

עיבוד תמונה

מrzah

פרופ' שמואל פלאג

מתרגלת | אביטל שפרן

דוויד קיסר שמידט

דניאל דיז'ב

עיבוד ספרתי של תמונות | 67829

כתב ע"י דוד קיסר-شمידט ודניאל דיצ'ב

12 בינואר 2022

מרצים | פרופסור שמואל פלאג

מתרגלת | אביטל שפרן



מצאתם שגיאה? ספרו לנו! שלחו לנו מייל לאחת מהכתובות שכאן:

daniel.deychev@mail.huji.ac.il

david.keisarschm@mail.huji.ac.il

© כל הזכויות שמורות לדוד קיסר-شمידט ודניאל דיצ'ב

لمורות שכן לנו באנט ביווית ואנו להתייחס לכיתוב בשורה הקודמת ברצינות

תוכן העניינים

13	I הקדמה
15	II ייצור תמונות
15	1 העין האנושית
16	2 קליטת תמונה דיגיטלית
20	III היסטוגרמה
20	3 שיווי היסטוגרמה
23	IV התמרת פורייה
23	4 מבוא
24	4.1 מספרים מרוכבים
24	4.2 קצר על תדרות
26	DFT - טרנספורמציה פורייה דיסקרטית 5
28	5.1 תוכנות
29	5.2 גלים לא סטציונרים Short Time FT - STFT
32	5.3 ספקטוגרמה
34	5.3.1 יישומים
35	V התמרת פורייה דו מימדית - שימוש בתמונות
41	6 הזזה של תמונה וההתמרת פורייה

43	7	7 תכונת הפירוק של פוריה
47	7.1	נגזרת של תמונה
48	7.2	שינויי טרנספורם פוריה לאחר טרנספורמציה על התמונה
51	7.3	Aliasing
54	VII	קונבולוציה
55	7.4	משפט הקונבולוציה
56	7.5	קונבולוציה דו מימדית
57	7.6	נגזרת באמצעות קונבולוציה
59	7.7	<i>Edges</i>
60	7.8	פלסיאן
61	7.9	חידוד באמצעות פלסיאן
63	VIII	פירמידות תמונה
65	8	פרמידה גאוסיינית
66	9	פרמידת פלסיאן
70	10	דחיסת פרמידה
70	11	תפירת תמונות
75	VIII	ניקוי רעים
76	11.1	פרמטר טשטוש גאוסייני - <i>Bilateral – Filtering</i>
80	11.2	<i>patch – methods</i>
82	11.3	<i>Google – Night – Sight</i>
84	IX	טרנספורמציות על תמונות

85	12	позוזות של תמונות
85	12.1	<i>Rolling – Shutter</i> אפקט
87	12.2	<i>Direct</i> שיטות
89	12.3	Lukas-Kanade
99	X	זרימה אופטית - Optical-Flow
105	13	פסיפס - ריצוף תמונות (Mosaicing)
112	13.1	ריצוף תמונות עם מספר נקודות מבט
115	13.2	פסיפס דינמי
119	XI	למידת מכונה
119	14	רשתות נוירוניים
127	15	רשתות קונבולוציה
131	15.1	<i>Pooling</i>
134	15.2	<i>Epoch</i>
134	15.3	Overfitting
137	XII	גאומטריה
137	15.4	Perspective Projection
139	15.5	TCC – Time To Collision
140	15.6	גאומטריה אפיפולרית - Epipolar Geometry
143	15.7	המטריצה הפנדנטלית - Fundamental Matrix
144	15.8	Depth From Stereo
146	15.9	Block Matching \approx Optical Flow
147	15.10	אלטרנטיבת לחישוב עומק - Kinect

148	נקודות נעלמות - Vanishing Points -	15.11
150	רמזים נוספים לעומק	15.12
152	XIII דחיסת תМОנות	
152	(Huffman Code)	16 קוד האפמן
155	Entropy – Encoding	16.1 דחיסת אנטרופיה –
160	Lempel – Ziv Compression(LZW)	17
162	Run Length Encoding	18
163	19 מערכות דחיסה	
164	Predictive Encoding	20
165	Pyramid Compression	21
167	Transform Compression	22
167	התמרת קוסינוס - DCT	22.1
170	JPEG	22.2 דחיסת
172	wavelets	22.3
178	XIV דחיסת סרטוניים	
179	Motion Vectors	23 וקטורי תנועה –
181	MPEG-23.1	הרחבות ל-
182	XV תМОנות בינהוית	
182	23.1.1 שימושים	شمונת

185	24 קשירות - Connectivity
188	גבולות 24.1
191	מרחקים 24.2
194	טרנספורמציה מרחק 24.3
195	דיאגרמת Voronoi - טריאנגולציה Delaunay 24.4
196	תכונות של אובייקטים ביןאריים 24.5
198	מורפולוגיה מתמטית 24.6
198	Structuring Element 24.7
206	מקסימום לוקלי בטרנספורמציה מרחק - Skeleton – Medial Axis 24.8
207	מציאת פינות 24.9
208	XVI נושאים נוספים
208	25 שאלות
208	26 העשרה
208	чисוב התמרת פוריה דו-מימדית עם כפל מטריצות 26.1
211	קואורדינטות הומוגניות 26.2
214	משפט ריאלי ועקרון המקסימום לערכים עצמיים (בקשר של אלגוריתם האריש) 26.3
217	27 סיכום בניית הפנורמה
218	28 סיכום קצר LukasKanade
220	סיכום קצר על פנורמות המורכבות משתמשי נקודות מבט זמניים שונים 29
221	XVII תרגולים
221	30 תרגול 1 - טרנספורמציות על תמונות, שיווי היסטוגרמה וקוונטיזציה

227	31 תרגול 2 - גלים, התמרת פוריה, SIFT וספקטrogramה
238	32 תרגול 3 - טרנספורם פוריה דו מימדי, נגזרת של תמונה, פילטרים
242	... נגזרת של תמונה 32.1
243	... פילטרים 32.2
243	... Low Pass 32.2.1
244	... High Pass 32.2.2
245	... Band Pass 32.2.3
245	... Gaussian 32.2.4
247	33 תרגול 4 - קונבולוציות, נגזרות, חידוד, לפלייאן ו-Edge Detection
262	34 תרגול 5 - פירמידות וטרנספורמציות 2D
278	35 תרגול 6 - ומציאות Warping Feature Points
292	36 תרגול 7 - תיאור הסביבה של נקודה - Descriptors
292	... Feature-Descriptors 36.1
293	... אלגוריתם Multi-Scale-Oriented-Patches - MOPS 36.2
296	... אלגוריתם SIFT 36.3
299	... אלגוריתם RANSAC 36.4
306	37 תרגול 8 - תפירה באמצעות Min-Cut-Blending, תכנון דינامي ו-Blending
307	... שיטה נאייבית 37.1
307	... תפירה עם Alpha/Pyramid-Blending 37.2
307	... Alpha-Blending 37.2.1
309	... Pyramid-Blending 37.2.2
311	... מציאת תפיר אופטימלי 37.3

316	38 תרגול 9 - Hough-Transform וצבע
316	Hough-Transform 38.1
322	מרחבי צבע - Spaces Color - 38.2
331	39 תרגול 10 - למידת מכונה ורשתות נוירונים
340	40 תרגול 11 - סוגים שונים של רשתות נוירונים
356	41 תרגול 12 - High Dynamic Range (HDR) - 12
365	42 תרגול 13 - סיכום וידאו (Video Summarization)
365	סיכום הווידאו לתמונה אחת - 42.1
368	סיכום הווידאו לוידאו קצר - 42.2
373	Clustered Synopsis Of Surveillance Video 42.3

רשימת אלגוריתמים

21	השוואה היסטוגרמית	1
22	השוואה היסטוגרמית מלאה	2
31	<i>STFT</i>	3
40	הצנת טרנספורם פורייה כתמונה	4
44	חישוב טרנספורם פורייה דו מימדי באמצעות הגרסה החוד מימדית	5
47	חישוב נגזרת של תמונה	6
52	הקטנת סיגנל	7
52	הקטנת תמונה	8
53	הגדלת תמונה	9
70	דוחיסת פרמידה	10
72	תפירת לפלייאן	11
73	תפירת לפלייאן	12
81	Non-Local-Means-(NLM)	13
91	LK איטרטיבי (מעבר v, u גדולים)	14
93	LK עם פרמידות (שימוש כשהנגורות בשתי התמונות לא דומות)	15
104	חישוב זרימה אופטית באמצעות פרמידות	16
105	חישוב זרימה אופטית באמצעות פרמידות עם LK	17
125	Stochastic Gradient Descent	18
134	Mini – Batch – Stochastic – Gradient – Descent	19
153	Huffman Encoding	20
161	Lempel – Ziv Compression(LZW)	21
161	Lempel – Ziv DeCompression(LZW)	22
187	One Pass Scanning For Connected Components	23
225	השוואה היסטוגרמית	24
260	Detection Edge Canny	25
262	הקטנה של תמונה (<i>Reduce</i>)	26
264	הגדלה של תמונה (<i>Expand</i>)	27
288	Harris Detector	28
317	Hough-Transform	29
332	Stochastic Gradient Descent	30
359	Gradient Domain HDR Compression	31

361	CNN Training Using Perceptual Loss	32
363	Generic Pipeline For Computer Vision Tasks	33

סיכום זה מוקדש לדודניאל.

עיבוד ספרתי של תמונות! מלא בשמחה, אהבה וגלידה ועומד להיות לנו كيف איז תתוכנו D:

יש לנו הרבה נושאים ביניהם: כל מיני! Lets jump right into it!

הקדמה

הרצאה 1

מהו עיבוד תМОונות? הכל מתחילה מצלמה שמסתכלת על אובייקט ויוצרת תМОונה. היום, אנחנו משתמשים בצלמות בטלפון שלנו, אך כמוות התמונות הגדולה ביותר שנוצרת, היא דוקא על ידי מצלמות אבטחה, שצלמות 7\24.

עיבוד תМОונות היום כולל גם עיבוד קול מתוך התמונה, שימוש באינטיליגנציה מלאכותית עבור רכבים אוטונומיים ועוד. נשאלת השאלה, לאיזה תחום נכנס עיבוד ספרתי של תМОונות? אלגוריתמים? גרפייה? ארכיטקטורת מחשב? רשותות? כל נושאים אלה הם נושאים קלאסיים במדעי המחשב בהם עסקנו עד סוף השנה ב. היום יש נושאים חדשים, כמו ניתוח דיבור, עיבוד שפה טבעיות, ראייה ממוחשבת, למידה عمוקה, אלה נושאים חדשים שקשורים ל-AI. הקורס שלנו נכנס לחلك של AI, וזה גם התחום העיקרי היום בהייטק.

ניתוח מידע

כאנחנו רואים תМОונות שונות, אנו נוטים למצוא את המשותף, למשל, כאשר נראה תМОונה שלנו לפני עשר שנים ותМОונה בעשוייה, נדע מיד שמדובר באותו בן אדם. יחד עם זאת, לא ברור איך המחשב יוכל להזאת זאת. אין זו בעיה רק בתחום זה, למשל, המשפט "התרגול מוכן לאכילה" בעל שתי משמעות - התרגול הוכן בתנור וعصיו אפשר לאכול אותו, או שהתרגול מוכן לאכול זרעים. כיצד המחשב יוכל לקבוע מה המשמעות של המשפט? באמצעות למידה عمוקה, יוכל לתת מענה לשתי בעיות אלו.

אחד התכונות הבסיסיות בניתוח מידע היא האפשרות לראות את המידע. למשל, חוות חכמות הן היו שבסוגות לראות; חידקים אינם חכמים, אך טורפים כן ועיניהם מקדימה כדי שיוכלו לראות את הטרפ, וטרפים לעומת זאת בעלי עיניים בצדדים כדי שיוכלו לראות את הטרפ. נרצה לאפשר למחשב לראות את הטרפ שלו.

יש שימושים רבים לעיבוד תМОונה, למשל, ניתן לקחת תМОונה מלאה בערפל ולהבהיר אותה. אפשר ל凱ר סרטוני מידע על ידי הצגת אלמנטים מסוימים באותו הזמן¹. מעבר לכך, זיהוי אובייקטים בראייה ממוחשבת, *Mobileye* למשל, זיהוי פנים, *Apple – faceId*.

¹ דבר זה למעשה פותח על ידי המרצה וסטודנטים מהקורס בעבר. לחברת קוראים *briefCam*

צעדים להמשך

קורסים דומים לקורס שלנו הם:

- גרפיקה ממוחשבת.
- עיבוד תМОנות רפואי.
- מבוא לרשומות נוירונים.

בسمスター הבא כצעד הבא המשך אפשרי:

- מבוא ללמידה מכונה.
- קורס ההמשך: ראייה ממוחשבת - גאומטריה של תМОות. למידה דו מימדית והפיכתה לתלת מימדית.
- רשתות נוירוניים לתМОות.
- שימוש בלמידה عمוקה - קורס חשוב למי שרווצה לעסוק בנושא.
- צילום חישובי - קורס קטן ו חמוץ.
- מבוא לאינטלייגנציה מלאכותית - קורס יחסית עתיק ומעניין.

חלק II

יצור תמונות

תמונות נוצרות מאור שמאיר סצנה (אובייקט כלשהו), ומוחזר מהסצנה. האור המוחזר נתפס על ידי המצלמה. עוצמת ההארה שאנו תופסים במצולמה מסומנת ב- I . מוקדם ההזרה של הסצנה מסומן ב- r , למשל עבור גוף לבן, מוקדם ההזרה 1, גוף שחור - מוקדם ההזרה אפס. עוצמת ההארה של הגוף מסומנת ב- L . מתקיים הקשר $r \cdot L = I$. למעשה, יש לנו כאן יחס ישיר בין התאורה לבין האור שmagiu למצולמה.

1 העין האנושית

העין שלנו קולטת אור בצורה מעניינת - האור עובר דרך הרשתית ועובד חיווט דרך עצבי הראייה שנמצאים לפני התאים. לעין שלנו יש גם כתם עיור שדרךו איןנו רואים. יש לנו:

- 10^8 קנים - אחראים לראייה שחור לבן.

- 10^7 מדוכים - אחראים לצבע אדום, כחול, ירוק.

↳ איפה יש רזולוציה גדולה יותר? בשחור לבן או בצבע? כמובן שבשחור לבן, אך השאלה פי כמה. נבחן כי ככל שליש מתוך המדוכים הם בצבע זהה ולכן הרזולוציה שמה קטנה פי 30 מהרזולוציה בשחור לבן שכן יש שם כ- $\frac{10^7}{3}$ מדוכים וביחס ל- 10^8 זה .30.

↳ מגיבים פחות טוב לאור בהשוואה לקנים, אך בלילה אנו רואים בעיקר שחור לבן.

- 10^4 עצבים שmagiuים למוח.

↳ מה שאומר שיש הבדלי סדר גודל ענקיים בין מה שmagiu למוח לבין מה שאחראי על העבודה המוקדם.

האור הנראה הוא חלק קטן מהספקטרום האלקטרומגנטי. התדר שלו הוא בטווח [350, 780] ננומטר, כאשר בקצת הגבוה של הסקלה נמצאות קרניות קוסמיות וקרני X (מחורים שחורים למשל) ובקצת הנמוך גלי רדיו ומיקרו.

נשאלת השאלה, מה הוביל אותנו לתוך זהה? התשובה היא שאלת הצבעים בכדור הארץ. המשמש למעשה מקרן קרינה באורך גל של לפחות 450 ננומטר. מזלונו האטמוספירה מסוגלת להעביר את קרני השמש ולכן הכוכב שלנו לא חשוך והתפתחה סקלה זו.

אחד התכונות המעניינות של העין היא שכאשר קולטן קולט אור, הוא מגביר את תגובתו אך גורם לקולטנים מסביבו להחלש. ככה העין מצמצמת אינפורמציה. מה המשמעות של זה? אם נעיר באור לבן על קולטן מסוים, הוא יחליש את הקולטנים מהצדדים שלו ולכן למרות שלא מוקרנו עליהם אור ויחשבו שיש חושך, הם יציגו אור מעט יותר בהירות. לעומת זאת, השחור לא

מדכא את הלבן, כי הלבן הוא האור הנקלט. בקצרה, השחור יותר שחרר לפני החלק שקלט את האור, והלבן יותר לבן אחריו. כולנו מכירים תമונות מתעטות בהן אנו מתרכזים באובייקט מסוים עם צבע מנוגד לאובייקט קרוב וכל המסביר מתחילה להבהב, הסיבה לכך היא שהאובייקטים בצבע הכהה (המנוגדים) מגבירים או מחלישים את האובייקטים מסביב. מעבר לכך, העין גורמת גם לשאלות בתמונות, כמו השוואת נוכחות בין גודלים זהים. הסיבה לכך היא שאנו רואים בצורה יחסית לסביבה.

2 קליטת תמונה数 דיגיטלית

- השלב הראשון בקלטת תמונה הוא קודם כל צילום סביבה תלת מימדית אל תוך תמונה דו מימדית באמצעות מצלמה. צילום זה עושה שימוש בפרספקטיבה ובאופטיקה רציפה.
- השלב השני הוא הפיכת התמונה למשהו בדיד ולא רציף.
 - ▷ נבחר מספר סופי של פיקסלים.
- השלב השלישי הוא קונטיזציה שנעודה גם לעבר מסקלה רציפה לסקלה בדידה.
 - ▷ נבחר מספר סופי של צבעים מתוך המספר הסופי של פיקסלים.

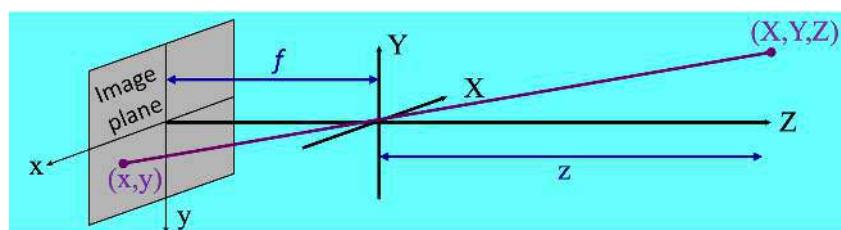
צילום תמונה

הטכנית הבסיסית היא קamera אובייקורה - האור שנכנס דרך מוקן הפוך על הקיר מצד שני.

מה שיוצר את התמונה הוא אוסף קרנויים שעוברות דרך חירר המצלמה.

- אוסף הקרנויים הוא כל הקרנויים שנכנסות דרך החירר, שכן הן היחידות שנופלות על הפלם.
- מפרק מוקד הוא המפרק בין החירר למשטח עליו נופלות الكرנויים. מסמנים אותו לרוב ב- f .

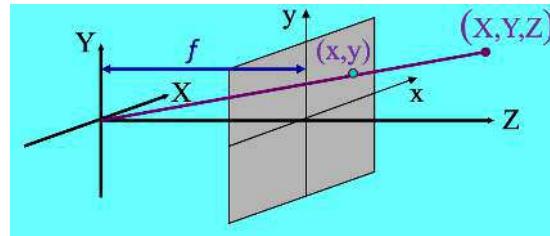
נניח כי יש נקודה (X, Y, Z) על האובייקט, נשאלת השאלה, כיצד לחשב את הנקודה (x, y) המתאימה לה על משטח המצלמה. קודם כל, נבחן כי משטח המצלמה נמצא מאחוריו החירר ולכן התמונה תתhape:



איור 1: קamera אובייקורה

ככה הפיזיקה עובדת. יחד עם זאת, לשם נוחות, נמקם את מישור המשטח מקדימה, לפני החיריר ולא אחוריו, ככה שהתמונה

לא תתהפק:



איור 2: המוחשת השינוי לשם נוחות

עתה נרצה להבין את הקשר בין (X, Y, Z) לבין (x, y) , ללא סימן, אלא רק גודל, שכן בשתי שיטות החסתכלות שהזכרנו הסימנים הפוכים בغالל היפוך התמונה ואי היפוכה. נבחן כי המשולש הנוצר $X-Z$, Y וחיבורם דומה למשולש הנוצר $-f-y$, f וחיבורם וולכן מותקיים היחס $\frac{y}{f} = \frac{Z}{Z-X}$. כלומר סך הכל:

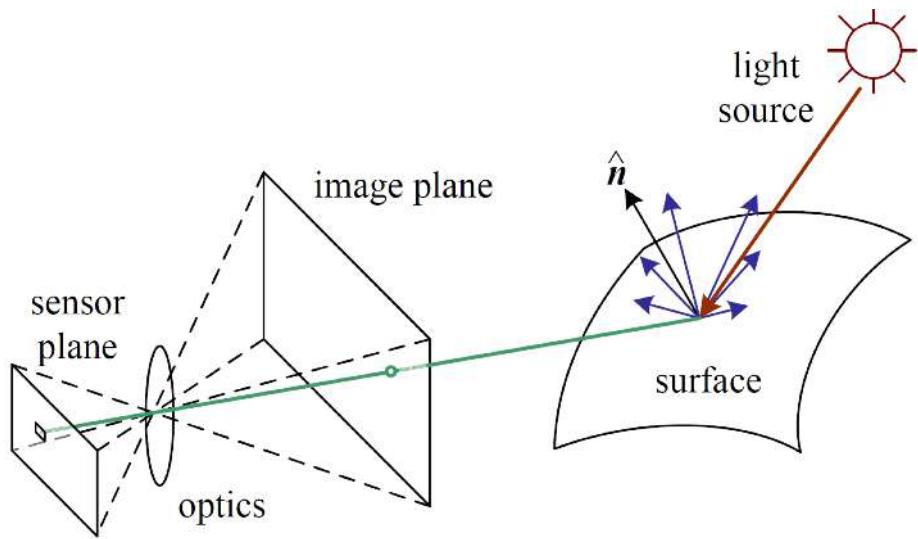
$$x = \frac{f}{Z} X y = \frac{f}{Z} Y$$

אם Z גדול, הגוף יותר קטן, ולבן y , x קטנים. אם המרחק מהחריר גדול, y , x גדולים. מתי נרצה להגדיל את f ? למשל, כשהנרצה להגדיל רזולוציה. אם נרצה לצלם כוכבים נגדיל את המרחב ככמה שנרויה מידע נקודתי, אבל נפסיד שדה ראייה רחב יותר:

• f גדולה - ראייה טלסקופית.

• f קטנה - שדה ראייה מאוד גדול.

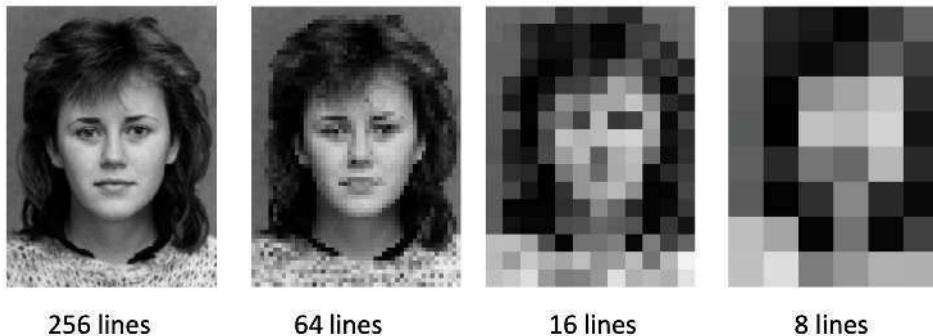
לסיכום, זה מה שקרה:



איור 3: סך כל שלב זה

בחירה מספר סופי של פיקסלים

תמונה נתן לצלם ברזולוציות שונות. לא תמיד אנחנו רוצחים שהתמונה תהיה באיכות $8k$ (המונע מידע בקובץ, קובץ גדול), ולא היינו רוצחים שהתמונה תהיה מורכבת מפיקסל בודד (מידע מינימלי ביותר). נרצה לבחור מספר שורות מסוים לתמונה שתתאים למטרת שלנו. נביט בהמחשה הבאה:



איור 4: אותה תמונה בהינתן מספר שורות שונה

הבחירה תעsha לצרכים שלנו, ולא בהכרח בהעדפה למספר שורות גדול יותר - זה מגדיל צרכית זכרון וזמן ריצה. הבחירה נעשית במכשיר - יש לנו מספר סופי של גלאים והם יוצרים את הרזולוציה.

בחירה מספר סופי של צבעים

בתוך המכלה יש למעשה שלוש שכבות שמטפלות באור:

1. השכבה הרכינה היא פילטר IR.
2. השכבה השנייה היא פילטר לצבע שmagu ל科尔טן - ריכוז האור כך שייפול על השטח הרגש לאור.
3. השכבה השלישית אינה מכירה צבעים, והופכת את האור שהיא קולטת לאוותות חמליים.

התוצאה היא תמונה פילטר שנראית פילטר Bayer, בה לכל פיקסל יש ערך שאומר האם הוא רגש לירוק, כחול או אדום, כאשר הירוק הוא הכי חשוב כי הוא הקרוב ביותר לסקלת שחורה לבן. אלה הפilters הצבעוניים שנמצאים בשתי השכבות. בכל תמונה יש לכל פיקסל כמה ערכים - RGB, אבל בתמונה שדיברנו עליה זה עתה, לכל פיקסל יש ערך אחד - R או G או B , מאיפה שני הערכים האחרים הגיעו? התשובה היא שהמצלמה המציאה אותם, זה עיבוד נוספת נוסף שהמצלמה עשוה לאחר שהיא מקבלת את האוותות החמליים. היא יכולה למשל לעשות ממווצע בין הצבעים של השכבים. במצלמות מושכללות יותר אפשר לקבל את התמונה המקורי הזה, ללא הפילטר; זה בדיקת מה שצלמים עושים כדי שיוכלו לעזרך את התמונה לבד. תמונה זו נקראת *תמונה raw*. הסוד המקzuוי של כל חברת מצלמות היא מה שיטות הפילטר שהם משתמשים בה.

נשאלת השאלה, למה משתמשים דווקא ב-*RGB*, התשובה היא שהעין שלנו רואה אדום וירוק ב-*RGB* ולכן אין סיבה להשתמש בעוד מימדי צבעים. במחקר לעומת זאת, כמו למשל תמונות לוויינים, יש 17 ערוצי צבע, כי אותם לא מעניינת התמונה עצמה אלא נתוני התמונה.

לבסוף אנו מקבלים מטריצה של ערכים שמייצגים צבע, ובאמצעות אופרטורים נוכל לבצע מניפולציות עליה. זהו למעשה עיבוד ספרתי.

ראית צבע - דרכים שונים לייצרת צבעים

נתון להקרין אורות אחד על השני. אם לא נקרין, הכל יהיה שחור. מרחב ראייה זה נקרא כאמור *RGB* דרך אחרת היא שימוש בדיו. ההבדל הוא שדיו בולע אור ולא מקרין אור. למשל, אם משתמש בסוגי דיו שונים, כל אחד מהם יבלע אורך גל אחר. למרחב ראייה זה קוראים *CYMK*.

כדי להציג תמונה במחשב מספיק להקרין אורות, אך כדי להדפיס תמונה? למעשה, כאשר אנחנו מדפיסים תמונה צבעונית יש המרה לתצוגת דיו מተצוגה מוקנית בתוך המדפסת.

