{1}{2}(1 \quad 0 \quad -1)$

- **טיפול בערכי החזרה מהנגזרת:** מכיוון שנגזרת מחזירה לנו ערכים שליליים ובתמונה אין ערכים שליליים, יש שתי דרכים לטפל בבעיה:

  פונקציית $ReLU$: נקח את כל הערכים השליליים באינפוט ונאפס אותם באוטפוט, וכל ערך שגדול מ 0 ישאר אוו הדבר.

  מתיחה: נמפה את הערך השלילי הגבוה ביותר לשחור. את החיובי ־ ללבן ואת 0 לאפור.

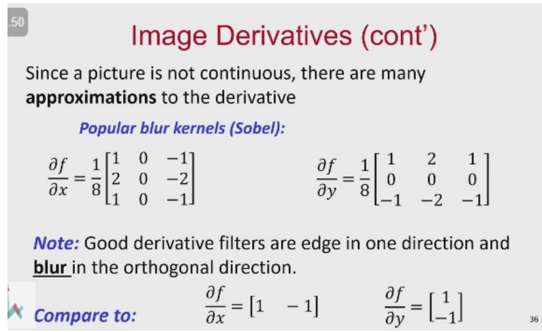
- **נגזרת לפי x או y:** כאשר נגזור לפי ציר ה x נקבל את על המעברים האנכיים. לעומת זאת כשנגזור לפי ציר ה y נקבל את כל המעברים האופקיים.

- **ממוצע הנגזרות בתמונה** שווה ל 0 בד"כ.

4.1.1 קרנל $Sobel$:

- **קרנל** $Sobel$ **אומדן לנגזרת:** נבצע ממוצע באופן הבא עם הקרנלים הגדולים יותר.
הרעיון - נמצע (כדי לנפחית רעשים) בניצב לנגזרת (ולא בכיוון הנגזרת, כדי לא להחליש את ה $edge$) ונחסיר ביניהם. כך נרוויח את הוצאת הרעש מהתמונה וה $edges$ לא ימחקו לנו.

- **הקרנלים:** ניתן לראות כי הקרנל הגדול מורכב מקרנל טשטוש על ציר ה x וגזירה על ציר ה y. ולהיפך.

- 

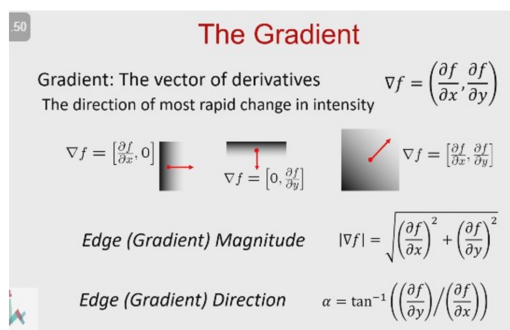
4.1.2 גרדיאנט:

- **גרדיאנט - מציאת השפה בתמונה:** כשנרצה מצוא את ה $edge$ בתמונה והיכן מתחלף משחור ללבן נוכל לחשב את הגרדיאנט (מטריצות הנגזרות החלקיות לפי x ולפי y לכל פיקסל). שזה המקום והזווית שבו מתרחש השינוי הגבוה ביותר התמונה. לאחר מכן נסתכל על הנגזרת לפי x ולפי y ונראה היכן מתרחש השינוי הגדול ביותר.

- **מגניטוד - אם נרצה לדעת היכן השפה עם השינוי הגדול יותר:** נקח את הערך המוחלט - שורש ריבוע הנגזרות. הוא לא יביא לנו את כיוון השינוי אלא את **עוצמת** השינוי בשני הצירים.

- **אם לא מעניין אותנו אם השינוי אופקי או אנכי אלא הכיון:** נחשב את הזוית עם השינוי הגדול ביותר עם tan^{-1}.

- **לסיכום:**

- 

- **המרכז של ה** $edge$: יהיה בשיא של הנגזרת הראשונה או היכן שהנגזרת השניה חוצה את ציר ה x ושווה ל 0 ($zero\ crossing$).

4.2 נגזרת שניה:

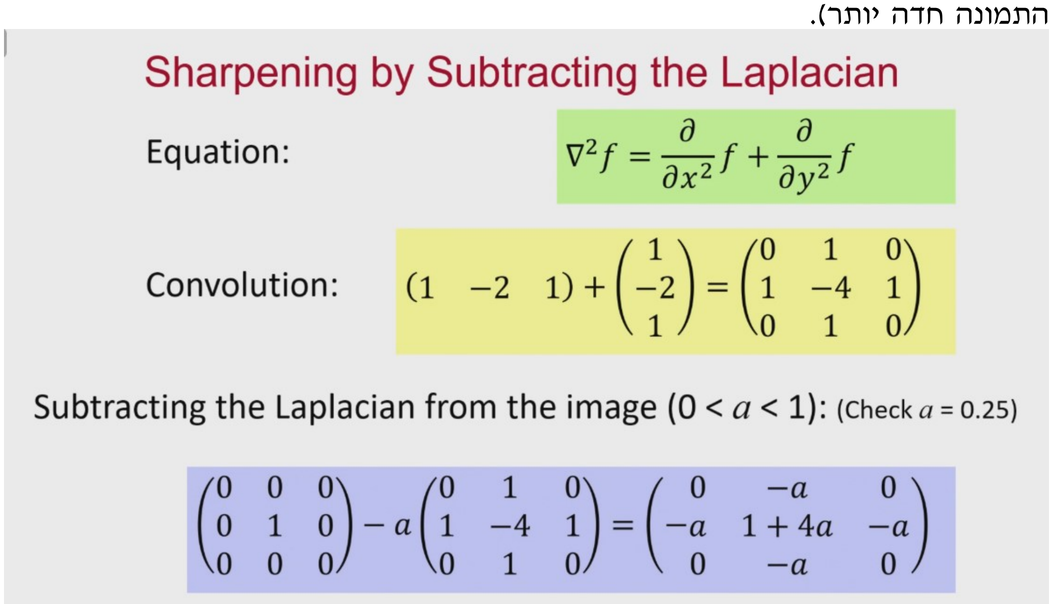
- **נגזרת שניה:** נעשה קונבולוציה עם $[1,-1]$ ונמצא את הנגזרת הראשונה, אח"כ נעשה על התוצאה קונבולוציה עם $[1,-1]$ שוב פעם, וזאת הנגזרת השניה.
  זה שקול לעשות על התמונה קונבולוציה עם $[1,-2,1]$ משום ש $f*(g*h)=(f*g)*h$.
  כאשר ה $kernel$ אופקי זה נגזרת לפי x, ואנכי זאת הנגזרת לפי y.

- **נגזרת שניה ורעש:** כפי שאמרנו נגזרת שניה מושפעת מאד מרעש, לכן כשנרצה לבצע נגזרת שניה - נחליק (נמצע) עם גאוסיאן (מקדמים בינומים) ורק לאחר מכן נבצע נגזרת ראשונה ושניה.

4.2.1 חידוד של תמונה $Sharpening$:

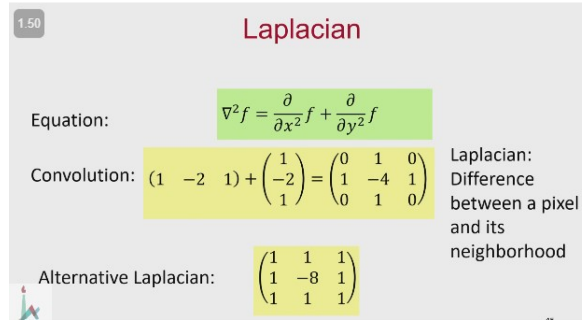
- **חידוד תמונה בעזרת נגזרת שניה:** נוכל לחשב נגזרת שניה, ולהחסיר את הנגזרת השניה מהתמונה. כך אני נפריד בין קצוות ונחדד את התמונה. מה שלפני השפה יהיה נמוך יותר, ומה שאחריה יהיה גבוה יותר.

- **נעשה זאת עם קונבולוציה:** ניתן לעשות את החידוד עם קונבולוציה ובחירת פרמטר חידוד a (ככל ש a גדול יותר התמונה חדה יותר).



4.2.2 לפלסיאן:

- **לפלסיאן:** סכום של הנגזרת השניה לפי x ולפי y. ניתן לעשות קונבולוציה של סכום הנגזרות החלקיות ישירות על התמונה כך-



- **לפלסיאן על תמונה אחידה:** תמונה שכל הפיקסלים שלה שווים ל k ונעשה עליה לפלסיאן, נקבל תמונה ששווה 0, כי אנו מחסירים את הפיקסלים אחד מהשני (בדומה לנגזרת).

4.2.3 $Edge\ Detection$:

- $zero\ crossing$: היא הנקודה בה הנגזרת השניה חוצה את האפס, נקודה זו מסמנת את מרכז ה $edge$.
- **בעיות של** $zero\ crossing$: אם נבצע החלקה (טשטוש כדי להיפטר מהרעש) ואח"כ נגזרת שניה, אנו נקבל עקומות סגורות בתמונת הפלט, משום שהקווים בנגזרת השניה הם קווי גובה.
- **האלגוריתם** $Canny\ Edge$: בא להתמודד עם הבעיות של $zero\ crossing$.

1: נטשטש את התמונה עם גאוסיאן.

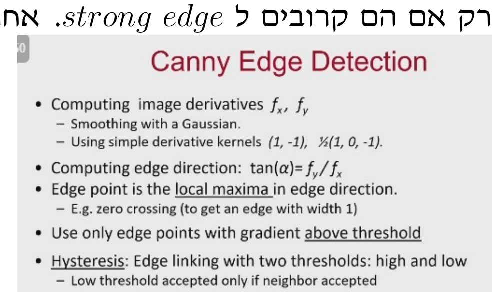
2: נחשב את הנגזרת הראשונה לפי שני הצירים.

3: נחשב את המגניטוד והזווית של הגרדיאנט.

4: נקח את הזוויות שקיבלנו, ונעשה להן קוונטיזציה ל 4 סוגים של זוויות - $0, 45, 90, 135$. כי אלה כיווני השינוי שיעניינו אותנו.

5: $non - maximum\ suppresion$: נגדיר סביבה. לאר מכן נעבור על כל פיקסל בתמונה, נצעד בכיוון הגרדיאנט שלו על כל הפיקסלים שבסביבה ומראים את אותו כיוון גרדיאנט (מבין 4 הזוויות שבחרנו), ונשאיר רק את הפיקסל עם הערך המקסימלי (כדי להתגבר על הטשטוש שעשינו בשלב 1) ואת כל השאר נאפס.

6: היסטרסיס: נקבע שני ספים, אחד גבוה T_1 ואחד נמוך T_2. נעבור על כל פיקסל ונבדוק אם ערך המגניטוד גדול מ T_1, אם כן - נגדיר את הפיקסל להיות 1 $(strong\ edge)$. פיקסל שיהיה נמוך מ T_2 יקבל 0 . כל הפיקסלים שבאמצע הטווח יוגדרו כ $week\ edges$ (כי אנחנו לא בטוחים לגביהם אם הם $edges$ או לא), נשים באותם הפיקסלים ערך 1 רק אם הם קרובים ל $strong\ edge$. אחרת - נשים בה ערך 0.



- CNN **- רשתות קונבולוציה:** נאמן רשת על פלטים וקלטים והקונבולוציות שהובילו לפלט. וכך כשנכניס לה קלט חדש היא תגיד לנו מה הקונבולוציה המתאימה בהתאם לפלט שאנו רוצים לקבל.

4.3 $Cross-Corolation$:

- $Cross-Corolation$: אופרטור שדומה מאד לקונבולוציה, אך אין לו את השיקוף. כי יש פלוסים בנוסחה ואנו עובדים עם הצמוד של הסיגנל f.

  הערה: בתמונות אין שום משמעות לצמוד (כי התמונה ממשית ולא מרוכבת) ולכן ההבדל היחיד הוא השיקוף.

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

- **מתי נשתמש ב** $Cross-Corolation$ - $Template\ Maching$**:** כשנרצה למצוא התאמה בין תמונות. נרצה להבין אם יש דמיון בין שתי תמונות אך אחת היא הזזה של השניה נשתמש ב $Cross-Corolation$.

  כשנרצה למצוא דפוס מסויים בתמונה, נוכל למצוא אותו בעזרת $Cross-Corolation$.

  כיצד נשתמש: נעבור עם הקרנל שלנו על כל התמונה ונראה היכן יש לנו את הערך הגבוה ביותר. שם יהיה מאצ' וזה המקום שאנו מחפשים.

- **משפט פורייה ו** $Cross-Corolation$**:**

$$\Phi(f \otimes g) = F^* \cdot G$$

5 פירמידות - $Image\ Pyramid$:

- **הרעיון:** נתונה לנו תמונה, ואנו נבנה מעליה עוד גרסאות מוקטנות שלה. נקטין בעזרת טשטוש, ונדגום.

- **מספר הרמות** יהיו $log(N)$ כאשר N היא גודל התמונה. כאשר בסיס הלוג הוא פקטור ההקטנה (אם נקטין פי 2 הבסיס של הלוג יהיה 2).

- **מספר הפקסלים בפירמידה:** עבור הקטנה בפקטור 2 אנו נגדיל את מספר הפיקסלים מ N^2 ל $1\frac{1}{3}N^2$.

- **כיצד נבצע הקטנה:** עבור בקטנה פי 2 - נעשה טשטוש עם קרנל גאוסיאני ודגימה של חצי מהפיקסלים.

  הקטנה שרירותית: נקטין באמצעות פוריה - נחשב פוריה, נגזור את התדרים הגבוהים ונשחזר.

- **למה נרצה לייצר פירמידות:**

1: כשנרצה לבצע חיפוש של אובייקט מסויים בתמונה, נוכל לחפש אותו ברמה העליונה של הפירמידה, להבין באיזה חלק של התמונה הוא נמצא, ואז לצמצם את החיפוש שלנו לאיזור הזה בלבד בתמונה המקורית, ע"י מעבר בכל אחת מהרמות. (לדוגמה חיפוש בניין ב $google\ Earth$).

2: כשיש לנו הרבה תמונות (מרכז מצלמות אבטחה) אזי כל אחת ממהתמונות תוצג בצורה מוקטנת, וכשנבחר אותה נוכל להציג אותה בצורתה המקורית.

- **יעילות:** לעיתים נעדיף לפרק את קונבולוציית הטשטוש הדו מימדית לשתי קונבולוציות ־ אופקית ואנכי בנפרד.

- **הגדלת תמונה עם קונבולוציה ־** $Expand$: נשים 0 בין כל שני פיקסלים, ושורת אפסים בין כל שתי שורות, ונטשטש עם הקרנל $[0.5, 1, 0.5]$.
נשים לב כי הקרנל נסכם ל 2 ולא ל 1 כמו בכל פעם, עשינו זאת כדי להעלות את הממוצע לאחר הכנסת האפסים, כדי שלא נקבל תמונה כהה מידיי.

- **קונבולוציה של הקטנה ב 2:** אם נרצה להקטין ב 2 ואחכ ב 4 וכו... נוכל להקטין עם קונבולוציה של מקדמים בינומים. קרנל של הקטנה ב 2 הוא $[1, 2, 1]$.

- **מה נעשה בקצוות:**
  1 ־ שיקוף של הפיקסל האחרון, נעדיף לא להשתמש בשיטה זו כי תמונות אינן ציקליות.
  2 ־ נרפד באפסים ונתייחס לכל פיקסל שמחוץ לתמונה כ 0.
  3 ־ נשכפל אתהפיקסל האחרון עד אינסוף.

5.1 פירמידת גאוסיאן ־ G:

- **פירמידת גאוסיאן:** פירמידה שמטושטשת בעזרת קרנלים שדומים לקרנל גאוסיאני ־ הקטנה ב 2 בכל שלב.
כאשר G_0 זאת התמונה המקורית ו G_1 זאת הקטנה ב 2, וכן הלאה עד G_n.

- **מה יקרה כשנחזור לגודל המקורי:** התמונה תראה מטושטשת ככל שנעלה ברמה, כי אנו נאבד את התדרים הגבוהים.

- **כיצד ייראה מימד התדר ־ פורייה:** בכל שלב אנו נאבד תדרים גבוהים, לכן בכל שלב הגאוסיאן של פורייה ייקטן (יהיה צר יותר).

5.2 פירמידת לפלסיאן ־ L:

- **פירמידת לפלסיאן:** פירמידה שבה בכל שלב אנו מבצעים פירמידת גאוסיאן ומחסירים ממנה את ההגדלה של השלב הבא. כך שכל תמונה תציג את מה שאיבדנו באותה הרמה (את קווי המתאר).

- **נחשב כך:** $L_0 = G_0 - Expand(G_1)$ כאשר G_0 זאת התמונה המקורית ו G_1 זאת הקטנה ב 2. ו $L_n = G_n$ (כי אין מה להחסיר ממנו).

- **נשים לב כי:** $L_n + L_{n-1} = G_{n-1}$ לכן אם נחבר את כל איברי הלפלסיאן נקבל את התמונה המקורית.

- **כיצד ייראה מימד התדר ־ פורייה:** בשלה הראשון (L_0) אנו נעשה $high\ pass$ ונישאר רק עם התדרים הגבוהים. לאחר מכן $(L_1 - L_{n-1})$ בכל שלב אנו נישאר עם $Band\ pass$־ נאבד את התדרים באמצע ונישאר רק עם הגבוהים, וגם הגאוסיאן יהיה צר יותר משלב לשלב. הרמה האחרונה (L_n) תכיל את התדרים הנמוכים ביותר.

- **מיזוג תמונות עם** $Spline$: אם נרצה לשלב שתי תמונות יחד כך שלא יראו את התפר בין שתי התמונות. נרצה מעבר רך בין שתי התמונות ונעשה זאת ע"י הנוסחה הבאה:

$$C(x, y) = h(x)A(x, y) + (1 - h(x))B(x, y)$$

נגדיר משקל $h(x)$ כך שבשלב ההתחלתי הוא יהיה 1 יותר אצל A ו 0 אצל B, ובכל שלב יירד משקל צ A ויעבור ל B. עד שבשלב הסופי המשקל יהיה 0 ל A ו 1 ל B.

5.2.1 שימושים:

1. **דחיסה:** זה אפשרי מכיוון שאנו שומרים את התדרים הגבוהים $(edges)$. נבצע על התמונה לפלסיאן, נעשה קוונטיזציה ל 3-5 רמות, נדחוס ונשחזר את התמונה.

2. **מיזוג של שני חצאי תמונות:** נבנה פירמידת לפלסיאן לכל אחת מהתמונות, ונבנה פירמידת מיזוג, כך שבכל רמה בפירמידה - צד שמאל יהיה של A וצד ימין של B. ובאמצע - אם מספר פיקסלים אי זוגי נקח את הממוצע של שתי התמונות.
כשנרצה לבנות את התמונה נחבר את כל התמונות בפירמידה יחד.

3. **מיזוג תמונות באופן שרירותי:** אם נרצה למזג את התמונות, אך לא חצי של שתי התמונות אלא לדוגמה לשים את הראש של אדם א' על גוף של אדם ב'.
נעשה פירמידות לפלסיאן לכל אחת מהתמונות - L_A, L_B.
נבנה מסיכה בינארית M ונבנה לה פירמידת גאוסיאן G_m.
עבור כל רמה נבנה פירמידה חדשה C עם המשקל של המסיכה M בעזרת הנוסחה הבאה:

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

6 טרנספורמציה על תמונות:

- **מוטיבציה:** אנו רוצים להרכיב תמונת פנורמה ממספר תמונות שצולמו כל אחת בזווית אחרת או ממרחק אחר. הזווית יכולה להיות על כל אחד מהצירים (x,y,z). נרצה להדביק אותן יחד כך שכל אחת תמשיך את האחרת.

- **מה חדש:** עד עכשיו שינינו את הערך של הפיקסל - שינינו צבע. בטרנספורמציות אנו לא ניגע בצבע הפיקסל אלא נמקם אותו במקום חדש בתמונה - נזיז אותו - נשנה קאורדינטות.

- **חוסר קומוטטביות:** הטרנספורמציות אינן קומוטטיביות ויש משמעות לסדר ההפעלה שלהן.

- **טרנספורמציות גלובאליות:** אנו נתעסק בטרנספורמציות על פרמטר - גלובאליות, עבור כל פיקסל נבצע את אותה הטרנספורמציה, שינוי קאורדינטה.

6.1 מטריצות טרנספורמציה וקאורדינטות הומוגניות:

- **מימוש:** ניתן לממש טרנספורציה באמצעות כפל במטריצה, נקח את הפיקסל ונכפול אותו במטריצה מגודל 2×2 או 3×3.

הערה: את רוב הטרנספורמציות ניתן לייצג במטריצת 2×2, אך מכיוון שיש טרנספורמציות שדורדות ייצוג במטריצת

3×3, אנו נייצג את כל הטרנספורמציות בייצוג של 3×3 כדי שנוכל להפעיל אותן אחת אחרי השניה. נעשה זאת ע"י הוספת קאורדינטת $w = 1$ ונרפד את שאר המטריצה ב - 0, למעט האינדקס $(3, 3)$ שנשים בו 1.

- **כיצד נעבור למטריצה של 3×3:** נעבור למערכת קאורדינטות חדשה שנקראת קאורדינטה הומוגנית, נעשה זאת ע"י הוספת קאורדינטה נוספת ששווה ל 1.
  כשנרצה לחזור לקאורדינטות הרגילות - נחלק את x ו y בקאורדינטה האחרונה - w.

- **תכונות של קאורדינטה הומוגנית:** כפל בסקלר לא משנה אותן.

- **כיצד נתייחס לקאורדינטות הומוגניות:**

$$(x, y, w) \Rightarrow \left(\frac{x}{w}, \frac{y}{w}\right)$$

היצוג הבא נקרא ייצוג אינסוף:

$$(x, y, w)$$

הייצוג $(0, 0, 0)$ הוא ייצוג שלא קיים.

6.2 סוגי טרנספורמציות:

6.2.1 טרנספורמציית הזזה - $translation$:

- **הזזה - $translation$:** נזיז את התמונה מספר פקסלים אנכי - ללמעלה \ למטה, ומספר פיקסלים לצדדים - ימינה \ שמאלה.

- **ייצוג בעזרת מטריצה:** את הטרנספורמציה הזו לא נוכל לייצג בעזרת מטריצת 2×2, אלא נצטרך לייצג אותה בעזרת מטריצת 3×3.

- **כיצד נעבור למטריצה של 3×3:** נעבור למערכת קאורדינטות חדשה שנקראת קאורדינטה הומוגנית, נעשה זאת ע"י הוספת קאורדינטה נוספת ששווה ל 1.
  כשנרצה לחזור לקאורדינטות הרגילות - נחלק את x ו y בקאורדינטה האחרונה - w.

- **כיצד תיראה המטריצה:**

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

כאשר t_x ו t_yמייצגים את ההזזה

- **מה יישמר:** זוויות, מרחקים.

6.2.2 טרנספורמציית כיווץ \ מתיחה - $Scaling$:

- **כיווץ או מתיחה -** $scaling$: זום אין או זום אאוט. נרחיב בצירים פי a או נכווץ פי a. ניתן לכווץ \ להרחיב כל ציר לפי פרמטר אחר $(non-uniform\ scaling)$.

- **כיצד נייצג במטריצה:** השינוי שנרצה לבצע - $(x,y) \Rightarrow (s_x \cdot x + s_y \cdot y)$.

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- **מה יקרה שנשתמש במטריצה ההופכית:** נפעיל את הטרנספורמציה ההפוכה - התמונה תחזור לגודל המקורי.

- **מה יישמר:** זוויות.

- **כיצד נמלא את החורים שנוצרו:** נלמד בשיעור הבא

6.2.3 טרנספורמציית סיבוב - $Rotation$:

- **מה נעשה:** נבצע על התמונה רוטציה - סיבוב. נרצה להזיז כל הפיקסל בזווית θ.

- **איך נשנה:**

$$x \Rightarrow x \cdot cos(\theta) - y \cdot sin(\theta)$$

$$y \Rightarrow x \cdot sin(\theta) + y \cdot cos(\theta)$$

- **המטריצה תיראה כך:**

$$\begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- **הערה:** הטרנספורמציה הינה לינארית, אנו מתייחסים לפונקציות sin, cos כאל סקלרים כי הזווית θ קבועה.

- **מה יקרה שנשתמש במטריצה ההופכית:** נפעיל את הטרנספורמציה ההפוכה - התמונה תחזור לזווית המקורית.

- **מה יישמר:** מרחקים יישמרו, זוויות יישתנו.

6.2.4 טרנספורמציית שיקוף - $Mirror$:

- **שיקוף - $Mirror$:** נשקף את התמונה

$$x \Rightarrow -x$$

$$y \Rightarrow -y$$

- **המטריצה תיראה כך:**

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

הערה: ניתן להוריד את ה "-" מאחד הצירים ולשקף רק סביב הציר השני.

6.2.5 טרנספורמציית $Shear$:

- **טרנספורמציית שיר - $shear$:** מתיחת קצוות התמונה הפיכת התמונה למעין מקבילית.
- **המטריצה תיראה כך:**

$$\begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

6.2.6 טרנספורמציית $Prespective$:

- **שינוי פרספקטיבה:** שינוי הפרספקטיבה בה צילמנו את התמונה. למשל פסי רכבת יכולים להיראות כאילו הם מתקרבים אחד לשני אם נצלם אותם מלמטה. מצד שני בצילום מלמעלה הם ייראו מקבילים.

טרנספורמציות של הרכבה:

6.2.7 טרנספורמציית $Rigid$:

- **מה נשלב:** טרנספורמציה המייצגת הזזה ורוטציה.
- **המטריצה תראה כך:**

$$\begin{bmatrix} cos(\theta) & -sin(\theta) & t_x \\ sin(\theta) & cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

- **מה נאבד:** האוריאנטציה תאבד לנו, הכיוון שבו אנו רואים את התמונה.
כל שאר הדברים יישמרו.

6.2.8 טרנספורמציית דמיון - $Similarity$:

- **מה נשלב:** נשלב יחד $rotation, translation, uniform\ scaling$. טרנספורמציה ריגידיט עם $scaling$.
- **מה נאבד:** נאבד אורינטרציה, וגדלים - אורכים בתמונה.
- **מתי נרצה להשתמש:** קורה לנו כשאנחנו מצלמים תמונה על ציר מסויים, אך אנו לא תמיד מצלמים מאותה הזווית.
- **המטריצה תיראה כך:**

$$\begin{bmatrix} s\cdot cos(\theta) & -s\cdot sin(\theta) & t_x \\ s\cdot sin(\theta) & s\cdot cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

6.2.9 אפינית - $Affine$:

- **מה נשלב:** נשלב כל הטרנספורמציות שראינו עד עכשיו. יש 6 פרמטרים.
- **המטריצה תראה כך:**

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

אנו לו יודעים מה כל פרמטר.

- **מה אנו מאבדים:** גדלים, אוריינטציה והזוויות ישתנו. מה שישאר זה קווים מקבילים ישארו מקבילים, ופורפורציות - יחסים בן איזורים ישארו.
- מצלמה לא יכולה לייצר הזזה אפינית. אך היא קירוב טוב להזזה פרוייקטיבית.

6.2.10 פרוייקטיבית - $Projective$:

- **מה נשלב:** שילוב של אפינית והטלה פרויקטיבית, לראשונה אנו נשנה את הקארדינטה השלישית - w, שמסמנת את אפקט העומק.
נשנה את זווית ההסתכלות, התמונה תיראה כמו הסתכלות על פסי רכבת - שינוי העומק של התמונה.
- **דרגות החופש:** טרנספורמציה זו היא עם הכי הרבה דרגות חופש מבין כל הטרנספורמציות.
- **מתי נשתמש:** באה לטפל במקרים שתמונות שונות צולמו מזוויות שונות.

- **המטריצה תראה כך:**

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

- **מה ישתנה:** הקווים המקבילים, הפורפורציות, הגדלים והזוויות ישתנו. הדבר היחיד שישמר זה קווים ישרים.
- **דוגמאות:** צל השמש הוא הטלה פרוייקטיבית. בנוסף כשנצלם שתי מונות לרוב ההטלה ביניהן תהיה פרוייקטיבית.
- כל הטרנספורמציות הן קירובים לטרנספורמציה הפרוייקטיבית.

השינויים בתמונה לפי טרנספורמציות ייראו כך:

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

6.3 תכונות של טרנספורמציה לינארית:

1. המרכז יתמפה למרכז - פיקסל שהיה במרכז, ישאר במרכז.
2. הקאורדינטה $(0,0)$ תישאר $(0,0)$.
3. קווים ישרים יישארו ישרים.
4. קווים מקבילים יישארו מקבילים.
5. היחסים בין אזורים שונים בתמונה נשמרים.
6. סגורות תחת הרכבה - ניתן להפעיל אותן אחת על \ אחרי השניה.
   חשוב לזכור כי הסדר כן משנה.
7. אם נפעיל את המטריצה ההופכית נחזור לתמונה המקורית.
8. כל הטרנספורמציות הפיכות.

6.4 כיצד נמצא את הטרנספורמציה המתאימה:

- **התאמת נקודות:** נמצא בכל תמונה 4 נקודות (כי יש לנו משוואה עם 8 נעלמים לכל היותר - טרנספורמציה פרודקטיבית), ונתאים אותן בין שתי התמונות. לאחר מכן נקח את הנקודות הישנות והחדשות ונמצא את הטרנספורמציה המתאימה ע"י פתרון משוואה.

- **בחירת הנקודות:** נעדיף תמיד לקחת הרבה יותר מ 4 פיקסלים בכל תמונה כדי שההתאמה תהיה מדוייקת יותר.

6.5 התאמה בין התמונות - $Warping$:

- נרצה לקחת את התמונה f ואת הטרנספורמציה T שמצאנו, וליצור תמונה חדשה g שהיא תוצאת הפעלת T על f.

6.5.1 דרך ראשונה - $Forward\ warping$:

- **הרעיון:** נעבור על כל פיקסל בתמונה f, ונשאל לאן הוא צריך לעבור בתמונה החדשה g. ונשבץ אותם במקומות המתאימים.

בעיות עם האלגוריתם:

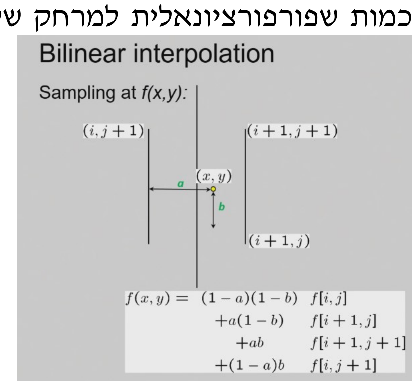
- **מה נעשה כשפיקסל עובר למיקום לא שלם:** (לדוגמה בטרנספורמציית סיבוב) נפזר את צבע הפיקסל הזה על הפיקסלים שסמוכים אליו.
  בסוף תהליך ה $warping$ ננרמל את התמונה כדי לוודא שלא חרגנו מהערכים ושינינו את הממוצע שלה יותר מידיי.

- **מה קורה כשנגדיל תמונה פי 2:** יהיו לנו חורים כי כל פיקסל יועתק למקום חדש ושאר המקומות יהיו ללא פיקסלים.

6.5.2 דרך שניה - $Backward\ warping$:

- **הרעיון:** נתחיל מהכיוון ההפוך, נלך לתמונה g ונשאל כל פיקסל שם מאיפה הוא הגיע. למעשה נעשה טרנספורמציה הפוכה (כי הן הפיכות) על g.

- **מה יקרה כשניפול בין שני פיקסלים:** (לדוגמה אם הטרנספורמציה הייתה הגדלה), נעשה אינטרפולציה (בד"כ בי לינארית) בין הפיקסלים הסמוכים - נקח שני שכנים מהצדדים ושני שכנים מלמעלה ולמטה, ונקח מכל אחד מהם כמות שפורפורציונאלית למרחק שלהם, כל שהם קרובים לפיקסל נקח מהם יותר.



- אם הטרנספורמציה הפיכה תמיד נעדיף לעבוד בשיטת $backward$, אם אינן הפיכות נעשה $forward$ מחוסר ברירה. בקורס כל הטרנספורמציות הפיכות לכן נעשה $backward$.

6.6 שיטות מבוססות פיצ'רים למציאת נקודות עניין בתמונות $Feature\ Points$:

- **מוטיבציה:** אנחנו מוצאים את ההזזה לפי פיצ'רים (מציאת נקודות), ולא כמו שלמדנו עד עכשיו - שיטות $pixel\ direct$ (קרוס קורלציה, בדיקה מתי הרוטציה מתלכדת). משום ש:
בתמונות עם הזזות מורכבות זה נהיה קשה לזהות בשיטות אלו.
בנוסף אם אין חפיפה בין התמונות, אנו נתקשה להתאים ביניהן.

- **כיצד ניצור תמונת פנורמה:**

1: נעבור על כל תמונה ונמצא בהן נקודות מעניינות - נקודות יחודיות שיהיה קל למצוא אותן בתמונה השניה (לא בהכרח נמצא אותן).

2: נתאר כל נקודה בצורה טובה, כדי שיהיה לנו קל להתאים אותן בין שתי התמונות.

3: נמצא את הזוגות המתאימים של הנקודות בשתי התמונות.

4: נשתמש בזוגות הנקודות כדי לעשות $align$ בין התמונות, ולזהות את הטרנספורמציה.

- **בעיות:** כיצד נמצא את אותן הנקודות בין שתי התמונות, ונוודא כי הן אכן אותן נקודות?

1: ניצור אלגוריתם רפטטיבי שמתנהג אותו הדבר על שתי התמונות, כך שהוא ימצא את אותן נקודות מעניינות בכל תמונה.

2: נרצה למצוא עבור כל נקודה נקודה אחת בדיוק שתהיה הזוג שלה, לכן נרצה לכל נקודה מזהה ייחודי, כדי שנוכל לוודא אותה ב 100%.

6.6.1 אלגוריתם $Harris\ corner$ למציאת פינות בתמונה:

- **הרעיון:** אנו נרצה למצוא חלון שבבירור לא נמצא כמה פעמים בתמונה, כך שהוא יהיה ייחודי. לכן נמצא פינות בתמונה - מפגש של שתי $edges$. נקח נקודות כאלה כדי להתאים אותן בין שתי התמונות.

- **האלגוריתם ייפעל כך:**

עבור כל פיקסל אנו נבדוק מהו השינוי u,v שלו עבור ככל שינוי אפשרי. נבדוק את השינוי בעזרת הנוסחה הבאה שתייצג את פונקציית השגיאה:

$$E\left(u,v\right)=\sum_{y}w(x,y)\left(I(x+u,y+v)-I(x,y)\right)^{2}$$

למעשה נקח את ריבועי המרחקים בין הפקסלים של החלון. כאשר I זאת התמונה המקורית, ו w זאת פונקציית חלון. כשנקבל שינויים קטנים - אנו באיזור חלק. אחרת - אנחנו בפינה.

- **האלגוריתם לא יעיל ונרצה לייעל אותו.**

נשים לב כי נוכל להסתכל על טור טיילור כדי לקרב את השינויים $x+u,y+v$. עבור u,v שינויים קטנים מאד. לכן

נוכל להסתכל על הביטוי הבא:

נגדיר מטריצה

$$M=\sum_{(x,y)\in w}\left[\begin{array}{c}\sum_{x,y}I_x\cdot I_t\\ \sum_{x,y}I_y\cdot I_t\end{array}\right]$$

ונעבוד עם הביטוי הבא:

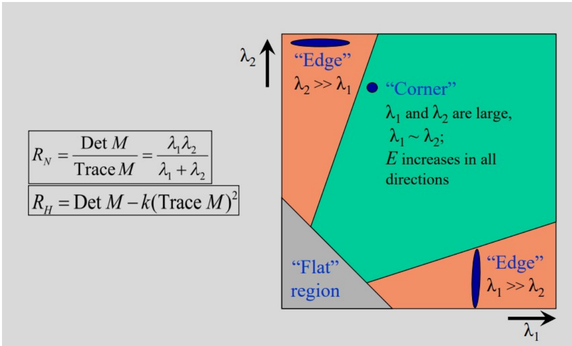
$$E(u,v)=(u,v)\,M\left(\begin{array}{c}u\\ v\end{array}\right)$$

- **נשים לב** כי מעניין אותנו ההזזה ספציפית ולא כל ההזזות, כי אם אנו עומדים על פינה - אז גם הזזה קטנה תגרום לשינוי גדול. לכן אנו יכולים להסתכל על טווח שבין u,v ממזערים וממקסמים.

  הביטויים שימזערו הם ע"ע של M הקטן ביותר, וה-ע"ע הגדול ביותר ימקסם את השינוי.

  נוכל למצוא את השינוים באופן הבא:

  נמצא את היחס בין הערכים העצמיים עבור כל פיקסל ($corner\ response$), ונגדיר $threshold$



- **האלגוריתם:**

  אחרי שנמצא את הע"ע והיחס, נבדוק איזה נקודות עברו את ה $threshold$, ונבחר אותן. ומתוכן עבור כל קובץ נקודות נקח את הקודה שנמצאת במקסימום מקומי.

  את הנקודות האלו נשווה בין שתי התמונות.

- **איזה הבדלים בין התמונות נזהה:** אנחנו נזהה שינויים של סיבוב, שינויים של תאורה (הוספת סקלר) כי אנו עובדים עם נגזרת.

  אך אנו נפספס שינויי סקאלה בין התמונות - אם נעשה זום-אין, אנו נפספס חלק מהפינות.

- **כיצד נוכל להתגבר על שינוי סקאלה?**

  נבבנה פירמידת G לתמונה ונחשב עבור כל תמונה את האלגוריתם של $Harris$.

  יכול להיות שנמצא את אותה נקודה בכמה רמות שונות של הפירמידה, כדי להתגבר על זה אנו נשכלל את האלגוריתם ונבדוק עבור כל נקודה האם היא מקסימום עבור כל רמה בפירמידה, ונשמור רק את הנקודה בסקאלה המקסימלית.

6.7 מציאת תיאור לנקודות התמונה ־ $Descreption$:

- אנו לא יכולים לתת תיאור לפיקסל לפי הצבע שלו או מיקום. בנוסף אנו רוצים להיות אינווריאנטים לשינויי סקאלה, תאורה, סיבוב רעש, טעויות קטנות של הזזה בפיקסל. לכן נרצה לתאר את **סביבת** הפיקסל, ושהתיאור לא יהיה ספציפי מידיי, אך גם לא כללי מידיי כדי שנוכל להתאים נכון.

- **רעיון ראשון:** נקח את הפיקסל שמייצג את הפינה, ואת הפיקסלים שסביבו, ונשווה מרחק אוקלידי או קרוס קורלציה. חסרונות: שינוי תאורה, רעש סיבובים יכולים להפריע לנו להתאמה.

- **רעיון שני ־ היסטוגרמות:** נחשב היסטוגרמה של הצבע או של הגרדיאנט לסביבה, ונוכל לחשב את ההיסטוגרמה של הסביבה.

  חסרון: גרדיאנט הוא תלוי כיוון, וצבע הוא תלוי תאורה.

6.7.1 אלגוריתם $MOPS - Multy\ scale\ oriented\ patches$:

- **אלגוריתם $MOPS$:** נחלק לשני חלקים. חלק ראשון נמצא את נקודות העניין, בעזת $Harris$, לאחר מכן נקח $patches$ של התמונה מתוך תמונה מטושטשת, כך נהיה אינווריאנטים לסיבוב וסקייל. שלב שני ־ ננרמל את ה $patch$ כדי להיות אינווריאנטים לשינויי תאורה.

- **שלב ראשון:** לאחר שנקבל את ההנקודות מהאלגוריתם של $multy\ scael\ Harris$, נוסיף לכל פינה את הכיוון שלה ־ אוריינטציה (כדי שנוכל לתקן ולהיות אינווריאנטים לסיבוב) בעזרת פרמטר θ.

- **מציאת הכיוון ־ אוריינטציה:** נוכל להשתמש בגרדיאנט. אך נקח טשטוש של הזווית של הגרדיאנט, כדי להיות אינווריאנטים שונות של רעש או הזזות במספר קטן של פיקסלים.

- **חיתוך $patch$:** לאחר שנמצא את הזווית, נחתוך חלון בגודל 40×40 אך ברזולוציה נמוכה יותר ־ בדגימות של כל 5 פיקסלים לאחר טשטוש, סביב הנקודה בכיוון האוריינטציה. כך אנו נהיה אינווריאנטים לסיבוב.

  כיצד נבצע: נעלה שתי רמות בפירמידה מעל החלון הנוכחי, ששם כבר התמונה מוקטנת ומטושטשת ונקח חלון בגודל 8×8.

- **נירמול ־ אינווריאנטיות לשינויי תאורה:** לאחר שנחתוך את החלון בכיוון הגרדיאנט, אנו ננרמל את ה $patch$ שלנו ע"י חיסור הממוצע וחלוקה בשונות.

- **איך נשווה בין שני $patches$:** נקח את הפאצ' בגודל 8×8 נפעיל עליו טרנספורמציה שתעביר אותו לווקטור של 64 ועליו נבדוק מרחק אוקלידי משאר הווקטורים.

  דרישות נוספות: נסתכל על היחס בין ההתאמות, נרצה לקחת את ההתאמה שהכי טובה מבין שאר ההתאמות. מכיוון שאם כל ההתאמות טובות באותה המידה, משמע שאף אחת מהן לא מתאימה. בנוסף נבחר גבול כך שרק דגימה שעברה את גבול ההתאמה תיחשב מתאימה.