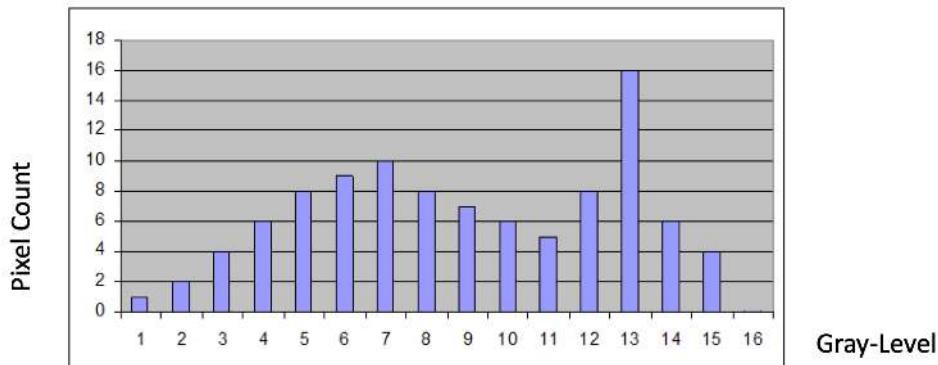
מרחב ראייה נוסף הוא של טלוויזיה ושמו *YIQ*. ה-Y מייצגת את השחור לבן ולמעשה מאפשרת לטלוויזיות שלא קלטו צבע, להמשיך לראות בשחור לבן גם כשעבورو לשידור בצבעים, ההמרה מ-*RGB* ל-*YIQ* היא כך (פשטוט טרנספורמציה ליניארית):

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

חלק III

ההיסטוגרמה

ההיסטוגרמה היא מדידה של כמות פיקסלים בעלי רמת צבע (או שחור לבן) מסוימת. ניתן להבטئ בתמונה הבאה להמחשה:



איור 5: דוגמה להיסטוגרמה

מכיוון שיש לנו שלושה מימדי צבע RGB , אנו עושים שלוש היסטוגמות, אחת לכל מימד. עליה השאלה, מה בנוגע לאייחודים שלהם, שכן יש לנו מרחב תלת מימדי של צבעים ולא רק שלושה צירים של צבע. התשובה היא שלא עושים היסטוגרמה תלת מימדית, מכיוון שיש יותר מדי אפשרויות. למשל, אם יש 256 ערכים לצבע אחד, אז במרחב תלת מימדי יהיה $2^{24} = 256^3$ אפשרויות, שזה יותר מדי.

אנו נניח בברירות מחדל שמדובר בשחור לבן, אלא אם צוין אחרת.

הבהרה בהיסטוגרמה תלת מימדית אנו יוצרים קובייה בגודל $256 \times 256 \times 256$ (כי יש 256 ערכים לכל טווח צבעים R, G, B) ואז אם אנחנו נתקלים בפיקסל (23, 4, 101) אנחנו מוסיפים +1 למקום המתאים בתיבת. בשולשת ההיסטוגמות הדו-מימדיות אנו יוצרים לכל ערך צבע R, G, B היסטוגרמה משלה. ערך ה- R נגיד יסתכל רק על הערך הראשון, ערך ה- R , בכל פיקסל ויבנה את הההיסטוגרמה לפיו.

3 שיווי היסטוגרמיה

נניח כי לכל צבע בתמונה יש אותו מספר של פיקסלים, אז ההיסטוגרמה הייתה נראית אחידה (העמודות היו בגובה קבועות). עתה נניח שיש לנו תמונה שימושת בטווח חסום של דרגות אפור, למשל [100, 150]. הינו רוצים למרוח את התמונה שתתאים לטווח המלא [0, 255], אבל זה רק במקרה של ההיסטוגרמה קבועה.

במקרה של ההיסטוגרמה קבועה בכל הסקלה, נוכל לסכום את הערכים ולקבל $\int_0^g p(x) dx = P(g)$ כאשר p היא ערך ההיסטוגרמה בערך האפור x . זו תהיה פונקציה ליניארית עולה. במקרה של [100, 150] קיבל גם סכימה $\int_0^g q(x) dx = Q(g)$

רק שכן קיבל עלייה רק מהנקודה 100. נשאלת השאלה, איך לעבור בין גרעף אחד לגראף המצוומצס? נסתכל על כל דרגת אפור g , נבדוק מהו $Q(g)$ ונמצאஇ g נתנו את אותו ערך בהיסטוגרמה האחידה, כלומר $(Q(g)) = P^{-1}(Q(g))$. זהו המעבר, באמצעות השוואה של סכימות ערכיו האפור.

עתה, נרצה שהמתיחה של התמונה תשמור על פרופורציה בין הפיקסלים וגם נקבל מגוון רחב יותר של ערכיו האפור. ערכים שכיחים יותר נפריד ליותר דרגות אפור, ואילו ערכים צמודים מעטים רק נזק לאורך הסקלה. אם רוב ערכיו האפור אינם שכיחים, שיווי היסטוגרמה אינה דרך.

נבחן כי היסטוגרמה דומה להיסטוגרמה אחידה אם ההיסטוגרמה המוצטברת שלה דומה לו לינארית.
נרשום אלגוריתם לביצוע שיווי היסטוגרמה:

נסמן ב- n_i את מספר הפיקסלים עבור ערך i (או היסטוגרמה) וב- $S_j = \sum_{i=0}^j n_i$ את מספר הפיקסלים עם ערכים i, \dots, j (או היסטוגרמה המוצטברת). לאחר שנחשב את ההיסטוגרמה המוצטברת, נעביר כל פיקסל מקורי j ל- S_j . נניח כי הסקלה היא $[K, \dots, 0]$ וכי יש לנו N פיקסלים.

באופן מדויק, בהינתן היסטוגרמה נבצע את הצעדים הבאים:

השוואה היסטוגרמית 1

1 : Histogram Equalization

2 : For each pixel i :

3 : change i into S_i

4 : Stretch (linear) new gray levels back to $[0, \dots, k]$:

5 :

First non zero S_m change to 0 and last value S_k into k // which means the effective value of S_i is

6 : $// S_i - S_m$ and the effective value of S_k is $S_k - S_m$

7 : change j back to scale but stretched $j = K \frac{S_j - S_m}{S_k - S_m}$

למעשה, הנרמול של j עם היחס $\frac{S_j - S_m}{S_k - S_m}$ זהה, יוצרת מתיחה של ההיסטוגרמה. נבהיר כי אנו מעגלים כלפי מטה את הערך החדש של j כי אנו צריכים להיות בקבוצה $\{0, \dots, K\}$. מעבר לכך, ניתן כי פיקסלים שערכם מעוגל לצבעים שונים, יMOVFO
לאוטו צבע לאחר ההשוואה. זאת לא בעיה, כי זה אומר שמדובר במספר מועט של פיקסלים, לפי הערך החדש של j .

בקיצור, ההיסטוגרמה מוחתת צבעים שימושיים בהם הרבה ומקבצת צבעים שימושיים בהם פחות.

לבסוף, נקבל כי בהינתן תמונה, האלגוריתם הבא יבצע עלייה השוואה היסטוגרמית:

Algorithm 2 השוואת היסטוגרמיות מלאה

- 1 : Given Image $I(x, y)$ create histogram H :
- 2 : For all x, y do $H(I(x, y)) = H(I(x, y)) + 1$
- 3 : Create cumulative histogram S :
- 4 : $S(0) = H(0)$
- 5 : For all k do $S(k+1) = S(k) + H(k+1)$
- 6 : Create Look Up Table (LUT) T :
- 7 : $T(k) = \text{round}\left(Z \frac{S_k - S_m}{S_Z - S_m}\right)$ // there might be a bug here. (hint: zero division)
- 8 : Apply LUT T to image I , get equalized image J :
- 9 : $J(x, y) = T(I(x, y))$

ה-LUT שימושית מאוד לשינוי התמונה. עתה נרצה לשים לב לכמה תכונות של השוואת היסטוגרמיות:

- ההשוואה היא מונוטונית.
- ▷ קלומר אם שני פיקסים בעלי ערכים עם יחס מסוים (גדול שווה או קטן שווה), היחס ישמר.
- אם נפעיל השוואה פשוט, מה יקרה?
- ▷ נקבל אותה תוצאה.
- מה המשמעות של הערך 127 לאחר ההשוואה?
- ▷ יש בדיקת חצי פיקסלים עם ערכים קטנה ממנו וחצי גדולים ממנו. קלומר אם במקור הפיקסל 17 חצי מהפיקסלים גדולים מ-17 וחצי קטנים ממנו, הפיקסל יעבור ל-127. דרגת האפור מייצגת את ההתפלגות המצטברת.
- מתי זה נכשל?
- ▷ עיתון הוא בשחור לבן. אם נביט בו עם זוכיות מגדלת נראה את כל הסיבים שלו (אליה רעים בצבעים שונים) ואמנם נבצע השוואת היסטוגרמיות נקבל המון צבעים נוספים שיוצרים לנו הרבה רעש ורקע. זה נובע מכך שהאותיות בהתפלגות נמכה ביחס לסייעים ולכך לא נקבל הופעה שלן.
- ▷ קלומר אם מראש אנו יודעים שיש רק שני צבעים וهمון רעים - שווי היסטוגרמיה רק יגביר את הسطייה ואת הרעים.

חלק VII

התמרת פורייה

4 מבוא

הרצאה 2

התמרת פורייה היא בעצם דרך להציג פונקציות. הפונקציה מוצגת כסכום של פונקציות \sin , כאשר כל מחומר יכול להיות בעל אמפליות ופaza כרצונו (φ , לפיסיקאים שבינינו). מסתבר שבאמצעות הכפלת בקבוע והזזה נוכל לקבל את כל הסיגנלים שאנו רוצים.

שינוי הפaza למעשה מזיז את הסיגנל ימינה (אם התוספת שלילית) ושמאלה (אם התוספת חיובית). שינוי האמפליות יתן לנו מרעת גדולה יותר.

בלינארית ראיינו פירוק של איבר במרחב וקטורי לצירוף לינארי של הבסיס, ראיינו שיש הרבה בסיסים ובסיסים "הכי טובים" הם האורתונורמלים. למשל, ניתן לרשום את המטריצה הבאה כ-

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

עתה נסתכל על בסיס $\left(\begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}, \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}, \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} \end{bmatrix}, \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \right)$, כאן הבסיס הוא לא אחד ואפס בכל פיקסל, אלא מלא כל אחד מארבעת הפיקסלים. איך אנחנו יודעים שהם בסיס? ניתן לראות שהם אורתוגונליים ואורתונורמלים. תכף נראה כי בסיס זה דומה לבסיס פורייה, מכיוון שככל איבר בסיס נתן לנו תמונה "אחדה" כביכול.

משפט. (פוריה, 1807)² כל פונקציה מחזוריים יכולה להיקתג כטכום משקל של \cos , \sin עם תדריות שוות.

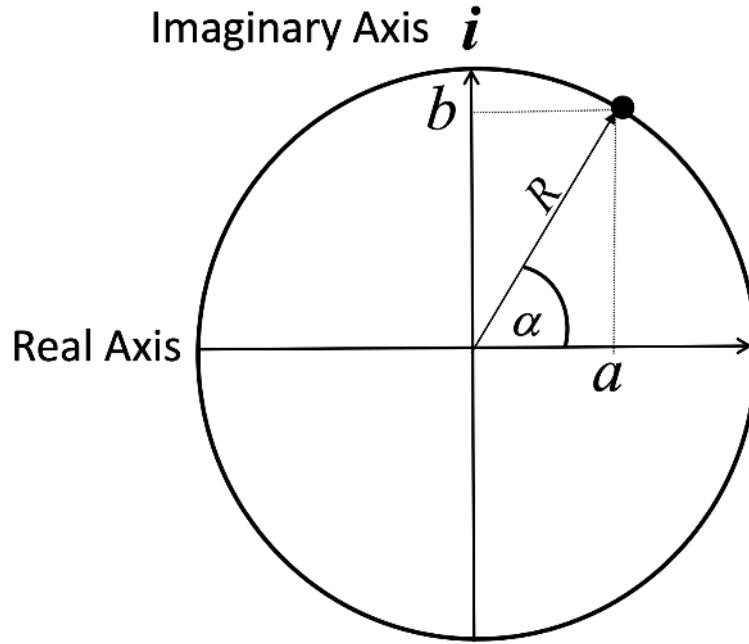
נשאלת השאלה, איך מין תמונה במציאות היא מחזוריית אינסופית? אין תמונה כזו. لكن נניח בחלוקת זה כי כל תמונה שוכפלת אינסוף פעמיים ימינה, שמאלה, לעללה ולמטה, וככה נקבל מחזוריות של התמונה. כלומר יש לנו פונקציה אינסופית מחזוריית שבסכל מחזור יש לנו את התמונה המקורית. זאת ההנחה לכל אורך הרצאה זו.

² אף אחד לאאמין לו, ולא הסכימו לתרגם את הכתבים שלו לאנגלית, זה קרה רק מאוחר יותר.

4.1 מספרים מרוכבים

בקשר של טרנספורמציה פורייה נוח לדבר על מספרים מרוכבים.

מספר מרוכב הוא מספר מהצורה $a + bi$ ונitin להגדירו באמצעות זהות אoilר $Re^{i\alpha} = R \cos(\alpha) + Ri \sin(\alpha)$. מתקיימים הקשרים $R = \sqrt{a^2 + b^2}$, $\alpha = \arctan\left(\frac{b}{a}\right)$:



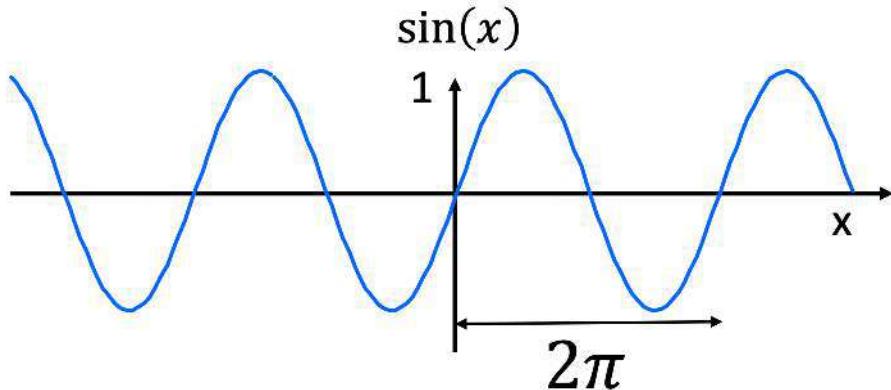
איור 6: המחשה גאומטרית לשתי הציגות

אנו מגדירים בנוסף $|a + bi| = \sqrt{a^2 + b^2}$ ואת הצמוד של המרוכב להיות $\bar{z} = a - bi$ מכאן נובע כי $|z| = |\bar{z}|$.

בנוסף, הייצוג של α נוח מאוד לכפל, שכן $R_1 e^{i\alpha_1} \cdot R_2 e^{i\alpha_2} = R_1 R_2 e^{i(\alpha_1 + \alpha_2)}$, ואילו לחיבור נוח הייצוג הבסיסי.

4.2 קצת על תזרירות

נביט בגרף הבא:

איור 7: גרפ' \sin

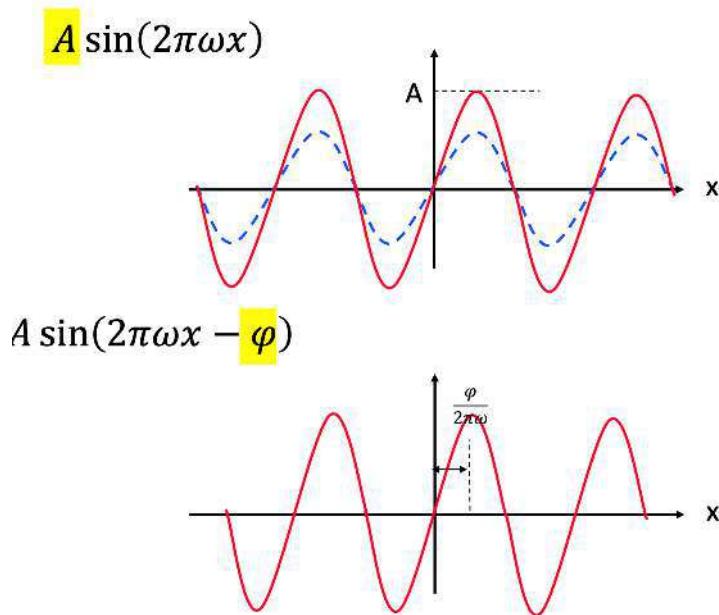
הגדרה. אורך הגל הוא אורך המוחזר המינימלי של הפונקציה.

הגדרה. תדרות של פונקציה מחזורי עם אורך גל T היא $\frac{1}{T}$, ומשמעותה כמה פעמים מוחזר הפונקציה נכנס ביחידת. תדרות מודדים ביחידת המידה $[Hz]$ - הרץ.

דוגמה. עבור x אורך הגל הוא 2π , התדרות היא $\frac{1}{2\pi} Hz$. עבור $\sin(ax)$ נקבל אורך גל $\frac{2\pi}{a}$ ותדרות $\frac{a}{2\pi} Hz$.

דוגמה. התדרות של $\sin(2\pi\omega x)$ היא ω ואורך הגל הוא $\frac{1}{\omega}$.

עבור פונקציות מחזוריות נוכל לשנות את הפאזה ואת האמפליטודה, לדוגמה עבור $\sin(2\pi\omega x - \varphi)$ נראה שהזינו את ($2\pi\omega x - \varphi$). כדי להוכיח זאת, נשאל מהו החיתוך החדש עם ציר ה- x ? נדרש כי $2\pi\omega x - \varphi = 0$ ומכאן $x = \frac{\varphi}{2\pi\omega}$ נדרש.



איור 8: המראה לשינויי פאזה ואמפליטודה

DFT 5

הרעין הכללי הוא קבלת וקטור קלט $(f(0), f(1), \dots, f(N-1))$ והוצאת וקטור פלט $(F(0), F(1), \dots, F(N-1))$ קלומר קיבנו N מספרים והחזנו N מספרים.

אם וקטור הקלט הוא וקטור בהצגת הבסיס הסטנדרטי, וקטור הפלט מיוצג בבסיס פורייה.

הчисוב הוא לפי הטרנספורמציה הבאה:

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

צריך לזכור את הנוסחה בעל פה, אבל בוואו נראה שהיא לא מפחידה כל כך.

למשל, $\bar{f}(0) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) e^0 = \frac{1}{N} \sum_{x=0}^{N-1} f(x)$ ממוצע חשבוני.

בנוסף, להז יש לנו טרנספורמציה הפוכה והיא

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

מה ההבדל?

- כאן אין מינוס באקספוננט ואין חלוקה ב- N .

\Leftrightarrow החלוקה ב- N לא תמיד מתרחשת בטרנספורמציה עצמה, אלא בהפוכה, לעיתים מחלקים בשתייהן ב- \sqrt{N} , זה לא

משנה, מה שחשוב זה שסך הכל חילקו ב- N .

למעשה, מה שקיבלו זה שוקטור הפלט נותנו את f כסכום משקלול של \cos , \sin עם מקדמים מרוכבים, אך התוצאה הסופית צריכה להיות ממשית, ולכן האיברים המרוכבים צריכים להצטמצם.

תוריגל. מה יקרה אם נבצע טרנספורמציה פורייה פעמיים? נשכח לרגע את הנרמול ב- $\frac{1}{N}$.

פתרון. נבחן כי ההבדל היחיד הוא הסימן באקספוננט, ולמעשה מתקיים כי התוצאה המותקנת היא $(x)^{-f}$, אבל מהי $\sin\left(\frac{2\pi u}{N}(N-x)\right) = -\sin\left(\frac{2\pi u}{N}x\right)$ וכי $\cos\left(\frac{2\pi u}{N}(N-x)\right) = \cos\left(-\frac{2\pi ux}{N}\right) = \cos\left(\frac{2\pi ux}{N}\right)$? $f(-x)$

$$e^{\frac{2\pi i ux}{N}} = e^{\frac{2\pi i u(N-x)}{N}}$$

ומכאן נובע כי

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

ולכן התוצאה היא פשוט היפוך הסדר של הערכים מוהסוו' להתחלה. למשל, אם קיבלנו בהתחלה $0, 1, 2, 3, 4$ נקבל עתה $4, 3, 2, 1, 0$. בambilים אחרים קיבלונו טרנספורמציה משוקפת.

שאלה מה הסיבות של טרנספורמציה פורייה?

תשובה יש לנו סכימה N פעמים $(F(0), F(1), \dots, F(N-1))$ כשבכל סכימה אנו סוכמים N איברים ולכן נקבל (n^2) .

זה זמן ריצה איטי מדי, שכן 10^6 איברים נתונים 10^{12} . יחד עם זאת, נבנה אלגוריתם בשם טרנספורמציה פורייה מהירה - FFT שקצר את סיבוכיות זמן הריצה ל- $O(n \log n)$. זאת הסיבה שהתחמש בהשתמש בטרנספורמציה פורייה.

نبיט עתה בביוטי

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

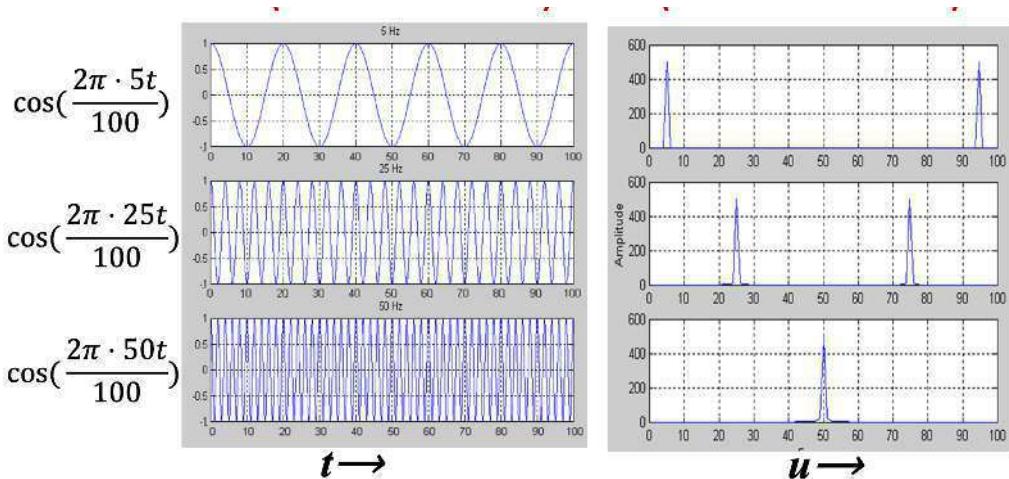
לכל u יש לנו וקטור בסיס שהוא תדר תלוי ב- x כלומר N איברים עם $x = 0, \dots, N-1$. למשל:

- עבור $0 = u$ קיבל 1.

$$\cdot \cos\left(\frac{2\pi x}{N}\right) + i \sin\left(\frac{2\pi x}{N}\right)$$

- וכן האלה.

בהצגה של פונקציה עם טרנספורמציה פורייה קיבל מספר סוגי תדרים. נביט למשל באյור הבא:



איור 9: דוגמה לאיך נראה טרנספורמציה פורייה

באյור הראשון, אנו רואים שב- $u = 5$ יש לנו עליה, מדוע? נבחן כי אורך הגל של פונקציה זו הוא $20 = \frac{100}{5}$ ולכן מסpter הפעים שהוא נכנס ב-100 הוא בדיק 5, כלומר תדרות הגל היא 5. אך מדוע בתדרות יש עליה? נזכיר כי

$e^{-\frac{2\pi i ux}{N}} = \cos\left(\frac{2\pi}{100} \cdot 5t\right) - i \sin\left(\frac{2\pi}{100} \cdot 5t\right)$ $F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) e^{-\frac{2\pi i ux}{N}}$

כאן הרכיב המשי הוא בדיק הפונקציה המקורית וכן קיבלנו את כל מה שאנו צריכים מהרכיב המשי ועלינו לאפס את הרכיב המודומה. זאת נעשה באמצעות $(u - N)$ ולבסוף קיבל אך ורק $f(t) = \cos\left(\frac{2\pi}{100}5t\right)$. במקרה אחר, זה או איבר $F(-x) = F(N-x)$ הבסיס היחיד של פונקציית שולוונטי ולכן היחיד שנסכם. העלייה ב-5-95 = 100 נובעת מהקשר שראינו ומכך צריך לאפס את הרכיב המודומה. דבר דומה קורה בדוגמה השנייה, יש סימטריה של תדרים.

אבל בדוגמה השלישי, אין יותר סימטריה, מודוע? נבחן כי התדרות של $\cos\left(\frac{2\pi \cdot 50t}{100}\right)$ היא 50. אם יש רכיב מודומה עבור $t \in \mathbb{Z}$, על כן, אין צורך לאפס את הרכיב המודומה, יותר על כן, מי שאמור לאפס $F(100-50) = 0$.

!F(50)

$$\begin{aligned} e^{-\frac{2\pi i ux}{N}} &= \cos\left(\frac{2\pi}{100} \cdot 50t\right) - i \sin\left(\frac{2\pi}{100} \cdot 50t\right) \\ &= \cos(\pi t) - i \sin(\pi t) = \cos(\pi t) \end{aligned}$$

שכן $\cos(\pi t) = 0$ לכל $t \in \mathbb{Z}$. על כן, אין צורך לאפס את הרכיב המודומה, יותר על כן, מי שאמור לאפס $F(100-50) = 0$.

!F(50)

מה מיוחד בתדר זה? זה למעשה התדר הכי גבוה! באופן כללי $\frac{N}{2}$ זה התדר הכי גבוה שנוכל להפיק מ- N דגימות. תדר זה נקרא **תדר ניוקויסט (Nyquist)**.

הערה. העולם שלנו לא בניו מפונקציות כל כך פשוטות, אך הוא כן בניו ממחסום שלחן.