6.7.2 אלגוריתם $SIFT - scale\ invariant\ feature\ transform$:

- **אלגוריתם** $SIFT$: האלגוריתם מוצא בעצמו את נקודות העניין, הוא לא משתמש ב $Harris$. הוא מתאים אוריינטציה בצורה שונה. נקח חלון סביב הנקודה, אך לא ניקח פיקסלים אלא נקח חלון של גרדיאנטים סביב הנקודה.

- **מציאת נקודות העניין:** הוא ממשתמש בפירמידת $DOG - difference\ of\ Gaussian$, פירמידה שבה אנו מטשטשים את התמונה ומחסרים, ללא הקטנת נתמונה משלב לשלב.
נטשטש עם הפילטר הבא:

$$G(x, y, k\sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-\left(x^2+y^2\right)}{2(k\sigma)^2}}$$

ובכל שלב של הפירמידה נעלה את הפרמטר k שמעלה את רמת הטשטוש. סה"כ נטשטש 5 פעמים (מספר רמות טשטוש נקראות - אוקטבה) ונקבל פירמידה עם 4 רמות הפרשים. לאחר מכן נקטין את כל הרמות ונטשטש מחדש.
לאחר מכן: נחפש בכל שלב בפירמידה נקודות קיצון. אך נשים לב שנקח כל נקודה רק מרמה אחת בפירמידה, הרמה בה הנקודה היא הכי קיצונית.

- **מציאת זווית האוריינטציה:** נקח חלון סביב הנקודה, ונבנה ספקטוגרמה בה יש לנו 36 רמות, כך שכל רמה מייצגת תזוזה ב 10 מעלות. נמשקל את הפיקסלים לפי המגניטוד שלהם - פיקסלים עם שינוי גדול יותר יתרמו משקל גדול יותר, בנוסף נמשקל פיקסלים גבוהים במשקל גבוה יותר.אח"כ נקח את הזווית של הפיקסל להיות הזווית הדומיננטית (הגבוהה ביותר) בספקטוגרמה.

- **חישוב הפיצ'ר ווקטור:** עבור כל נקודה נקח חלון של גרדיאנטים בגודל 16×16 כך שניתן משקל גאוסיאני לפיקסלים (רחוקים יקבלו משקל קטן יותר).
שימוש באוריינטציה: עבור כל נקודה בחלון, נחסר ממנה את הזווית שמצאנו בחלק הקודם.
את כל הנגזרות האלה נחלק ל 4×4 קבוצות, ועבור כל קבוצה נבנה היסטוגרמה של 8 כיווני גרדיאנט. נשטח את כל ההיסטוגרמות של הקבוצות להיות ווקטור באורך 128, ווקטור זה ייצג את האיזור.

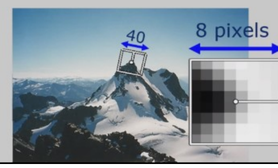
- **אינווריאנטיות:** סקאלה - אנחנו עובדים תמיד על הסקאלה הנכונה. תאורה - אנחנו עובדים עם נגזרות. סיבוב - החזרנו את הזווית תטא.

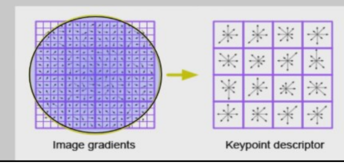
- **נרמול הווקטור:** נקח את הפיצ'ר ווקטור, וננרמל אותו להיות ווקטור יחידה. לאחר מכן נעשה לו $cliping$ לכל הערכים שמעל 0.2, כדי להתמודד עם שינויי תאורה קיצוניים. לאחר מכן ננרמל שוב את הווקטור לווקטור יחידה.

- **מציאת נקודות מתאימות:** באותו האופן כמו אלגוריתם $MOPS$.

השוואה בין $MOPS$ ל $SIFT$:

	MOPS	SIFT
Feature Point Detection	Scale-invariant Harris (x, y, s)	Difference of Gaussians local extrema (x, y, s)
Orientation	Blurred gradient direction (x, y, s, θ)	Histogram of weighted gradient directions (x, y, s, θ)
Patch + Scale invariance	8×8 pixels from level $s + 2$ in pyramid ($\cong 40 \times 40$ on level s)	16×16 gradient from scale s, separated to 4×4 histograms of 8 orientations
Rotation invariance	Patch taken at orientation θ	Orientation θ subtracted from gradients direction
Intensity invariance	$patch = \frac{patch - mean}{STD}$	Normalize to unit vector + clip at 0.2
Distance metric	$\frac{1-NN}{2-NN}$ (Euclidian)	$\frac{1-NN}{2-NN}$ (Euclidian)





6.8 חיבור הנקודות - $Matcing$:

- **הנחות:**
  1: יש לנו תיאורים לנקודות עבור כל אחת מהתמונות.
  2: עבור כל פיצ'ר בכל תמונה נשאל מי k השכנים הכי קרובים.
  3: נבחר 2 פאצ'ים להיות זוג, אם הם עם התאמה הכי גבוהה בשתי התמונות (כדי להימנע מהתאמות שגויותשבהן כל הנקודות קרובות באותה המידה).

- **התמודדות עם התאמות שגויות:** למרות שעשינו הכל כדי להתאים על הצד הטוב ביותר, עדיין יהיו לנו טעויות בהתאמות ($outliers$ - רעש, הסתרות, לא חד משמעויות, טעויות בשלבים קודמים). לכן נרצה שיטה רובאסטית שתדע להתמודד עם $outliers$.

6.8.1 אלגוריתם $RANSAC - Random\ sample\ consensus$:

- **האלגוריתם:** אלגוריתם איטרטיבי שרץ N איטרציות, שמחפש את המודל שיש קונצנזיוס לגביו. אנו מניחים כי רוב הדאטה הוא נכון, והדאטה הלא נכון לא מסכים על שום דבר. לכן המודל הנכון יהיה זה שהרוב מסכים עליו.

- **צעד ראשון:** נדגום באופן רנדומלי בכל שלב את מינימום הנקודות שאנו צריכים כדי להתאים את המודל (עבור תמונות 4 נקודות).

- **צעד שני:** נמצא מודל שמספק את מינימום הנקודות.

- **צעד שלישי:** נעבור על שאר הנקודות ונבדוק כמה מהן מסכימות עם המודל שמצאנו. עבור סף δ אם המרחק של שאר הנקודות קטן מ δ אזי הן מסכימות.

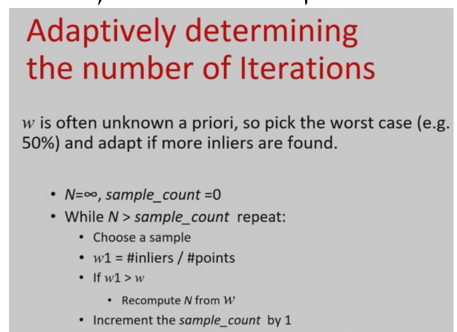
- **פתרון:** נחזיר את הפתרון שהכי הרבה נקודות הסכימו עליו.

- **כמה איטרציות נעשה:** עבור מספר נקודות מינימלי שווה ל S נקודות, כך שאחוז ה $inliers$ שלהן הוא w, ואנו רוצים דיוק של p אחוז הצלחה. נעשה N איטרציות, כדי שבוודאות נעשה איטרציה אחת טובה שמבוססת כולה על $inliers$.

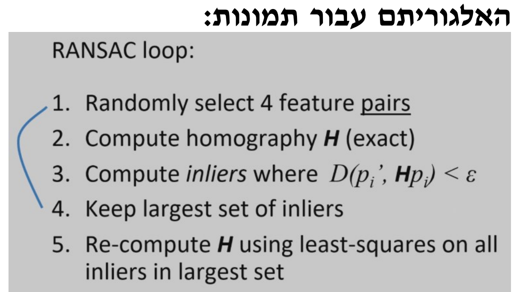
כיצד נמצא את N:

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

- **איך נריץ את האלגוריתם:** נתחיל מ $w = 50\%$, ונריץ את האלגוריתם. אם מצאנו קונצנזיוס שגדול מ w - נעדכן את w להיות הקונצנזיוס שמצאנו, נוריד את מספר האיטרציות ונחשב שוב את האלגוריתם.



- **האלגוריתם עבור תמונות:**



6.9 $Rolling\ Shutter$:

- **מה זה:** המצלמות הישנות צילמו בשיטת CCD, בה כל הפריים מצולם בפעם אחת. כיום מצלמות הפלרפון מצלמות ע"י קריאת שורות, קוראות את כל השורות מלמעלה ללמטה מספר פעמים. לדוגמה - סרטון של פריים 24 תמונות לשניה יקרא 24 מטריצות מלמעלה ללמטה בשניה.

- **חיסרון:** אם נצלם פלופלור מסתובב אנו נאבד את התמונה המקורית והתמונה לא תשקף את המציאות.

- **ייתרונות:**

1: נוכל להסיק מתוך התמונה מה מהירות הסיבוב של הפלופלור.

2: אם נצלם תוך כדי נסיעה נוכל להבין לפי הזוויות של העצמים הקרובים והרחוקים מה המרחק ביניהם.
3: אם נצלם מיתר, נוכל לדעת איך הוא זז ונוכל לשחזר את המיזיקה שנוגנה.

- אם נרצה לקבל תמונה ללא טשטוש נצטרך זמן חשיפה קטן מאד בכל תמונה.

6.10 שימוש בטרנספורמציה על תמונות:

אנו נניח כי אין *Rolling shutter* וכי כל התמונה מצולמת במכה אחת.

כיצד נשלב שתי תמונות:

6.10.1 שיטה ראשונה - שילוב בעזרת נקודה אחת:

נקח שתי נקודות דומות - אחת בכל תמונה, ונבדוק כמה הן זזו אחת מהשניה.
הבעיה היא שזה ימצא לנו רק הזזה של התמונות ולא נוכל לקלוט כל שינוי. לכן נעדיף לקחת יותר נקודות ולמצע אותן.

6.10.2 שיטה שניה - חישוב:

- **השיטה נקראת** *Direct Method*: נחשב את סכום ריבועי ההפרשים של שתי הנקודות (אחת מכל תמונה)

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

כך נמצא את ההזזה שתביא לנו את ההפרש הכי קטן.
אנו נחשב **ממוצע** על האיזור החופף עבור כל פיקסל, כדי לפתור בעיות כשאין איזור חפיפה. בנוסף נבחר איזור חפיפה גדול.

- **סיבוכיות:** מספר הפיקסלים בתמונה. אם נבדוק עבור כל הזזה - הסיבוכיות תגדל.

- **נוסחה שקולה** - נמקסם את ה *cross correlation*:

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

- **נוסחת** *cross correlation* **מנורמלת:** נחסר את ממוצע התמונה מכל אחת מהתמונות (כדי להזניח חיבור של קבוע), ונחלק בסטיית התקן (כדי להזניח כפל בקבוע).

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

הקורלציה המנורמלת עוזרת לנו למשל להתאים בין אובייקט מסויים שצולם בשמש ובצל.

- **מתי נשתמש בקורולציה מנורמלת:** כשנרצה לעקוב אחר גוף מסויים בפריימים שונים. או כדי למצוא אובייקט מסויים בתמונה.

- **כיצד נבצע את החיפוש:** נחפש בעזרת פירמידה, כך נוכל לחפש בתמונה המוקטנת ולמצוא את ההזזה בצורה יעילה יותר.

- **בעיות של חיפוש בעזרת קורלציה:**

1: הדיוק הוא של פיקסלים שלמים. לא נוכל לזהות הזזה של מאית פיקסל.

2: הסיבוכיות היא אקספוננציאלית במספר הפיקסלים - N^2, ואם התמונה מורכבת יותר לדוגמה עם רוטציה - הסיבוכיות עולה ל N^3.

6.10.3 שיטת $Lucas\ Kanade\,(LK)$:

- **הרעיון:** נסתכל על טור טיילור מסדר ראשון, כלומר במקום להסתכל בנוסחה על $I_2(x+u,y+v)$ נסתכל על טור טיילור של הפונקציה כך:

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

למעשה אנו רוצים למזער את הטעות.

כאשר I_x,I_y הן הנגזרות לפי x,y, ו I_t היא הנגזרת בזמן - הפרש הפיקסלים $I_t(x,y)=I_2(x,y)-I_1(x+u,y+v)$.

- **איך נמזער את הטעות:** נגזור כל פיקסל לפי x ולפי y ונשווה ל 0.

- **הערה:** אם התמונה הייתה הזזה - u,v שווים לכל התמונה, אחרת (בסיבוב למשל) לכל פיקסל יש u,v אחרים.

- **לאחר חישוב הנגזרות נכפול בנעלמים $[u,v]$ ונמצא את ההזזה:**

$$\begin{bmatrix}\sum_{x,y}I_x\cdot I_x & \sum_{x,y}I_x\cdot I_y\\ \sum_{x,y}I_y\cdot I_x & \sum_{x,y}I_y\cdot I_y\end{bmatrix}\cdot\begin{bmatrix}u\\ v\end{bmatrix}=-\begin{bmatrix}\sum_{x,y}I_x\cdot I_t\\ \sum_{x,y}I_y\cdot I_t\end{bmatrix}$$

את I_x,I_y נחשב על אחת מהתמונות. ואת I_t נחשב באמצעות שתי התמנות $I_t(x,y)=I_2(x,y)-I_1(x+u,y+v)$.

- **למשוואות יהיה פתרון כש:** רק כשיש להן ע"ע שגדולים מ 0.

- **מתי נקבל ע"ע שווים ל 0:** יהיה למטריצה שני ע"ע ששווה ל 0 אם התמונה כולה חלקה כי אז הנגזרות שוות ל 0.
אם ה x שווה ל 0 (כל שורה אחידה) הנגזרת ב x תקבל ע"ע ששווה ל 0.
אם ה y שווה ל 0 (כל עמודה אחידה) הנגזרת ב y תקבל ע"ע ששווה ל 0.

- **הע"ע יהיו גבוהים:** בפינה או בנקודה בודדת.

- **הערה:** נוכל למצוא את ההזזה רק אם היא **קטנה.**

- **האלגוריתם:** שיטת גרדיאנט דיסנט.

1: נחשב את המטריצה הבאה על אחת מהתמונות

$$A = \begin{bmatrix} \sum_{x,y} I_x \cdot I_x & \sum_{x,y} I_x \cdot I_y \\ \sum_{x,y} I_y \cdot I_x & \sum_{x,y} I_y \cdot I_y \end{bmatrix}$$

2: נחשב את המטריצה הבאה בעזרת שתי התמונות כך $I_t(x,y) = I_2(x,y) - I_1(x+u, y+v)$ ננחש בהתחלה ש $u = v = 0$:

$$b = \begin{bmatrix} \sum_{x,y} I_x \cdot I_t \\ \sum_{x,y} I_y \cdot I_t \end{bmatrix}$$

3: נחשב $d = \frac{I_t}{I_x}$, כאשר d מסמן את ההפרש

$$A \cdot \begin{bmatrix} du \\ dv \end{bmatrix} = -b$$

4: נעדכן:

$$u+ = du, \quad v+ = dv$$

5: נזיז את התמונה הראשונה לקראת התמונה השניה ב u, v.

הערה: אם מצאנו הזזה של 0.5 פיקסל, ואח"כ הזזה של 0.25 פיקסל, נזיז בפעם אחת ב 0.45 כי כל הזזה מטשטשת את התמונה.

6: נחזור בצורה איטרטיבית, עד שניתכנס.

- **הערה:** כשיש הזזה גדולה אנחנו לא נתכנס, לכן נעשה פרמידה ואח"כ נפעיל את האלגוריתם.

כי הפירמידה תטשטש את התמונות, וכך הנגזרות שלהן יהיו שוות גם אם ההזזה גדולה. נתחיל מהרמה הקטנה ביותר ונעלה.

- **רמת הדיוק:** גבוהה מאד, גם ברמת מאית הפיקסל.

- **סיבוכיות:** מהירות החיפוש היא גבוה ביותר, גם בתמונות גדולות.

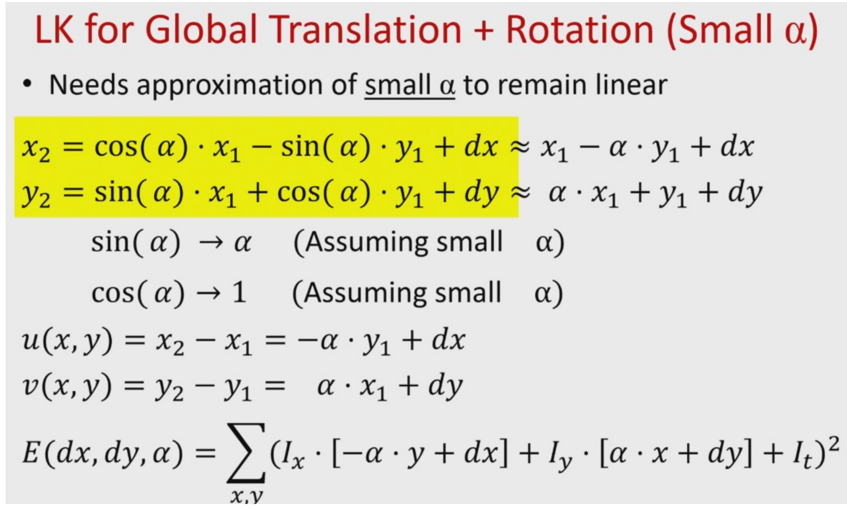
- **הנוסחאות של** $Lucas\ Kanade$ **להזזה וזום:**

$$x_2 = s \cdot x_1 + dx \quad \Rightarrow \quad u(x,y) = x_2 - x_1 = (s-1) \cdot x + dx$$
$$y_2 = s \cdot y_1 + dy \quad \Rightarrow \quad v(x,y) = y_2 - y_1 = (s-1) \cdot y + dy$$

פונקציית השגיאה:

$$E(dx, dy, s) = \sum_{x,y} \left(I_x \cdot [(s-1) \cdot x + dx] + I_y \cdot [(s-1) \cdot y + dy] + I_t\right)^2$$

- **הנוסחאות של** $Lucas\ Kanade$ **להזזה וסיבוב:**



אחרי שנפתור את השוואות נחזור להסתכל על sin ו cos כרגיל.

הערה: פירמידה לא עוזרת לנו בסיבוב.

- **אפקט** $Motion\ parallax$: כשאנו מצלמים תמונה תוך כדי תנועה, עצמים שקרובים יותר למצלמה ינועו מהר יותר.

לוקאס קנאדה על תמונה כזו: יזיז את התמונה, במספר שבו רוב הפיקסלים בתמונה זזים.

6.10.4 $Optical\ Flow$:

- שיטה זו באה לפתור הזזה של תמונות שזזות באפקט $Motion\ parallax$, ולטפל בהן באופן טוב יותר מאשר LK.

- **הרעיון:** נחשב את התנועה של כל פיקסל בנפרד.

- **הנחות:** נניח כי הצבעים לא משתנים בין התמונות. בנוסף נניח כי ההזזה קטנה.

- **בעיית אפקט** $Apertur$: מתרחשת כשאנחנו בודקים תנועה בחלונות קטנים של התמונה. אנו נראה את הכיוון הכללי של החלק, אך לא נוכל להיות בטוחים שזו אכן התנועה המדוייקת. למשל תנועה של אובייקט שנע שמאלה ולמעלה, אנו נראה כי הוא זז שמאלה בלבד.

לכן: נצטרך לזכור זאת כשנעשה $Optical\ Flow$.

חישוב בעזרת קורלציה:

- **חישוב בעזרת קורלציה:** נקח חלון סביב הפיקסל, ונמצא את הקורלציה הגבוה ביותר בתמונה הבאה. זה יהיה ה $Optical\ Flow$ של הפיקסל הזה.

יעילות: נגביל את החיפוש למיקום שבו אנו חושבים שהפיקסל נמצא.

גודל החלון: נרצה חלון גדול כדי שתהיה לנו מספיק אינפורמציה, מצד שני נרצה שהוא יהיה קטן כי שלא נכניס לחלון מספר אובייקטים שזזים באופן שונה.

שימוש בפירמידות: גם כאן נשתמש בפירמידות כדי לא לחפש בכל התמונה. נתחיל מהרמה הקטנה ביותר, ונמשיך בעזרת בתוצאה שקיבלנו לרמה שמעליה.

התוצאה הטובה ביותר: תתקבל באיזור עם נגזרות גבוהות, שם נוכל למצוא את הקורלציה המדוייקת.

חישוב בעזרת LK:

- **חישוב בעזרת LK:** נוכל לקחת חלונות בתמונה ולחשב על כל חלון LK. כאשר הסכום σ הוא רק על החלון הספציפי. כדי להימנע מאפקט $Apertur$ נחליק את התמונה.
  שימוש בפירמידות: גם כאן נשתמש בפירמידות כדי לא לחפש בכל התמונה. נתחיל מהרמה הקטנה ביותר, ונמשיך בעזרת בתוצאה שקיבלנו לרמה שמעליה.

6.11 ייצור תמונת פנורמה:

- **מימוש:** נבחר תמונת ייחוס ונתאים לה את התמונה מימינה וכן הלאה. כדי למצוא את הייחוס בין תמונות שאינן צמודות - נכפיל את המטריצות שלהן עם המטריצות של התמונות שבניהן.
  לאחר מכן נעשה $warping$ - נזיז אתה תמונות ביחס לתמונה שבחרנו, ונחבר אותן יחד.

6.11.1 שיטות שונות להדבקת התמונות:

- **האתגר**: אנו רוצים להדביק את התמונות כך שלא נראה שיש תפר בניהן, אלא שהמעבר ייראה חלק ככל היותר.

- **אפקט $gosting$:** אפקט שקורה כשאנחנו מחברים מספר תמונות בהן יש תזוזה של אובייקט. כך כשנחבר אנו נראה את האובייקט נמרח.

- **השיטה הנאיבית:** נקח את מרכז התמונות, ומכל תמונה נקח את החלק שלה שהכי קרוב אל המרכז שבחרנו. ונדביק אותן יחד.

- **שיטת החלקה - $\alpha - blending$:** נרצה לטשטש את המעבר בין התמונות. נעשה זאת בעזרת מסכה לא בינארית, השיטה הזאת נקראת $Feathering$.
  החסרון: יש לנו קושי לבחור את הפרמטר α הנכון (המשקל שנקח מכל תמונה בכל שלב - קצב השינוי).

- **הדבקה בעזרת פירמידות:** נקח פירמידת לפלסיאןשל כל תמונה ונתפור כל רמה של תדרים (רמת פירמידה) בנפרד.

6.11.2 הדבקה בעזרת תכנון דינאמי:

- **מוטיבציה:** נרצה להתמודד עם אפקט $gosting$, נעשה זאת בעזרת מציאת המיקום הטוב ביותר לחתוך את התמונות (התפר), כך שאם יש אובייקט שזז נקח את האחת מהתמונות שבה נמצא האובייקט ומשאר התמונות נוריד את האובייקט שזז. כלומר נסתכל על כל אזור חפיפה, ונקח רק אחד מכל התמונות.

- **המימוש:** נעשה זאת בעזרת חיתוך דינאמי. נתחיל מהשורה הראשונה ונשאל עבור כל שורה איפה הכי זול לנו מבחינת $error$ להעביר את החתך, ונזכור את התוצאה עבור השורה הבאה.

- **נחשב את השגיאה כך:** בין שני אזורי החפיפה בשתי התמונות.

- **לאחר מכן נמלא את הטבלה באופן הבא:** עבור כל פיקסל

$$E[i,j] = e(i,j) + \min(E[i-1,j-1], E[i-1,j], E[i-1,j+1])$$

כשהמינימום מייצג את שלשת הפיקסלים הבאים: הפיקסל מעל, מימין למעלה ומשמאל למעלה.

- **לבסוף:** נלך לשורה האחרונה ונבחר את הפיקסל המינימלי, ונעלה למעלה בשורות.
- **סיבוכיות:** $O(n)$ץ

6.11.3 הדבקה בעזרת $min\ cut$:

- **מוטיבציה:** נרצה להתמודד עם אפקט $gosting$. **בנוסף** תכנון דינאמי לא מאפשר לנו לקחת חתכים מורכבים, אלא חתכים אנכיים בלבד.
- **המימוש:** נגדיר רשת זרימה בארופן הבא ־ נתייחס אל החלקים שנקח מהתמונה הראשונה בתור המקור של הרשת, לחלקים שנקח מהתמונה השניה בתור הבור, הקודקודים יהיו הפיקסלים באיזורי החפיפה, ומשקלי הצלעות יהיו ההפרשים בין התמונות.
  נרצה למצוא את: החתך המינימלי ברשת.
- **נגדיר את משקלי הצלעות כך:**

$$W(p1, p2,\ \mathrm{A},\ \mathrm{B}) = \|\mathrm{A}(p1) - \mathrm{B}(p1)\| + \|\mathrm{A}(p2) - \mathrm{B}(p2)\|$$

כאשר $p1, p2$ הם קודקודים (סמוכים?), ו A, B הן התמונות. נשאל כמה שתי התמונות מסכימות עבור שני הקודקודים. אם התוצאה מינימלית (הסכמה גבוהה) נרצה להעביר שם את החתך.

- **לבסוף:** מה שנמצא משמאל לחתך שייך לתמונה הראשונה, ומה שנמצא מימין לחתך נקח מהתמונה השניה.

6.11.4 הדבקה בעזרת יריעה:

- **קרני אור של מצלמה:** כל מצלמה מצלמת את הסצנה עם קרני אור שמסתכלום בזווית ימינה, ישר ושמאלה.
- **איך נחבר את הקרניים:** נעבור על כל התמונות ומכל התמונות נקח את הקרניים שמסתכלות ימינה ונחבר אותן לתמונה אחת. לאחר מכן נעשה אותו דבר לקרניים שמסתכלות ימינה ושמאלה, לאחר מכן נחבר את שלשת התמונות.
- **תפירת תמונות עם יריעה:** כשנרצה לתפור מספר פריימים עם שינוי בכל פריים, לדוגמה סרטון של עץ שענפיו זזים ברוח. נצטרך להגדיר יריעה ממנה נקח כל פעם חלקים מפריימים מתקדמים יותר וחלקים מפריימים מאוחרים יותר, כדי ליצור אפקט של תזוזה.
  כלומר ניקח סטריפים שונים מזמנים שונים.

7 ניקוי רעשים:

- **המטרה:** נרצה להוריד את הרעש מבלי לפגוע בתמונות.
- **מתי נקבל תמונות רועשות:** צילום בחושך, אולטרסאונד.

- **איך נבדוק את איכות הניקוי:** עבור תמונה מורעשת y, תמונה מקורית x , ותמונה לאחר ניקוי $\hat{x}$.

1: נחשב $n = y - \hat{x}$. אם קיבלנו תמונה **ללא קווי מתאר** אזי ניקינו את הרעש כמו שצריך, אך אם זלגו פרטים מהתמונה המקורית - הניקוי היה לא טוב.

2: $mean\ square\ error$ - נחשב באופן הבא:

$$MSE = ||\hat{x} - x||_2^2$$

ככל שהתוצאה קרובה יותר ל 0 אזי התמונה נקייה יותר.

3: $PSNR$ שיטה שנמצאת בשימוש בעיבוד אותות

$$PSNR = 20 \cdot log_{10} \frac{255}{\sqrt{MSE}}$$

7.1 טשטוש בעזרת $Biliteral\ Filter$:

- **הרעיון:** נרצה לטשטש רק במקומות שאין לנו $edges$. נעשה זאת באופן הבא - כשנראה $edge$ נטשטש רק מצד שמאל שלו, ולאחר מכן כשנעבור אותו נטשטש רק מצד ימין שלו.

- **הנוסחה עבור** $gray\ scale$: נטשטש עם גאוסיאן, אך ניתן משקל גבוה עם למרחק מהפיקסל $(\mathbf{p} - \mathbf{q})$ וגם למרחק בדרגת האפור $(I_{\mathbf{p}} - I_{\mathbf{q}})$ -

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_{\mathrm{s}}}(\|\mathbf{p} - \mathbf{q}\|) G_{\sigma_{\mathrm{r}}}\left(|I_{\mathbf{p}} - I_{\mathbf{q}}|\right) I_{\mathbf{q}}$$

נשים לב: כי הוספנו גורם נרמול $\frac{1}{W_{\mathbf{p}}}$.

סימונים: σ_{s} מסמן את גודל הגאוסיאן במרחב, ו σ_{r} מסמן את גודל הגאוסיאן בצבע.

- **כיצד נבחר את גודל הגאוסיאנים:**

גאוסיאן במרחב: נבחר 2% מגודל אלכסון התמונה.

גאוסיאן בצבע: ממוצע או חציון של הגרדיאנטים של התמונה.

- **הנוסחה עבור תמונת** RGB: נסתכל על ווקטור הצבע C_q כעל תלת ממדי, ועל המרחק כמרחק ווקטורי.

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_{\mathrm{s}}}(\|\mathbf{p} - \mathbf{q}\|) G_{\sigma_{\mathrm{r}}}\left(|C_{\mathbf{p}} - C_{\mathbf{q}}|\right) C_{\mathbf{q}}$$

- **החסרון - סיבוכיות חישוב:** מכיוון שאנו צריכים לנרמל במשקל כל פיקסל - $\frac{1}{W_{\mathbf{p}}}$, הקושי החישובי הוא גדול.

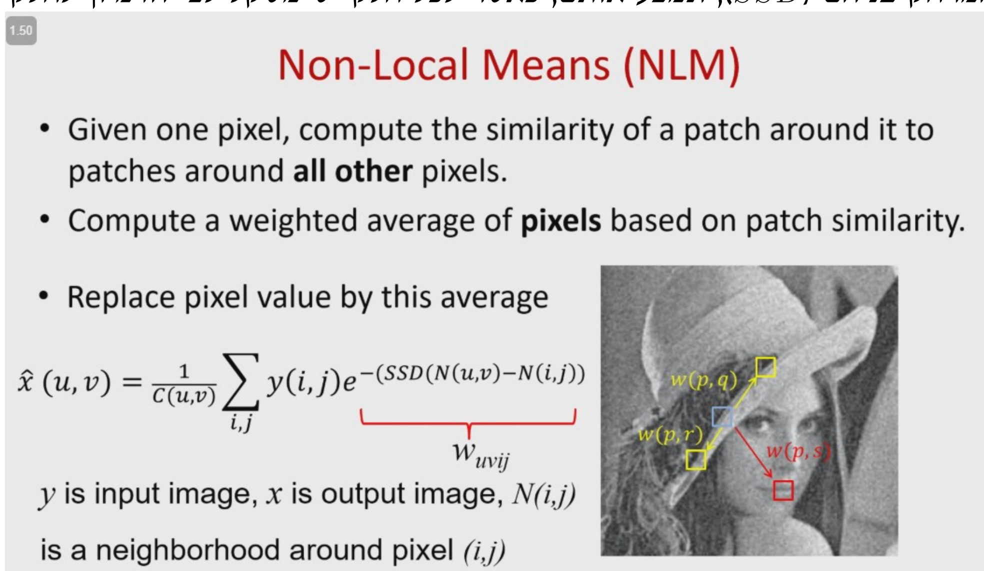
7.2 טשטוש בעזרת טלאי - $Patch\ Methods$:

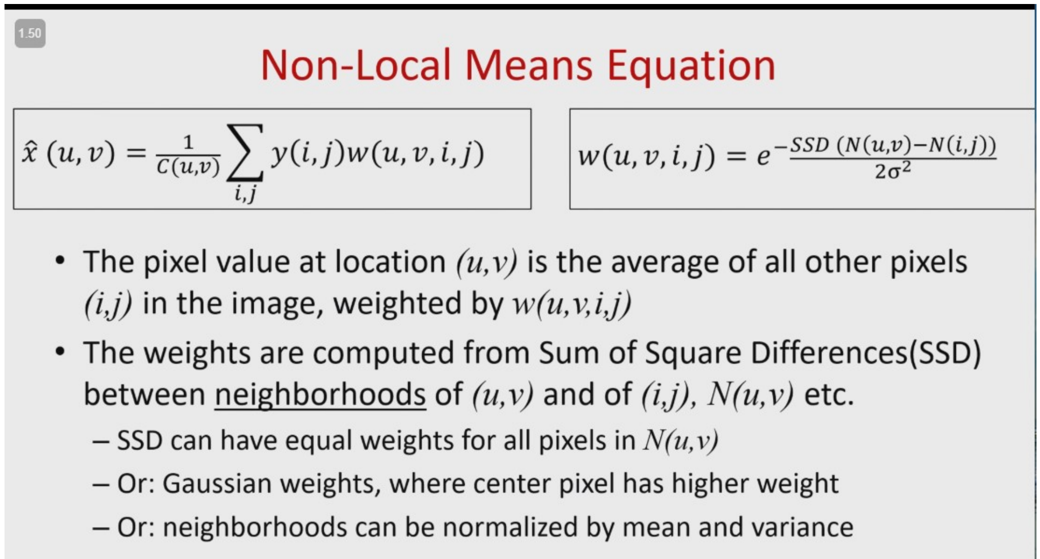
- **סימונים:** $y(t)$ - תמונות סטטיות בזמנים שונים.
- **הנחה:** אנו מניחים כי הרעש הוא גאוסיאני והממוצע שלו שווה ל 0.
- **מוטיבציה:** נשתמש כאזר יש לנו מלא תמונות של אותו האובייקט, אך כל תמונה עם רעש שונה. אנו נרצה למצע כמה שיותר תמונות יחד, כך כל תמונה תהיה עם רעש שונה ולכן הממוצע של התמונות יוריד את הרעש.
- **הנוסחה:**

$$Var(\bar{X}) = \left(\frac{1}{n}\right)^2 Var\left(X_1 + X_2 + \cdots + X_n\right) = \frac{\sigma^2}{n}$$

כיצד נעשה זאת על תמונה שיש לנו רק עותק בודד שלה:

- **הרעיון:** יש לנו בכל תמונה כמה חלקים שהם דומים באותה התמונה, לכן נקח את כל החלקים הדומים ונמצע אותם.
- **כיצד נבצע - שיטת** NLM: נקח חלון עבור כל פיקסל שנרצה למצע, נעבור על כל החלונות בתמונה ונבדוק מה המרחק בניהם (SSD), ונמצע אותם, כאשר לכל חלק יש משקל לפי הדמיון לחלק המקורי.





8 אלגוריתם $Hough\ Transform$ למציאת צורות גיאומטריות בתמונות:

8.1 שיטה ראשונה:

- **מה נרצה:** נרצה למצוא צורות גיאומטריות בתמונה, קווים עיגולי ועוד. נעשה זאת בדומה לאלגוריתם $RANSAC$ ע"י הסכמה של הפיקסלים על הצורה המתאימה.

- **הקושי:** יש בתמונה רעשים, הסתרות

- **שלב ראשון:** נעביר את התמונה לתמונת $edges$ ע"י האלגוריתם של $canny$ או כל אלגוריתם אחר.

- **שלב שני:** נקח את כל הפיקסלים בתמונת האדג' שמסכימים על הצורה שאותה אנחנו מחפשים.
נעבור על כל האפשרויות לצורה שאנחנו מחפשים (לדוגמה כל הסיבובים של קו ישר) ונשאל את כל הפיקסלים על איזה קו ישר הם מסכימים.

- **מטריצת פרמטרי המרחב:** למעשה כדי לייצר קו ישר אנחנו צריכים למצוא את המשוואה $y = ax + b$ כאשר a, b הם הפרמטרים שאנחנו מחפשים.
ניצור מטריצת מרחב פרמטרים שמאותחלת כולה לאפסים, כך שהעמודות ייצגו את b והשורות את a, וכל נקודה בתמונה תייצג את (a_i, b_j). ונמפה כל קו בתמונה לנקודה במרחב הפרמטרים (כל נקודה שפיקסל מסכים עליה יתווסף לה - 1).

- **לאן הפיקסל יצביע:** נשים לב כי פיקסל x, y יכול להצביע למלא קווים שונים שעוברים דרכו, אך מתקיים: $b = y - ax$ לכן כל נקודה תצביע לקו ישר אחד במרחב הפרמטרים.

- **משפט הדואליות ב $Hough$:** קו בתמונה ממופה לנקודה במרחב הפרמטרים. ונקודה בתמונה ממופה לקו במרחב הפרמטרים.

- **לבסוף:** נסתכל על הנקודה במרחב הפרמטרים שהכי הרבה קווים עברו דרכה, ונבחר את הישר הזה להיות הישר שאותו חיפשנו (הנקודה שקיבלה ערך מקסימלי), ניתן לעשות זאת באמצעות $threshold$.

- **בעיות:**
רפרזנטציה: אנו לא יכולים לייצג קווים אנכיים.
גודל הטבלה: מה הערכים של a, b
הרזולוציה: מה הקפיצות של a, b.

8.2 שיטה שניה:

- **שיטה שניה - קאורדינטות פולאריות:** ניתן לייצג קו ישר באמצעות המרחק מהרשית - d. והזווית ביחס לציר ה x:

$$d = x \cdot \cos(\theta) + y \cdot \sin(\theta)$$

כעת נקודה לא תתמפה לקו ישר, אך עדיין כל הסינוסים יתלכדו לאותה הנקודה.
נבחר $threshold$ וכל הנקודות שעברו אותו ייכנסו בתור קו.

- **מה פתרנו:**
רפרזנטציה: נוכל לשים קווים מאונכים (זווית 0).
גודל הטבלה: הטווח כעת הוא d - גודל התמונה. θ - זווית של 360 מעלות.
הרזולוציה: נוכל לבחור את הקפיצות של d להיות קפיצות של פיקסלים, ושל θ - להיות קפיצות של מעלות או חלקי מעלה.

- **מה נעשה שיש לנו d, θ לא שלמים:** נעשה קוונטיזציה ונשלח אותם לערך הקרוב ביותר.

- **נשים לב:** כי האלגוריתם מחזיר לנו קווים אנסופיים. לכן נצטרך לחזור לתמונה ולהבין אילו חלקים של הקו רלוונטים לנו.

8.3 שיפורים לאלגוריתם:

- **שינוי ראשון -** $Smoothing$: נרצה לטשטש את תמונת ה $edges$ ולעשות הצבעה שתלויה בעוצמה. (במקום תמונה בינארית).

- **כיצד נבחר את המקסימום המקומי:** אנו נרצה להימנע מכך שהמקסימומים שנבחר יהיו שניהם מאותו האיזור. לכן נבצע $non\ max$ ונבחר את המקסימום הלוקאלי מכל סביבה.

- **כיצד נייעל:** כדי לחסוך בהצבעות נרצה להשתמש בגרדיאנט. כשאנו מסתכלים על פיקסל מסויים במרחב ההצבעות, נסתכל על הגרדיאנט וכך נוכל למצוא את הקו הבודד שעובר דרכו. באופן זה לא נצטרך לבדוק את כל הקווים שיכולים לעבור דרך הנקודה.
כך יהיו לנו לא הצבעות לאיזור מסויים במרחב ההצבעות, משם נקח את המקסימום הלוקאלי.

8.4 מציאת מעגל:

- **הרעיון:** נרצה אלגוריתם למציאת צורות עגולות בתמונה. נבנה מרחב פרמטרים (a,b,r) כאשר a,b הם מרכז המעגל, ו r הוא הרדיוס.
  נעבור על כל הפיקסלים בתמונת ה $edges$, ונבקש מהם להצביע למשוואת מעגל מסויימת a,b,r.

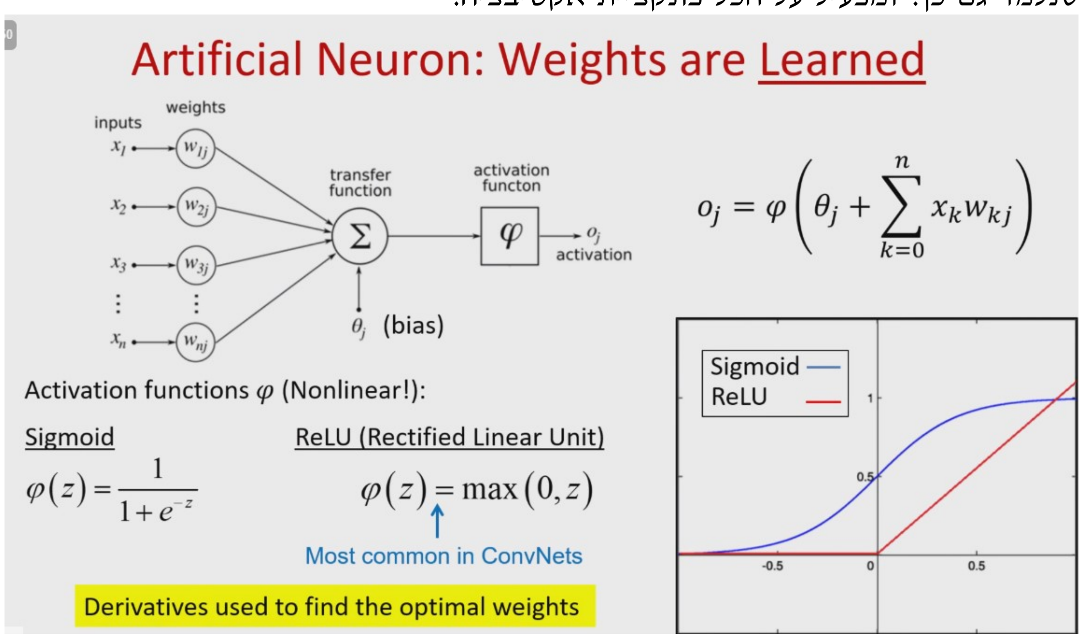
- **בעיות:**
  גודל הטבלה: כעת הטבלה היא n^3.
  מספר ההצבעות: יש לנו n^2 (הצבעה ל a,b שיקבעו את r) הצבעות וזה מלא.