5.1 תכונות

$$. F(u) = F(u + N) .1$$

$$. F(u) = F^*(-u) = F^*(N-u) .2$$

$$. (a+bi)^* = (a-bi) .3$$

$$. |F(u)| = |F(-u)| .4$$

$$. F(-u) = F(N-u) .5$$

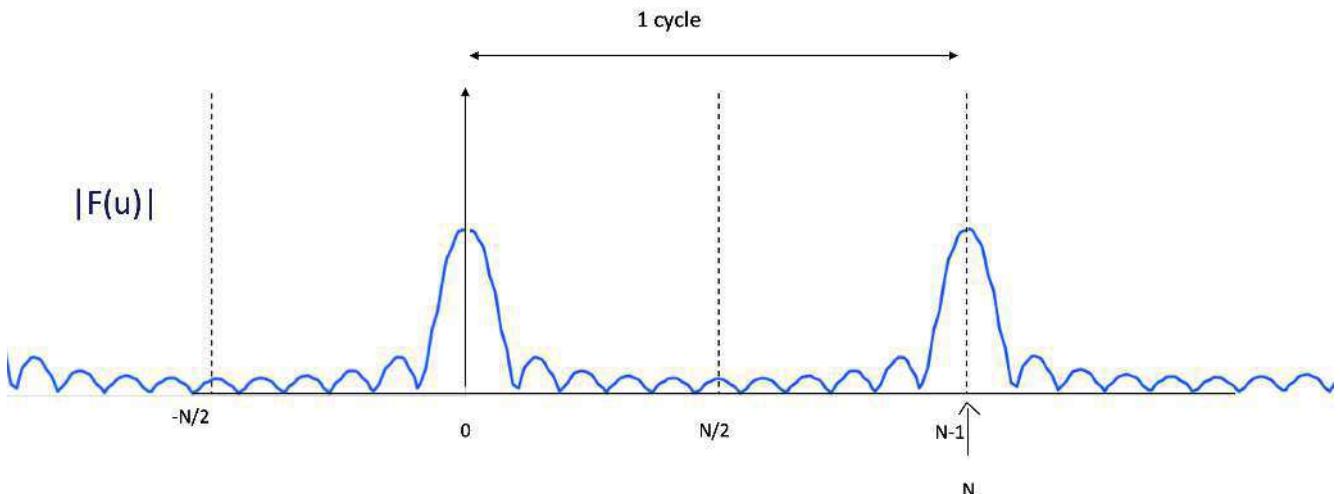
$$. f(-u) = f(N-u) .6$$

דוגמא. $|F(6)| = |F^*(-6)| = |F^*(250)|$ וכך $F(6) = F^*(-6) = F^*(250)$ וכך $F(6) = F(262)$ ואילך $N = 256$

יתר על כן,

- מכיוון שקיבלנו מלכתחילה N מספרים ממשיים וקיבלנו חזקה N מספרים **מרובבים**, קיבלנו למעשה $2N$ מספרים ממשיים חזקה, ואכן, זה יותר מדי (הרי איך מ- N דגימות בלבד ניתן להפיק כל הרבה מידע?).

- ▷ מספיק לבחור את $F(0), F(1), \dots, F\left(\frac{N}{2}\right)$ ולהסיק את שאר האיברים עם הצמוד.
- התדר הכי גבוה בטרנספורמציה הוא $u = n$.
- ▷ כל $0 \neq u$ יתן כפל בסינוס וкосינוס שرك יקטינו את הביטוי הכללי, ואילו אם $0 = u$ קיבלנו 1 בכל מקרה שכן הכל חיובי.
- טרנספורמציה פוריה היא מחזוריית, שכן היא מורכבת מפונקציות מחזוריות.
- מספיק להסתכל על $F(0), F(1), \dots, F\left(\frac{N}{2}\right), F(-1), \dots, F\left(-\frac{N}{2} + 1\right)$ שכן מהם ניתן להסיק את colum. לכן לעתים נסתכל על הגרף בצורה הבאה:

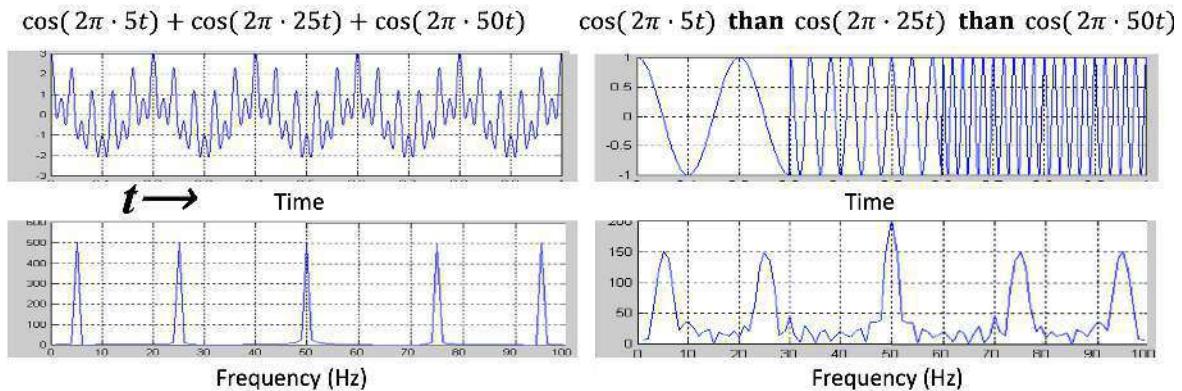


איור 10: 0 נותן את הערך הכי גבוה ולכן שמו אותו במרכז. הכל מחזורי ולכן מספיק להסתכל על הטווות $\left[\frac{-N}{2}, \frac{N}{2}\right]$

הערה. בגלל ש-sinc פונקציה חלקה, קשה להציג התנוגות של מדרגות ולכן פונקציות כאלה כוללות סכום אינסופי של sinc, לכן מעבר חד בתמונה נותן הרבה תדרים רבים.

5.2 גלים לא סטציונריים Short Time FT - STFT

نبיט בפונקציה הבאה:



איור 11: סיגנל עם תדריות משתנה

פונקציה שאינה עם אורך גל קבוע, נקראת פונקציה לא סטציונרית. מה הבעה בביטוי טרנספורמציה פורייה רגילה? הבעה היא שהחוצאה תהיה לפי תדרות ולא לפי זמן, וככה לא יוכל לאיזה חלק כל עלייה מתיחסת. אם כך, נבצע טרנספורמציה פורייה על ידי חלוקה לאזורים בהם היא כן סטציונרית. ככלומר בכל חלון בו יש תדרות קבועה, נקבל תדרים מותאימים.

מתי משתמשים בדבר זהה? בדיבוב, כאשר מנתחים קול, התדרים משתנים. בניתוח זה, אנחנו צריכים לבחור מספר דגימות בשנייה ולקבל גל מסוים. בעיבוד קול علينا לבחור את הדברים הבאים:

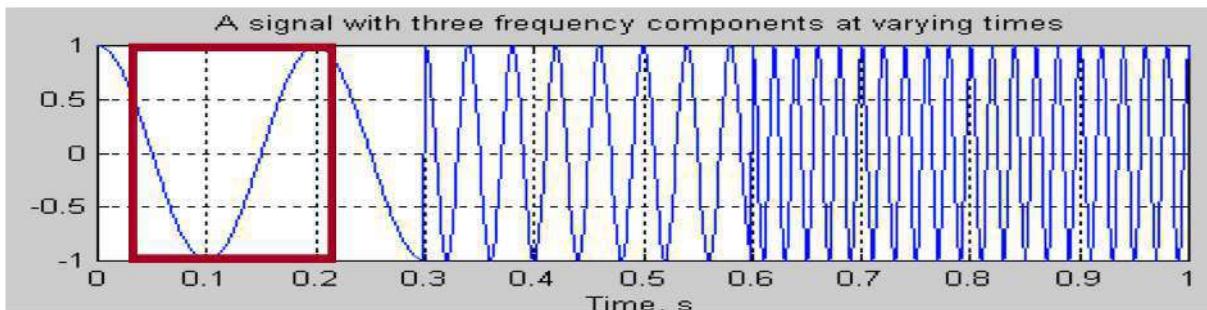
- כמה צפוף לשים את הדגימות.
- כמה ביטים לייצוג כל דגימה.

מאוד טבעי לפרק את הגל הזה לסכום של תדרים פשוטים. למעשה, באמצעות טרנספורמציה פורייה אפשר להוכיח כי על מנת שנשמעת האות הקול, הדגימה צריכה להיות לפחות פי 2 מהמקסימום של התדרות ככלומר אם אנחנו רוצים להיות מסוגלים לשמעו [Hz] 20,000 [Hz] 40,000 דגימות לפחות בתדרות של [Hz] 40,000, 40,000 דגימות בשנייה. לכל ערך דגימה אנו בוחרים מספר ביטים לייצוגו, בתמונה בצד שמאל, אנו בוחרים מספר ביטים לפיקסל.

יש הקובלות רבות בין תמונות לבין קול.

כדי לבצע טרנספורם פורייה, נבחן כי חלוקה לאזורים היא לא דבר פשוט שכן קשה לוחות תדריות המשתנות. לכן, אנו בוחרים פונקציית חלון (n) שמתאפסת בכל מקום שאינו בתוך הטווח, או בעלת התפלגות מסוימת, ומזינים אותו קצר בכל פעם כהה שככל איזור סטציונירי מקבל טרנספורם פורייה ממש עצמו.

אנו בוחרים את הראשית על ידי זהה של w לחalon המתאים או הכפלה בסיגנל, כך נקבל לבדוק את הדגימה הרצiosa בצד שמושקלת. פונקציית החalon מושקלת את הדגימה, ככלומר היא תנתן חשיבות לאזורים שונים בתוך החalon. לאחר שסימנו עם החalon, אנו עוברים לאחד הבא, עד שהגענו לאחד האחרון. כמובן, הכל בדי. זה נראה בערך כך:



איור 12: דוגמה לחלון בסיגנל לא סטציונרי

נוכל לרשום אלגוריתם לביצוע תהליך זה, לו אנו קוראים STFT:

Algorithm 3 STFT

- 1 : Choose a window $w(n)$ of finite length
 - 2 : Place the window on top of the signal at $t = 0$
 - 3 : Truncate the signal by multiplication with this window
 - 4 : Compute the FT of the truncated signal, save results
 - 5 : Incrementally slide the window to the right
 - 6 : Go back to step 3, until window reaches the end of the signal.
-

יתכנו כל מיני פונקציות חלון, הפונקציות הבאות ממחישות את התפקידים העיקריים שלחן:



איור 13: חלון משולש וחלון מרובע - המרובע נותן חשיבות זהה לכל האיברים בחלון, אך המשולש מממשקל אותם.

למעשה, אנו מקבלים טרנספורמציות פורייה חדשה, על תמונה :

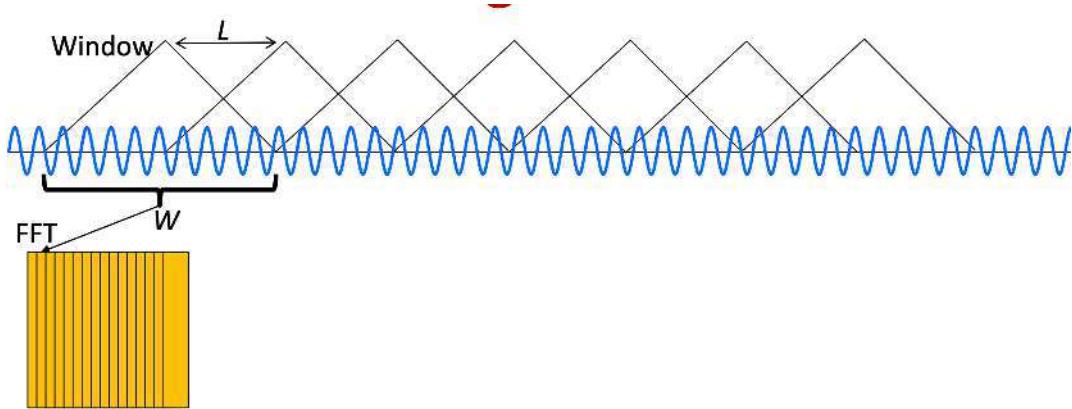
$$F(n, u) = \sum_{m=-\infty}^{\infty} f(m) w(n-m) e^{-\frac{2\pi i un}{N}}$$

והטרנספורמציה ההפוכה:

$$f(n) = K \sum_{p=-\infty}^{\infty} \sum_{u=0}^{N-1} F(pL, u) e^{\frac{2\pi i un}{N}}$$

כאשר בכל צעד אנו מدلגים על טווח L ועוברים לחלון הבא.

בפועל, מה שאנו עושים נראה כך:



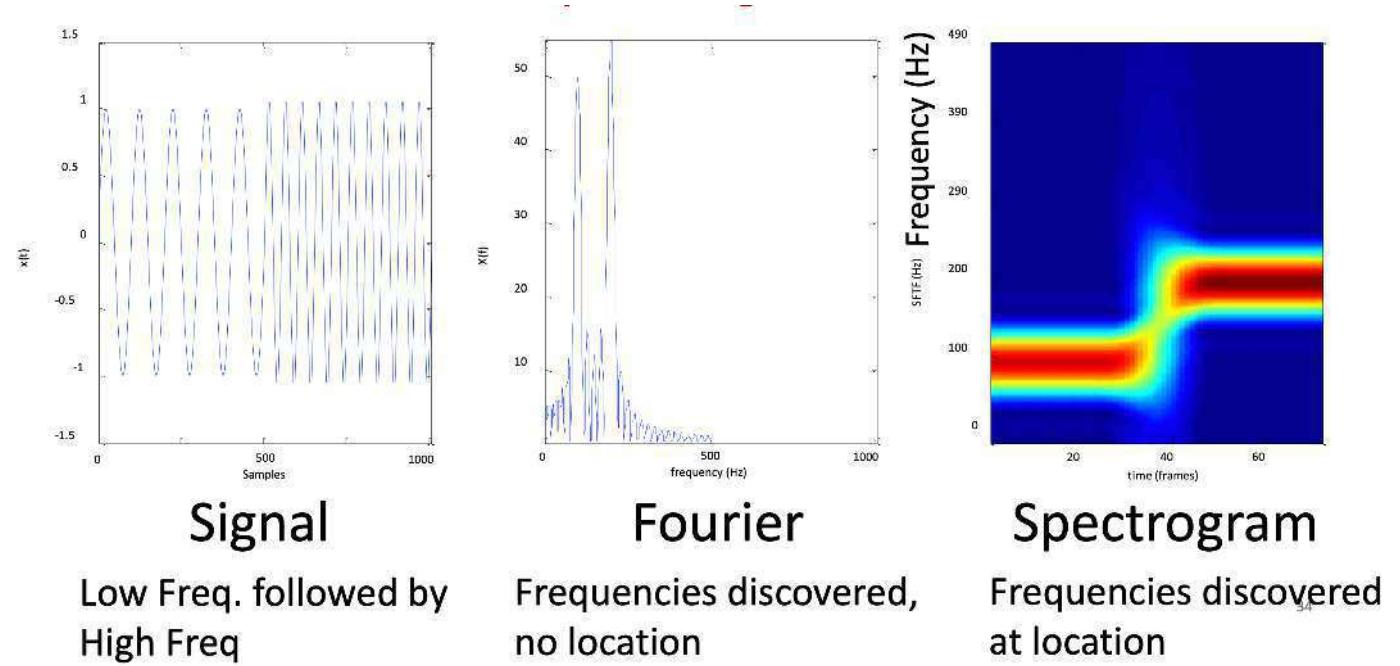
איור 14: חלוקה לחלונות דגימה. יש חפיפה בין החלונות זה לזה, שכן אנו רוצים לתפוס את האזוריים הסטציונריים, ולא לדגום כל חלק מאזור פעם אחת.

נעזר בסימונים הבאים:

- W - אורך החלון
 - ▷ מספר המקדמים השונים מאפס ב- w
- L - מספר הדגימות בין שני חלונות חופפים.
- מכאן החפיפה בין שני חלונות היא $L - W$.

5.3 ספקטוגרפיה

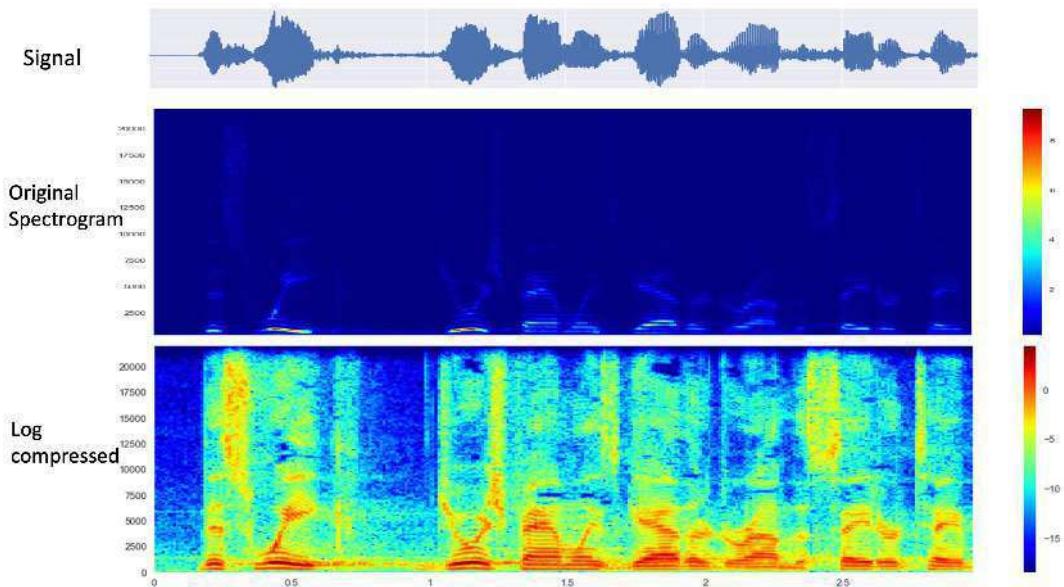
כלי שימושי להציגת STFT הוא גраф ספקטוגרפיה שמציג תדריות כתלות בזמן ולא עצמה כתלות בתדרות. באמצעות גراف זה נוכל ממש להציג את STFT עם חלוקה מדויקת לאזוריים. זה נראה כך:



איור 15: ספקטוגרפיה ביחס להצגה רגילה של טרנספורם פורייה והסיגנל

הגרף לעיטים יהיה לא ברור, ככלומר עם קונטרסט נמוך ולבסוף עליו טרנספורמציה לוגריתמית כדי להעלות קונטרסט, ככלומר

$$\text{נבע} (\log (|F(u)| + 1)$$



איור 16: הצגה ברורה של ספקטוגרפיה עם פרישה על כל דרגות האפור. בעצם אנו רואים את תדרי הקול כתלות בזמן

למעשה ביצור ספקטוגרפיה אנחנו צריכים לנקוט בחשבון את הפרמטר הבא:

• אורך החלון:

- ▷ אם החלון קטן נקבל דיוק רב יותר, אבל לא נוכל לחשב תדרים בצורה טובה.
- ▷ אם החלון גדול מדי, נקבל הרבה תדרים אבל נאבד דיוק.
- ▷ אנו מקבלים רזולוציה של תדריות עם אורך חלון גדול במחיר של הפרדת זמנים גרועה (התדריות ישתנו והגל אינו סטציוני).

5.3.1 "שומים"

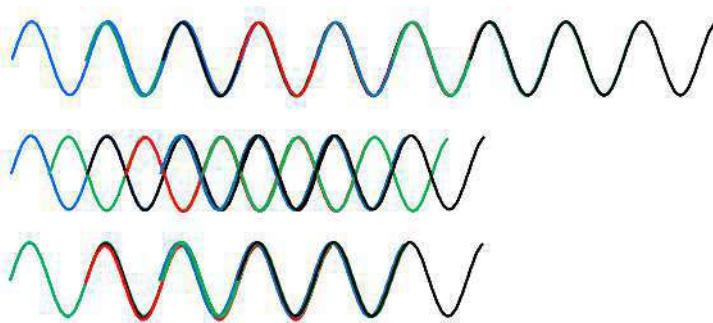
נזכיר עתה שימוש מעניין לכלי זה - *FastForward*. למשל, אם נרץ פי 2 יותר מהר, התדריות תגדל פי 2 והcoil ישנה. מה נעשה כדי שלא יקרה? יש כמה דרכים להתמודד עם הבעיה:

1. נוכל לדגום כל נקודה שנייה ולוטר בעצם על חצי מהנקודות.

(א) או במקרה זאת, להקטין את הקפיצות בין העמודות בספקטוגרפיה, נשאר עם אותם תדרים (כפי הרוי הספקטוגרפיה מציגה תדרים), אבל אנו עלולים לקבל בעיות של פאות.

2. נוכל להחליט שזה בסדר שייהו תדרים גבוהים ממה שהוא קודם. במקרה זה, נגיד למערכת שתדר הדגימה הוא כפול לדוגמה [Hz] 16 [קילוהץ], שכן אז היא תיקח 16,000 דגימות ותציג אותן בשנייה במקום 8,000, בכל פעם. כמובן במקומות מסוימים יכול למכסימום ומינימום s פעמיים בשנייה, הוא יבצע זאת $2s$ פעמיים בשנייה ולכן התדריות גdale פי 2.

בפתרון 1 לעבעה, שינוי המרווח בין הדגימות יוצר בעיה. הבעיה היא שהשינוי הזה משנה את הפאה של כל האותות, שכן אנו מזיזים את כל האותות מבלי להתחשב בפאה המקורית שלהם. לכן כשنبנה את הגל החדש, החיבור בין האותות שבכל חלון עלול ליצור התאבכות הורסתה. על כן, נדרש לחשב את הפאה ולתקן אותה לכל תדר, כמובן, מיד לאחר שביצעוו שינוי ב מהירות הצגת הגל נבצע תיקון פאה לכל חלון.



איור 17: תיקון פאות.

בשורה הראשונה: החלונות המקוריים.

בשורה השנייה: החלונות לאחר הקטנת הקפיצות בספקטוגרפיה.

בשורה השלישית: האות לאחר תיקון הפאה.

התמרת פורייה דו מימדית - שימוש בתמונות

הרצאה 3

הערה. הפרק הנוכחי כולל המון מלל ויחזר על אותן נקודות כמה פעמים במיללים שונים, כדי להדגיש ולבהיר דברים, כך שהנושא יהיה מובן לקורא.

במושיק יש מכשיר בשם *Equalizer* אליו אפשר לשנות את עוצמת הבסיס או עוצמה של הtones הגבוהים וכו'. כיצד המכשיר עובד?

ראשית ה-*STFT* מפעיל *Equalizer* - יוצר ספקטוגרמה שכך קול לא בעל תדרות אחידה. המכשיר יוצר, מהקלט של המשתמש, וקטור عمودה המתאים לתדרים שרוצים להגבר/להנמק. לדוגמה עברו וקטור הקלט

$$(1.4, 1.2, 1, 0.9, 0.7, 0.7, 0.9, 1, 1.2, 1.4)$$

ניתן לראות שאנו מגדילים את התדרים הגבוהים והנמוכים, כי אנו כופלים אותם בפקטורי גדול מ-1. לדוגמה, התדר הנמוך ביותר כפול ב-1.4. את התדרים האמצעיים אנו דוחקים מחלשים, כי אנו כופלים אותם במספר קטן מ-1. את הוקטור הזה אנו כופלים בכל עמודה כדי לשנות את התדר, כל עמודה שקיבלו מה-*STFT* בכל נקודות זמן, ועשויים את הטרנספורמציה ההיפוכית *ISTFT* כדי לקבל את הסיגナル עם העוצמות החדשות.

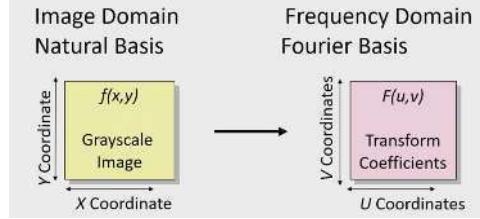


איור 18: ממשק של *Equalizer*

תמונה, בניגוד לסאונד, היא אובייקט דו-מימדי. נסמן את התמונה בתור פונק' $f(x, y)$ המתאימה לכל קווארדיינטה דרגת אפור (לא נתעסק עכשו עם RGB).

נבעץ על f התמרת פורייה דיסקרטית דו-מימדית, $F(u, v) = DFT(2D - F)$, ונקבל פונק' מרוכבת (u, v) - כאשר הבחירה לסמן את הקואורדינטות בתור u, v היא כדי שייהי מובן שמדובר על תדרים.

זכור שאנו מתיחסים לתמונה f בתור פונק' מהזויות, קרי מנחים שהתמונה f משוכפלת אינסוף פעמים לכל הצדדים. מהגדרת F ומהנהנה ש- f מהזורה, קיבל שgem F מהזורה.



איור 19: התמרת פורייה על תמונה

התמרת פורייה דיסקרטית דו-מימדית, וההיפוכית שלה, מוגדרות כדלקמן:

$$\begin{aligned} -\infty < \forall u, v < \infty \quad F(u, v) &= \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) e^{-\frac{2\pi i (ux+vy)}{N}} \\ -\infty < \forall x, y < \infty \quad f(x, y) &= \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u, v) e^{\frac{2\pi i (ux+vy)}{N}} \end{aligned}$$

הערה עבור תמונה לא-ריבועית, בגודל $N \times M$ ההתמורה תראה כדלקמן (שימו לב שבאופן שריורי בחרנו לנរמל רק בההתמורה הפוכה):

$$\begin{aligned} -\infty < \forall u, v < \infty \quad F(u, v) &= \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x, y) e^{-2\pi i \left(\frac{ux}{N} + \frac{vy}{M} \right)} \\ -\infty < \forall x, y < \infty \quad f(x, y) &= \frac{1}{NM} \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} F(u, v) e^{2\pi i \left(\frac{ux}{N} + \frac{vy}{M} \right)} \end{aligned}$$

הערה. אטם עלולים להגיד שמדובר, מטריצה $N \times N$ היא פשוט וקטור באורך N^2 ולכן אין סיבה להגיד לההתמורה פורייה חדשה למקורה הדו-מימדי.

עם זאת, כשאנחנו מסתכלים בעניינים שלנו על תמונה אנחנו אף פעם לא מסתכלים עליה כעמודה אחת של פיקסלים אלא כמטריצה, וגם יש הבדל בין התמורה חד-מימדית לדו-מימדית: במקרה הדו-מימדי יש לנו שני תדרים, v, u , האחד מותאים לעמודות של התמונה והאחר לשורות של התמונה - רוחב ולגובה.

בஹמשך להערה, נשים לב שיש מספר הבדלים בין התמרת פורייה חד-מימדית לדו-מימדית:

- יש לנו עתה שתי קואורדינטות לפונק' במקומות אחדת - לדוגמה $F(u, v)$ במקום (u)
- בחזקה של e כתוב עתה $uy + ux$, במקום רק ux במקרה החד-מימדי
- במקרה הדו-מימדי אנו כופלים ב- $\frac{1}{N}$ גם בההתמורה וגם בההתמורה ההיפוכית.

↳ הסיבה לכך היא שיש לנו N^2 איברים, אך אנו צריכים סה"כ לחלק ב- N^2 ולשם נוחות ספציפית במקרה זה החליטנו לחלק בכל התמורה ב- $\frac{1}{N}$ (אך אין חובה לעשות זאת כך).

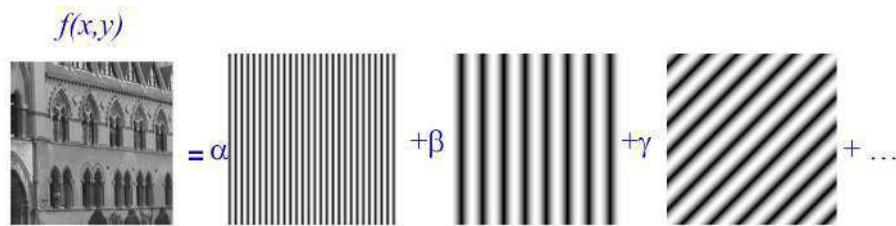
בדומה להתרמת פוריה חד-מימדית, גם כאן $(0, 0)$ מקבל ערך הקשור במוצע הערכים של f :

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

כאשר \bar{f} הוא מוצע הערכים של f . אם היינו בוחרים לחלק ב- N^2 בהגדלה של F אז היינו מקבלים

פונק' הבסיס של התרמת פוריה דו-מימדית הן מהצורה $e^{\frac{2\pi i(ux+vy)}{N}}$, כאשר כל צמד (x, y) נותן פונק' בסיס אחרת. לתמונה בגודל 5×5 לדוגמה, קיבל 25 פונק' בסיס - כל פונק' היא מטריצה בגודל 5×5 .

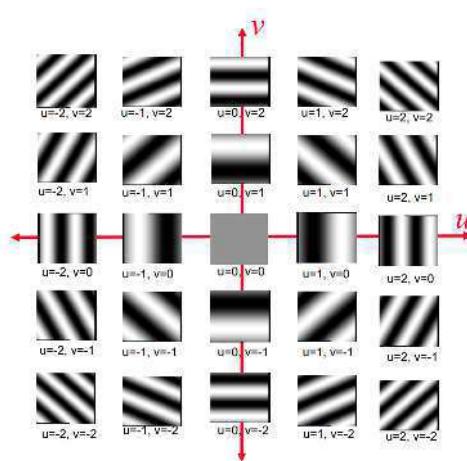
בפונקציות הבסיס הללו אנו משתמשים כדי לבצע התרמת פוריה דו-מימדית הפוכה ($2D-IDFT$), כל תמונה שהיא ניתן לייצג כסכום משוקל של פונקציות הבסיס הללו.



איור 20: כל תמונה ניתן לייצג כסכום משוקל (צירוף לינארי) של בסיסי פוריה

להלן ויזואлизציה של הבסיסים עבור תמונה 5×5 , לכל ערכי $u, v \leq 2$. באירנו אנו רואים רק את החלק המדומה, $\sin\left(\frac{2\pi}{N}(ux + vy)\right)$

באירור 25 תונות קטנות, כל אחת מתאימה לפונק' בסיס אחרת. צבע שחור מותאים לערך 1 (המקסימום של \sin) ובן מותאים ל-1 (המינימום של \sin). אפור מותאים ל-0. עבור $u = 0, v = 0$ ניתן לראות כיצד הפונק' מקבלת 0 (אפור) בכל מקום, זאת כי החלק המדומה עבור $0 = u = v$ הוא $\sin 0 = 0$



איור 21: המראה של בסיס פוריה לתמונה בגודל 5×5

הערה. נדגש זאת שוב, כל אחת מ-25 התמונות באירור הנ"ל הן פונק' **שונות**, שסודרו בהתאם לערכי $h-u, v$ שייצרו כל פונק'.

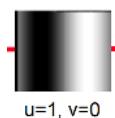
התמונות לא מוצגות כאן כדי להוות פונק' אחת גדולה (למרות שהסימטריה היפה של המכלול גורמת לעין שלנו לרצות להסתכל על כל התמונה כפונק' אחד).

לאלה מכם שעדיין לא הבינו עד הסוף מה התמונה מייצגת, נציג דוגמה נוספת. נתמקד ב- $v=0$, $u=1$.

פונק' הבסיס שלנו תראה כך:

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

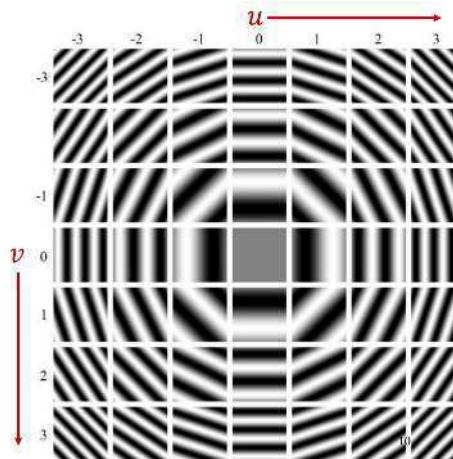
בתמונה זו אנו מציגים רק את החלק המדומה, $\sin\left(\frac{2\pi ux}{N}\right)$. ניתן לראות כיצד בIOR של התמונה, בו $0 = x$, אנו מקבלים ערך אפור, 0, אחר כך עולים ל-1, יורדים ל-1 – ואז חוזרים ל-0. בדיקék כך מתנהג \sin !



IOR 22: פונקציית בסיס של פורייה עבור $u = 1, v = 0$

לסיום, עבור תמונה בגודל 5×5 (מ"ז מミיד 25) אנו מקבלים את 25 פונק' הבסיס של פורייה, כל אחת מהן היא מטריצה 5×5 , כאשר כל תמונה בגודל 5×5 בעולם ניתן לייצג כסכום משוקלל של בסיסי הפורייה, כאשר כל איבר בבסיס מוכפל במקדם הפורייה המרוכב המתאים לו - (u, v, F) , שמשנה את האמפליטודה והפזה של פונק' הבסיס אותה הוא כופל. זה למעשה מה שאנו עושים כשאנחנו מפעילים התמרת פורייה הפוכה, $.2D - IDFT$.

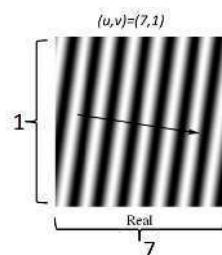
אם נסתכל על בסיס פורייה לתמונות בגודל 7×7 , עם כל הערכים $u, v \leq 3$, קיבל פונק' בסיס שנראות כך:



איור 23: המחשה של בסיס פורייה לתרמונה בגודל 7×7

שימו לב שככל שהגדלים (ערך מוחלט) של התדרים u, v יותר גדולים אנו מקבלים שינויים יותר חדים. הפעם יש לנו $7 \cdot 7 = 49$ פונק' בסיס כי התרמונה היא בגודל 7×7 ולכן זה המימד של המ"ז.

נשים לב שיש לנו 2 שני תדרים: u בכיוון x ו- v בכיוון y . מה זה אומר? אם לדוגמה $(u, v) = (7, 1)$ אז בכיוון x יש לנו 7 גלים ובכיוון y גל אחד.



איור 24: המחשה של בסיס פורייה עבור $(u, v) = (7, 1)$

כלומר, אם נסתכל על שורה אחת של פיקסלים בתמונה (ציר x) אז היא תבצע 7 מחזוריים ביחידת הזמן בתמונה. אם נסתכל על ציר y , על אף שהרבה יותר קשה לראות, כי ציר x בולט יותר, היא תבצע מחזור אחד. ניסינו להציג זאת באיור הבא, שמסתכל על עמודה ספציפית בתמונה הקודמת. שימו לב כיצד אנו עושים מעבר משוחר לבן חזרה לשחור פעמיים בנגדות לציר ה- x שהספיק לעשות זאת 7 פעמים.



איור 25: המחשה של בסיס פורייה עבור $(v, u) = (7, 1)$. עמודה ספציפית.

נזכיר כי להתרמת פורייה, $F(u) = R(u) + iI(u)$, אנו מגדירים מספר תכונות:

$$1. |F(u)| = \sqrt{R^2(u) + I^2(u)} \quad (\text{ספקטרום פורייה})$$

$$2. \alpha(u) = \tan^{-1} \left(\frac{I(u)}{R(u)} \right) \quad (\text{פазת פורייה})$$

$$. F(u) = |F(u)| \exp(i\alpha) \quad (\text{תחת הגדרות אלה})$$

נרצה להציג את טרנספורם פורייה כתמונה, וכך נעבור לפיה האלגוריתם הבא:

הציגת טרנספורם פורייה כתמונה 4

- 1 : Compute 2d transform Fourier of the image $F(u, v)$ // Not uniformly colored
 - 2 : Compute $\log(|F(u)| + 1)$ // Log transform to be able to see low level grey scale values
 - 3 : Scale to full grey-level range
 - 4 : Move $(u = 0, v = 0)$ to center of image (shift by $\frac{N}{2}$)
-

אנו משתמשים ב- $\log(|F(u)| + 1)$ ולא ב- $|F(u)|$ כי עבור u נמוך, אך עבור u גבוה $|F(u)|$ מאוד גבוהה. בשלב

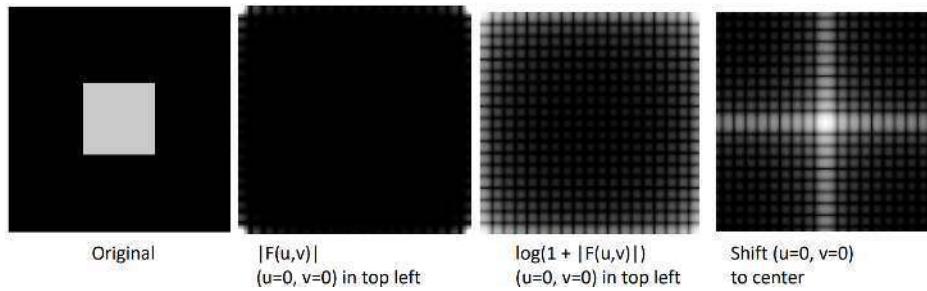
2 אנו מוחתמים את הערכים כך שייפרשו את כל טווח ערכי האפור שיש לנו.

אנו מוסיפים שלב 3 בו אנו מזיזים את התדר $(u = 0, v = 0)$ למרכז התמונה (הזהה ב- $\frac{N}{2}$).

דוגמה. להלן דוגמה לדריכים להקטין את הטווח של פורייה מ-[0..100] ל-[0..10].

המקורי F	0	1	2	4	100
חלוקת ב-10	0	0	0	0	10
$\log(1 + F)$	0	0.69	1.01	1.61	4.62
מתיחה לטווח [0..10]	0	1	2	4	10

אנו רואים שהדרך המיטבית היא באמצעות שימוש ב- $\log(1 + |F|)$.



איור 26: חישוב התרמת פורייה על תמונה אמיתית, יחד עם שלבי ה-*downscaling* לוגריטמי והזזה של $(u = 0, v = 0)$ למרכז התמונה.

שאלה מתי הערך $|F(0, 0)|$ מקבל ערך מקסימלי?

תשובה שככל המספרים חיוביים. בסאונד, בנגדות תמונה, חלק מהערכים הם שליליים אך בדרך כלל ($u = 0, v = 0$) קיבל את הערך 0, שיהיה הממוצע של הערכים החיוביים והשליליים.

6 הזזה של תמונה והתרמת פורייה

אנו יכולים לבצע הזזה (Translation) לתמונה ואז f, F יושפעו באופן הבא:

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

$$F(u - u_0, v - v_0) \iff f(x, y) e^{\frac{2\pi i(u_0 x + v_0 y)}{N}}$$

כאשר \iff מציין שהנוסחה החדשה דומה לנוסחה המקורית, יחד עם כפל בפקטור הנוסף הרשום כאן.

נראה זאת פורמלית, ישירות מההגדרה של התרמת פורייה והתרמת פורייה ההופוכה:

טענה. תהי $f(x, y) = f(x - x_0, y - y_0)$ ו пуנק' דיסקרטית בשני משתנים ויהיו $x_0, y_0 \in \mathbb{N}$. נגידיר F_f , $F_{f_{shifted}}$ ונסמן $\forall u, v \in \mathbb{N}, |F_{f_{orig}}(u, v)| = |F_{f_{shifted}}(u, v)|$ בהתאם. אזי

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

$$\begin{aligned}
 F_{f_{shifted}}(u, v) &= \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f_{shifted}(x, y) e^{-\frac{2\pi i(ux+vy)}{N}} = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x - x_0, y - y_0) e^{-\frac{2\pi i(ux+vy)}{N}} \\
 &\stackrel{\text{שניאי אינדקסים}}{=} \frac{1}{N} \sum_{x=-x_0}^{N-1-x_0} \sum_{y=-y_0}^{N-1-y_0} f(x, y) e^{-\frac{2\pi i(u(x+x_0)+v(y+y_0))}{N}} \\
 &= \frac{1}{N} \sum_{x=-x_0}^{N-1-x_0} \sum_{y=-y_0}^{N-1-y_0} f(x, y) e^{-\frac{2\pi i(ux+vy)}{N}} \cdot e^{-\frac{2\pi i(ux_0+vy_0)}{N}} \\
 &= e^{-\frac{2\pi i(ux_0+vy_0)}{N}} \cdot \frac{1}{N} \sum_{x=-x_0}^{N-1-x_0} \sum_{y=-y_0}^{N-1-y_0} f(x, y) e^{-\frac{2\pi i(ux+vy)}{N}} \\
 &\stackrel{(*)}{=} e^{-\frac{2\pi i(ux_0+vy_0)}{N}} \cdot \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) e^{-\frac{2\pi i(ux+vy)}{N}} \\
 &= e^{-\frac{2\pi i(ux_0+vy_0)}{N}} \cdot F_f(u, v)
 \end{aligned}$$

אבל הוואיל ו- $\forall x \in \mathbb{R}, \sin^2 x + \cos^2 x = 1$ ו- מתקיים

$$\left| e^{\frac{2\pi i(ux_0+vy_0)}{N}} \right| = \sqrt{\cos^2 \left(e^{\frac{2\pi i(ux_0+vy_0)}{N}} \right) + \sin^2 \left(e^{\frac{2\pi i(ux_0+vy_0)}{N}} \right)} = 1$$

$$\text{לכן } |F_{f_{shifted}}(u, v)| = \left| e^{-\frac{2\pi i(ux_0+vy_0)}{N}} \cdot F_f(u, v) \right| = |F_f(u, v)|$$

(*) נובע מכך שהתמרת פורייה היא פונק' מחזוריות, שכן $f(-x_0) = f(N - x_0)$ ולכן

$$\begin{aligned}
 \sum_{y=-y_0}^{N-1-y_0} f(x, y) e^{-\frac{2\pi i(ux+vy)}{N}} &= \sum_{y=0}^{N-1-y_0} f(x, y) e^{-\frac{2\pi i(ux+vy)}{N}} + \sum_{y=-y_0}^{-1} f(x, y) e^{-\frac{2\pi i(ux+vy)}{N}} \\
 &= \sum_{y=0}^{N-1-y_0} f(x, y) e^{-\frac{2\pi i(ux+vy)}{N}} + \sum_{y=1}^{y_0} f(x, -y) e^{-\frac{2\pi i(ux-vy)}{N}} \\
 &= \sum_{y=0}^{N-1-y_0} f(x, y) e^{-\frac{2\pi i(ux+vy)}{N}} + \sum_{y=1}^{y_0} f(x, N-y) e^{-\frac{2\pi i(ux+v(N-y))}{N}} \\
 &= \sum_{y=0}^{N-1-y_0} f(x, y) e^{-\frac{2\pi i(ux+vy)}{N}} + \sum_{y=N-y_0}^{N-1} f(x, y) e^{-\frac{2\pi i(ux+vy)}{N}} \\
 &= \sum_{y=0}^{N-1} f(x, y) e^{-\frac{2\pi i(ux+vy)}{N}}
 \end{aligned}$$

ובאופן סימטרי נסיק את זה עבור הסכימה ההפוליה. רק נעיר כי מתקיים כי

$$\begin{aligned} e^{\frac{-2\pi i(ux+v(N-y))}{N}} &= \cos\left(\frac{-2\pi(ux+v(N-y))}{N}\right) + i \sin\left(\frac{-2\pi(ux+v(N-y))}{N}\right) \\ &= \cos\left(\frac{-2\pi(ux-vy)}{N} - 2\pi v\right) + i \sin\left(\frac{-2\pi(ux+vy)}{N} - 2\pi v\right) \\ &= \cos\left(\frac{-2\pi(ux-vy)}{N}\right) + i \sin\left(\frac{-2\pi(ux-vy)}{N}\right) \\ &= e^{\frac{-2\pi i(ux-vy)}{N}} \end{aligned}$$

השתמשנו זאת באחד המעברים.

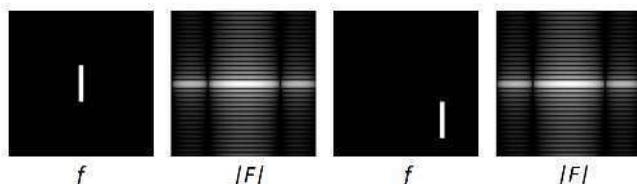
□

בזאת סיימנו את החוכחה.

אינטואיטיבית, המשמעות של $f(x - x_0, y - y_0)$ היא שאנו מזיזים את f ימינה ב- x_0 ולמטה (תלויל לאן פונה ציר ה- y) ב- y_0 , ובאופן דומה עבור F .

מכך ש- $|F_{f_{shifted}}(u, v)| = |F(u, v)|$ אנו למדים כי הזהה של תמונה לא משנה את הערך המוחלט של התמרת הפורייה - את הספקטרום - אך ההתמרה עצמה משתנה. זה כי כל הערכים ישנו רק את **הפaza**, אבל לא את **הספקטרום**.

עבור הזהה של התמרת פורייה, אנו כבר לא נקבל ערכים ממשיים וההתמונה תהפוך למורכבת, שכן אנו לא יכולים להזיז התמרת פורייה.



איור 27: שתי התמונות השמאליות: תמונה וספקטרום פורייה שלה. שתי התמונות הימניות: התמונה המקורי מוזזת, וספקטרום פורייה שלה שלא השתנה. לא נאמר לנו, עם זאת, ש- F - עצמה לא השתנה בהזזה - הפaza השתנה, פשטוט הספקטרום לא.

7 תכונות הפירוק של פורייה

נרצה לחשב את טרנספורם פורייה הדו מימי' באמצעות הגרסה החד מימדי'.

נישום

$$\begin{aligned}
F(u, v) &= \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) e^{-\frac{2\pi i (ux+vy)}{N}} = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) e^{-\frac{2\pi i vy}{N}} \cdot e^{-\frac{2\pi i ux}{N}} \\
&= \frac{1}{N} \sum_{x=0}^{N-1} e^{\frac{-2\pi i ux}{N}} \underbrace{\sum_{y=0}^{N-1} f(x, y) e^{-\frac{2\pi i vy}{N}}}_{F_x(v)} = \frac{1}{N} \sum_{x=0}^{N-1} e^{\frac{-2\pi i ux}{N}} F_x(v)
\end{aligned}$$

כלומר

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

במילים, לכל עמודה אנחנו מחשבים את התמרת פורייה שלה, אז אנחנו רצים ב- $\sum_{x=0}^{N-1}$ על השורות, כלומר עושים התמרת פורייה על השורות.

הערה. (v) הוא מעין היבריד - הוא תלוי באינדקס שורה x ובתדר v , ולא רק שני אינדקסים או שני תדרים. לכן חשוב

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

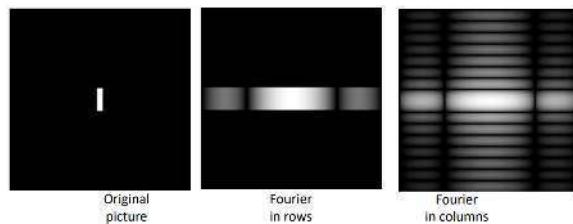
מכאן אנו למדים של מנגנון בוצע התמרת פורייה דו-מימדית, מספיק לדעת לעשות התמרת פורייה חד-מימדית, להפעיל אותה על העמודות, אז על השורות (או להפוך, התהליך סימטרי).

чисוב טרנספורם פורייה דו מימדי באמצעות הגרסה החד מימדית 5 אלגוריתם

- 1 : Compute transform Fourier for each row $\sum_{y=0}^{N-1} f(x, y) e^{\frac{-2\pi i vy}{N}}$ denote the result at a horizontal frequency v and a column x by $F_x(u)$
 - 2 : Compute transform Fourier for each row by using $F_x(u)$: $F(u, v) = \frac{1}{N} \sum_{x=0}^{N-1} e^{\frac{-2\pi i ux}{N}} F_x(v)$
-

נשים לב שבעת חישוב התמרת הפורייה **חד-מימדית** (v) F_x , בהגדירה שלנו אנו לא מחלקים ב- N , בעוד שבעת החישוב של התמרת הפורייה **חד-מימדית** (v) $F(u, v)$ (באמצעות $F_x(v)$) אנחנו כן מחלקים ב- N . כלומר עבור שתי פונק' שמייצגות התמרת פורייה חד-מימדית, פעם חילקו ב- N ופעם לא. עקרונית אנחנו יכולים כן לחלק בשניהם אבל אז יש לשים לב שלא צריך לחלק ב- N בתמרת פורייה ההופוכה.

טענה (ללא הוכחה) ניתן להשתמש בתמרת פורייה חד-מימדית כדי לחשב התמרת פורייה מכל מימד.



איור 28: התמרת פורייה בשורות ובעמודות

נשים לב שכאשר שורה שלמה היא שחורה, כלומר, כלומר 0, אז גם התמרת פורייה היא 0. בנוסף, בפורייה בשורות יש לנו התאפסות בעממיים בשחור (שתי עמודות שחורות) ואז בעמודות יש לנו הרבה יותר התאפסות: המון המון שורות שחורות. כלומר, השינויים בההתמרת הפורייה כאן בכיוון u יותר גדול מאשר ב- x .

תכונות

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

$$F(u, v) = F^*(-u, -v) \bullet$$

$$|F(u, v)| = |F(-u, -v)| \Leftrightarrow \text{כמסקנה גם}$$

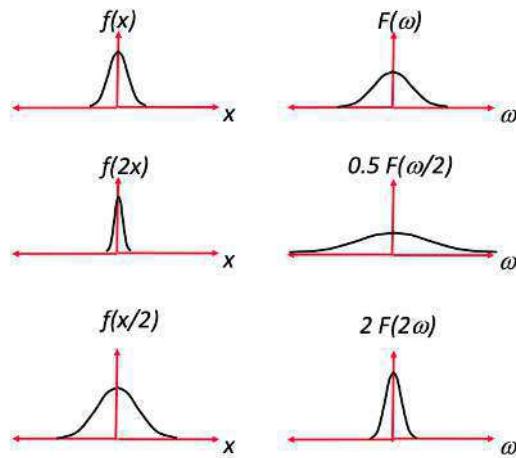
$$\Phi(f_1(x, y) + f_2(x, y)) = \Phi(f_1(x, y)) + \Phi(f_2(x, y)) \bullet$$

$$\Phi(a \cdot f(x, y)) = a \cdot \Phi(f(x, y)) \bullet$$

$$\Phi(f(ax, by)) = \frac{1}{|ab|} \cdot F\left(\frac{u}{a}, \frac{v}{b}\right) \bullet$$

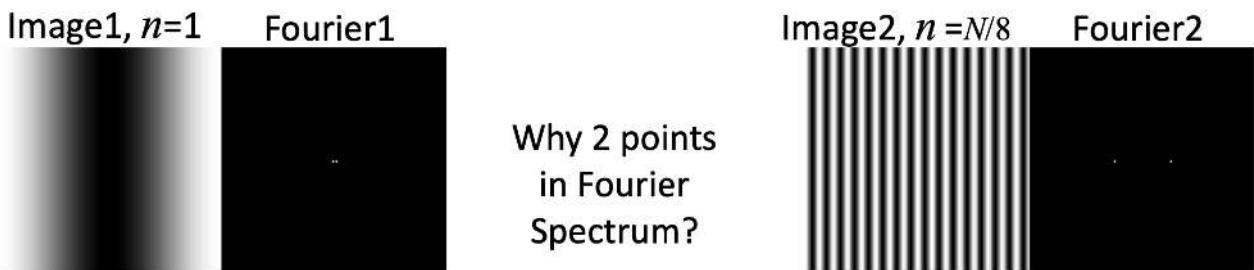
\Leftrightarrow תכונה זו מאוד אינטואיטיבית, הרי אם הקפינו את התמונה בפקטור a, b אנו נראה הכל מכובץ יותר ולכן התדר

יהיה חזק יותר, כלומר $F\left(\frac{u}{a}, \frac{v}{b}\right)$. נביט באיור הבא להמחשה:



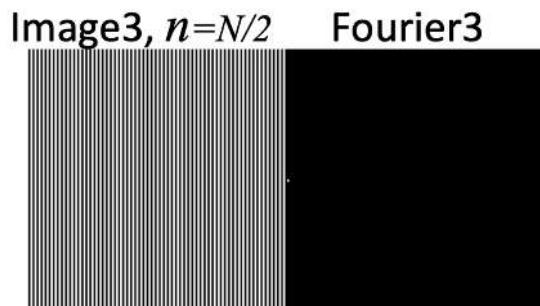
איור 29: כאן כאשר הכפלנו ב-2 קיבלנו תדר גבוה יותר (הת הפרשות גבוהה יותר, לא אמפליטודה של פורייה). כאשר חילקנו ב-2 קיבלנו תדר עם ערכים קרובים יותר לאפס.

נביט באյור הבא כדי לראות איך נראה פורייה של תמונה:



איור 30: התמונה השמאלית היא בתדר של 1, וולכן אנו מקבלים את שתי הנקודות. התמונה הימנית היא בתדר 8 ואנו מקבלים תדר גבוה יותר ועוד הפעם שתי נקודות. מדוע שתי נקודות? נזכיר כי $F^*(v, -u) = F(u, v)$ וולכן קיבל גם את האיבר הצמוד וגם את האיבר הרגיל, שסכומם ביחס יצמצמו אחד את השני כשבנוסף את הטרנספורמציה ההיפוכית.

لتوضעה זו יש מקרה קטן מיוחד:



איור 31: מדוע כאן יש רק נקודה אחת? כאן יש לנו תדר שהוא $\frac{N}{2}$ ולכן הפורייה הוא $\cos(\pi x) + i \sin(\pi x) = \cos \pi x$ ומכאן אנו מקבלים כי הצמוד שווה לאיבר המקורי.

7.1 נגזרת של תמונה

כיצד נגזרת תמונה? מדובר בפונקציה דיסקרטית ולכן הדבר בעיתי שכן היא לא רציפה. יחד עם זאת, באמצעות הצגה עם

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

$$\begin{aligned} f'(x) &= \sum_u F(u) e^{\frac{2\pi u i x}{N}} \cdot \frac{2\pi i u}{N} \\ &= \frac{2\pi i}{N} \sum_u u F(u) e^{\frac{2\pi u i x}{N}} \end{aligned}$$

כאשר הגזירה כאן היא כמובן לפי x .

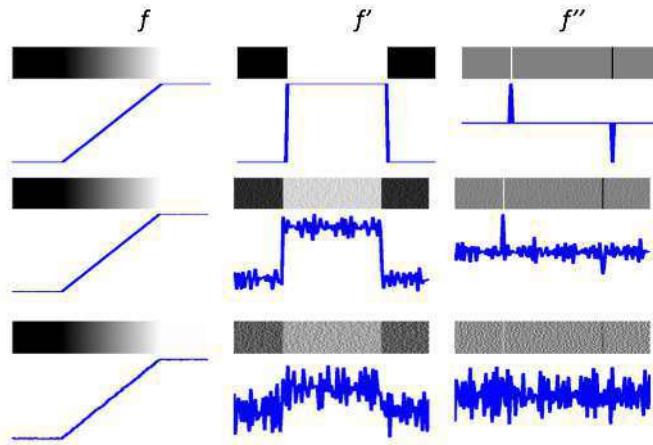
מה קיבלנו? קיבלנו למעשה ביטוי מאוד קרוב לטרנספורם פוריה ההפוך הרגיל, רק שאנו כופלים ב- $-u$ כל תדר. מה זה אומר? תדרים גבוהים, כמו רעשים למשל, יהיו קרייטיים כאשר נחשב נגזרת. במקרה אחרות קיבלנו כי $(x)'$ היא מuin טרנספורם פוריה ההפוך שימושו 매우 מרעים. באותו האופן עבור פוריה דו מימדי ניתן לגזר לפיקס או לפיבר ולקבל כפל ב- $-u$ או ב- $-v$ בהתאם ובכך רגשות לרעשים מכל אחד מהמים בתאמה. מכאן נסיק את האלגוריתם הבא:

Algorithm 6 חישוב נגזרת של תמונה

- 1 : x Derivative of f
 - 2 : Compute the Fourier transform F
 - 3 : Multiply Fourier coefficient $F(u, v)$ by $\frac{2\pi i}{N}u$
 - 4 : Compute the inverse Fourier transform
 - 5 : y Derivative of f
 - 6 : Compute the Fourier transform F
 - 7 : Multiply Fourier coefficient $F(u, v)$ by $\frac{2\pi i}{N}v$
 - 8 : Compute the inverse Fourier transform
-

נבחן כי לאחר הגזירה $DC = F(0)$ מתאפס.