- **ייעול:** נשתמש בגרדיאנט כך שעבור כל רדיוס r, הפיקסלים יצביעו רק ל a,b שנמצאים בכיוון הגרדיאנט, כך כל נקודה תצביע לשני מעגלים.
  צמצמנו את ההצבעות ל $O(n)$.

- **מימוש:** באותו האופן של מציאת קווים, נתחיל מתמונת ה $edges$ ונחפש הצבעות של כל פיקסל, עד שנמצא נקודות מקסימום מקומי שכל ההצבעות מתלכדות אליהן.

9 רשתות נוירונים:

- **רשת נוירונים:** כל נוירון מקבל וקטור קלט, מכפיל את הקלטים במשקולות w שנלמדו, סוכם, ומוסיף להם $bias$ שנלמד גם כן. ומפעיל על הכל פונקציית אקטיבציה.



- **תהליך האימון:** נרצה למצוא את המשקולות w ואת ה $bias$ עבור כל נוירון.

- **משפט:** מספיקה לנו רשת עם שכבה אחת אך עם מספר גדול של נוירונים, כדי לפתור כל פונקציה.

- $one\ hot\ vector$: דרך לתרגם אותיות למספרים, ווקטור של אפסים שמכיל 1 במיקום המתאים.

- **רשת** $AutoEncoder$**:** רשת שבה השכבה מכילה פחות קודקודים מהאינפוט והאאוטפוט. ממשת להצפנה.
שימוש בתמונות: משמשת לניקוי רעשים. רעש נופיע באופן אקראי בכל פיקסל, לכן אם נעבור דרך מספר קטן של קודקודים הרעש לא יעבור וייעלם.

- **פונקציית** $softMax$**:** פונקצייה שמנרמלת את ווקטור האאוטפוט. נשתמש בה בבעיות קלסיפיקציה, היא נותנת לנו הסתברות לקבל כל תוצאה.

- **פונקציית אקטיביציה:** פונקציה שכל ניירון מפעיל על הקלט, היא עוזרת לנו ללמוד פונקציות לא לינאריות.

9.1 פונקציות $Loss$ למדידת השגיאה, ותהליך הלמידה:

9.1.1 פונקציות $Loss$ שונות:

- **פונקציית** $Loss_{0-1}$**:** אנו נמדוד את ביצועי הרשת בעזרת פונקציית $error$, ככל שהיא נמוכה יותר הרשת טובה יותר.

$$L_{0-1}(f,S) = \frac{\text{number of times } f\left(x_i\right) \neq y_i}{|S|}$$

- **פונקציית** $accuracy$**:** סופרת את מספר התוצאות שצדקנו עליהן. ככל שהיא גבוה יותר הרשת טובה יותר.

- **פונקציית** MSE**:** מרחק ריבועי בין הפרדיקציה לתיוג האמיתי.

$$L_{MSE}(f,S) = \frac{1}{|S|}\sum_i \left\|f\left(x_i\right) - y_i\right\|^2$$

- **פונקציית Cross Entropy Loss:** פונקציה זו מתאימה לבעיות קלסיפיקציה. הרשת מחזירה לנו ווקטור הסתברות, כך שכל כניסה i בווקטור מייצגת הסתברות לקבל את התיוג ה i.

- **המטרה:** נרצה למצוא את ה $Loss$ המינימלי, כדי למצוא את המינימום אנו נרצה לגזור.

- **מציאת מרחק בין ווקטורים:**

$$\text{Cosine Similarity: } \frac{\sum_x \left(I_1(x) \cdot I_2(x)\right)}{\|I_1(x)\| + \|I_2(x)\|}$$

- **פונקציית Perceptual Loss:** פונקצייה המתאימה למציאת לדמיון בין תמונות. היא בודקת תכונות של התמונה ולא דמיון בין פיקסלים. משום שאם נמדוד דמיון בין פקסלים אזי שתי תמונות מוזזות יקבלו $Loss$ גבוה למרות שהן דומות.
כיצד נעשה זאת: נשווה בין מאפייני התמונות ברשת נוירונים. נקח רשת VGG מאומנת, נקח את האקטיביציות מכל מיני שכבות שונות. נעשה אותו הדבר לשתי התמונות ונשווה בניהן. אם הם קיבלו אאוטפוט דומה בשכבות דומות (נמדוד באמצעות $L2-Loss$) אזי הן יקבלו $Loss$ נמוך.

9.1.2 תהליך הלמידה:

- **כיצד נמצא את ה $Loss$ המינימלי:** נשתמש באלגוריתם גרדיאנט דיסנט. נחשב את הגרדיאנט, ונלך בכיוון ההפוך (נלך בכיוון ההפוך לשינוי הגבוה ביותר.) באופן זה אנו נמזער את פונקציית ה $Loss$. עבור פרמטרים θ נלמד אותם כך:

$$\theta_i = \theta_{i-1} - \eta \nabla f_{\theta_{i-1}}$$

כאשר η הוא $learning\ rate$ וואומר לנו מהו גודל הצעד. זהו **היפר פרמטר** שאנו קובעים והוא לא נלמד.

- **אלגוריתם** SGD: נדגום תת סט של הדאטה $(epoch)$, ועליו נחשב את ה $Loss$ ואת הגרדיאנט, ונאמן עליו את המודל כדי למצוא את הפרמטרים.
גודל ה $batch$ שנבחר גם הוא היפר פרמטר.

- **בעיית התאמת יתר -** $overfitting$: אם נשאף להגיע ל $Loss$ המינימלי, אחנו נתאים את המודל שלנו טוב מאד לסט האימון, אך הוא לא יכליל טוב לעולם האמיתי.
כיצד נוודא שלא עשינו $overfitting$: נפצל את הדאטה ל $train, test, validation$. ונבדוק את הפרמטרים שלמדנו מה $training$ על ה $validation$. לבסוף נמדוד על ה $test$.

9.2 רשתות קונבולוציה:

- **רשת קונבולוציה:** רשת מלאה, בה כל שכבה היא קונבולוציה של **חלון** של השכבה שלפניה, **המשקולות הן קבועים** בשכבה הרשת.

- **רשתות קונבולוציה לתמונות:** נקח את התמונה x בתור קלט, ופילטר w (חלון) בגודל מסויים המגדיר משקולות. נכפול את הפיקסלים שבחלון במשקולות הפילטר ונסכום, זה יהיה הפלט של הנוירון הבא.
ניתן להגדיר מספר פילטרים בגדלים שונים ומשקולות שונות, וכך ליצור מספר שכבות.
נחשב:

$$w^T x + b$$

למעשה: כל נוירון תלוי באיזר קטן מהשכבה שלפניו (החלון).
מה נלמד: את הפילטר

- **מימד השכבה:** כל שכבה תהיה קטנה יותר מהשכבה שלפניה, משום שאנו רוצים שהפילטר לא יחרוג מגבולות התמונה \ השכבה הקודמת. המימד השלישי של השכבה יהיה מספר הפילטרים שיצרו אותה.
לדוגמה עבור תמונה של 32×32 ופילטר של 5×5, השכבה שנקבל היא 28×28. אם נפעיל 6 פילטרים המימד של השכבה יהיה $28 \times 28 \times 6$.

- **בין שכבה לשכבה:** נפעיל פונקציית אקטיביציה $ReLU$.

- $stribe$**:** הפרמטר שבו אנחנו קופצים עם החלון מפילטר לפילטר, $stribe = 2$ - זאת אומרת שנעבור על פיקסל כן פיקסל לא.

- **ריפוד באפסים:** אם לא נרצה להקטין את התמונה ואת השכבה הבאה, נרפד את גבולות הפילטר שמחוץ לתמונה באפסים.

- **שכבת פולינג -** $Pooling\ Layer$**:** המטרה היא להקטין את מימד האינפוט, על פני הרוחב והגובה (מבלי לגעת במספר ה $chanels$). אנחנו לוקחים את השכבה ומקטינים אותה.
  1 - $Max\ poolimg$**:** נעשה מקסימום כדי להשיג אי לינאריות, ניקח מכל סביבה את האלמנט המקסימלי.
  2 - $Average\ poolimg$**:** נקח חלון על כל התמונה ולבסוף נחשב ממוצע של החלון.
  היפר פרמטר: יהיה גודל החלון, אין שום פרמטר שאנחנו רוצים ללמוד.

- **היפר פרמטרים של הרשת:** עומק (מספר הפילטרים לכל שכבה), רוחב השכבה, סוג השכבה, גודל החלונות, מה ה $stribe$.
  פרמטרים של האימון: מה שיטת האופטימיזציה, מהו צעד הגרדיאנט ($learning\ rate$), איך נתחיל.

- **בחירת משקולות:** בד"כ נבחר בהתחלה באופן רנדומלי, עד שהמספרים מתכנסים.

- **אימון רשת:** נבחר תחילה דגימות, נריץ את הרשת עליהן ונקבל את השגיאה, נחשב את הגרדיאנט בעזרת $Backprop$ לבסוף נעדכן את הפרמטרים ע"י הגרדיאנט.

- **שיטת** $DropOut$ **למניעת אוברפיט:** בכל $epoch$ נזרוק חלק מהנוירונים ולא נשתמש בהם. כך שנמנע מנויירונים להסתמך על נוירונים אחרים, נוירונים אלו הם נוירונים שבד"כ לומדים מבנים סבוכים לאוד שהם למעשה $overfitt$.

9.3 שיפור יכולות של רשתות:

- **Transfer learning:** כשחוקרים ניתחו רשתות הם גילו כי השכבות הראשונות ברשת שומרות מידע כללי על הדאטה. לכן ניתן לעשת שימוש בשכבו הראשונות של רשתות מאומנות, ולאמן רק את השכבות העמוקות יותר שלהן על דאטה חדש.

- **קונבולוציית** 1×1**:** חוקרים של גוגל הוסיפו לרשתות שלהם קונבולוציות של 1×1, באופן זה אנחנו משנים את מימד העומק של התמונה - הצ'אנל.
  למה זה טוב: נוכל לקחת אינפוט של $n \times m \times c$ (כאשר c מייצג את מספר הצ'אנלים), ולעשות קרנל של 1×1 ולהוציא אאוטפוט של $n \times m \times c/2$. כך נצמצם את כמות הצ'אנלים.
  1: זה עוזר לנו לתמצת, ולהבין מה קורה בכל אחד מהצ'אנלים.
  2: בנוסף הן יכולות לשמש כ $fully\ connected$ - זה שקול לטנזור של $1 \times 1 \times c$ והיא דומה למטריצת $fully\ connected$. כך נוכל לשחק עם גודל האינפוט. (זה פחות שימושי להקטנת גודל האינפוט).

- רשת $ResNet$: החוקרים זיהו כי קיימת בעיה ברשתות עמוקות. כשאנו לומדי את הגרדיאנטים של רשת עמודה וחוזרים להחלה, לפעמים אנחנו כופלים בערכי גרדיאנט שקטנים מ 1 (בעיית Vanishing gradient). לכן כשנגיע לשכבה הראשונה הערכים יהיו 0 כי אנחנו כופלים מלא ערכים קטנים. הפתרו היה לעבוד עם בלוקים - Residual Block.

- **שיטת Residual Block:** הבלוק יכיל שכבות וקונבולוציות. לסוף הבלוק (השכבה האחרונה של הבלוק) אנחנו נחבר את האינפוט. השכבה האחרונה תסכום את האינפוט המקורי והאינפוט שעבר בשכבות, וזה מה שיעבור לשכבה הבאה.

למה זה פותר את בעיית Vanishing gradient: בבכל פעם שנגזור שכבה אנחנו נגזור את $x + f(x)$ כך הגרדיאנט לא יהיה קטן מידיי כי יש לנו את x. כך אנחנו נגיע לתחילת הרשת עם משקל שונה מ 0.
בנוסף - נוכל לעבוד מרשת רדודה לעמוקה. נגדיר את כל הבלוקים אך ניתן להם בהתחלה פרמטרים נמוכים, כך בהתחלה הרשת לא תתחשב בהם.אך בכל שלב שהרשת תתפתח, היא תתן להן משקולות גדולים יותר, וכך היא תגדל ותהיה עמוקה יותר.

9.4 מודלים גנרטיבים:

המשימה שלנו היא $unsupervised$, אין לנו תיוג, אלא נותנים לרשת ללמוד מלא תמונות ולאחר מכן מבקשים ממנה להוציא תמונה מסויימת שהוא ייצר בעצמו. המודל יסומן ב G

דרישות מהמודל:

1. $Sufficient$: כלומר שיהיה מסוגל לייצר את כל התמונות בטריינינג סט. הוא מקבל כאינפוט ווקטור Z שנדגם רנדומלית, והוא יחזיר לנו $G(Z) = x$, כאשר x היא תמונה רנדומלית, שהרנדומליות שלה נובעת מהרנדומיות של הווקטור Z.

2. $Compactnes$: כל Z שנדגום יחזיר לנו תמוהנ טבעית. שלא יהיה לנו קודי לייצר את התמונה

סוגי מודלים:

9.4.1 מודל Auto-Regressive:

- **הרעיון:** אנו מייצרים תמונה ע"י חיזוי של הפיקסל הבא. נחזה רנדומלית את הפיקסל הראשון, ולאחר מכן נמשיך עם הפיקסל הבא שההסתברות שלו גובה.
בתהליך הלמידה: נלמד את ההסתברות לקבל את הפיקסל ה i בהינתן הפיקסלים $0...1-i$.

- **איך נלמד:**

$$p(x) = \prod_{i=1}^{n^2} p\left(x_i \mid x_1, \ldots, x_{i-1}\right)$$

- **חסרון:** המודל הזה איטי בייצור תמונות כי הוא מייצר פיקסל אחרי פיקסל. אך הוא מודל טוב למודל שפה (מודל N- גראם).

9.4.2 מודל VAE - variational autoencoders:

- **הרעיון:** יש לנו מצפין ומפענח. נכתוב רשת שמורכבת מ $encoder$ ו $decoder$. ה $encoder$ לוקח את התמונה ודוחס אותה לייצוג וקטורי, והמטרה היא למצוא את הייצוג הטוב ביותר שמתאר את התמונה בצורה הטובה ביותר. בנוסף נאמן $decoder$ שבהינתן וקטור הוא מחזיר לנו את התמונה המקורית. כך נוכל לממדוד את ביצועי ה $encoder$. (אנחנו לא צריכים את ה $decoder$ לאחר תהליך האימון).

- נשתמש ברשת זו לבעיות קלסיפיקציה, למדוד דימיון בין תמונות בעזרת דמיון הווקטורים.

- **כיצד נשתמש למודל גנרטיבי:** ניצור VAE, ונגדיר מרחב מהתפלגות שנחליט עליה שיהיה מרחב ממנו הווקטורים מתפלגים. כך נדע איך לדגום מהמרחב, ונוכל לדגום ווקטורים לשלוח אותם ל $decoder$, והוא ייצר לנו את התמונה.
שינוי ברשת: ה $encoder$ כעת יוציא לנו ממוצע ושונות μ, σ, ולא ווקטור. והם יתנו לנו את ההתפלגות, מתוכה אנחנו נדגום ווקטור שנשלח כאינפוט ל $decoder$ שייצר לנו תמונה.
איך יראה ה $Loss$: **1** - יהיה לנו reconstruction loss - כי אנחנו רוצים לוודא שהווקטור טוב ומקודד טוב. **2** - בנוסף יהיה לנו $KL-loss$ - ששומר על הסדר, ומוודא כי הווקטור שנדגם קרוב להתפלגות הנורמלית.

9.4.3 מודל GAN – generative adversarial networks:

- **הרעיון:** יש לנו שתי רשתות שמאמנות אחת את השניה. יש רשת יוצרת - G שמקבלת ווקטור Z מהתפלגות נורמלית והיא מייצרת תמונה מההתפלגות. בנוסף יש לנו רשת D שמקבלת תמונות ומדרגת איזו תמונה אמיתית ואיזו לא. כך הרשת D מאמנת את הרשת G, ע"י זה שהיא נותנת לה ניקוד על התמונות שהיא ייצרה ונראות אמיתיות.

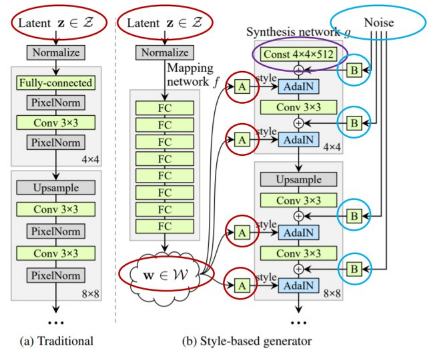
$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}(\boldsymbol{x})[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - [\mathbb{D}(\boldsymbol{Z}))]$$

למעשה G תנסה למנם את ה $Loss$, ו D מנסה למקסם אותה.

- **תהליך הלמידה:** בהתחלה נתן ל G לייצר תמונות באופן רנדומלי ונתן ל D לדרג אותן. D יצליח למצוא את ההבדל בקלות. לאחר מכן נחזור ל G ונגיד לו שהוא נכשל ושיוציא תמונות טובות יותר, וגם אותן ניתן ל D לבדוק ולהכריע אם הן אמיתיות. וכן הלאה וכן הלאה, עד שנסיים, ניפטר מ D נישאר עם G.

- **חסרונות:**
1: האימון לא מתכנס, ה $Loss$ לא מגיע לתוצאה קבועה.
2: $Mos\ collapse$: אם G למד לצייר תמונה אחת ממש טוב הוא יתקע וייצר בכל פעם את אותה התמונה כדי לנצח את D.
  3: **Diminished gradient**: הרשת D טובה מידיי, ואז לא משנה מה G יעשה אנחנו לא נצליח לנצח.
  4: היפרפרמטרים משנים לנו ממש את התוצאות.

- $StyleGAN$: רשת שפותחה ע"י אינבידיה, הרעיון הוא לייצר תמונות דומות עם שינויים קטנים. הם השתמשו ב $stile\ transfer$ - העברת סגנון של תמונה אחת לתמונה אחרת. אז הם לקחו תמונה 1, והתייחסו אל ה $content$ של התמונה (זהות הבנאדם, זווית הפנים) בתור הסטייל, וניצור דברים סטוכסטים ע"י רעש (הזזת השיער או הוספת נמשים) שהם למעשה ה $style$ של התמונה.
שינוי ארכיטקטורת G: כל כמה שכבות נתן לו רמז לסטייל של התמונה שאנו רוצים לייצר. נקח את ווקטור Z, נעביר אותו בתוך רשת f שתחזיר לנו ווקטור חדש W. את הווקטור W אנחנו נזריק ל G אחרי כל שכבה עם רעש שונה שנוסיף לו בכל פעם, והוא הולך לרמז לרשת מה לייצר.



9.4.4 מודל Denoising Diffusion:

- **הרעיון:** הרשת לומדת לנקות תהליך מרקובי של הרעשה של תמונה. נוסיף לתמונה קצת רעש בכל שלב, לפי התפלגות גאוסיאנית שאומרת לנו איזה רעש להוסיף בכל שלב, עד שלבסוף נקבל תמונה רועשת. במהלך האימון נלמד אותו לנקות רעש משלב i לשלב ה $i-1$. לבסוף נקבל רשת שמנקה מתמונה מורעשת נתונה קצת מהרעש.

- **איך נייצר תמונה:** נדגום רעש מוחלט, ונתחיל לנקות אותו קצת קצת עד שהרשת תייצר לנו תמונה חדשה.

10 גיאומטריה:

- כשאנחנו מצלמים את העולם, אז עבור כל נקודה (X,Y,Z) בעולם (כאשר Z מייצג את מימד העומק), אנחנו נעבור לנקודה (fX,fY,Z), שנמצאת על קו ישר שעובר ממרחב העולם, דרך החריר, עד למרחב התמונה. כדי לחשב את השינוי נוכל לקחת 6 נקודות, ולבדוק לאן כל נקודה במרחב האמיתי, עברה במרחב התמונה. סה"כ נצטרך 6 נקודות שיתנו לנו 12 משוואות, משום שיש לנו 11 דרגות חופש במטריצת המעבר M.

- **נוסחאות למעבר:** ממימד העולם למימד התמונה.

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

- **כיצד נחשב את המטריצה** M: נבחר 6 נקודות, נבדוק את הנקודה המקורית (X,Y,Z) והשנוי (u,v) ונחשב באופן הבא

$$u_i = \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$
$$v_i = \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$

- **חישוב זמן להתנגשות - TTC:** נרצה לחשב את הזמן שעובר כשאנו נייחים עד שאנחנו מנגשים בעצם זז. או שאנחנו בתנועה והעצם נייד. עבור הפריים ראשון שצולם x_1והפריים השני שצולם x_2 נחשב כך:

$$\text{TTC } = \frac{x_2}{x_2 - x_1} = \frac{fX/Z_2}{fX/Z_2 - fX/Z_1} = \frac{1/Z_2}{1/Z_2 - 1/Z_1} = \frac{Z_1}{Z_1 - Z_2}$$

כך שאם הפריים גדל פי 2, אנחנו נתנגש בפריים הבא.

10.1 גיאומטריה עם שתי מצלמות:

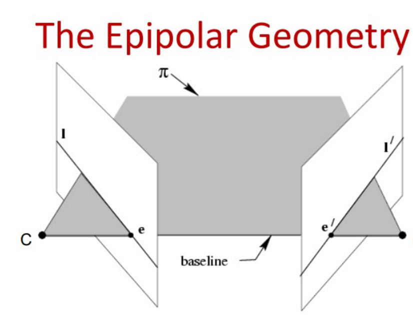
- **כיצד נעבוד עם שתי מצלמות:** נקודה במצלמה הראשונה תגדיר לנו קו בעולם (משום שכל נקודה נדגמת מקו ישר שעובר דרך החריר). וקו ישר בעולם יגדיר קו ישר במצלמה השניה, הקו נקרא "**קו הפיפולארי**" .

- **המישור ההפיפולארי:** כשיש לנו 2 מצלמות, הן יגידירו מישור הפיפולארי שמוגדר משני קווים הפיפולארים אחד לכל מצלמה.

- **בייס ליין:** המרחק בין שתי המצלמות נקרא הבייס ליין (קו שמחבר את שתי המצלמות).

- **אפיפיול:** המקום שבו הבייסליין חותך את התמונה (התמונה של המצלמה השניה במצלמה הראשונה).
המישור הזה יכול להיות פיזי (אם נניח את החריר בגובה המישור), אך זה לא משנה. הוא מוגדר ע"י שני מרכיז המצלמה ונקודה בעולם שעוברת דרכו.
נשים לב: כל מישור הפיפולארי כולל את הבייסליין (כל המישורים מסתובבים סביב הבייס ליין), וכולם נחתכים בנקודה e (המקום שהבייסליין חותך את המצלמות).

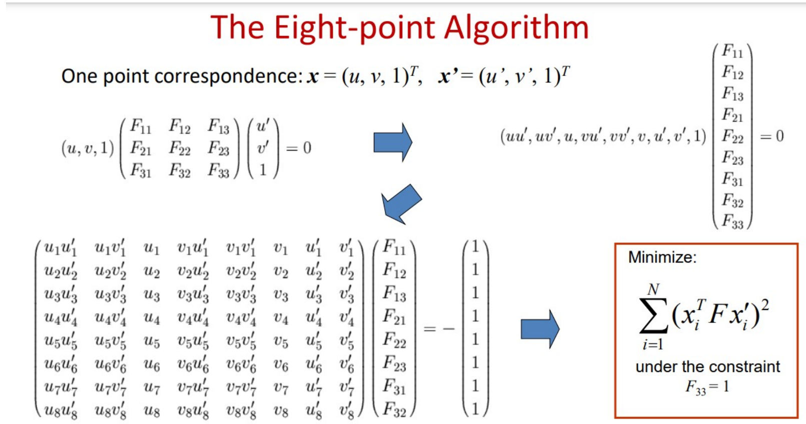
- **הנקודה** e: מיקום שבו מצלמה 1 רואה את השניה. והמקום שבו כל הקווים האפי פולארים נתכים בו באחת המצלמות.

- 

10.2 מציאת נקודות דומות בתמונה שצולמה משתי מצלמות - סטריאו:

- **הרעיון:** נרצה לנתח את העולם התלת מימדי ולהסיק פרספקטיבה ועומק שישקפו את המציאו.

- בהינתן מטריצה פונדימנטלית F, ונקודות בתמונה x נוכל למצוא את הקו האפיפולרי בתמונה x' כך שהנקודה תימצא עליו. כך נחסוך בחיפוש ונחפש על קו, במקום על כל המרחב . נעשה זאת באופן הבא:



- **מציאת המטריצה F:** באמצעות 8 נקודות.
- **פרלקס - $Parallax$:** כשאנו מצלמים שתי תמונות של סצנה אחת משתי מצלמות שונות. השינוי של הסצנה בין שתי התמונות נקרא פרלקס.

  כיצד נחשב:

$$d = x_l - x_r = \frac{fb}{Z}$$

נשים לב: כי ככל ש Z גדול יותר (האובייקט המצולם רחוק יותר) אזי הפרלקס קטן יותר.

10.3 אופק בתמונה - Vanishing point וחישוב עומקים:

- **נקודת האופק של תמונה - Vanishing point:** כשנרצה למצוא את האופק בתמונה נוכל לחפש זאת ע"י שני קווים מקבילים שנמצאים בתמונה.

  משפט: כל הקווים המקבילים בתמונה יפגשו באותה נקודה v באופק. (לדוגמה פסי רכבת).
- **Vertical Vanishing point:** כשנצלם מלמעה ללמטה או להיפך אזי ה Vanishing point יהיה ורטיקלי.
- **שיטות נוספות לחישוב עומק:** ניתוח התאורה, הסתרות, מרחק באמצעות לייזר ועוד

11 דחיסת תמונות:

מוטיבציה: נרצה להוריד את מספר הדיסקים שהתמונה תופסת בדיסק או שסרט תופס בדיסק, בכדי שתתפוס פחות מקום.

נתבסס על ההנחות:

1. פיקסלים סמוכים בתמונה דומים.
2. פריימים סמוכים בסרט, דומים.

דמיון בפיקסלים שכנים: לרוב פיקסלים סמוכים דומי למעט ב $edegs$. אך מספר ה $edges$ בתמונה הוא נמוך

דחיסה ללא אובדן: דחיסת zip, אף ביט לא משתנה. לרוב לא נשתמש בזה לתמונות, אלא רק לתמונות רפואיות.
דחיסת Lossy compression: דחיסה שבה נאבד קצת פרטים, לדוגמה דרגת אפור של צבע הפיקסל.

11.1 דחיסת הופמן קוד:

זאת דחיסה ללא אובדן.
הרעיון: נעשה היסטוגרמה של הסימנים בתמונה, לאחר מכן נבנה קודים באורך משתנה. כך שצבע שנמצא הרבה בתמונה - נקודד עם ביט 1, צבע נדיר יותר יקבל קידוד ארוך יותר. באופן זההקובץ ייצוג בפחות ביטים.

מימוש: אם יש לנו $a_1..a_n$ צבעים, והסתברויות $p_1,..p_n$ (ערך בהסטוגרמה). נבנה עץ, כל דרגות האפור יהיו עלים בעץ, ואנחנו נחבר שני עלים עם ההסתברויות הכי נמוכות לקודקוד אחד שיהיה סכום ההסתברויות. לאחר מכן נחבר כל שני קודקודים עם ההסתברויות הכי נמוכות. כך כל צבע יהיה עלה, לכל עלה יהיה מסלול מהשורש עד העלה, והקידוד של המסלול יהיה הקוד.
סה"כ: נעבור על הקובץ פעמיים.

דוגמה:

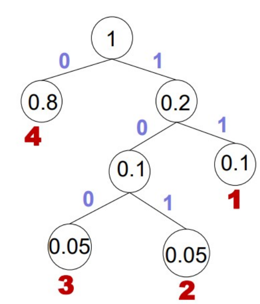
Huffman Coding: Variable Length

Pixel Value	1	2	3	4
probability	0.1	0.05	0.05	0.8

Pixel Value	1	2 3	4
probability	0.1	0.1	0.8

Pixel Value	123	4
probability	0.2	0.8

Pixel Value	1234
probability	1



נשים לב: אנו נדע שהגענו לסוף הקידוד אם הוא נגמר. אך מה נעשה במצבים בהם לא רצינו את כל הקידוד ולכן עצרנו באמצע והוא לא נגמר. אז מה שיקרה זה שבקידוד הבא אנחנו נמישך מאמצע הקידוד הקודם, זה קצת יעשה לנו בלגן, אך אחרי מססר קטן של קידודים אנחנו נחזור למיקום המקורי.

אם לכל הסימנים יש את אותה ההסתברות: הפמן קוד לא יחסו לנו כלום.

הופמן קוד לתמונה בינארית: אם יש לנו תמונה בינארית שרובה 1 ־ים והאשר אפסים. נוכל לעבוד עם מקבץ של ביטים ולהסתכל על הרצפים. לדוגמה נוכל לקחת רצף של 8 ביטים ולקודד אותו, באופן זה יהיו לנו מלא בלוקים שכולם 1.

שוני בין תמונות שמכילות אותם אובייקטים: אם יש לנו שתי תמונות שמכילות את אותם אובייקטים אך בסדר שונה. ההופמן קוד של שתי התמונות יהיה שווה משום שההיסטוגרמה שווה.

11.1.1 אנטרופיה - entropy encoding:

entropy encoding: דחיסה לפי האנטרופיה (וודאות בערכי הפיקסל), אם יש לנו הסתברות p (היסטוגרמה) על מאורע k (ערך מסויים לפיקסל). מה האינפורמציה שנקבל כשיגידו לנו שמאורע מסויים קרה, כלומר כמה האינפורמציה תחדש לנו משהו שלא ידענו. אז אם $p(k)$ גבוהה - האינפורמציה לא עזרה לנו הרבה (היא מועטת). אך אם ההסברות נמוכה והמאורע קרה - קיבלנו הרבה אינפורמציה שעזרה לנו, כי חשבנו שהמאורע לא צריך לקרות הסתברותית.
האנטרופיה היא סכום האינפורמציה של כל הערכים

$$\text{Entropy } = -\sum_{k=0}^{n} p(k)\ln p(k)$$

ההתפלגות תקבל את האנטרופיה הכי גבוהה כשלכולם יש את אותה ההסתברות (כש $p(k)=\frac{1}{n}$ כשיש n איברים).
ההסתברות עם האנטרופיה הנמוכה ביותר היא כשאנחנו בוודאות יודעים שכל דרגות האפור שוות לדרגה i מסויימת.

11.2 דחיסת Lempel-Ziv (LZW):

הרעיון: בונה על חזרות של סדרות בתמונה, ולא על ההסטוגרמה. ברגע שיש לנו הרבה חזרות אנחנו נדחוס ביעילות הרבה יותר גבוהה. לכן היא טובה לדחיסת טקסטים.
סה"כ: אנו עוברים על הקובץ פעם אחת בלבד.

מימוש: נעבור על הקובץ ונבנה טבלת מחזורות שראינו עד עכשיו, ברגע שנראה את אחת מהמחרונות שוב - נחליף אותה במספר התא שבה נמצאת המחזורת בטבלה.
אין צורך להעביר את הטבלה משום שהמקבל בונה אותה גם כן במעבר על הקובץ. אך את תחילת הטבלה - סימנים קבועים (0-255) אנחנו כן נעביר.
האלגוריתם:

LZW Encoding Algorithm

```
Initialize dictionary with all single characters (0..255)
P = first input character
while not end of input stream
    C = next input character
    if PC is in dictionary          // concatenation
      P = PC                        // concatenation
    else                            //PC not in dictionary
      output the code for P
      add PC to dictionary          // concatenation
      P = C;
end while
output code for P
```

11.3 דחיסת Run Length:

הרעיון: משמשת בעיקר לפקס, אנחנו מניחים כי רוב הטקסט הוא לבן, וחלק מהנקודות שחורות. לכן נכתוב את מספר הביטים של הצבע שיש לנו בכל פעם.
מתי זה ייכשל: כשיש לנו תמונה רועשת ויש הרבה החלפות בין לבן לשחור.

11.4 מערכות דחיסה:

אלגוריתם לדחיסה:

1. נעשה לתמונה טרנספורמציה שתביא אותה למצב נח לדחיסה.
2. נעשה קוונטיזציה למקדמים.
3. נעשה קידוד.
4. נשמור את התמונה.

אלגוריתם לשחזור:

1. נקרא את התמונה.
2. נעשה קקידוד הפוך לקידוד הדחיסה.
3. נעשה טרנספורמציה הפוכה.
4. נשמור את התמונה

נשים לב: אובדן האינפורמציה מתרחש בקוונטיזציה.

כיצד נמדוד שגיאה: נחסר מהתמונה המקורית ונחשב RSS או SNR. אך לעיתים אנחנו לא נראה שינוי ו RSS יהיה גבוה. לכן נוכל למדוד עם $perceptial\ Loss$ שמודד דמיון בין תמונות בעזרת רשת.

איזו טרנספורמציה נעשה:

1. נוכל לחשוב על נגזרת (קונבולוציה עם $[1,-1]$) - $edge\ detector$.
   אם נעשה נגזרת ועליה נעשה הופמן קוד ללא קוונטיזציה: התמונה תצא דחוסה כתלות באנטרופיה של ההיסטוגרמה. האנטרופיה אחרי הנגזרת תהיה יותר קטנה - לכן, תהיה לנו דחיסה טובה יותר.
   זה לא יעבוד על תמונה מורעשת.
   איך נשחזר: נשלח את הפיקסל הראשון ואת הנגזרת. באופן זה נוכל לשחזר את ההפרשים בעזרת הנגזרת.
2. **קוונטיזציה אופטימלית:** נעשה קוונטיבציה למספר נמוך יותר של דרגות אפור, כך נדחוס בדרגות האפור.

3. **דחיסת פירמידה:** נבנה פרמידת לפלסיאן, ונעשה לה קוונטיזציה רק בין 3-5 דרגות אפור. לאחר מכן נעשה דחיסת אנטרופיה של LZW.
למה זה עובד: הפירמידה עושה פרדיקציה טובה של הרמה מתחת, ורוב הרמה מלאה באפסים. יש לנו אנטרופיה נמוכה. יש קוונטיזציה חזקה.

הערה: דחיסת קובץ בפעם השניה מוציאה את הקורלציה (דמיון) מתוך הקובץ והיא לא תתן לנו יצרון נוסף.

11.5 $JPEG$:

מה זה: דחיסה שמשמשת לדיסת תמונות.

שלבי הדחיסה:

1. נחלק את התמונות לבלוקים זרים אחד לשני, בגודל $n \times n$.
2. עבור כל ריבוע נבצע טרנספורמציית DCT (טרנספורמציית cos).
3. נעשה קוונטיזציה למקדמים של DCT שקיבלנו.
4. נצפין עם הופמן.

למה נשתמש ב DCT: משום שהיא יוצרת שכפול של התמונה אך באמצעות שיקוף בקצוות. היתרון הוא כי בקצוות הפיקסלים יהיו דומים אחד לשני ולכן נוכל ליצור סדרה רציפה. לעומת שכפול של פוריה שמצמיד את הקצה השני של התמונה ויוצר סדרה מוזרה.
מה אנחנו נעביר: למעשה אנחנו נשמור על התדרים הנמוכים, וניפטר מהגבוהים.

טרנספורם cos ייראה כך:

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

עבור

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{if } u = 0 \\ \sqrt{\frac{2}{N}} & \text{otherwise} \end{cases}$$

ועבור הצורה הדו מימדית:

$$C(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\left(\frac{(2x+1)u\pi}{2N}\right) \cos\left(\frac{(2y+1)v\pi}{2N}\right)$$

הטרנספורם ההפוך ייראה כך:

$$f(x) = C(u)\alpha(u)\cos\left(\frac{(2x+1)u\pi}{2N}\right)$$

הטרנספורם ההפוך הדו מימדי ייראה כך:

$$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} C(u,v)\alpha(u)\alpha(v)\cos\left(\frac{(2x+1)u\pi}{2N}\right)\cos\left(\frac{(2y+1)v\pi}{2N}\right)$$

ופונקציית הבסיס תראה כך: (כל התמונות בעולם הם סכום של פונקציות הבסיס)

$$\alpha(u)\alpha(v)\cos\left(\frac{(2x+1)u\pi}{2N}\right)\cos\left(\frac{(2y+1)v\pi}{2N}\right)$$

איך זה יעבוד בפועל: טרנספורם DCT יגיד לנו בכמה להכפיל כל אחת מפונקציות הבסיס (בדומה לטרנספורם פורייה).
אם נקבל מקדמים נמוכים אנחנו נאפס אותם, וזאת בעצם הקוונטיזציה.

כיצד נעשה $JPEG$ עם צבע:

1. נעביר את התמונה לייצוג של $Y'CbCr$. כאשר Y זה ה $Intensity$, ו Cb, Cr זה ערוצי הצבע.
2. נדחוס כל אחד מהערוצים לחוד.
3. נעשה קוונטיזציה - נחלק את כל הערכים במטריצה בערך מסויים - n ונקח את הערך השלם שקיבלנו. בשחזור נכפיל כל אחד מהערכים ב n.
   לחלופין ניתן לחלק ב n לקחת את הערך השלם ולהכפיל ב n.

11.5.1 דחיסה נוספת - $JPEG$ 2000:

הרעיון: במקום לחלק את התמונה לחלונות, אנחנו נממש כמו דחיסת פירמידה, אך במקום פירמידה נשתמש ב Wavelet Functions.
Wavelet Functions: זוגות של $low-pass$ ו $high-pass$, זה חוסך לנו את ההוספה של הפירמידה לתמונה.

מימוש: נקח את הסיגנל, נעשה לו $high-pass$ ונדגום, אח"כ נעשה לו $low-pass$ ונדגום.
כך חצי מהפיקסלים ייוצגו ב low וחצי ב $high$.

11.6 דחיסת וידאו - $MPEG$:

הרעיון: אנחנו לא צריכים לשמור את כל הפריימים אלא רק את מה שהשתנה ביניהם.

כיצד נבצע:

1. **חיסור:** נוכל לדעת מה השתנה ע"י חיסור בין הפריימים.
חסרון: יש רעש ושוני בין התמונות, בנוסף יכול להיות שזזנו קצת

2. **התאמת תזוזה:** נעבור עם פאצ'ים על התמונה ונבדוק לאן הייתה תזוזה. נעבור על כל ריבוע בתמונה השניה, ונראה מאיפה הוא הגיע בתמונה הראשונה. נעשה זאת ע"י ווקטור, שיגיד לנו לאן הריבוע הכי דומה שאומר לנו מהיכן לקחנו את הריבוע, כלומר - כמה להזיז את הריבוע הנוכחי כדי שתאים לתמונה המקורית.
לבסוף: נחשב את ווקטור ההזזות שאומר לנו מה היה השינוי של כל ריבוע ($motion\ vector$). לאחר מכן נעשה חיסור בין הפריים המוזז לפריים הראשון, כדי לדעת מה היה השינוי בכל פריים (MCD). ונשלח את ווקטור ההזזות ($motion\ vector$) ואת ווקטור החיסורים בין הפריימים (MCD).
נשים לב: כי ככל שהפריימים יותר דומים הדחיסה יותר טובה.

שני מודים של דחיסה:

1. **דחיסה עם bit rate קבוע,** לדוגמה אם אנחנו יודעים שהערוץ חסום ואנחנו רוצים לשדר ברוחב הערוץ. אזי אם כל התמונה היא MCD (הפרש גבוה - יש הרבה הזזות), אנחנו לא נוכל לשדר את כל ההפרשים לכן נראה תמונה מטושטשת.

2. במקרה שנשמור את התמונה לקובץ ולא אכפת לנו משידור אז יהיו פריימים שידחסו יותר, ופריימים שידחסו פחות.

נשים לב כי יש לנו תלות כי כל הפריימים תלויים בפריים הקודם + ההפרשים. לכן אם פריים אחד מתקלקל כל הסרט נהרס.

כדי להתמודד עם הבעיה: אנחנו נשלח מידיי פעם פריים שלם שייקרא i, ושאר הפריימים שתלויים בו ייקראו p. פריים שאחריו ישלח i פריים, ייקרא b פריים.

12 צבעים:

את העולם אנחנו רואים בצבעים שמיוצגים ע"י אורכי גל בטווח $[350, 720]$. כל הצבעים מיוצגים ע"י שלשת צבעי בסיס RGB (יש עוד צבעי בסיס).

ניתן לייצג כל צבע (אורך גל) באמצעות צירוף לינארי של צבעי הבסיס. לכן נסדר מטריצה בגודל $3 \times 720 - 350$ שתגיד לנו עבור אורך הגל ה i מה מקדמי צבעי הבסיס שצריך כדי ליצור את הצבע ה i. כשמחשב ירצה להציג צבע i מסויים, הוא יכפיל את המקדמים שלו שמופיעים במטריצה ב RGB כדי לקבל את הצבע.

ניסוי שנערך הראה כי יש צבעים שכדי לייצג אום אחנו צריכים להחסיר מאחד מצבעי הבסיס, דבר שלא אפשרי פיזיקלית.

12.1 מרחבי צבע:

12.1.1 מרחב הצבע XYZ:

מרחב הצבע XYZ: מרחב צבע בו יש הפרדה בין הצבע לבין התאורה (בהירות) שלו.

- Y מייצג את ה $Luminates$ - תאורה.
- XZ ייצגו את הצבע - $chrumaticity$.

המוטיבציה לייצור מרחב הצבע הזה:

- כך נפטרנו מערכים שליליים, ב RGB בכדי לייצג צבעים מסויימים היינו צריכים ערך שלילי של צבע מסויים.
- סיבה נוספת היא שיהיה יותר קל לייצג צבעים כשיש הפרדה בין הצבע לבהירות שלו. בעוד שב RGB אנחנו צריכים לייצג כל צבע בעזרת שלשת הערוצים ויהיה הבדל בין אדום בהיר לכהה, בייצוג של XYZ כל צבעי האדום יהיו מיוצגים בעזרת אותם XZ ורק ערך ה Y יישתנה.