נרצה לתת המלצה למשמעות הרעשים בגרף הנגזרת:

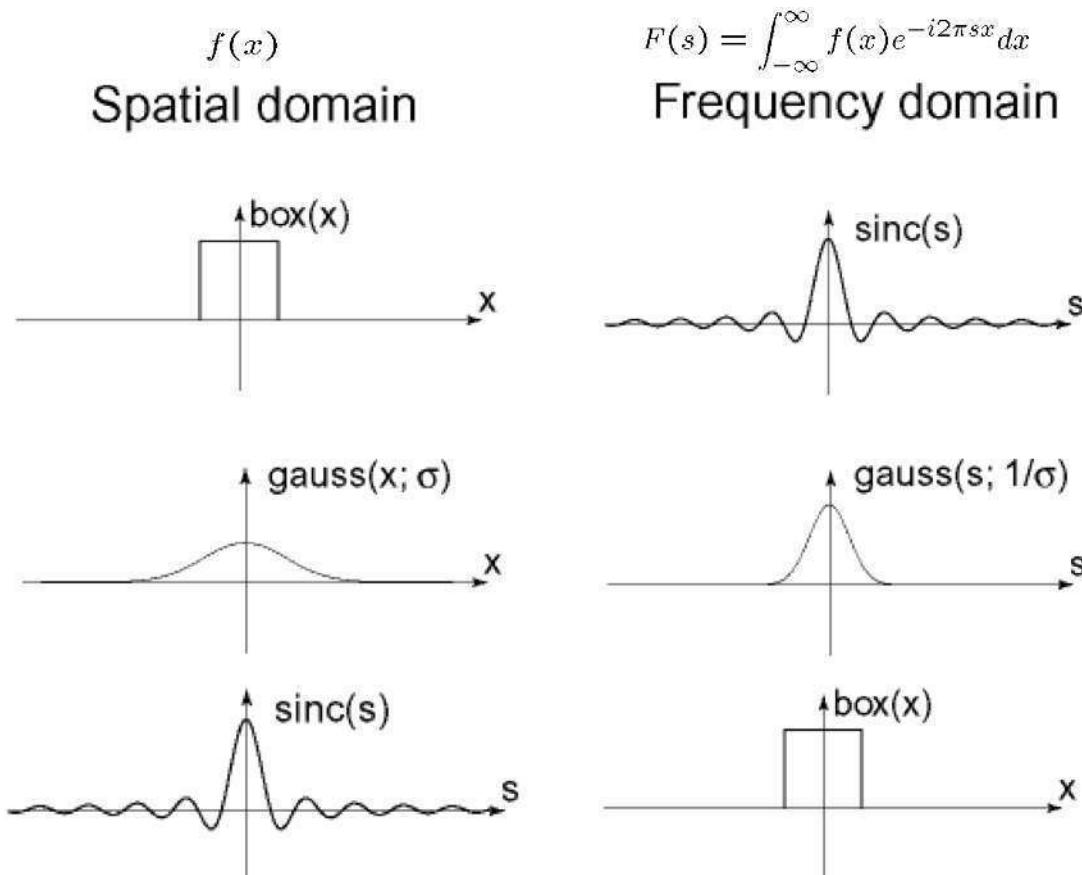


איור 32: המונה הראשונה ללא רעשים, בעלת נגזרת ללא רשי רקע. התמונה השניה עם מעט רעשים, בעלת נגזרת עם יחסית הרבה רעשים בתדריות גבוהה. התמונה השלישית עם הרבה רעשים, בעלת נגזרת עם המון רשי רקע. הנגזרות השניות מושפעות מכך שימושותית וכן כל נגזרת מסדר a תושפע אף יותר. במילים אחרות, הנגזרות מגבירות את הרעשיס! על כן, כאשר מפעילים אופרטורים בתמונות צריך להיזהר.

7.2. שינוי טרנספורם פוריה לאחר טרנספורמציה על התמונה

נציג עתה תמונה $RECT$, שהיא תמונה שחורה עם בלוק לבן בחלק ממנה.

מתתקבל כי הפוריה הוא מעין גאוסיין המוגדר כ- $\text{sinc}(x) = \frac{\sin(\pi\omega)}{\pi\omega}$. זה נראה בדיקות כך:



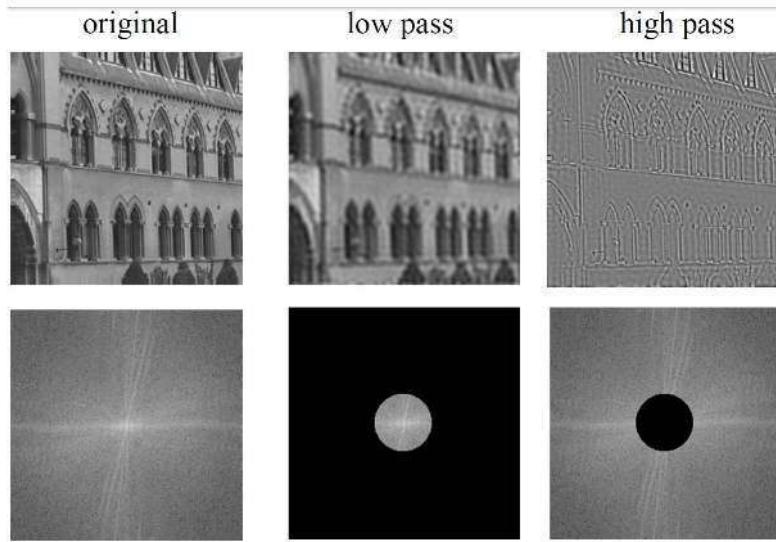
איור 33: לעללה ניתן לראות את הגרסה הרציפה של טרנספורם פוריה, אנה שכחו אותה. מדובר הטרנספורם פוריה של sinc הוא rect ? ראיינו כי ביצוע טרנספורם פוריה משקף את הפונקציה, אך מכיוון ש-rect סימטרית קיבל אותו הדבר.

או לעומת זאת נרצה להיפטר מתדרים גבוהים או נמוכים. לשם כך השתמש בפילטרים שונים:

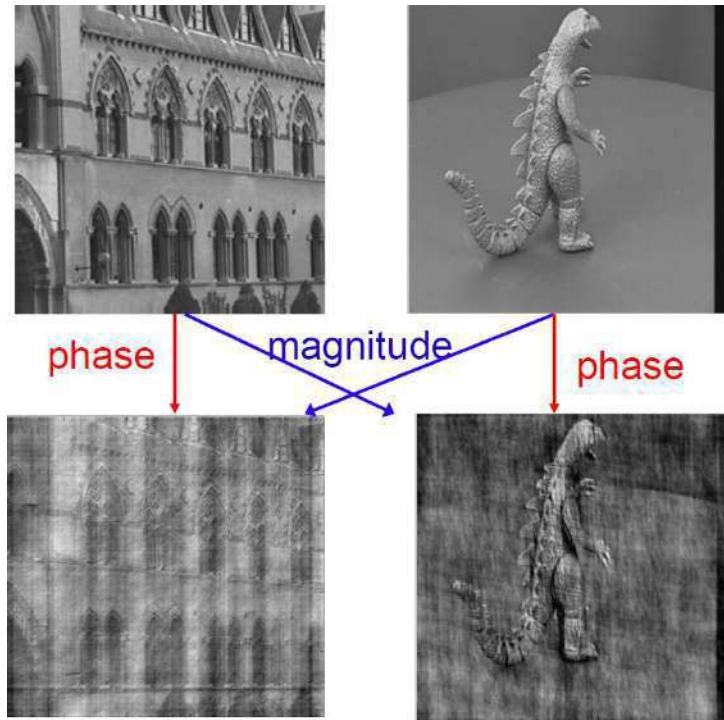
low - pass - נחליש את התדרים הגבוהים שהתקבלו בטרנספורם פוריה. ככה קיבל תמונה מטושטשת.

high - pass - נחליש את התדרים הנמוכים שהתקבלו בטרנספורם פוריה. ככה קיבל תמונה עם יותר דיוק אך נאבד תדרים נמוכים.

نبיט בתמונה הבאה להמחשה:

איור 34: המראה *high/low pass*

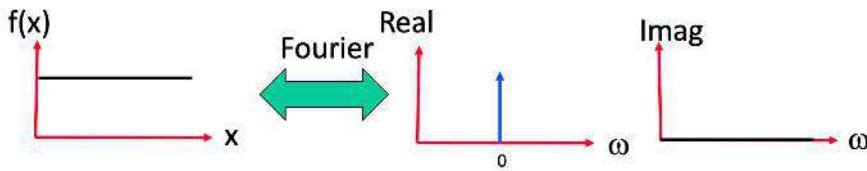
יתר על כן, את התמונה מייצגת גם מגנטודה וגם פאזה. כמו שאמרנו קודם, המגנטודה לא משתנה לאחר היסט, ולכן לא מספיק לייצג תמונה איתה. צריך גם את הפאזה, נביט בתמונה הבאה להמחשה:



איור 35: התמונה השנייה שמורכבת משילוב של הפאזה שלה אבל מגנטודה של תמונה אחרת עדין מובנת, למראות שהרבה פחות. התמונה השנייה באוטו האופן התמונה השנייה גם יחסית ברורה. זה רק מדגיש כמה הפאזה חשובה ביצוג התמונה, למראות שאינה ויזואלית.

דוגמה. מהו הפוריה של $f(x) = 1$? זו פונקציה קבועה ולכן הממוצע הוא 1 ולכן אופקי או

אנכי ולקן קיבל אך ורק קפיצה ב-0.



אייר 36: התמונה השנייה שמורכבת משילוב של הפאזה שלה אבל מגנטודה של תמונה אחרת עדין מובנת, למרות שהרבה פחות. התמונה השנייה באוטו האופן התמונה השנייה גם יחסית ברורה. זה רק מדגיש כמה הפאזה חשובה בייצוג התמונה, למרות שאינה ויזואלית.

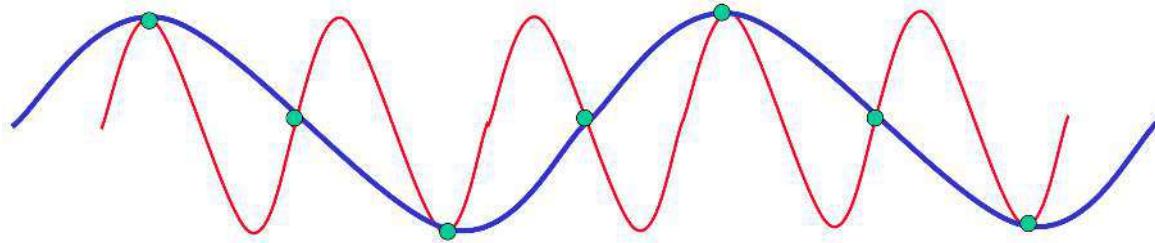
שאלת נשאלת שאלה, למה משתמשים בטרנספורם פוריה בעיבוד תמונות?

תשובה לא משתמשים, אבל זה עוזר להבין את החומר.

Aliasing 7.3

לעתים אנו דוגמים מעט מדי מתחוםנו ואז כשאנו מחשבים פוריה אנו מפסידים תדרים רבים ומתקבלים בסוף תוצאה לא טובה.

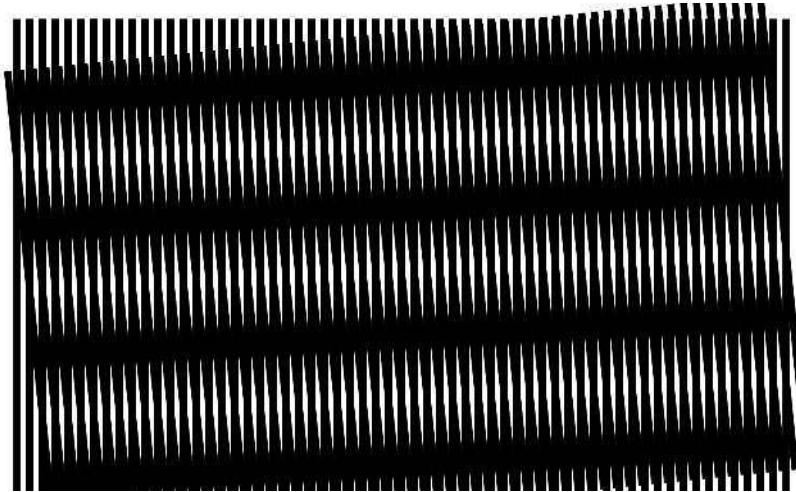
נביט בדוגמה הבאה:



אייר 37: הדגימות מרמזות לגל סינוס אחר בתדרות נמוכה יותר, למרות שהתדר המקורי היה בעל תדרות גבוהות יותר.

כדי להבטיח שזה לא קורה, נדגם מספק, כמה מספיק? אם אנו רוצים תדר מקסימלי f נדגים $f/2$ דגימות לפחות, ראיינו זאת. לדוגמה משמעות גדולה, למשל, אם אנו מסוגלים לראות אובייקטים בתמונה גם אם נקטין אותה פי 10 נעדיף להקטין אותה כי זה יחסוך זמן ריצה, אך כאן צריך להיזהר מאד. נניח שביצענו הקטנה פי 10, יש עדין תדרים גבוהים שאנו עלולים לאבד בתופעת *aliasing*, ולכן נרצה להיפתר מהם. כיצד? נחשב את הפוריה ונאפס את התדרים הגבוהים. כיצד נקטין פי 10? יש מי שיעשיך לדגום כל פיקסל שני, אך אנו עלולים לאבד המון מידע, למשל אם הפיקסלים הם שחור לבן, נקבל תמונה שחורה בלבד או לבן בלבד במקום אפורה. לכן אנו קודם נפתרים מהתדרים הגבוהים ורק לאחר מכן דוגמים כראוננו.

בטלויזיה למשל לא רוצים שאנשים יגעו עם חולצות פסים:



איור 38: אם לא דוגמים מספיק טוב מקבילים אפקטים כאלה (אפקט moire).

מדוע? בדיק בಗל תופעה זו, יש דגימה חלקית שמשנה את איך שהחולצה נראית בפועל. כדי להקטין סיגנל השתמש בפוריה:

הקטנת סיגנל 7

- 1 : Compute Fourier (N samples to N coefficients)
 - 2 : Bring $u = 0$ to center (FFT shift)
 - 3 : Crop Fourier from N to $\frac{N}{2}$ coefficients -
remove $\frac{N}{2}$ high frequencies // now there are only $-\frac{N}{4}$ to $\frac{N}{4} - 1$
 - 4 : Compute Inverse Fourier ($\frac{N}{2}$ coefficients to $\frac{N}{2}$ Samples)
-

כדי "להקטין תמונה" השתמש באלגוריתם זה:

הקטנת תמונה 8

- 1 : Blur and Subsample every 2^{nd} pixel in every 2^{nd} row
 - 2 : Bring $u = 0$ to center (FFT shift)
 - 3 : Reduce Fourier signal
 - 4 : Compute Inverse Fourier ($\frac{N}{2}$ coefficients to $\frac{N}{2}$ Samples)
-

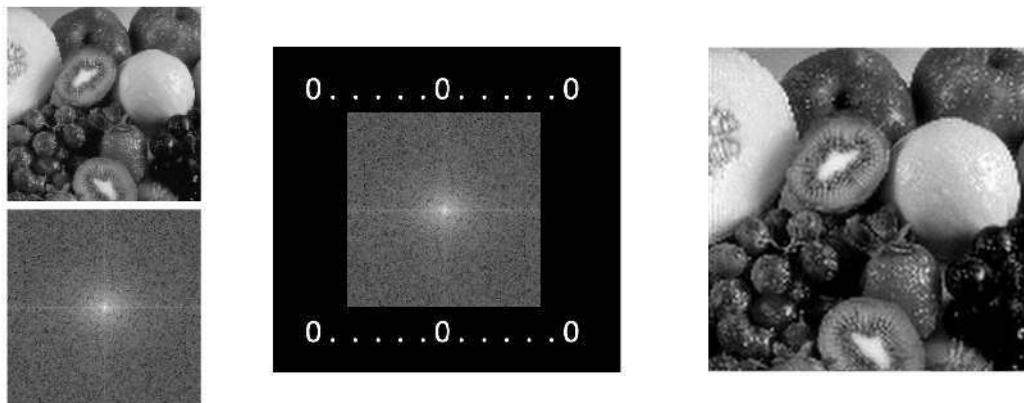
כמובן עולה השאלה מה קורה לאחר שנחשב את הטרנספורם הפוך, האם נאבד רמת דיוק? כו, אך זה יהיה פחות מוגש ממחיקת $\frac{N}{2}$ והישארות עם N דוגימות.

להלן אלגוריתם להגדלה של תמונה באמצעות ריפוד תדרים גבוהים ב-0:

Algorithm 9 הגדלת תמונה

-
- 1 : Compute Fourier ($N \times N$)
 - 2 : Pad Fourier with zeros (e.g. $2N \times 2N$)
 - 3 : Compute Inverse Fourier ($2N \times 2N$)
-

להלן דוגמה לפלט לאחר הפעלת האלגוריתם:



איור 39: ריפוד הפורייה באפסים ולקיחת התמורה הההפוכה נוותנת לנו הגדלה של התמונה המקורי

קונבולוציה

הרצאה 4

קונבולוציה היא אופרטור שמופעל על התמונה בצורה אחידה, כלומר אנו לא לוקחים תוצאה מחישוב קודם על פיקסל מסוים, אלא מחשבים על כל הפיקסלים בתמונה. למשל, הפיכת כל פיקסל לממוצע הפיקסלים:

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

לקחנו תמונה g ועבורי כל פיקסל y, x הפעלו משקלות וקיבנו טשטוש. כאשר $f = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$. למשל, ניתן את פונקציית הזזה. בעצם כל אייר במטריצה f מייצג משקלות לפיקסל מסביבנו. יתר על כן, אנו רוצים שיתקיים $\sum_{i,j} f(i, j) = 1$ שכן איןנו רוצים לחזור מטוחה התמונה.

שאלה מהו ממוצע התמונה לאחר ביצוע קונבולוציה?

תשובה נשאר אותו דבר כי סכום המשקלות תמיד אחד.

הערה. כאשר אנו מבצעים קונבולוציה עם קרנל שיוצר ממוצע אנו מטשטשים את התמונה, כאשר אנו משקללים כל פיקסל אנו

$$\frac{1}{5} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

יצרים ממוצע משוקלל שמטשטש פחות, כמו גאוסיין. הקונבולוציה הבאה מטשטשת

שאלה למשל, עבור $(f * g)(x, y) = ?$ זיהוי

תשובה זיהוי זהה של g . ניתן לחשב ולראות כי $h(6) = 1, h(7) = -1$

כשמחשבים קונבולוציות צריך להחליט איך מטפלים בגבולות:

- אם f מוגדרת על $[5, \dots, 0, -1] f(-1) = ?$ מהי?

↳ נציג $f(-1) = 0 = f(9)$ כלומר מוחז לקרנל הכל אפס.

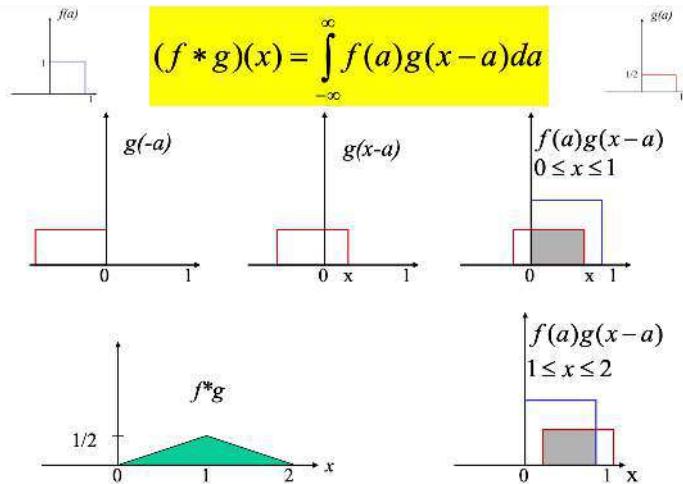
↳ נוכל להציג ציקליות כלומר $f(-3) = f(9) = f(3) \pmod{6}$ החישוב של האינדקס הוא $9 \pmod{6}$.

↳ או שיקוף, כלומר $f(-a) = f(a)$

הkonvolוציה יכולה להיות רציפה:

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

השיקוף מאפשר לנו לבצע זהה של הKernel על התמונה ולסכום כאשר בכל פעם שעוברים לנקודה הבאה איזים עוד קצת ימינה:



איור 40: כאן השיקוף של g ביחס ל- y מאפשר להציג אותו בכל פעם כאשר עוברים ל- $-x$ גדול יותר. במקרה זה יש לנו קונבולוציה שהיא rect על והותzáה היא משולש, שכן כאשר $x = 1$ קיבלנו שטח מקסימלי (החלק האפור) בין שתי הפונקציות, כאשר $x > 1$ השטח קטן.

7.4 משפט הקונבולוציה

משפט. תהיו f, g שתי תמונות, אז $(f * g) = F * G = \Phi(f) \cdot \Phi(g)$. כלומר גם מתקיים $\Phi(f \cdot g) = \Phi(f) \cdot \Phi(g)$.

זה מזכיר את סיבוכיות החישוב $n \log n$ עם fft .

מכאן נסיק כי במקומות לבצע פילטר עם קונבולוציה, נוכל לעשות פורייה.

תבוננות

1. קומוטטיביות: $f * g = g * f$
2. אסוציאטיביות: $f * (g * h) = (f * g) * h$
3. דיסטריביטיביות: $f * (g + h) = f * g + f * h$
4. מכאן נסיק כי זו פעולה לינארית מעל סדרות חד ממדיות או מעל תמונות דו ממדיות. לכן ניתן להציג כל קונבולוציה כמטריצה.

דוגמה. כיצד נציג את הקונבולוציה $(1, 2, 1)^T$ על $\frac{1}{4}(x_1, x_2, \dots, x_6)$ כלומר להפעיל עליהם קוונטולוציה, על כן נפעיל לפי השיטה הציקלית. נשלים את ה الكرnel $-\frac{1}{4}(1, 2, 1, 0, 0, 0)$. ונקבל את המטריצה

$$\frac{1}{4} \begin{bmatrix} 2 & 1 & 0 & 0 & 0 & 1 \\ 1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 & 2 & 1 \\ 1 & 0 & 0 & 0 & 1 & 2 \end{bmatrix}$$

זו מטריצה שכל שורה שלה היא שיפט ימינה של העמודה הקודמת.

7.5 קוונטולוציה דו מינימלית

$$h(x, y) = \sum_{k=0}^n \sum_{l=0}^m f(k, l) g(x - k, y - l)$$

כאן אנו למשה משקפים את ה الكرNEL סבב ציר x לאחר מכן סבב ציר y (הסדר לא משנה) ואז שמים את האמצע של ה الكرNEL על האיבר הראשון בתמונה, סוכנים, מתקדים קדימה וכן הלאה. כאשר אנחנו יוצאים מגבולות ה الكرNEL או התמונה או מרפידים עם אפסים.

$$\cdot \begin{bmatrix} 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

למשל,

בנוסף,

$$\begin{aligned} \begin{bmatrix} 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \end{bmatrix} &= \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \\ \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} &= \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \\ \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} &= \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \end{aligned}$$

כמו מתקדים ביןומים! למה זה טוב? כי אפשר לחשב אותן ביעילות על ידי ה קודמים עם זהות פסקל. יתר על כן, הוא למעשה rect, לכן אנו מסוגלים לחשב rect ביעילות.

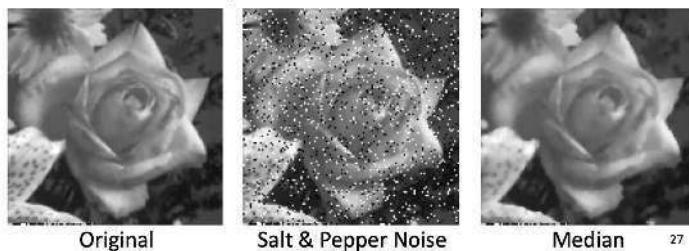
קוונטולוציות מסווגות להוריד רעשים על ידי טשטוש של התמונה. אחת הדרכים היא לעשות קוונטולוציה עם rect, אבל

במשפט הקונבולוציה זה כמו לעשות פורייה הפוך על מכפלות הפוריריה שזה בדיק כמו לעשות פורייה הפוך הכפלה ב- $sinc$ שכן

$$\Phi(\text{rect}) = \text{sinc}$$

טשטוש כאשר אנו בוחרים קרנל גדול יותר שמליטש יותר נקבל תמונה יותר מטושטשת. כי אנו בוחרים ממוצע של כל השכנים (והרחוקים מהם).

חציון ניתן לנוקות רעש באמצעות חציון שכן הוא מושפע פחות מערכי קיצון, ככלمر נבחר את החציון של כל קבוצה פיקסלים במקומות הפיקסל עצמו (הקבוצה היא השכנים של הפיקסל). ככה אנו מקבלים פחות רעש.



אייר 41: פיזרנו רעש על התמונה והחציון הפחית אותו משמעותית ביחס לממוצע שהוא מורה אותו על התמונה.

לכן כאשר נרצה לנוקות רעש:

- ממוצע - נאבד מידע ונקבל מריחה.
- חציון - נדרש ערכי קיצון

7.6 נגזרת באמצעות קוונבולוציה

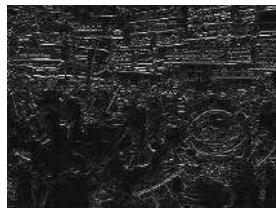
הנגזרת מוגדרת ע"י

$$\frac{\partial f(i,j)}{\partial x} = \lim_{\varepsilon \rightarrow 0} \frac{f(i,j) - f(i-\varepsilon,j)}{\varepsilon} \approx f(i,j) - f(i-1,j) \approx \frac{f(i+1,j) - f(i-1,j)}{2}$$

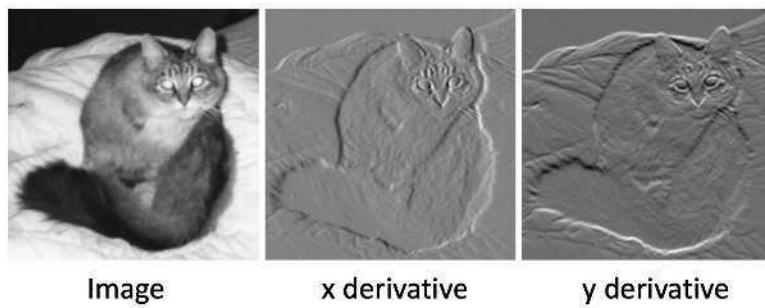
שכן בעולם הדיסקרטי המינימום ל- ε הוא 1 (אין משמעות לкопיצה של חצי פיקסל, התמונה מוגדרת על ערכים שלמים בלבד).

נבחן כי דבר זה ניתן להשיג על ידי קוונבולוציה עם $\begin{bmatrix} 1 & -1 \end{bmatrix}$, המינוס אחד מימין בגלל השיקוף. או במקרה השני $\cdot \frac{1}{2} \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$.

הנגזרת מורידה את דרגות האפור של התמונה מטה וכן נקבל תמונה שחורה יותר, אבל עם מקומות של מעברים חדים בולטים יותר, שכן בדרך כלל הפיקסלים דומים אחד לשני, אבל במקרים חדים ההפרש של הנגזרת יהיה מספר גדול בערכו המוחלט, לכן נראה הבדל.

איור 42: זו הנגזרת של התמונה לאחר שהפעלנו עליה $ReLU$ כלומר זרקנו ערכים שליליים.

לא זריקת ערכים אלה, אפשר לקבל מצבים כאלה:

Zero is grey. No absolute value. 35

איור 43: מעברים חדים מוגדים בתמונה

שאלה מה הממוצע לאחר הנגזרת?

תשובה בתמונה רגילה, כל נגזרת שעולה, גם יורדת לאחר מכון ולכון נקבל ממוצע אפס. מעבר לכך, אפשר להבחן כי $f(i, j) - f(i, j - 1) - f(i, j - 2) - f(i, j) + f(i, j - 1) - f(i, j - 2)$ מלבד בעמודה הראשונה בה אין אפשרות להחסיר, אלא אם בוחרים להחסיר אפס או את העמודה האחורונה בצורה ציקלית. מכאן נקבל כי קיבל את סכום איברי העמודה האחורונה, אלא אם החישוב הוא ציקלי.

לפנינו שאנו גוזרים אנו צריים לוודא שאין רעשין, כי הנגזרת מגדילה רעשין כידוע. על כן יש קוונטוליזית Sobel שמפחיתה רעשין בחישוב הנגזרת. למעשה אנו מבצעים גאוסיין על התמונה בכיוון הניצב לכיוון הנגזרת ואז מסויציאטיביות אנו מחשבים

את הקוונטוליזה של הגאוסיין והנגזרת ורק אז מחשבים על התמונה. למשל עבור ציר x : נקבל גאוסיין $\frac{1}{4} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$ ונגזרת

$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$. כאן נבחין כי החישוב הוא כמו לעשות קוונטוליזה עם $\frac{1}{2} \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$ והקוונטוליזה אחד על השני תנתן

הגאוסיין על התא הראשון בנגזרת, לעבור לתא הבא, ולהרכיב מהעמודות מטריצה. לא לשוכח לנרטל ב- $\frac{1}{8}$ שנובע מה- $\frac{1}{4} \cdot \frac{1}{2}$.

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

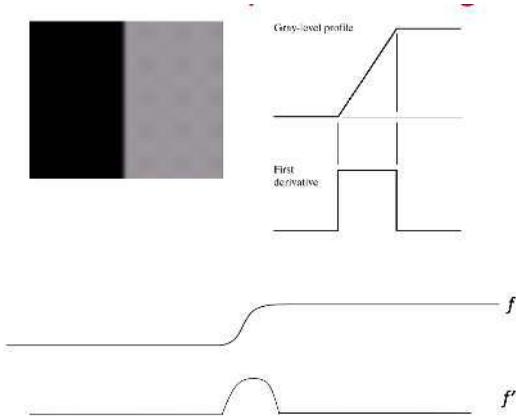
עבור ציר y קיבל באופן דומה

$$\cdot \frac{\partial f}{\partial x} = [1, -1], \frac{\partial f}{\partial y} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

זאת בהשוויה לנגזרות הרגילים

Edges 7.7

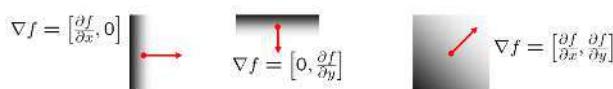
בתמונה יתכו מעברים חדים בין מקומות ונרצה לזהות אותם כדי להגבר את קונטראסטו. כדי לזהות זאת משתמש בנגזרת.



איור 44: מעבר בין שחור לאפור, הנגזרת מקבלת ערך קיצון

ננצל את השינוי הזה בנגזרת כדי להגדיר משחו חזק יותר, הגרדיאנט.

גרדיאנט מוגדר להיות $\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right)$. הוא למעשה אומר לנו לאיזה כיוון מקסימלי התמונה משתנה.



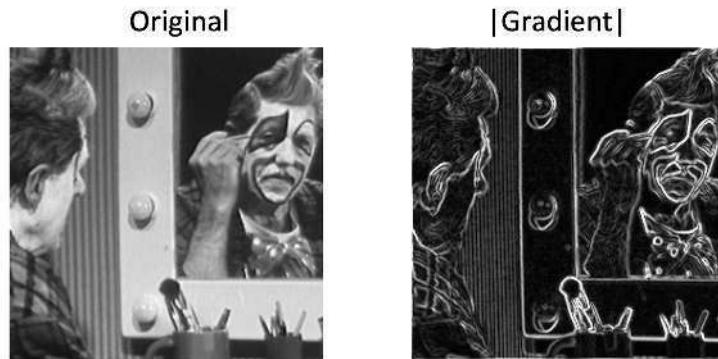
איור 45: שינוי משחזר לבן הוא חיובי

לגרדיאנט כמורן יש מגנטודה וכיוון:

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

$$\alpha = \arctan \left(\frac{\left(\frac{\partial f}{\partial y}\right)}{\left(\frac{\partial f}{\partial x}\right)} \right) \bullet$$

כמו הנзорות, רק יותר, הוא יבליט לנו שינויים חדים בתמונה. נציג את המגניטודה למשל:



אייר 46: המגניטודה של הגרדיאנט מציגה לנו שינויים חדים בתמונה. כאן אין ערכים שליליים ושהור הוא 0.

7.8 לפלייאן

הנגזרת השנייה מציגה רעשים בצורה חזקה יותר ומאפשרת לזהות *Edges* טוב יותר מהנגזרת השנייה. נחשב נגורת זו

$$(1, -1) * (1, -1) = (1, -2, 1)$$

$$\begin{pmatrix} 1 \\ -1 \end{pmatrix} * \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}$$

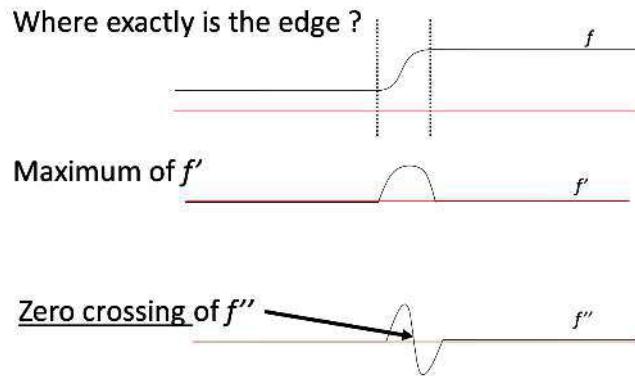
ונקבל שסכום הנגורות (עם ריפוד באפסים) הוא

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

לסכום הנגורות זהה אלו קוראים לפלייאן.

פלסייאן מוגדר להיות $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

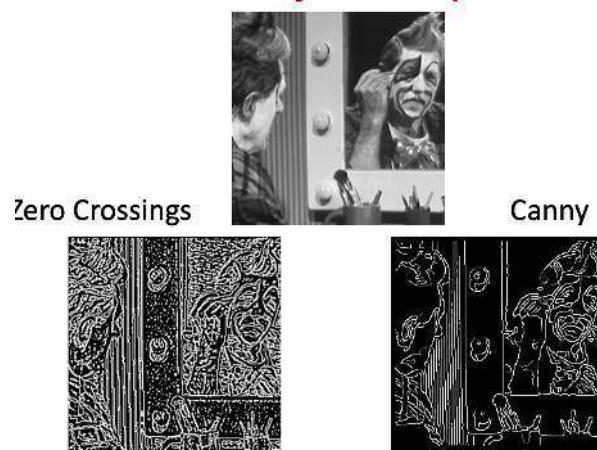
עם הצגה אחרת של הנגורות. צורה אלטרנטיבית היא $\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$



איור 47: הנגזרת השנייה מציגה בדיק את המעבר בין הצבעים. נקודת זו נקראת *Edge* ואנו מכנים תופעה זו *Zero-Crossing*

על כן נרצה ליזהות נקודות אלה. הבעיה היא שהנגזרת השנייה מאוד רועשת ולכן נדרש, למשל באמצעות גאוסיין. אבל עדיין, אנו מפסידים מידע מעבר לכך, הנגזרת השנייה מציגה קווי גובה של התמונה ולכן תמיד תנתן לולוות סגורות.

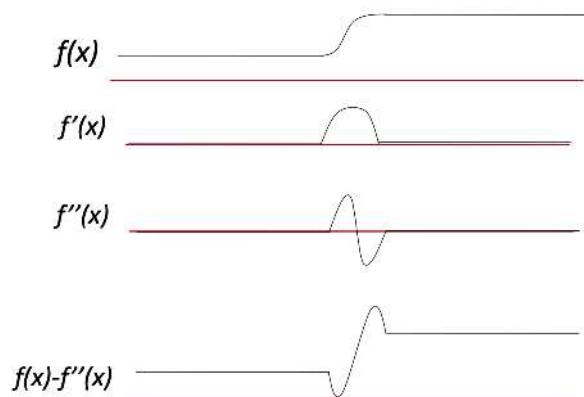
כדי לבצע זאת بصورة מיטבית יש את האלגוריתם של *Canny*. באמצעותו קיבל תוצאה טובה:



איור 48: מקבלים לא רק לולוות, אלא קוים שלמים של *Edges*, ללא רעשים. ככה זיהינו *Edges*

7.9 חיזוד באמצעות לפלייאן

הינו יכולים גם להחסיר את הלפלסיאן, שכן ההפרש בין הפונקציה לנגזרת השנייה מגדיל הפרשיים בין נקודות רחוקות שאנו לא מסוגלים להבדיל ביניהן:



איור 49: נוצר הפרש גדול יותר מה שמנגדיל את הניגוד ויוצר חידוד

התוצאה תראה כך:

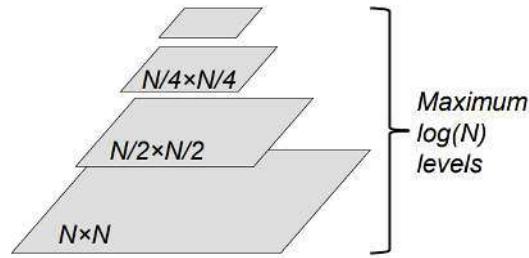


איור 50: חידוד התמונה באמצעות החסרת פלסיין

פירמידות תמונה

הרצאה 5

הרעין בפירמידה של תמונות הוא לקחת תמונה ולהקטין אותה כל פעם פי 2 ברוחב ובאורך, כלומר פי 4 סה"כ. מכאן שגם גודל התמונה המקורי היה $N \times N$ הרי כמות התמונות המksamילית שנוכל לקבל בפירמידה היא $\log_2 N$, כאשר התמונה האחרונה תהא בגודל 1×1 .



איור 51: המיחה לפירמידה

מספר הpixels בתמונה יהיו בקירוב

$$N^2 + \frac{1}{4}N^2 + \frac{1}{16}N^2 + \dots + 1 = 1\frac{1}{3}N^2$$

כפי סכום של סדרה הנדסית, $S_n = \frac{a_1 \cdot (q^n - 1)}{q - 1}$, מתקיים

$$1 + \frac{1}{4} + \frac{1}{4^2} + \dots + \frac{1}{4^{\log_2 N}} = \frac{\left(\frac{1}{4}\right)^{\log_2 N + 1} - 1}{\frac{1}{4} - 1} = \frac{4}{3} \cdot \left(1 - \left(\frac{1}{4}\right)^{\log_2 N + 1}\right) \xrightarrow{n \rightarrow \infty} \frac{4}{3}$$

הערה $1\frac{1}{3}N^2$ זה לא צזה רחוק מ- N^2 , שכן יצירת פירמידה לא תזבז לנו המון מקומם.

שאלה ומה בכלל אנחנו צריכים פירמידות?

תשובה אפשר לדוגמה להשתמש בפירמידות לשם חיפיש:

אם ברצוננו למצוא אובייקט בגודל 32×32 בתמונה שהיא 128×128 , אז ההתאמה בכפלים (כמות המיקומות בהם האובייקט יכול להופיע) הוא 2^{26} . אך אם נבנה פירמידה, ונחפש את האובייקט בגרסתו המוקטנת בתמונה הקטנה,

נוצרך לחפש פחות, וכך נדע את האзор בו נמצא האובייקט בתמונה הגדולה, ונדע שם צריך לחפש. דבר זה יחסוק לנו המון כח חיפוש.

דוגמה/המחשה גוגל ארת: אם נרצה למצוא את בנין רוטברג בתמונה של כדור הארץ, לא נמצא אותו בחיים. אז נסתכל על תמונה מוקטנת, נלק לマזראת התיכון. אז נלק לישראל ואז ירושלים ואז האוניברסיטה, ולבסוף נמצא אותו.

איך בונים פירמידה?

נזכיר איך משנים גודל של תמונה.

הקטנה:

1. טשטוש

(א) לדוגמה קונבולוציה עם קרנל כלשהו.

$$\text{אפשר קרナル } 3 \times 3 \text{ כמו } \begin{pmatrix} \frac{1}{2} & \frac{2}{4} & \frac{1}{2} \\ \frac{1}{2} & \frac{4}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{2}{4} & \frac{1}{2} \end{pmatrix} \text{ או אפלו יותר גדול.)}$$

2. נדגם כל פיקסל שני בכל שורה שנייה

נשים לב שאת קונבולוציות הטשטוש המסויימות שהראינו כאן ניתן להציג את קרナル אופקי קונבולוציה עם קרナル אנכי. זה עדיף על קונבולוציה עם מטריצה 5×5 , כי לעשות קונבולוציה אופקית עולה לנו 5 כפלים, ואז קונבולוציה אופקית גם 5 כפלים - ביחד $5 + 5 = 10$ כפלים. קונבולוציה עם מטריצה 5×5 עולה 25 כפלים.

הגדלה:

הגדלה היא תהליך שהפוך במוחותנו להקטנה, אך גם כמעט הפוך לו באלגוריתם:

1. בין כל שני פיקסלים נשים אפס

2. נטשטו (כדי לקבל תמונה נורמלית, עם האפסים זה יראה מוזר)

(א) שימו לב שבטשטוש בהקטנה רצינו שסכום האיברים בקרナル יהיה שווה לאחד בכל קונבולוציה (נירמול). כאן לא נרצה את זה כי כל פיקסל שני הוא אפס: אם נרמל התמונה תהיה מאד כהה, אז לא נעשה זאת.

(ב) דוגמה: אם נעשה זירו-פאידיניג ואז טשטוש עם $\left(1, \frac{1}{2}, \frac{1}{2}\right)$ נקבל (מחישוב ישיר) בכל מקום שיש בו אפס את ממוצע של שני הפיקסלים לידו, ובכל מקום שאין בו אפס נקבל את הערך המקורי. لكن תוצאה זו תהיה טוביה.

משתמשים בדר"כ בקרנלים ביןימאליים לטשטוש, יותר קלים להבנה וחישוב מנגאוסיין ואסימפטוטית הם מאוד דומים.

בנוספּ, קרナル ביןומי הוא קונבולוציה של (1, 1) עם עצמו המונע פעמיים.

$$\begin{array}{ccccccc} & 1 & 2 & 1 & & /4 \\ 1 & 4 & 6 & 4 & 1 & & /16 \\ 1 & 6 & 15 & 20 & 15 & 6 & 1 & /64 \end{array}$$

$$(1\ 1)\ * \dots^{2k} \ * \ (1\ 1) \ /2^{2k}$$

אליה המקדים הבינוימים האי-זוגיים. עם המקדים הזוגיים בדר"כ לא עובדים כי אין להם איבר אמצעי. במקדים האי-זוגיים האיבר האמצעי יהיה אינדקס 0 שלו.

מה עושים בקצוות? אף פעם לא ציקלי! בפירמידה לא עושים ציקליות, זה טוב בפורייה אבל לא כאן. מה כן נעשה? אפשר לשחק. אם לדוגמה סט הנתונים שלנו הוא

0 2 0 4 5

נוכל **לשקר סביב האיבר האחרון**, ואז האיבר הראשון הראשון שיתווסף משמאל יהיה 2:

...4 2 0 2 4 5

או שנוכל **לשקר אחרי האיבר האחרון**, ואז האיבר הראשון שיתווסף יהיה 0:

...4 2 0 0 2 4 5

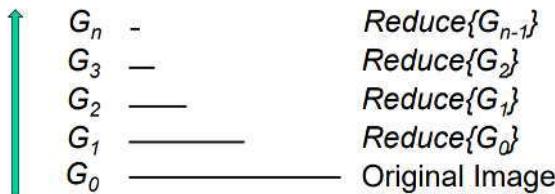
או שאפשר **לשבל את האיבר האחרון**:

...0 0 0 2 0 4 5

כמובן שצורך לטפל במקרה גם מצד ימין אך לא המחשנו זאת כאן. השורה התחתונה היא שיש המונע דרכיהם לטפל בקצוות, כמו תמיד אין אחת "נכונה" אבסולוטית.

8 פרמידה גאוסיינית

פרמידה גאוסיינית היא הקטנה איטרטיבית של התמונה. נגדיר את הפעולה $\text{Reduce}\{im\}$ להיות הקטנה של im כפי שתארנו קודם. הפרמידה הגאוסיינית תראה כבאיר.



איור 52: פרמידה גאוסיינית

הערה נזכיר שוב שאנחנו רוצים שקרנל הטשטוש שלנו יהיה ניתן לפרק לkernel אנסי ואופקי, כדי שנחשוך בכמה פעולות הכפל שאנו מבצעים.

אם ניקח תМОונות מהפרמידה ונגידיל אותן חזרה הן יהיו מאוד מטושטשות. אך המטרה שלנו היא שכשאנחנו הולכים ומקטינים את התמונות ישארו רק התדרים הנמוכנים.



אייר 53: כאשר אנו לוקחים תמונה, ומושווים אותה להגדלה של תמונה מהפרמידה (תמונה מוקטנת מטושטשת) אז נקבל שהתמונה שהגדלנו היא מטושטשת ביחס לתמונה המקורית. ככל שהתמונה קטנה יותר היא יותר מטושטשת.

9 פרמידת לפלייאן

פרמידת לפלייאן - החסירה בין שתי רמות של פרמידה גאוסיינית.
אבל כל שתי רמות הן בגודל **שונה**, לכן ניקח את הקטנה ונגידיל אותה (כפי שהראינו איך לעשות קודם) ורק אז נחסר. בrama الأخيرة אין לנו מה להחסיר שכן בסוף ראש פרמידת הפלייאן תהיה G_n .

Gaussian	Laplacian
G_n	L_n -
G_3	L_3 —
G_2	L_2 ——
G_1	L_1 ———
G_0	L_0 —————
	$G_n = \text{Reduce}\{G_3\}$
	$G_3 = \text{Expand}\{G_4\}$
	$G_2 = \text{Expand}\{G_5\}$
	$G_1 = \text{Expand}\{G_2\}$
	$G_0 = \text{Expand}\{G_1\}$

איור 54: פרמיידת לפלייאן

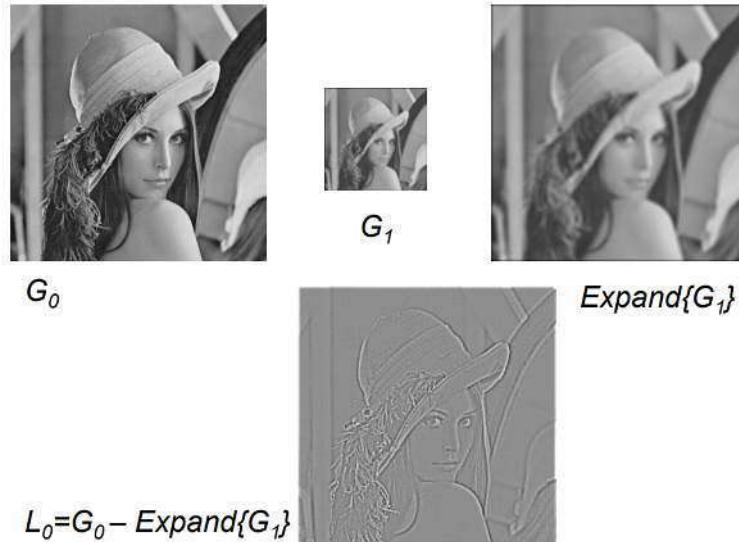
נשים לב שמתקיים לכל n (לאחר הגדלה של L_n כדי שתיה בגודל של (L_{n-1})):

$$L_n + L_{n-1} = \text{expand}\{L_n\} + L_{n-1} = \text{Expand}(G_n) + (G_{n-1} - \text{Expand}\{G_n\}) = G_{n-1}$$

$$\text{מכאן ניתן להסיק ובירט } \sum_{i=0}^n L_i = G_0 \quad \sum_{i=k}^n L_i = G_k$$

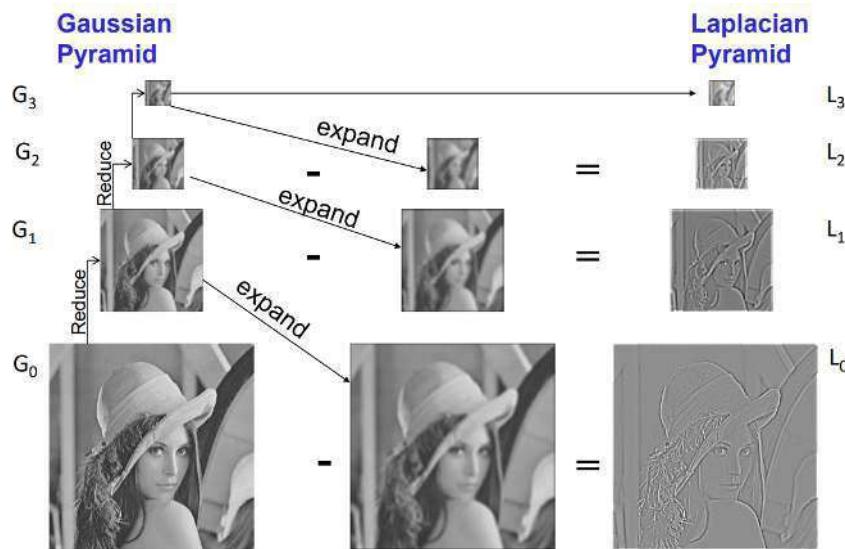
נשים לב ש- G_0 זו התמונה המקורית, אך פרמיידת הלפלסייאן מחזיקה בתוכה את כל המידע על התמונה.

הערה ניתן להסתכל על זה כפעולה אנלוגית להתרמת פורייה בפרמיידות. זה פירוק של תמונה לתדרים ברמות שונות. ברגע שאנחנו אוספים את כל התדרים הללו אנו מקבלים את התמונה המקורית.



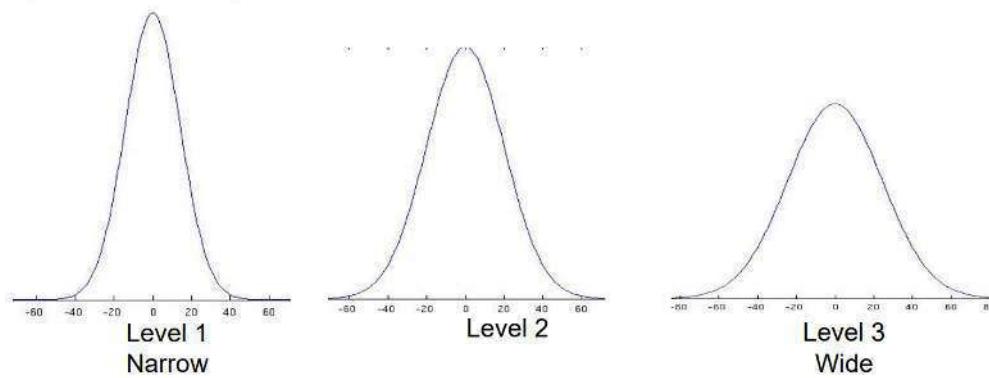
איור 55: האיבר הראשון בפרמיידת לפלייאן על תמונה קונקרטית. נשים לב שאנחנו מקבלים edges , מה שמסביר את השם פרמיידת "לפלסייאן".

نبיט באיור שלמעלה. באזורי האחדים הטשטוש לא משנה כלום, שכן ההפרש (הלפלסייאן) נותן 0 באזוריים הללו. במקומות עם מעברים חדים נקבל ערך גבוה. לכן קוראים לזה לפלייאן, זהו המיצוע של הסביבה: זו הסביבה המטושטשת בחותם הפיקסל בתמונה עצמה.



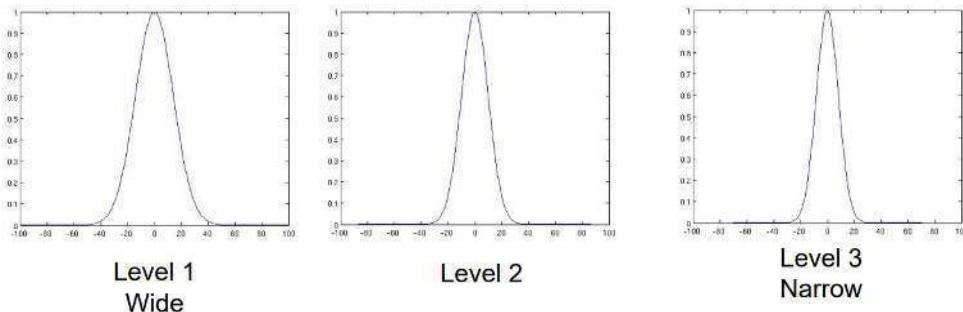
איור 56: פרמיידת לפלייאן שלמה על תמונה קונקרטית

זכור שבפרמיידה גאוסיינית בrama הראשונה היטשטוש הוא עם גaussien ואז ככל שאנו מתקדמים היטשטוש יותר חזק.



איור 57: בכל רמה בפרמיידה גאוסיינית אנו מפעילים על התמונה קרנל של גaussien רחב יותר

לפי משפט הקונבולוציה פירמידה גאוסיינית היא כמו כפל התמונה בקרנל גאוסייני (במרחב התדר).
כל שהקרナル הגaussienי רחב יותר הוא גם נמוך יותר כי סכום המקדמים הוא אחד והיטשטוש חזק יותר.
אכן, התמרת הפורייה של הקרナル הכי רחוב (הכי מיטשטש) הוא הכי דק: משאיר הכי מעט תדרים גבוהים.