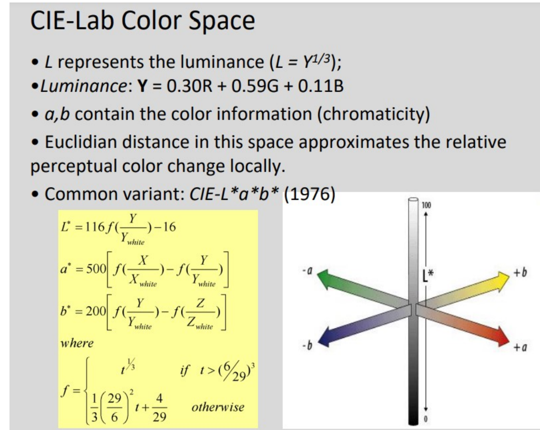
מהם צבעי הבסיס: בכדי להיפטר מהערכים השליליים כמו בייצוג של RGB, אנחנו נקח את צבעי הבסיס בייצוג זה להיות צבעים מחוץ לטווח אורכי הגל הנראה.
הרעיון הוא ליצור מרחב צבע היפותטי, כך כל מסך מחשב יכול למפות מכל מרחבי הצבע ל XYZ. מעין שפה משותפת שניתן לתרגם אליה את כל הצבעים, אך הוא היפותטי ומשמש למטרות תיאורתיות בלבד.

למה מרחב זה נוח יותר: למעשה מרחב הצבע הזה הוא תלת ממדי (X,Y,Z), אך מכיוון ש $X+Y+Z=1$ אזי Z תלוי ב X,Y ולכן ניתן להסתכל עליו כמרחב דוד מימדי.

מגבלת הצבעים של המרחב - gamuts:
במרחב זה כל הצבעים שנמצאים על קו שנמתח בין שני צבעים, יכולים להתקבל ע"י צירוף לינארי של הצבעים שבקצוות. אותו הדבר לגבי שלשה של צבעים, כל הצבעים שבאמצע המשולש יכולים להתקבל ע"י צ"ל של הצבעים שבקודקודי המשולש.
לכן: אם נבחר את צבעי הבסיס להיות RGB אנחנו לא נצליח לחסות את כל הטווח הנראה, כי המשולש שהם יוצרים קטן מכל המרחב. כדי לייצג את כל הצבעים נצטרך לקחת צבעי בסיס שנמצאים מחוץ לטווח הנראה. אך פיזיקלית אנחנו לא יכולים לממש את זה כי אחרת אנחנו לא נראה את הצבע שיוקרן למסך.
כל מכשיר מייצג את הצבעים לפי צבעי הבסיס שהוא בחר.
דמיון בין צבעים: ישנם צבעים במרחב שהם שונים, אך לנו הן נראים דומים. לכן הייצרנים לא צריכים להתאמץ ולתפוס אתכל הצבעים, אלא רק את הצבע שהכי קרוב אליהם ונראה לנו כאותו הצבע.

12.1.2 מרחב הצבע Lab:

מרחב צבע בו Lמייצג את השינוי בתאורה, ו abמייצגים את הישנוי בצבע.
קיים מרחב Lab^* שהוא לא לינארי.

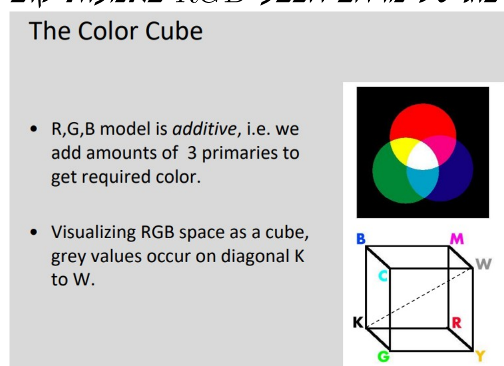


12.1.3 מרחב הצבע YIQ:

כבר ראינו את המרחב הזה, זהו מרחב שמיוצג באמצעות מטריצת מעבר ממרחב הצבע RGB.

12.1.4 RGB Cube:

ייצוג של מרחב הצבע RGB באמעות קוביה, באופן זה צבעים דומים עם תאורה שונה יהיו קרובים בקוביה.



ייצוג תמונות באמצעות הקוביה: ניתן לייצג תמונות בעזרת הקוביה שתשמש מעין ספקטוגרמה. בכל נקודה על הקוביה נסמן את אחוז החלק שלקחנו מהנקודה הזאת כדי לייצר את התמונה.

שוויון בין תמונות: ניתן להשוות בין תמונות על בסיס הסצנות. אם נרצה לחפש כמה תמונות דומות באגר, נוכל לחפש לפי הסטוגרמות צבע של התמונות.

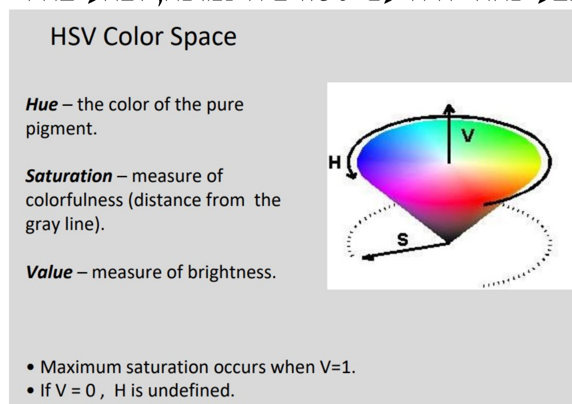
אפקט הקווים הישרים: חוקרים שמו לב כי כאשר אנחנו מגבירים בהירות של צבע מסויים בקוביה, הערך שלו גדל בקו ישר. הם השתמשו בתמונה הזאת בכדי לתקן תמונות בהן הגענו לקצה הקוביה והקו נמרח על הדופן משום שהוא הגיע לסטורציה. במקרה כזה נוכל להאריך את הקו משום שאנחנו יודעים שהוא צריך להיות קו ישר, ולתקן אותו בחזרה שיהיה בגבולות הקוביה. באופן כזה אנחנו מרוויחים פרטים שהפסדנו בגלל רגישות החיישן שמוגבלת למקסימום של 255.
למעשה: אלו פרטים שלא נמצאים בצילום, אך אנחנו משערים שהם נמצאים שם על בסיס התמונה והמשך הקו. באותו האופן ניתן לתקן אזורים שרופים בתמונה.

12.1.5 מרחב הצבע HSV:

מרחב צבע שמיוצג כחרוט שמורכב משלשה דברים:

- $H - Hue$: גוון הצבע, היקף החרוט ־ כל נקודה בהיקף מייצגת גוון צבע.
- $S - Saturation$: המרחק שלנו מהמרכז, מייצג את הסטורציה כמה הצבע רווי (חזק) או דהוי. (צבע לבן יהיה)
- $V - value$: הגובה, מייצג כמה הצבע בהיר לעומת כהה.

צבע כהה יהיה עם סטורציה נמוכה, וצהע בהיר יהיה עם סטורציה גבוהה. הצבע הלבן יהיה רווי מאד.



12.1.6 מרחב הצבע $CMYK$:

מרחב צבע של מדפסות, והוא שונה מכל מרחבי הצבע שראינו עד עכשיו.
במדפסת אנחנו מדפיסים על דף לבן לכן אנחנו צריכים להדפיס בצבעים שיבלעו לנו את אורכי הגל.
הצבעים שיש הם: צהוב כחול וורוד. בד"כ יש גם שחור כי במדפסת אנחנו משתמשים מלא בשחור.
למעשה במרחב זה אנחנו מחסרים צבע ולא מוסיפים צבע.

13 תמונות HDR:

מתי נשתמש ב HDR: כשיש לנו שתי סצנות עם תאורה שונה, לדוגמה אם אנחנו מצלמים מתוך חדר חשוך את החוץ המואר. אנחנו נרצה שנוכל לראות בסצנה גם את החדר מבלי שהאובייקטים שמחוץ לחדר יישרפו, וגם שנוכל לצלם את הבחוץ מבלי שהחדר ייראה חשוך מידיי. כלומר נרצה לעשות אדפטציה לסצנה.

הגדרה ־ טווח דינמי ־ Dynamic range: הטווח בין הערך המינימלי למקסימלי. עד עכשיו דיברנו על טווח של 0־255
.בנוסף מספר הביטי שהשתמשנו כדי לתאר תמונה היה 8 ביטים.
אך אנחנו יכולים ליצור תמונות עם טווח יותר גדול, ועם יותר ביטים.
תמונות עם טווח גדול יותר נקראות HDR, בעוד תמונות עם טווח רגיל נקראות LDR.
בכדי להציג תמונת HDR אנחנו צריכים שהמסך יתמוך בהצגה הזאת ־ יתמכו ב 16 ביט או יותר. **לחלופין** קיימת אופציה

לצמצם את הטווח ולהציג את התמונה במסך רגיל.
דרכים ליצירת תמונות HDR:

1. נצלם מלא תמונות LDR עם חשיפה שונה, לאחר מכן נשלב אותן לתמונת HDR.
2. מצלמה עם חיישן שיכול לצלם בטווח גדול יותר והוא מצלם ב HDR.

כיצד נאחד את התמונות:

1. אפשרות ראשונה היא להדביק את הסצנות הטובות מכל תמונה ע"י הדבקת פירמידות של התמונות.
חסרונות: זה לא נראה כ"כ טוב, משום שהתמונה לא נראית מציאותית.

13.1 איך נצמצם את הטווח חזרה ל LDR:

13.1.1 שיטות גלובאליות:

נוכל לעשות $clip$ לטווח 0-255, או שיווי הסטוגרמה. אך זה ייראה מאוד לא טוב, התמונה תהיה שרופה.
זה לא עובד כי: יש אזורים שונים שצריך להתייחס אליהם אחרת, אם נעשה אגוריתם אחד שעושה את אותה הפעולה על כל התמונה אנחנו נהפוך פרטים מסויימים בתמונה ללא מציאותיים.

13.1.2 שיטות לוקליות:

הרעיון: ניצור את תמונת הגרדיאנטים, וכשיהיה לנו שינוי תאורה בין הבפנים לבחוץ, יהיה לנו גרדיאנט גבוה מאד. כך נוכ לזהות את שני החלקים השונים בתמונה.

מימוש:

1. נפעיל לוג על התמונה, כי כך אנחנו רואים את שינויי התאורה בעולם. בנוסף יותר קל לחפש גרדיאנטים על לוג.
2. לאחר מכן נייצר גרדיאנטים חדשים.
3. נשחזר את התמונה מהגרדיאנטים שייצרנו.
4. נעשה על התמונה exp (כדי לבטל את הלוג).
5. לבסוף נקבל את התמונה החדשה.

איך ניצור גרדיאנטים חדשים: נחליף את הגרדיאנטים הקיימים, ע"י הכפלה במטריצה (Attenuation map) שתעלה או תוריד ערך של כל פיקסל.
הרעיון במפה: נרצה להוריד את הגרדיאנטים הגבוהים, ולהגביה את הנמוכים. נעשה זאת כך:

$$\varphi_k(x,y) = \frac{\alpha}{\|\nabla H_k(x,y)\|}\left(\frac{\|\nabla H_k(x,y)\|}{\alpha}\right)^{\beta} = \frac{\alpha^{1-\beta}}{\|\nabla H_k(x,y)\|^{1-\beta}}$$

עבור:

$$\beta = [0.8, 0.9]$$
$$\alpha = 0.1 \cdot \text{ mean } (\|\nabla H_k(x,y)\|)$$

כך שאם יש גרדיאנט שגדול מהממוצע נכפיל אותו בערך שקטן מ 1 ונוריד את ערכו. ואם יש גרדיאנט שקטן מהממוצע נכפיל אותו בערך שגדול מ 1 ונגדיל אותם.

13.1.3 שיטות מבוססות רשתות:

הבעיה עם רשתות היא שאין לנו ground-truth. אנחנו לא יודעים לאיזו תמונה להשוות.
לכן נחלק לשתי גישות:

1. **גישה ראשונה תהיה** $unsupervise$**:** הרשת תקבל תמונה, ותוציא תמונה בטווח $0 - 255$.
פונקציית ה $Loss$ **תהיה:** $perceptual\ loss$ נשתמש ברשת vgg מאומנת, שתקבל את שתי התמונות ותשמור על הפיצ'רים של התמונה המקורית.

2. **גישה** $supervise$**:**
שיטה ראשונה: ניצור את תמונת המטרה שלנו ־ תמונת LDR ע"י האלגוריתמים הקלאסים שלמדנו. לאחר מכן נתייג את התמונה הטובה ביותר. לאחר מכן נאמן את הרשת על התמונות המתוייגות.
שיטה נוספת היא: ליצור תמונת LDR בעזרת פוטושופ שתהיה התיוג של התמונה המקורית.
חסרון: אם אנחנו מתייגים לפי האלגוריתמים הקלאסים, הכי טוב שהרשת תצליח היא כמו האלגוריתמים הקלאסים, כי זה מה שהיא מכירה.
נשתמש בפונקציית $l_2 - Loss$.
נוכל להוסיף דיסקרימינטור כמו במודל גנרטיבי שינסה להבחין בין התמונות וישפר את הרשת.

14 תמונות בינאריות:

הגדרה: תמונה בינארית היא תונה המכילה שני ערכים בלבד, כשאחד מייצג את הלבן, ואחד את השחור.
ייצוג: ניתן לייצג באמצעות מטריצה, או באמצעות מערך ששומר את המקומות של הפיקסלים הלבנים, או מערכך ששומר רצפים לבנים ושחורים.

14.1 סוגי תמונות בינאריות:

- **סגמנטציה:** תמונה שחורה עם רקע לבן או להיפך.
- **תמונה עם דרגת אפור שמודפסת** היא גם תמונה בינארית. כדי ליצג תמונות דרגות אפור אנחנו נייצג את הצבע עם גודל הנקודה השחורה, או משחק עם צפיפות הנקודות.

מעבר מתמונת דרגות אפור לשחור לבן: כשנרצה לעבור מתמונת דרגת אפור לתמונה בינארי עם אובייקט שחור על רקע לבן. נבחר סף, ונגדיר שכל פיקסל מעליו יהיה לבן, וכל פיקסל מתחתיו יהיה שחור.
כיצד נבחר את הסף:

- **נחשב הסטוגרמה** ונקח את העמק הכי נמוך בהסטוגרמה ־ לא תמיד עובד.

- **נשתמש בגרדיאנטים.**
שרך ראשונה: נחשב את הנגזרות, ונקח את הפיקסלים עם הנגזרות הגדולות ביותר (ה $edges$), נבדוק מה דרגת האפור שלהן, וזה יהיה הסף.
דרך נוספת: נעשה סריקה של כל הספים האפשריים (עבור כל דרגת אפור), ונבדוק עבור כל סף אם הגבול שלו מתאים למקומות שיש נגזרות גדולות בתמונה. לבסוף נקח את הסף שמתאים הכי טוב לנגזרות.

14.2 Segmentation:

כשנרצה להפוך תמונות דרגות אפור או תמונות תבעוניות לתמונות סגמנטציה. נוכל לעשות זאת בכמה דרכים.
עבודה עם בקטריות: נתונה תמונה של בקטריות ורוצים לספור כמה בקטריות יש בתמונה, לכן נעבור לסגמנטציה. אך יש בעיה נוספת והיא חורים שקיימים בתמונה, נוכל להתמודד על הבעיה ע"י זה שנצבע כל פיקסל לבן שיש לידו פיקסל שחור ־ בצבע שחור. לאחר מכן נצבע כל פיקסל שחור שיש לידו לבן ־ בלבן.
סגמנטציה לתמונה צבעונית: נוכל להשתמש ברשתות כדי לקחת תמונה של אדם על רגע צבעוני (צילום חוץ) ולצבוע רק את הדמות בשחור.
סגמנטציות נוססות צובעות אובייקטים שונים בתמונה בצבעים שונים.

14.3 קשירות בין אובייקטים:

קשירות בין אובייקטים:
נרצה לדעת האם שני פיקסלים שונים בתמונה שייכי לאותו האובייקט.
חיפוש קינקטביות: נקח פיקסל ונסתכל על השכנים שלו.

- שיטה אחת מסתכלת על השכנים שמעליו ומהצדדים (רכיב ־ 4).

- שיטה שניה מסתכלת על כל הפיקסלים מקיפים אותו (רכיב ־ 8).

הגדרות: אם שני פיקסלים באותה הקבוצה הם יקיימו את התנאים הבאים:

1. **רפלקסיביות:** כל פיקסל בקבוצה עם עצמו.

2. **סימטריה:** אם x קשור ל y, אזי גם y קשור ל x.

3. **טרנזטביות:** אם x קשור ל y, ו y קשור ל z אזי גם x קשור ל z.

כך נוכל לחשב קשירויות, נחפש מסלול בין x, y וכך נוכל לדעת אם הם קשורים.

הערה: כל מסלול שקשור ברכיב - 4 קשור גם ברכיב 8. אך ההפך לא בהכרח נכון.

איך נמצא את כל רכיבי הקשירות בתמונה:

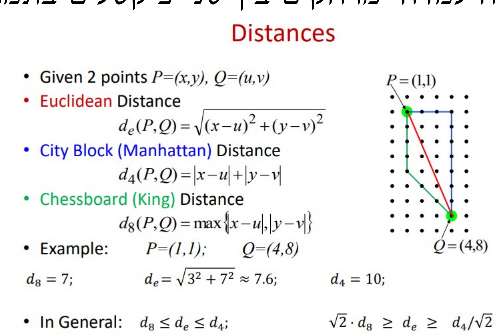
- נעבור פעם אחת על התמונה נבחר פיקסל ברכיב ונצבע אותו בצבע מסויים, אם יש פיקסלים שמתאימים לו (יש ביניהם מסלול) נצבע אותם באותו הצבע.
- אם גילינו פיקסל אחר שלא מחובר לאובייקט - נצבע אותו בצבע אחר.
- אם גילינו לבסוף שיש צבעים שכן מחוברים - נשמור בצד כי שני הצבעים שייכים לאותו אובייקט. ונמשיך בסריקה.
- אחרי שעברנו על כל התמונה, כל הפיקסלים צבועים ויש לנו מפה שאומרת איזה שני צבעים מחוברים יחד. נוכל לחבר את כל הצבעים ששייכים לאותו אובייקט יחד, בעזרת האלגוריתם $union\ find$.
- מספר הצבעים הוא מספר רכיבי הקשירות שיש לנו.

14.3.1 עקומות:

- **קונקטיביות שונה לרקע ולאובייקטים:** כשיש לנו צורה סגורה (עקומה), אנחנו לא נמדוד את רכיבי הקשירות באובייקט וברקע באותה השיטה. אלא נמדוד את הרקע ב רכיב 8 או רכיב 4, ואת האובייקט נמדוד בשיטה האחרת.
אם לא נעשה כך - נגיע לסתירה עם משפט עקומת ג'ורדן.
- **גבול של עקומה:** קבוצת כל הפיקסלים שהשכנים שלהם נמצאים ברקע.
אם יש לנו קבוצה C שהיא גבול לפי קשירות 8, נקח את הרקע לפי קשירות 4 ולהיפך.
- **Chain code:** דרך לייצוג קווים של עקומות בתמונה. נוכל לסמן את כל הכיוונים שאנחנו יכולים לזוז מפיקסל מסויים במספרים 8-0 נגד כיוון השעון. נתחיל מפיקסל מסויים, ונכתוב מערך של מספרים שמייצגים את הקו של העקומה, לפי הצעדים שלנו מהפיקסל הנוכחי לפיקסל הבא.
סיבוב ב 90 מעלות ניתן לעשות ע"י החסר או הוספת 2.

14.4 מרחקים:

נרצה למדוד מרחקים בין שני פיקסלים בתמונה, יש כמה שיטות:



הערה: שלשת המרחקים הם מטריקות - חיוביים, סימטרים ומקיימים את אש"ם.
הוכחה והפרכה של מטריקות נעשה בעזרת הוכחת או הפרכת אש"מ.

שוויון בין המרחקים: כשנמדוד מרחקים אנכיים או אופקיי יהיה שוויון בין שלשת הפונקציות למדידת מרחק.

Distance map/ Distance Transforms: נקח סט של נקודות ונחשב את המרחקים של שאר הפיקסלים בתמונה מסט הנקודות. המפה משתנה לפי הפונקציות שאיתן נחליט לחשב את המרחק.

Voronoi Diagram: יש לנו אוסף של נקודות, אנחנו רוצים להגדיר תאים מסביב לכל נקודה, כך שהתא יהיה קרוב לאותה הנקודה יותר מכל נקודה אחרת.
לכל שתי נקודות שיש לשטח בניה צלע, ניתן לחבר את הצלע ואז נקבל ייצוג של הנקודות באמצעות משולשים.
Delaunay Triangulation : חלוקה של אוסף נקודות בעזרת משולשים. כל שכל נקודה היא קודקוד. ניתן לפתור את הבעיה הזאת באמצעות Voronoi Diagram וחיבור הנקודות.