איור 58: בכל רמה בפרמיידת גאוסיינית אנו מפעילים כופלים **במרחב התדר** את התמונה בגאוסיין צר יותר

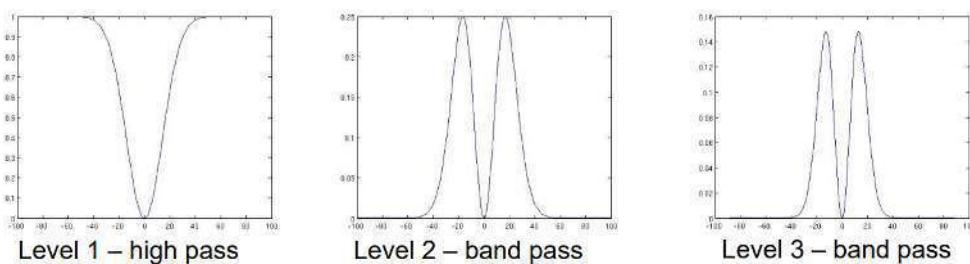
מכאן שבחפרמיידת לפלייאן, כאשר אנו מחסירים מהתמונה המטושטשת אנו מחסירים את התדרים הנמוכים ומשאירים רק את התדרים הגבוהים.

בפרט, ברמה הראשונה, יש לנו את כל התדרים, ואז אנחנו מחסירים את התמונה המטושטשת וישארו רק התדרים הגבוהים מהתמונה המקורית.

ובאופן כללי, כל רמה בלפלסייאן היא תוצאה של הכפלה (**במרחב התדר**) של התמונה בפונק' שהיא הפרש בין שני קרנלים גאוסייניים עוקבים.

דוגמה נוספת ברמה השנייה אנו לוקחים את התמונה המטושטשת ומחסירים תמונה עד יותר מטושטשת - אנחנו מחסירים את התדרים הנמוכים מבין אלה שנשארו. ככלمر ברמה השנייה אנחנו לא מחסרים (מורדים) את התדרים שהיו הקיימים בתמונה המקורית, אלה כבר ירדו ברמה הקודמת, ולא את התדרים הגבוהים, אותם אנו משאירים. על כן, הפילטר הוא *band – pass* - משאירים רק את התדרים שיותר באמצע.

זה מה שקרה גם בכל רמה אחרת, רק עם *band – pass* שונה כל פעם.



איור 59: הקרנלים בפרמיידת לפלייאן הם הפרשים בין שני גאוסיינים עוקבים בהם השתמשנו לחישוב הפרמיידת הגאוסיינית (ראו איור קודם)

לסיכום, בפרמיידת לפלייאן רק הרמה הראשונה משתמשת בקרנל *high – pass*, ושאר הרמות הן *band – pass* ורק הרמה האחרונה היא *low – pass* (כי שם אנחנו כבר לא מחסרים כלום ולוקחים את השכבה מהפרמיידת הגאוסיינית).

לשם הבירהה, הסיבה שרק הרמה הראשונה היא מיוחדת ויש בה *high – pass*, היא שברמה האפס (התמונה המקורית) אנו לא עושים שום טיפול. אפשר לחשב על זה שבמרחב התדר כפלו את התמונה בפונק' שhaiia 1 בכל מקום (כלומר התמונה לא

משתנה), לכן 1 פחות גאוסיאן תיתן לנו פונק' כמו ב-1 *Level high-pass* בתמונה הקודמת, שהוא *פילטר - הפונק' ב-1*. מתקבל ערך נמוך בערכים נמוכים, הקרים לאפס (שם התדרים נמוכים), וערך גבוה בערכים גבוהים (שם התדרים גבוהים). בשאר המיקומות יש לנו חיסור של גאוסיאנים, שימוש מלבד הרמה האחרונה בה אין חיסור ולוקחים פשוט את התמונה מהפרמידה הגaussית).

10 דחיסת פרמידה

ניתן לבצע דחיסה של תמונה באמצעות פרמידת פלסיין באופן הבא:

Algorithm 10 דחיסת פרמידה

- 1 : Build a Laplacian Pyramid
 - 2 : Quantize pyramid values to 3-5 values
 - Optimal Quantization
 - 3 : Compress using Entropy Compression
 - (Huffman, Lempel-Ziv)
 - 4 : Reconstruct normally
-

נשים לב שלפרמידת פלסיין הערכים יכולים לנوع בין -256 ל- $+256$, לכן בשלב 2 לשם הדחיסה נבצע קוונטיזציה אופטימלית. מסתבר שאם נעשה קוונטיזציה כזו אז לא נוכל לראות בעין הבדל בין התמונות, כי קוונטיזציה תנידן לנו אילו אזורים יותר כהים ואילו יותר מטושטשים, ומה שחשוב לעין שלנו זה שם שהיא בהיר ישאר בהיר ומה שהיא כהה ישאר כהה.

המשמעות של שלב 4 היא שאם כל פיקסל בתמונה היה מיוצג בדרגות אפור מלא א' אחרி קוונטיזציה והאפקן יכול להיות שאנו משתמש בהרבה פחות מביט אחד לפיקסל, כלומר המידע של בית אחד משמש לכמה פיקסלים באמצעות קידוד חכם ויש להפעיל אלגוריתם כדי להתאים את המידע לפיקסלים ולבנות את התמונה מחדש.

שיטת זו הובילה לשיטה הטובה ביותר לדחיסת תמונות שקיימות כיום, ושמה *wavelets*.

11 תפירת תמונות

אם נרצה לחבר שתי תמונות A , B ביחיד לתמונה חדשה C , כך שהחצי השמאלי הוא של A והחצי הימני הוא של B , נוכל להשתמש באלגוריתם נאיבי פשוט יחבר חצי של A וחצי של B , אבל אז יראו את "קו התפר", מעבר חד וברור אייפה שהיבנו את שתי התמונות.

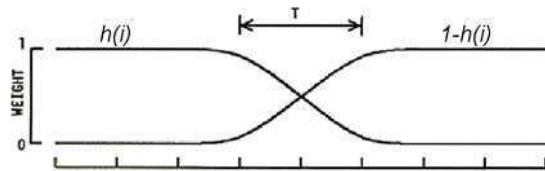


איור 60: תפירה נאיבית של שתי תמונות - קו תפָר בולט מואוד באמצע

אפשר לחולפן להשתמש ב-*spline*: פונק' שאומרת מתי להשתמש בכמה מהפיקסל ב-*A* וכמה מהפיקסל ב-*B*. נקבל שכל שורה y , ערכי התמונה *C* הם

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

כאשר h היא פונק' משקל כמתואר באיור. כך אם נלק משמאל לימין, בהתחלה הפיקסלים יהיו רק של *A*, אחר כך כשהגעיל לאזור האמצע הם יעברו בהדרגותיות להיות *B* ואז בסוף כל הפיקסלים מימין יהיו רק של *B*.

איור 61: פונק' משקל לצורך תפירת תמונות בשיטה של מעבר *spline*

בעיה באמצע אנחנו נראה חיבור של שתי התמונות. אם לדוגמה ב-*A* יש באמצע קווים מסווג אחד וב-*B* קווים מסווג אחר, אז באמצע של *C* נראה קווים משני הסוגים ביחד כאשר כל קו מופיע בחצי עוצמה.

תפירת לפלייאן

בhinintן שתי תמונות *A*, *B* שברצוננו לתפור, נבנה את פרמידות הלפלסיאן שלהם: L_a , L_b , L_c בה כל רמה Moravec במחציתה מ- L_a ומחציתה מ- L_b : מימין יהיה L_a , משמאל L_b ובמרכזו (אם כמה הפיקסלים Ai-זוגית) יהיה ממוצעו של L_a , L_b .

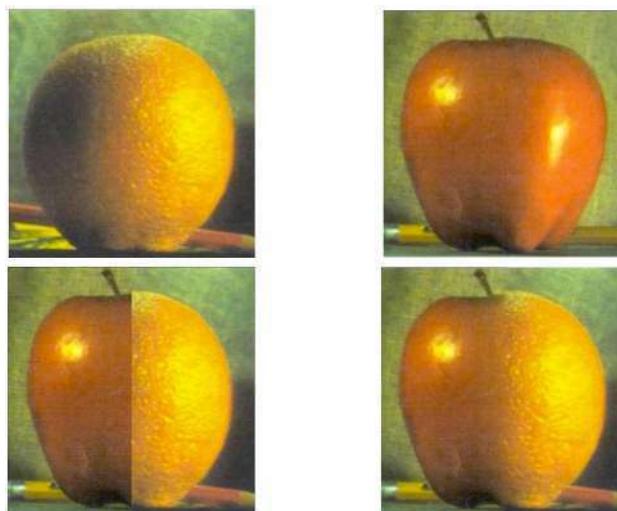
נסכום את כל הרמות של L_c (תוק שגדיל את הרמות כדי שני המחוברים בסכום יהיו באותו גודל) ונקבל את התמונה *C*.

Algorithm 11 תפירת לפלייאן**Multiresolution Pyramid Spline**

- 1 : Construct Laplacian Pyramids L_a, L_b
- 2 : Create a third Laplacian Pyramid L_c where for each level k:

$$L_c(i, j) = \begin{cases} L_a(i, j) & i < \frac{\text{width}}{2} \\ \frac{L_a(i, j) + L_b(i, j)}{2} & i = \frac{\text{width}}{2} \\ L_b(i, j) & i > \frac{\text{width}}{2} \end{cases}$$
- 3 : Sum all levels of L_c to get the blended image
 - resize levels of L_c so the summands are of the same size

נביט עתה בדוגמה לשימוש באלגוריתם. נשים לב ששמו את התפוח והתפוז בתמונות כך שהרגע מאחוריהם יהיה אותו דבר והם יהיו באותו גובה (מעין דרישת קדם במקרה זה כדי שזה יראה יפה). משמאלו למיטה ניתן לראות את המיזוג המשתמש באלגוריתם נאיבי, ומימין למיטה את המיזוג המשתמש בתפירת לפלייאן.



איור 62: דוגמה לשימוש באלגוריתם תפירת הלפלסיאן

תפירת תמונות בצורה שרירותית

בhinintן שתי תמונות A, B ומסיכה ביןארית M , נרצה לפתור את A, B לפי המסיכה M (M אומרת איזה אזור יהיה של A ואיזה של B , אם בפיקסל מסוים M הוא 1, אז אזור זה "שייך" ל- A).

הערה אנו בכוונה אומרים ש- M מגדירה אילו אזורים "שייכים" לאיזו תמונה, ולא אומרים ש- M אומרת אילו פיקסלים שייכים לאיזו תמונה, כי באזורי התפר דברים עלולים להיות יותר מעורבים (זה טוב, כך אנו לא רואים מעבר חד) או אם $M = 1$ בפיקסל מסוים לא בהכרח קיבל את הפיקסל של A באותו מקום.

לשם השגת המטרה שלנו נעבד לפי האלגוריתם הבא

Algorithm 12 תפירת לפלייאן

Multiresolution Pyramid Spline

- 1 : Construct Laplacian Pyramids L_a, L_b
 - 2 : Construct a Gaussian Pyramid G_m from mask M
 - 3 : Create a third Laplacian Pyramid L_c where for each level k:

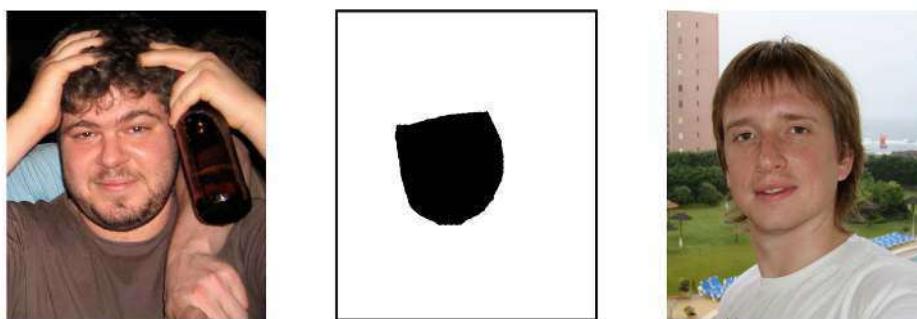
$$L_c(i, j) = G_m(i, j)L_a(i, j) + (1 - G_m(i, j))L_b(i, j)$$
 - 3 : Sum all levels of L_c to get the blended image
 - resize levels of L_c so the summands are of the same size
-

נשים לב שככל פיקסל (j, i) בכל רמה ב- L_c נוצר באמצעות ממוצע ממושקל של הפיקסלים המתאימים (קרי, אותו (j, i) ואותה רמה בפרמידה) ב- b , L_a, L_b כאשר המשקלות הן של G_m .

הערה הסיבה לחישוב לצור פרמידה אנטינית מ-M היא כדי שנדע בכל רמה אם לחת פיקסלים מ- L_a או מ- L_b , שכן כל רמה בפרמידה קטנה מהתמונות המקוריות A, B , והמסיכה M היא בגודל התמונות המקוריות. לכן הקטנה של המסיכה בכל רמה תניד לנו אילו פיקסלים לחת בכל רמה.

הערה למראות שברמה הראשונה $G_0 = M$ היא מסיכה ביןארית, כשתקדם ברמות G_i כבר לא תהיה ביןארית, וזה יגיד לנו שאנחנו לא לוחכים באוthon רמה רק A או רק B , אלא שילוב של השניים. אינטואיטיבית זה יוצר מעין אפקט של *fade – in*.

نبיט בדוגמה להפעלת האלגוריתם על שתי תמונות עם המסיכה שנמצאת במרכז האיור:



איור 63: דוגמה לתמונות עליהן נפעיל את האלגוריתם

תוצאה מלאגוריתם נאיבי קיבל בחלק העליון של האיור הבא. את תוצאתה האלגוריתם שלנו ניתן לראות בחלק התחתיו של האיור.



איור 64: למעלה: תוצאת האלגוריתם הנאיבי - רואים את התפירה בבירור.
למטה: תוצאת אלגוריתם תפירת הגאוסיינט - "ניתוח פלסטי לא נורמלי".

ניקוי רעים

הרצאה 6

מהו רעש בתמונה? רעים נוצרים כאשר חישבי או רעלרים נדלים כאשר הם פולטים אלקטرونים כתוצאה מאור, הרבה פעמים באופן רנדומלי ללא קשר לאור. כאשר האור נМОך (סצנה חשוכה) הרעש בדר'כ יהיה יותר גדול.

אחת השאלות שעולות הן כיצד לנוקוט את הרעים?

הדרך **הכי פשוטה** שראינו הייתה טשטוש, שכן הטשטוש מבטל את הרעש, אבל יוצר טשטוש לתמונה. האם יש שיטות טובות יותר? דרך נוספת הייתה שימוש בחץון, הוא מנקה, לא מטשטש, עם תופעות לוואי משלו.

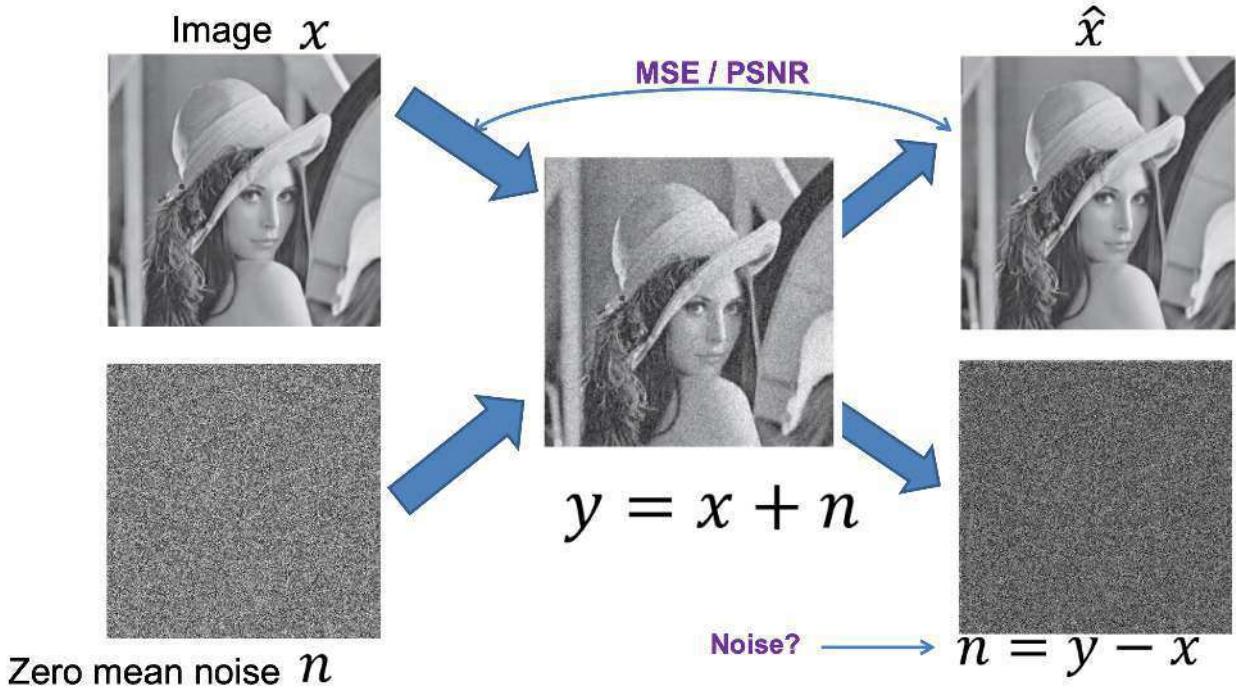
דוגמה. נסיף רעש לתמונה x כך שנקבל תמונה חדשה $n + x = y$, כאשר n הוא הרעש. התמונה תמיד עם ערכים חיוביים והרעש יכול להיות עם ערכים חיוביים או שליליים.

אם נפעיל על התמונה הרועשת אלגוריתם ניקוי רעים, נctrיך דרך לבדוק כמה היא טובה.

אם **יש** לנו את התמונה המקורית (לפני הוספת הרעש), נוכל להשוות אליה את החדש, \hat{x} .

אם אין לנו תמונה מקורית בלי רעש, נוכל לחסר מהתמונה הרועשת המקורית את התמונה שיצאה מהאלגוריתם ולהסתכל על ההפרש $x - \hat{x}$. במצב אידיאלי בו ניקנו את כל הרעים, ורק אותם, נקבל $n = y - x$.

אם אנחנו מצלחים לאחות פרטים של התמונה המקורית בתמונה ההפרש זה אומר שנפטרנו מפרטים חשובים. אם נראה רק רעש זה אומר שכנהראה נפטרנו רק מרעש. לאותם פרטים, אנו קוראים, כי שראינו *Edges*, ואותם לדנו איך להזות. בדוגמה הנ"ל אנו רואים קווי מתאר של האישה בתמונה, ולא רק רעים רנדומליים, ככלומר נפטרנו מפרטים מהתמונה המקורית וזה לא טוב.



איור 65: תהליכי הניקוי הנאיבי

שאלה האם נוכל להשווות את ה-edges במערכות האלגוריתם של Canny?

אם יש לנו את התמונה המקורית נוכל להסתכל על ה-error בין התמונות. זה בדר"כ יהיה המבחן ב-Testing.

נוכל להשתמש למשל בפונק' ה- MSE - Error Square Mean . $MSE = \frac{1}{n} \sum_{i=1}^n \|x_i - \hat{x}_i\|^2$

דרך אחרת, היא להשתמש ב-PSNR : Peak Signal – to noise ratio . $PSNR = 20 \cdot \log_{10} \frac{255}{\sqrt{MSE}}$

אם הרעש גבוה מאוד אז $PSNR = 0$. זאת כי \sqrt{MSE} קיבל לכל היתר 255 (שכן זה טווח הערכים ושורש הטיעות הריבועית

הממוצע לא יכול להיות גדול מהערך המקסימלי, 255, ואז במקרה זה קיבל 0 . $\frac{255}{\sqrt{MSE}} = 1 \Rightarrow 20 \cdot \log_{10} \frac{255}{\sqrt{MSE}} = 0$

אם אין רעש אז PSNR מקבל ערך מאד גבוה ($\frac{255}{\sqrt{MSE}} = \infty$ שווה לאינסוף).

הנחה שעבדנו אותה היא שהרעש הוא רעש אדיטיבי גaussiano לבן - הגרנו רעש בכל פיקסל.

Gaussian Noise (AWGN)

המשמעות של רעש לבן היא שככל התדרים מופיעים בעוצמה שווה - הפורייה נראה כמו תמונה קבועה לבן בצורה אחתיה.

רעש כחול לעומת זאת זה רעש בעל יותר תדרים גבוהים מנמוכים ורעש ורוד מכיל יותר נमוכים מגובאים.

11.1 פרמטר טשטוש גאוסיאני - Bilateral – Filtering

נניח כי p היא נקודה בתמונה I עם ערך אפור I_p .

הטשטוש הגaussiano הוא $GB[I]_p = \sum_{q \in S} I_q \cdot G_\sigma(\|p - q\|)$ כאשר S מצין את חלון הטשטוש, כאשר הנקודות קרובות ל- p מקבלים משקל גדול ולהפך. G_σ הוא הגaussiano שמטשטש בחלון עם פרמטר הטשטוש σ , דהיינו החלון בתמונה בו q נמצא.

ס גדול זה טשטוש גדול (גאוסיאן רחב), כי החלון גדול, ס קטן זה גאוסיאן קטן וכאן טשטוש קל. נציין כי הכתיבה $\sum_{q \in S}$ משמעותה סכימה על כל הפיקסלים בתמונה באמצעות האינדקס q . נזכיר כי הערך $G_\sigma(\|p - q\|)$ הוא הפעלה של הגאוסיאן על $\|p - q\|$ וכן כשלוקודות קרובות אחת לשניה, הנורמה קטנה ולכן הגאוסיאן עליה גדול. מכאן נקודת קרובות מקבלות משקל גדול יותר. דבר זה נראה כך:

Gaussian Blur Parameter

Assume p is a point in Image I with gray level I_p

$$\text{Gaussian Blur: } GB[I]_p = \sum_{q \in S} G_\sigma(\|p - q\|) I_q$$

size of the blur window

small σ



weak smoothing



input

large σ



strong smoothing

איור 66: טשטוש גאוסיאני עבור ס-ות שונות, ככל שהוא יותר כך הטשטוש חזק יותר

הערה. פורמלית, הKernel הגאוסיאני החד-מיידי G_σ מוגדר ע"י $G_\sigma = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right)$. לעומת G_σ הוא התפלגות גאוסיאנית עם סטיית תקן σ . (נציין גם שהKernel הדוו-מיידי הוא $(\frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right))$ פרקטית, במקומות גאוסיאן אנחנו משתמשים במקדים בינהיים.

עם זאת, עליה הבעה הבא:

בעיה. אם נעשה טשטוש עם גאוסיאן איזומטרי בין $edges$ אנחנו נפגע ב-edges. לכן ב-edges נרצה או לא לטשטש בכלל או לטשטש לחוד הצד של $edges$ ולהוד הצד אחר. אם נדע איפה ה-edge נוכל לעשות זאת.

הערה בניסוח אחר, GB היא בלתי תלואה בתוכן התמונה. ההשפעה של כל פיקסל על פיקסל אחר תלואה אך ורק **במרקח** בינוים בתמונה, ולא **בערכים** של התמונה.

הweeney כדי לפתור את הבעיה הוא לנקוט פונק' חילקה שתלויה גם בתוכן התמונה, ולא רק במרקח הנקודות, על ידי פיקסלים שהרמת אפור שלהם לא רוחקה מהצבע של הפיקסל עצמו. בכל אזור נבצע טשטוש שונה, כדי לא לפגוע ב-edges. אנחנו רוצים להתחשב בכל נקודה בפיקסל ממנו התחלנו כאשר ביצעוו מיצוע.

על כן, נשימוש בממוצע משוקל של הפיקסלים, בהתאם לכמה הצבעים דומים (נתחשב יותר בצבעים דומים) - זה מה שקרה בילאטרל: מקום אינפורמציה אחד הוא המרחק בין הפיקסלים, ומוקם האינפורמציה השני הוא המרקח בין הצבעים של הפיקסלים.

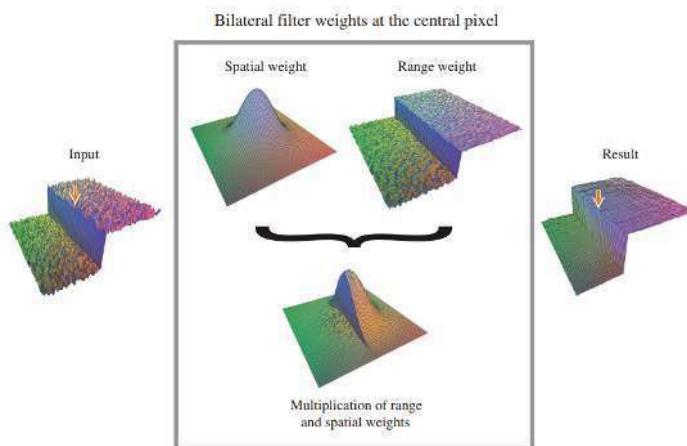
$$BF[I]_p = \underbrace{\frac{1}{W_p}}_{\text{Normalization-Factor}} \sum_{q \in S} \underbrace{G_{\sigma_s}(\|p - q\|)}_{\text{Space-Weight}} \cdot \underbrace{G_{\sigma_r}(|I_p - I_q|)}_{\text{Color-Weight}} \cdot I_q$$

לבדיל מגאוסיאן, בו סכום המשקלות תמיד היה 1 ולבצע טשטוש לא שינה את בהירות התמונה, כאן בכלל אלמנט ההפרש החדש, יהיה שינוי בסכום המשקלות והוא לא יהיה 1.

נזכיר שוב כי הגאוסיאנים פועלים על הנורמות, לכן נקודות במרחקים קטנים וצבעים דומים יקבלו משקל גדול יותר. על כן, עבור כל פיקסל נחלק בסכום המשקלות (בULONG) כדי לוודא שסכום המשקלות הוא 1. זה החלק של $\frac{1}{W_p}$.

הערה W_p מוגדר ע"י מה שמבטיח שסכום המשקלות הוא 1.

הערה σ_s, σ_r מצינים את מידת ה-*filtering* על I . המשווהה המגדירה את $BF[I]_p$ היא סכום ממושך מנורמל בו הוא פונק' משקל גאוסיאנית (כפי שראינו) שמשמפלת את המרחק (המרחיב). לעומת גאוסיאן במרחב התמונה, G_{σ_r} הוא גאוסיאן שמשמפל את טווח הערכים, שמקטין את השגיאה של פיקסל q על p אם הערכים שלהם בתמונה, I_p, I_q , שונים. לעומת גאוסיאן במרחב הצבעים.



איור 67: ויזואלייזציה להפעלת אלגוריתם BF .