14.5 מורפולוגיה מתמטית - פיקסלים כקבוצות:

נתייחס ל

15 סיכום וידאו - Video Summarization:

למה זה חשוב: רוב הדאטה הדיגיטלי בעולם הוא ווידאו של מצלמות אבטחה. אנחנו נרצה לחלץ מידע מהוידאו באופן יעיל, כך שנוכל להוציא מידע רלוונטי באופן מהיר.

15.1 מעבר מוידאו לתמונה - Synopsis Mosaic :

ביצוע: נעבור מייצוג של פריימים, לייצוג של סצנות. נהפוך את כל הוידאו לתמונה אחת, שמייצגת את כל התזוזות של האובייקטים והאובייקטים הסטטים.

1. **אובייקטים סטטים spatial information:** תחילה ניצור מתמונות הרקע שלא השתנה תמונת פנורמה (אם המצלמה קבועה, אין צורך לייצר פנורמה אלא לקחת את הרקע כמו שהוא) סטטית. כל האובייקטים שלא זזו בסרטון, ייוצגו כפנורמה של אלמנטים סטטיים.

2. **אובייקטים שזזים temporal information:** התנועה של האובייקט נמשכך על פני כמה פריימים. אנחנו נרצה לאחד את כל התנועה לפריים אחד. לכן נחשב את המסלול שהאובייקט עשה לאורך זמן ונצייר אותו על הסצנה הסטטית (נוכל להוסיף קו שמתאר את מסלול התנועה של האובייקט).

3. **מידע גיאומטרי - Geometric information:** נשמור את הטרנספורמציה בין הפריימים.

כיצד נוריד את כל האובייקטים שזזים: אנחנו רוצים להוריד את האובייקטים הזזים כדי להכין פנורמה מהרקע. נעשה זאת ע"י זה שנתייחס לאובייקטים זזים כרעש. כך נוכל להוציא את כל האובייקטים הזזים, עם חציון או ממוצע טמפורלי (לפי זמן ההופעה שלו בסצנה). כך ננקה את האובייקטים הזזים מהרקע.

איך נוסיף את האובייקטים שזזים:
תחילה נצטרך להבין מי האובייקטים שזזים. נעשה את זה על ידי חיסור הפריימים אחד מהשני כדי להבין מי זז.
איך נחשב את המסלול של האובייקט: נחסר את הפריימים מהתמונה הסטטית, כל אובייקט שיופיע בתמונת החיסור הוא אובייקט שזז. לאחר מכן נעקוב אחרי מסלול התנועה שלו ונדביק אותו על התמונה (או שנצייר קו במקום שהוא עבר).

15.2 מעבר מוידאו לוידאו קצר יותר - Video Synopsis:

רעיון נוסף הוא לצמצם את תנועת האובייקטים בסרט הארוך, לסרט קצר שמכיל את תנועת האובייקטים.

הנחות: דינמיקה מול סדר כרונולוגי -
אנחנו נשמור על הדינמיקה, כלומר אובייקט שזז מהר בסרט המקורי, יזוז מהר גם בסרט הקצר. אך אם שני אנשים שונים היו במקום אחד בזמנים שונים, אנחנו נציג בסיכום כאילו הם היו באותו הזמן, והיו באותה סצנה יחד.

הביצוע:

1. נזהה את האובייקטים שזזים בוידאו $(tubes)$, ונשמור את המידע עליהם בדאטה סט מסויים. על כל אובייקט שזז נשמור מאיפה לאיפה הוא זז ומתי - באיזה פרימים הוא הופיע ומה הוא עשה בסצנה.

2. שלב השאילתות: נבקש תמצית באורך זמן מסויים, ואנחנו נדחוס את האינפורמציה ביחס לזמן שבוקש בשאילתה.

דרישות: נתייחס את הקלט שלנו כווליום תלת מימדי $I = (x, y, t)$ (כאשר t מייצג את הזמן בסרטון), ואנחנו נוציא ווידאו חדש - $S = (x, y, t)$ ונדרוש ש -

- הוא יהיה קצר ככל הניתן (מבלי שאובייקטים יעלו אחד על השני).
- מקסימום מהדינמיקה \ פעילות מהוידאו המקורי תיכנס ל S.
- אנחנו לא נריץ את הסרטון, כדי שהדינמיקה תישמר.
- נדרוש שהתפרים בין הפריימים ייראו טוב. ששינויי תאורה לא יפריעו.

מימוש בפועל:

- **כיצד נזהה אובייקטים זזים:** נניח כי האובייקטים המעניינים הם האובייקטים שזזים. (זה לא בהכרח נכון, כי יכול להיות אדם סטטי שיעניין אותנו, או עץ שזז ברוח שלא יעניין אותנו).
אנחנו נתאר את האובייקטים שזזים בתוא $tubes$ - צינורות, בתוך הווליום התלת מימדי. ונשמור דאטה בייס של כל ה $tubes$ ואחכ נסדר אותם יחד בסצנה.

- **תחילה ניצור רקע** - הרקע שניצור יהיה רקע משתנה לאורך זמן, כלומר ניצור רקע שהוא בעצמו סרטון באורך הסרטון המקורי. כדי למצוא את הרקע האמיתי אנחנו נצטרך לחשב את הפיקסל ביחס לסביבה של,ו כדי להתגבר על שינויי תאורה. כלומר - בגלל שיש לנו סרטון לאורך היום הפיקסל של הרקע משתנה במהלך היום בגלל שינויי תאורה, ואנחנו צריכים להבין כי הפיקסל הוא אותו פיקסל, למרות שצבעו השתנה. לכן אנחנו נקח טווח של 10 דקות לפני ואחרי הפריים, ונגדיר את צבע הפיקסל לפי ההיסטוגרמה הטמפורלית.

הגדרה - היסטוגרמה טמפורלית: נקח פיקסל מסויים לפי זמן, ונסתכל על ההיסטוגרמת הצבעים שלו לאורך הזמן. נגדיר את הצבע שלו להיות הצבע עם ה bin הכי גבוה בהיסטוגרמה.

- **איך ניצור את ה** $tubes$ **:** לכל פריים בוידאו נמצא פונקציה f שמחזירה 1 לכל פיקסל בפריים שהיא חושבת שהוא אובייקט, ו 0 אחרת.
- נעשה זאת באמצעות מזעור השגיאה הבאה:

$$E\left(f_t\right)=\sum_{r \in I_t} E_1\left(f_t(r)\right)+\lambda \sum_{r \in I_t, s \in N(r)} E_2\left(f_t(r), f_t(s)\right)$$

עבור E_1 - data term: פיקסלים ששונים מהרקע - f תחזיר עבורם 1 , ועבור הדומים היא תחזיר 0.
עבור E_2 - smoothness term: אם יש שני פיקסלים צמודים עם צבע דומה - הם שייכים לאותו מקום (או שניהם רקע או שניהם אובייקט).
כאשר r הוא הפיקסל שאנחנו בודקים, ו s היא הסביבה של r.

למעשה מה שנקבל זה מין $mask$ עבור כל פריים, כך שהמקומות של האובייקטים יסומנו ב 1 - $blob$.
ניצור את ה $tube$ על ידי זה שנקח שני $blobs$ סמוכים שכל אחד מהם מייצג אובייקט, וניצור מהם רכיב קשירות אחד.
ייצוג של ה $tube$: נייצג אותו ע"י פריים התחלה ופריים סוף. בנוסף נשמור פונקציה שתאפיין אותו - characteristic function והיא תעזור לנו להבין את האינטרקציה בין $tubes$ שונים.

$$\mathcal{X}_b(x, y, t)= \begin{cases}\|I(x, y, t)-B(x, y, t)\|, & t \in t_b \\ 0, & o.w.\end{cases}$$

- **יצירת הוידאו:** ניצור אינטרקציה בין ה $tubes$ השונים. נלך לכל $tube$ ונפעיל על הזמן שלהם פונקציה M שתתן עבורם זמן חדש.

  הפונקציה M תוגדר ע"י בעיית אופטימיזיה של הכנסת כמה שיותר אלמנטים, וכמה שפחות התנגשויות.
  בדיקת התנגשויות: נכפיל את שני הפונקציות של ה $tubes$:

$$E_c\left(\hat{b}, \widehat{b'}\right)=\sum_{\substack{x, y \\ t \in \hat{t}_b \cap \hat{t}_{b'}}} x_{\hat{b}}(x, y, t) \mathcal{X}_{\widehat{b'}}(x, y, t)$$

כך שאם אין חפיפה בזמנים החדשים - המכפלה שלהם תהיה 0. אם הם ממש חופפים הערך יהיה גבוה.
כאשר $\hat{t}_b \cap \hat{t}_{b'}$ הוא הזמן בסרטון החדש ששניהם מופיעים בסרטון.
בנוסף, יש לנו פרמטר שמגדיר כמה אנחנו מאפשרים התנגשויות. ככל שיש יותר התנגשויות הוישאו יותר קצר.

התמודדות עם אובייקטים שביצעו תנועה ארוכה: אנחנו לא נרצה להגביל מלמטה את אורך הוידאו לפי האובייקט שזז הכי הרבה זמן בסרטון. לכן אם יש לנו אובייקט שזז הרבה זמן בסרטון, אנחנו נחתוך את התנועה שלו, גם אם זה לא נכון מבחינת הדינמיקה האמיתי של התנועה שלו במהלך הסרטון המקורי.

15.3 איחוד תנועות - Clustered Synopsis:

הרעיון: נקבץ את כל האובייקטים הדומים. דומים וויזואלית, או דומים מכיוון שהם שזזים באותו כיוון תנועה, ונציג אותן יחד.

בדיקת דמיון ויזואלי: בודקים דמיון בין פיצ'רים של אובייקטים (לדוגמה - נציג את כל האנשים, ואחכ את כל הרכבים).

$$S_{d_{ij}} = \frac{1}{2N}\left(\sum_k \left|S_k^i - \tilde{S}_k^j\right| + \sum_k \left|S_k^j - \tilde{S}_k^i\right|\right)$$

בדיקת דמיון של תנועה: עבור שני $tubes$ נבדוק אם אחד מהם הוא הזזה של השני, והם מתלכדים לאחר הפעלת ההזזה. (לדוגמה - נציג את כל המכוניות שנוסעות ימינה, ואחכ את כל המכוניות שזזות שמאלה)

16 סופר רזולוציה:

הרעיון: יש לנו תמונה קטנה ברזולוציה טובה, ואנחנו רוצים להגדיל את התמונה ולהעלות את הרזולוציה שלה.

דרישות:

1. כשנקטין את התמונה נקבל את התמונה המקורית ברוזולוציה טובה.
2. ההגדלה תראה טוב ברזולוציה גבוה.

שלבים: תחילה נגדיל את התמונה, אך לאחר ההגדלה אנחנו נקבל תמונה מטושטשת. נרצה להוסיף את התדרים הגבוהים כדי לחדד את התמונה.
נעבור על פאצ'ים מטושטשים בתמונה המוגדלת, ונשפר אותם לפאצ'ים חדים.

כיצד נחדד את הפאצ'ים:

16.1 באמצעות רשתות:

באמצעות למידה עמוקה, נסתכל על הרבה תמונות העולם, ונויא מהן פאצ'ים מחודדים ומטושטשים. כך נכין ספריה עם זוגות מתאימים של מטושטש x_i ומחודד y_i, הרשת תלמד את הפיצ'רים של הפאצ'ים המחודדים והמטושטשים והיא תלמד

כיצד להחליף את המטושטש בחד.

מימוש: נקטין את התמונה ונגדיל אותה בחזרה, ולאחר מכן נחסר אותה מהתמונה המקורית, כך נקבל את התדרים שנאבדים לנו בהחסרה. ואלו יהיו הפאצ'ים של התדרים הגבוהים ־ y_i.
בנוסף, נחפש פיצ'רים שהם $edges$ אנכיים אופקיים (נגזרות ־ לפלסיאן) סה"כ 4 מספרים ־ x_i. אח"כ נכין ספריה כך שכל פעם שיהיה לנו פאצ' עם אותן הנגזרות (x_i) נחליף אותן עם הפאצ' המחודד (y_i).

ייעול:

- במקום לעבור על כל הפאצ'ים בתמונה, נעבר רק על אלו שמכלים פינות ־ אלו שליש להם תדר גבוה.
- ננרמל ניגודיות.

16.1.1 אימון רשת על תמונה אחת:

ניתן לאמן רשת על תמונה אחת ע"י זה שנאמן את הרשת על פאצ'ים. לכל תמונה יש מלא פאצ'ים לכן יהיה לנו מלא מידע. בנוסף ניתן לסובב את התמונה או לשנות את הצבע שלה. כך יידמה לרשת שיש מלא דאטה שונה.

16.2 שיטה נוספת ־ ללא רשתות נוירונים:

נקח את התמונה המקורית ונוציא ממנה שתי תמונות ־ אחת של התדרים הגבוהים ואחת של הנמוכים.
לאחר מכן נלך לתמונה המוגדלת ונוציא ממנה פאצ' שאנחנו רוצים לחדד. נחפש פאצ' מתאים לו התמונה המקורית המטושטשת (אנחנו יודעים איפה לחפש כי זה באותה סביבה פחות או יותר), לאחר מכן נמצא את הפאצ' המתאים בתמונת התדרים הגבוהים (של התמונה המקורית).
ובפאצ'ים האלו נשתמש כדי לחדד את התמונה הגדולה.

ייתרון: יותר מהיר כי אנחנו מסתכלים על פחות פאצ'ים (רק בסביבה של הפאצ' שאנחנו רוצים לשפר).
הרעיון: אם אנחנו מסתכלים על סביבה קטנה של $edge$, להקטין ולעשות $crop$ זה אותו הדבר, לכן הפאצ'ים מהתמונה המקורית יעזרו לנו לשפר את התמונה הגדולה (התדר הגבוה מהתמונה הקטנה יהיה אותו תדר גבוה גם להתמונה המוגדלת), יותר מאשר אם נשתמש בספריה חיצונית.

16.3 סופר רזולוציה בהינתן מספר תמונות קלט:

מה נרצה: יש לנו מספר תמונות ברזולוציה נמוכה של אותה הסצנה, בשהמצלמה בתזוזה. אנחנו רוצים להפיק מהן תמונה אחת ברזולוציה גבוהה.

הרעיון: יש לנו כמה תמונות כך שכל אחת מצלמת מזווית אחרת, אם נחבר בין כל שני חצאי פיקסלים של התמונה ה i את חצאי הפיקסלים מהתמונה ה $i+1$ אנחנו נצליח למלא את המקומות המתאימים כדי להשיג רזולוציה גבוהה יותר. הבעיה

היא כי התמונות הן לא בהכרח ההזזה של חצאי פיקסלים אחת של השניה, אלא סיבוב או הזזה של יותר או פחות.

מימוש: נתמודד עם הטרנזופרמציות שיש לנו, נשים לב כי יש לנו טשטוש שונה בין פיקסלים באותה התמונה ובין תמונות. בנוסף יש לנו רעש.

העברת תמונה גדולה ברזולוציה גבוה לתמונה קטנה ברזולוציה נמוכה:

1. נצטרך למצוא את הטרנספורמציה - F_k
2. לטשטש - H_k.
3. נדגום כל פיקסל שני (אם מקטינים בפקטור 2) - D_k.

לאחר התהליך הזה נקבל את התמונה המוקטנ עם הרעש. כלומר מתקיים:

$$LR = D_k H_k F_k \cdot HR$$

לכן, אם נוכל למצוא את הטרנספורמציה, הטשטוש וההגדלה נוכל להגיע מתמונה מוקטנת ומטושטשת לתמונה גדולה ובהירה. נניח כי יש לנו את פקטור הטשטוש ואת הטרנספורמציה, אנו יודעים מהי ההגדלה שאנחנו מחפשים, לכן נוכל לחשב את HR.

נרצה למצוא את X שממזער לנו את המשוואה הבאה:

$$E(\underline{X}) = \sum_{i=1}^{N} \|P_i(\underline{X}) - y[i]\|^2$$

עבור N תמונות קטנות, כאשר $P_i(\underline{X})$ הוא ההטלה של X ל $y[i]$.

- **איך נמצא את ההזזה:** נבחר אחת מתמונות ה y שרירותית, ונקח אותה להיות תמונת ציר, ונראה מה ההזזה של כל התמונות אליה. את הטרנספורמציות נייצג באמצעות מטריצות.
- **איך נמצא את הטשטוש:** אם הטשטוש אחיד, נוכל למצוא את הטשטוש בעזרת קונבולוציה, שגם אותה ניתן לייצג באמצעות כפל מטריצה.
- **איך נמצא את הדגימה:** אם אנחנו דוגמים כל מסויים כדי להקטין, אנחנו יכולים לייצג זאת באמצעות כפל מטריצה גם כן.

כעת, נניח כי מטריצות הטשטוש והדגימה של כל התמונות שוות, כי הן צולמו כולן באותה המצלמה.

נוכל לחשב את הנוסחה הבאה ולמצוא את X:

$$\left\{\underline{Y}_k = \mathbf{D}_k \mathbf{H}_k \mathbf{F}_k \underline{X} + \underline{V}_k, \underline{V}_k \sim \mathbf{N}\left\{0, \sigma_n^2\right\}\right\}_{k=1}^{N}$$

יש לנו את כל הפרמטרים במטריצות מגודל $n^2 \times n^2$.

חסרונות:

- **טשטוש:** פעול הטשטוש היא לא הפיכה, כי יש אינסוף פתרונות למשוואת הטשטוש, על אף שיש לנו מלא תמונות מהן ניתן ללמוד את הטשטוש.

דרכים נוספות לפתרון:

1. נחפש את X כך שאם נסמלץ את ההקטנה, נקבל את ההפרש הקטן ביותר בין הסימולציה לתמונה הקטנה המקורית.

$$\underline{X} = \arg\min_{\underline{X}} \sum_{k=1}^{N} \|\mathbf{DHF}_k \underline{X} - \underline{Y}_k\|^2$$

2. נוסיף עונש על תדרים שלא אמורים להיות בתמונה המקורית. נוכל להגדיר את $A\{\underline{X}\}$ להיות הגרדיאנט.

$$\underline{X} = \arg\min_{\underline{X}} \sum_{k=1}^{N} \|\mathbf{DHFF}_k \underline{X} - \underline{Y}_k\|^2 + \lambda A\{\underline{X}\}$$

איך נפתור את מערכת המשוואות - Back Projection:

1. ננחש את התמונה הגדולה, נעשה סימולציה לתמונה הקטנה ונקבל את ההפרש.
2. לאחר מכן נתקן את X כדי שהשגיאה תהיה קטנה.
לדוגמה - אם פיקבל מסויים הגיע מ 10 פיקסלים, והשגיאה שלו הייתה 10, אזי וריד לכל אחד מהפיקסלים המקוריים 1.
3. נחזור על השלבים עד שהשגיאה תקטן, והאלגוריתם יתכנס.

חסרון: צריך להתאים את ההזזה בין מלא תמונות, יש מלא מקום לטעויות.

פתרון נוסף: במקום למזער את השגיאה בריבוע, נמזער את הערך המוחלט של השגיאה ונקח את החציון במקום את הממוצע.

16.4 סופר רזולוציה לתמונות צבעוניות:

נמיר את התמונה לייצוג אחר, ונעבוד על הערוץ של ה $intencity$. הוא הערוץ שמשנה ולא הערוץ של הצבע.
לאחר מכן נתקן את הסטטיסטיקות של הצבע.

16.5 תמונות עם הזזה:

אם יש לנו תמונות שאנחנו יודעים בוודאות שהן הזזה אחת של השניה, מספיק לחשב הזזה וטשטוש.
תהליך לא איטרטיבי - מציאת ההזזה:

1. עבור L_2: נגדיל את כל התמונות הקטנות, נעשה להן רגיסטרציה (נזיז אותן למיקום המדוייק אחת לקראת השניה) לפי התנועה שחישבנו, ונמצע אותן.
2. אם אנחנו רוצים למזער את L_1: נעשה אליימנט, רגיסטרציה, ונקח חציון.

תהיך איטרטיבי להוצאת הטשטוש:

1. מזעור הטשטוש נעשה באופן הבא:

$$\varepsilon^2(\underline{X}) = \|\mathbf{H}\underline{X} - \underline{Z}\|_1 + \lambda A\{\underline{X}\}$$

$$\underline{\hat{X}}_{n+1} = \underline{X}_n - \beta\left\{\mathbf{H}^T \operatorname{sign}\left(\mathbf{H}\underline{\hat{X}}_n - \underline{Z}_k\right) + \lambda\frac{\partial}{\partial\underline{\hat{X}}_n}A\left\{\underline{\hat{X}}_n\right\}\right\}$$