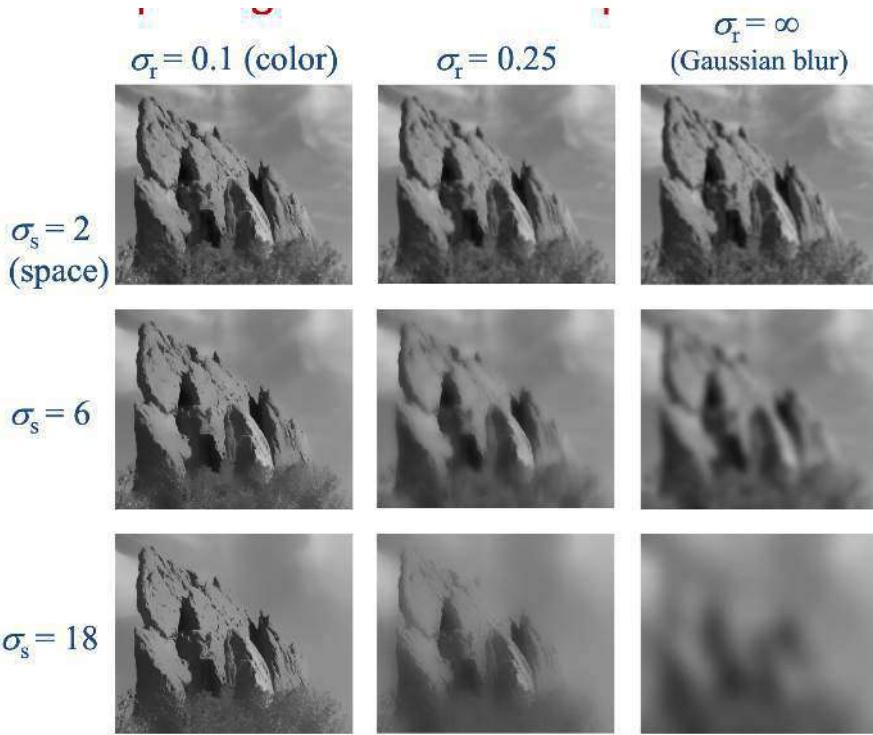
נשים לב שהמשקלות בנוסחה של BF ספציפית לכל פיקסל لكن אין אלגוריתם יעיל כדי לחשב את BF .

נסכם: יש לנו שני פרמטרים:

σ_s - כמו אנחנו נכנסים לתוך התמונה - כמו הernal טשטוש גדול. •

σ_r - אמפליטודה מינימלית של *Edge*, דהיינו כמו אנחנו רוצים שהצבעים יהיו שונים אחד מהשני. •

אפשר לשחק איתם ולקבל את התוצאות הבאות:



איור 48: כאשר אנו מתחשבים בדמיון הצבאים, אנחנו מקבלים טשטוש חלש יותר למרות הגaussien הגדל, בעוד אם אנו מאפירים הפרש נדל בין הצבאים אנו מקבלים טשטוש חזק מדי. כאשר $\sigma_r \rightarrow \infty$ אנו מקבלים שהגaussien במרחב הצביע G_{σ_r} הופך לשטוח, וכך לא משפיע על המשקלות, על כן אנו מקבלים פילטר גaussien רגיל במרחב התמונה שמטשטש אותה. אם σ_s גדול מדי ביחס ל- σ_r , הטשטוש יהיה חזק מאוד.

נבחן כי בגבול של הקربה בצבאים, $\sigma_r = 0$, ואז אנו עושים ממוצע בין אותם הצבאים וכך נזכה לקבל בדיקת אותו דבר (לא יהיה שינוי ולא יהיה טשטוש), על כן נרצה בכל זאת לתת הפרש צבאים מסוימים ולקחת $0 > \sigma_r$.

האלגוריתם כאמור לא יעיל כי צריך לנរמל משקלות בכל פיקסל, אין FFT . נעיר כי במקרה של כמה מרחבי צבע הנוסחה נשארת באותה צורה רק עם נורמה תלת ממדית ובמקרה I_q יהיה לנו C_q שהוא וקטור תלת-ממדי של RGB . דהיינו,

$$BF[I]_p = \underbrace{\frac{1}{W_p}}_{\text{Normalization-Factor}} \sum_{q \in S} \underbrace{G_{\sigma_s}(\|p - q\|)}_{\text{Space-Weight}} \cdot \underbrace{G_{\sigma_r}(\|C_p - C_q\|)}_{\text{Color-Weight}} C_q$$

הערה המרכיב הוא עדין וקטור $D - 2$ אבל עכשו הצביע הוא וקטור $D - 3$, שכן ה- $BF - D$ – 5.

בשילובו לחציו (*median*) סיגנון זה עובד טוב משמעותית:



איור 69: ניתן לראות שהחציון משאייר הרבה רעש בהשוואה ל-*Bilateral – filter* שמנקה יותר

patch – methods 11.2

ב-*BF* ניסינו להסתכל על פיקסלים קרובים אליו שגם דומים לו בצבע. עם זאת, להסתכל על פיקסל זה בעייתי כי אם הפיקסל הוא רעש זה לא משנה אם הוא דומה לי או לא. במקרה להסתכל על פיקסל ספציפי, נסתכל על כל הסביבה. נוכל לבחור סביבה בגודל 3×3 או 5×5 (או גודל אחר) ונבדוק האם הסביבה של הפיקסל שלנו דומה לסביבה אחרת בתמונה. לשיבובות אלה קוראים *Patch*.

נניח כי נתונה תמונה סטטistica – סרט רועש. כל *frame* בסרט יכול לקבל רעש אחר ולכון נקבל שהרעש משתנה בזמן. **איך נבטל אותו בהנתן שהצנה קבועה?** נניח כי הרעש הוא *mean – zero*, נוכל לסקום את כל התמונות ביחד (הסטנות בכל חלק), או לפחות חלק מהן, וכך הרעש יתבטל, נגרם (כדי לא לקבל מספרים גדולים מדי) ונקבל תמונה ללא רעש.

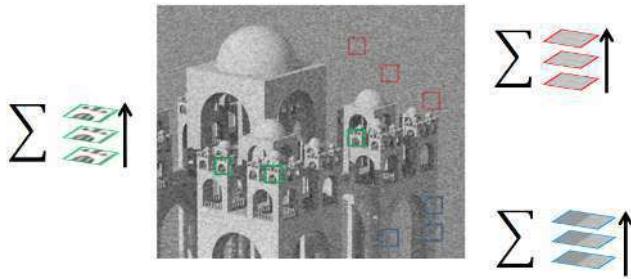
שיטת זו עובדת משתני סיבובות:

- (i) הרעש הוא *mean – zero*
- (ii) מסתבר שמתקיים הקשר

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

דהיינו השונות של \bar{X} (הממוצע) קטנה, ולכון הרעש שווה לאפס ככל שהוא מוחברים יותר תמונות.

כאנחנו משתמשים על תמונה בודדת, אין לנו הרבה תמונות למצער כמו במקרה הקודם, אבל יש לנו **אזורים דומים**. מצאו שם ניקח תמונות רגילים וניקח *patch-patch*-ים, נראה שלרוב ה-*patch*-ים דומים במקומות אחרים בתמונה. דהיינו לא צריך תמונות נוספות כדי להיפטר מהרעש כמו בסרט הסטטי, אלא מספיק לבחור סביבות דומות בתוך התמונה.



איור 70: יש מגוון אזורים, בתמונה שנראים מאוד דומה זה לזה, אותם ניתן למצוע כדי להיפטר מרעש. צריך למשקל אותם כי ה-*patch*-ים לא בהכרח זחים לحلוטין.

השיטה שלנו:

- נחפש דומים בתמונה. במקומות *patch* שבחרנו, נחליף אותו במשמעות של כל מה שמצאנו.

↳ המשקל של ה-*patch* הוא לא זהה - המשקל תלוי בכמה ה-*patch*-ים דומים.
↳ דומים קבלו משקל גבוה יותר ולהפך.

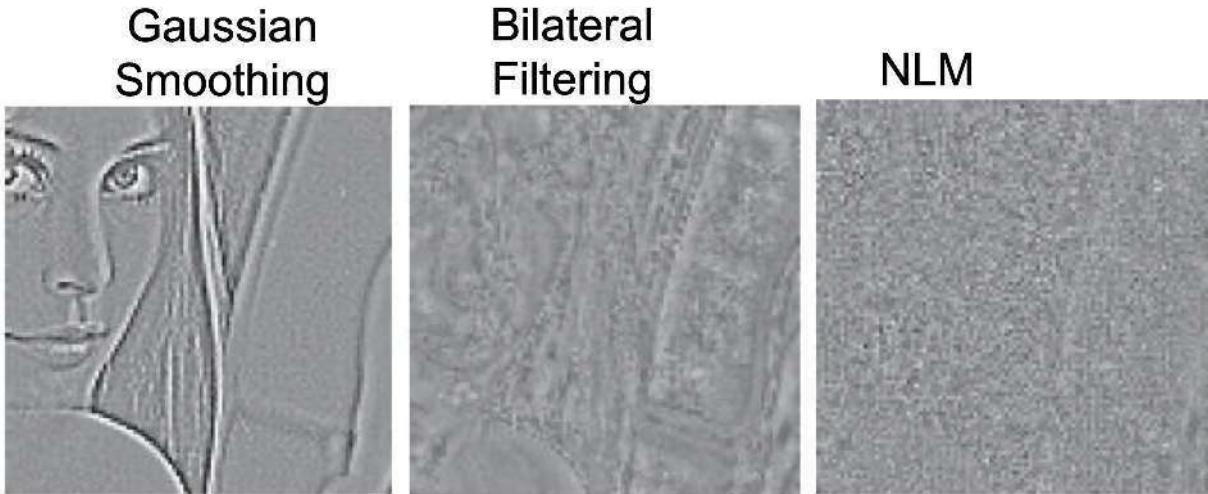
לשיטתה זו קוראים (Non-Local-Means-(NLM) ופורמלית היא תעבור לפיה האלגוריתם הבא:

Algorithm 13 Non-Local-Means-(NLM)

- 1 : Given one pixel, compute the similarity of a patch around it to patches around **all other** pixels.
- 2 : Compute a weighted average of **pixels** based on patch similarity
- 3 : Replace pixel value by this average

$$\hat{x}(u, v) = \frac{1}{C(u, v)} \sum_{i,j} y(i, j) e^{-\frac{SSD(N(u, v) - N(i, j))}{2\sigma^2}}$$

באלגוריתם הנ"ל y היא תמונה הקלט, \hat{x} הוא תמונה הפלט, $N(i, j)$ הוא סביבה (*patch*) של j , i (למשל בגודל 3×3). w היא פונק' המשקל - באופן כללי אנו לוקחים סכום ריבועים אחד, אבל אפשר למשקל גם אותם. $C(u, v)$ - קבוע נרמול, זה סכום כל המשקלות כדי שסכום הכל הן יסכו ל-1. כמו כן, SSD הוא Sum-of-Square-Difference בין ה-*patch*-ים. בקצרה, אנו ממשקלים כל פיקסל לפי ה-*patch*-ים ומנורמלים. אנו משתמשים על כל הפיקסלים למרוחות שבפועל אנו נשארים רק עם *pathces* בעלי התאמה גבוהה, זה טוב, כי אנו מקבלים כי אנו משתמשים על כל הפיקסלים על כל הפיקסלים בתמונה ולא רק על הפיקסלים מסביב לפיקסל שלנו. *Non – Local Mean*



איור 71: לשם השוואה, החסרת התמונה לאחר גאוסיין משaira הרבה *bilateral ,edges* משאיר קצר, אבל *NLM* משאיר כמעט כלום.

נסכם כי:

- טשטוש גאוסיאני - מורח את הרעש על התמונה
- טשטוש בילטרלי - יכול להעלים רעים שהם *PepperSalt*(Clomer לא אדיטיביים. מטשטש כשמדבר ברעש אדיטיביים).
- *NLM* - מעלים רעים אדיטיביים ועובד טוב כשמדבר ברעש שהוא *ZeroMean*.

Google – Night – Sight 11.3

טלפון ה-3 Pixel של גולץ הציג טכנולוגיה חדשה לצילום בחושך.
נזכר כי הפתרון שאנו מכירים להציג ערכיהם נוכחים וצפופים הוא *Correction* – γ , אך נקבל רעש, כי אין חשיפה כמעט לאור.

הפתרון של צלמים לאור חלש הוא חשיפה ארוכה – פתיחת הצמצם לזמן רב (נגיד 10 שניות לעומת, מאית/אלפית – שנייה בחשיפה רגילה).

בטלפון רגיל, הבעיה בחשיפה ארוכה לאור היא שהצמצם זו (בגלל שהיד שלנו לא יציבה לדוגמה), התמונה זהה, ומתקבלות תוצאות מרוחקת למראינו אלא אם אין תזוזות. לכן כשפותחים את הצמצם לזמן ארוך משתמשים בחזובה.

גולץ השתמשו בפתרון אחר: אנו בוחרים הרבה הרבעה דגימות, כלומר הרבעה תמונות בזמן שהצמצם פתוח (בגולץ לוקח בין 6 ל-20 תמונות במשך 6 שניות כדי להתגבר על הטשטוש שהצמצם יוצר בחשיפה הארוכה זו). זמן החשיפה נקבע לפי התמונה: משתמשים בחישון gyro כדי לראות כמה היד זהה לדוגמה, והסתכלו כמה הגופים בתמונה זו. ככל שהייתה יותר תזוזה השאירו את הצמצם לחשיפה יותר ארוכה.

לאחר מכן, (לפניהם ממצאים אותם) אנו מותאים בין התמונות שמצאו ועושים להן זהה כדי שייתאימו אחד לשני. עתה, אנו מחפשים *Patches* דומים בין התמונה שלנו לכל שאר התמונות, שכן התמונות הן אותה סצנה רק בזמן שונה. ה-*patch* לא יהיה בדיק באוטו מקום בגל התזוזה אך אם נחשף יכול להיות שנמצא *patch* מותאים באותו מקום טיפה אחר בתמונה השנייה בבדיקה בגל התזוזה זו.

כשמשלבים אותם ביחד, עם עוד פעולות בלבד מיצוע Multi-frame-Super-resolution-approach) שזה סוג של *AI*, תיקון צבע – (וכו) התוצאות יכולות להיות מאוד מרשימות, למשל מקור חשוק עם אור מלפטופים בלבד, והושגה התמונה הבאה:

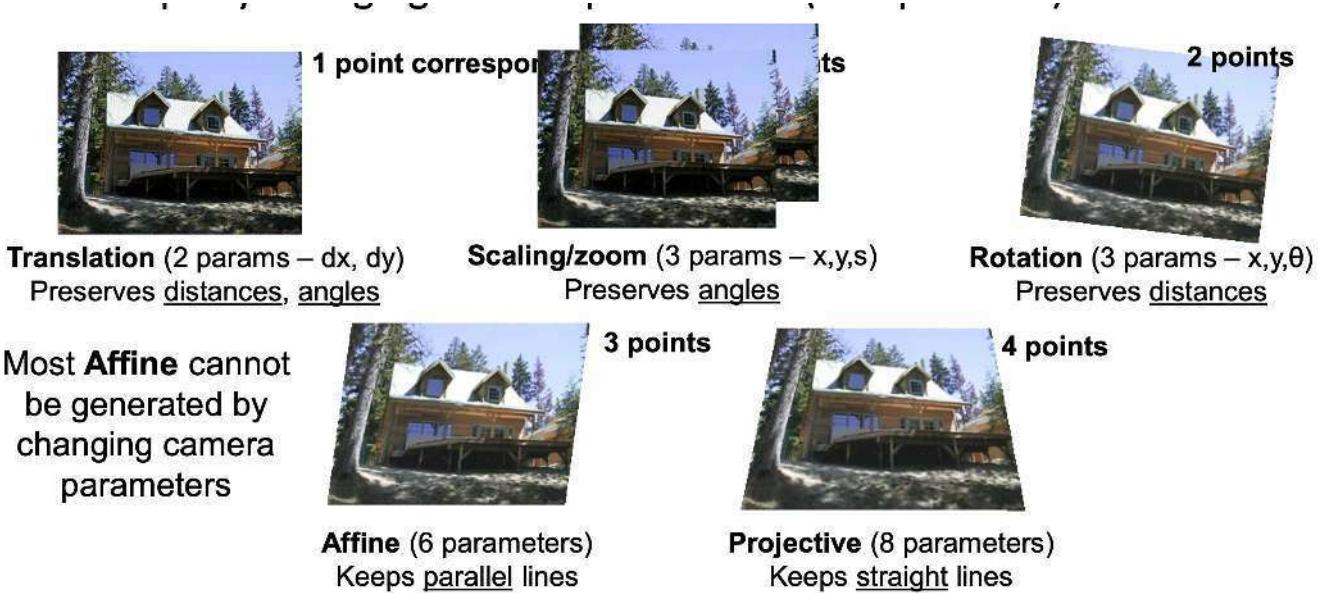


איור 72: הכל בזכות החתימה בין ה-*Patches*

בהמשך נלמד במפורט על הגדרת *patch* מתאים לכל נקודה, וההתאמות *patch*-ים אלה ל-*patch* מתאים בתמונות אחרות.

טרנספורמציות על תמונות

כשדיברנו בתרגול על טרנספורמציות על תמונות, תהינו איך בהינתן שתי תמונות נוכל לגלוות את הפרמטרים של כל טרנספורמציה.
במקרה הבא נוכל לגלוות אותם בדרך הבאה:



איור 73: במקרה של סיבוב אנו צריכים 3 פרמטרים - θ , y , x ש כוללים בתוכם את נקודת המרכז (x, y), שכן צריך לדעת ביחס לאיזו נקודה אנו מסובבים, וכן צריך 3 מושוואות, שנוכל להסיק מ-2 נקודות כי הן יתנו לנו 4 מושוואות. במקרה של הגדלה או הקטנה צריך גם 2 נקודות כי אנו רוצחים לדעת שלושה פרמטרים (s) כאשר (x, y, s) נקודת המרכז. במקרה של שינוי מיקום מספק (dx, dy) וכן צריך נקודה אחת. במקרה של הטלה יש 8 פרמטרים וכן צריך 4 נקודות. במקרה של טרנספורמציה אפינית, 6 פרמטרים וכן צריך 3 נקודות.

בעיה שעולה מתיאור זה היא מציאת הנקודות.

הערה את רוב הטרנס' האפיניות לא ניתן להשיג באמצעות שינויים פרמטרים של המצלמה (סיבוב המצלמה, שינוי מיקום, התרחקות וכו').

המטרה שלנו היא למשה להיות מסוגלים ליצור פנורמות. בפרקimos הבאים נלמד כל שלב ושלב בתהליך. לפני כן, נציג את השלבים:

- מציאת נקודות מיוחדות בכל תמונה אותן ננסה להתאים לנקודות כנ"ל בתמונה אחרת.
- יצירת *Descriptors* לכל נקודה שייעזרו לנו להתאים בין הנקודות בצורה מיטבית.
- התאמה בין ה-*Descriptors* כך שלכל נקודה תהיה נקודה מתאימה בתמונה האחרת.

▷ מיציאת ההומוגרפיה שבעזרת הגענו מהתמונה הראשונה לשנייה.

- איחוד התמונות שצילמנו.

כאמור, נלמד במפורט כל שלב.

12 תוצאות של תמונות

נציג שיטות לחישוב תוצאות של תמונות.

12.1 אפקט Rolling – Shutter

כל מה שראינו עד עכשיו אינו רלוונטי למצלמות שבטלפונים שלנו. הצלום שאנו חשבנו לעלי, הוא צילום של התמונה בנקודת זמן ייחוד, אבל המצלמות בטלפונים הם מסוג *Rolling – Shutter* שסורק את התמונה שורה שורה (זאת ב涅יגוד *Global – Shutter*, שתופסים את כל התמונה בחת-אחת). אנחנו נתונים לכל שורה חשיפה קטנה כדי שלא נקבל מריחה ודבר זה מאפשר גמישות בתוכנת התמונה. אותן מצלמות שקולות הכל ביחס זמן אחד נקראות *CCD*.

ב-*Rolling – Shutter* שהזמן בו קלטונו אויר בכל שורה אחרת ולכן זה עלול ליצור עיונות בתמונה, זה יוצר תמונות מאד מעניינת, למשל עברו פרופולו, אם נגדיר זמן דגימה (קרי, זמן חשיפה) מספיק קטן לכל שורה, אבל לא קטן מדי, וכך יכול תמונה לא מושתשת הנראית כך (אם זמן החשיפה ארוך מדי, אז אנחנו ניחשף לסיבוב שלם של הפרופולו ואז נראה טשטוש שמורכב מכל הסיבוב):



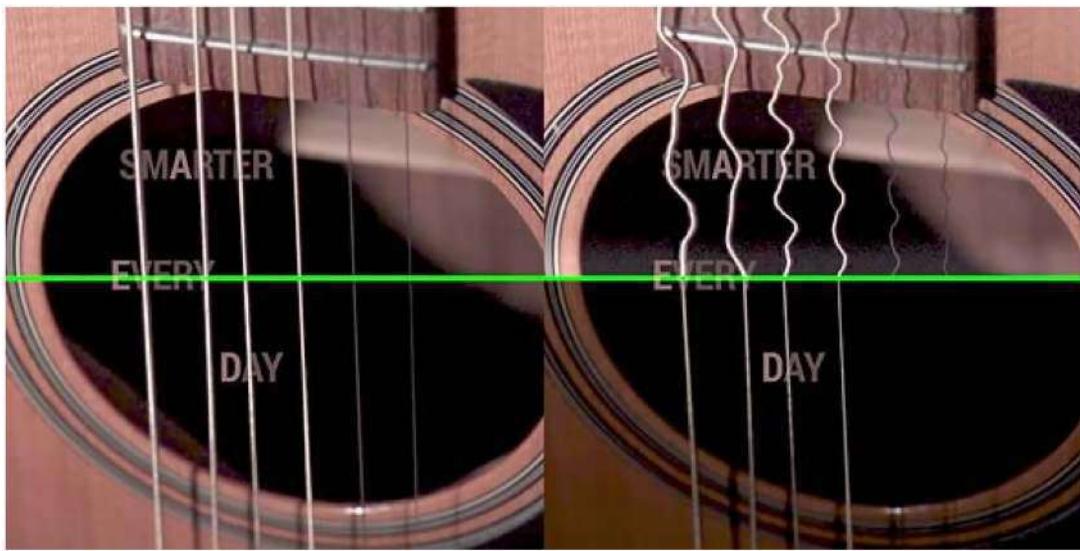
איור 74: במקומות כנפיים, קיבלנו הופעה מוזרה אבל גם מעניינת של כנף.

מלבד זאת, יש תופעות מעניינות, נבייט בתמונה הבאה:



איור 75: ככל שהאובייקט קרוב יותר לצלמתה, כך ההשפעה של *Shutter* חזקה יותר, ומיידה גם על מהירות גבוהה יותר (במקרה זה הצלם נע לכיוון שמאל). בambilים אחרים אובייקטים קרובים "זדים" מהר יותר בתמונות.

על ידי מדידת 24,000 שורות לשנייה, ופריטה על גיטרה, נקבל את התמונה הבאה:



איור 76: מבחיננו מדובר בתוצאה לא טובה, ונרצה להקטין את החשיפה של כל שורה, דהיינו למדוד יותר שורות בשנייה. אבל, קיבלו תוצאה די מגניבה.

ישנן דרכים להמודד עם *Rolling – Shutter*, ולא עוסק איתם. עבורנו העיקר לדעת שהתופעה הזו קיימת וכשנרצה לעבוד

עם תמונות זוזות נדרש לפתור את העניין הזה.

עתה נוכל להמשיך בנושאנו.

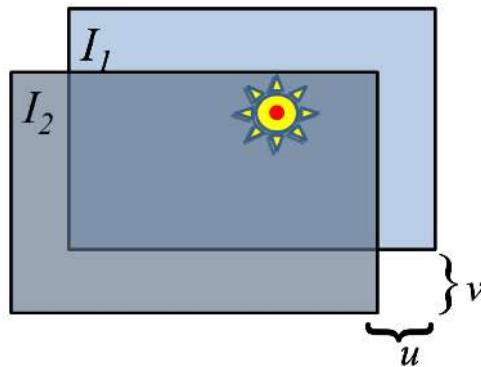
12.2 Direct שיטות

כאמור אחת המטרות שלנו היא למצוא את ההומוגרפיה שמשנה את התמונה לתמונה השנייה שקיבלו. על כן علينا למצוא פרמטרים מתאימים, כמו זווית סיבוב, או מרחק זהה, או שניי פרופורציות. דרך לעשות זאת, היא באמצעות שיטות יישורות "Direct"

מהן שיטות Direct?

הגדրתית, שיטות לחישוב ישיר, הן שיטות להערכת תנואה/צורה, המשחזרות את הפרמטרים הלא-ידועים יישורות מכניות מדידות בכל פיקסל בתמונה. בשיטות אלו אנו מנסים לצמצם פונק' שגיאה המבוססת על כל הפיקסלים בתמונה שיטה זו באה בנגדו לשיטה מבוססת נקודות אפיון (points Feature) בה אנו מוצאים מספר נקודות מסוימות מכל תמונה ומשחזרים ומנתחים את ההתאמות בין שתי נק' בשתי תמונות כדי לקבוע את הצורה וה坦ועה של הטרנס. בשיטות אלו אנו מצמצמים פונק' שגיאה המבוססת על מרחוקים בין מספר נק' אפיון מסוימות.

עתה, נרצה לחשב תזואה של תמונות, לשם פשוטות. לאחר מכן נעשו זאת לטרנספורמציות יותר מסובכות:



אייר 77: איך נחשב את u , v ?

כדי לחשב אותם מספיק למצוא נקודה אחת. אבל ראיינו כי באמצעות מספר נקודות משתנה כלשהו למצוא טרנספורמציה אפינית, הטלה וכו'.

עליה השאלה, האם אפשר לעשות זאת ללא לקיחת נקודות מהתמונה, הרי אם אין לנו אלגוריתםיעיל להתאמת, אולי כדאי לבצע חישוב ישיר? התשובה היא כן. אנו יכולים למצוא את הפרמטר (v, u) שיתן את ההזאה הכי טובה של שתי התמונות. למשל עבור I_1, I_2 נחסיר פיקסל וגעלה בריבוע. נקבל מינימיזציה לפונקציית השגיאה הבאה:

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

שתתן את u , v הרצויים.

ההפרש אף פעם לא אפס, בגלל רעשים ובגלל הזרזות שהן לא תמיד פיקסל שלם (אם אנחנו נזיז בחצי פיקסל את התמונה אנחנו נציג נס麤 להפעיל איזושי מניפולציה על הפיקסלים בתמונה ולא נשמר על הערכים המקוריים).

נרצה להקטין את הסכום (u, v) . עולה השאלה, מתי סכום ההפרש יהיה הכי קטן? נבחן כי כשאנו נזיז את התמונה אחת על השניה, דהיינו אוזר החיפוי לא קיים ולכן סכימה בכלל, השגיאה היא אפס, לכן צריך להיזהר, שלא נקבל פרמטרים חסרי משמעות כלשהי. על כן, כדי להתגבר על המקרה של אוזר חיפוי קטן, אנחנו עושים ממוצע על אוזר החיפוי. למה העלה בריבוע? כדי לשמר על ערכים חיוביים, למה ללא ערך מוחלט? כי נרצה לאוזר לאחר מכן. באופן דומה, אפשר לחפש על כל הפרמטרים, גם על הזרה וסיבוב. איך נעשה חיפוש מהיר להזזה? כמה מקומות צריכים לחפש בתמונה בטוחות.

$$\cdot 10^4 = 100 \cdot 100 \pm 100$$

הערה כשאנו אומרים "כמה מקומות צריכים לחפש בתמונה בטוחות", אנו מתייחסים שייצאנו מנקודת הנחה שבשתי התמונות הללו יש איזושי אוזר משותף, נגיד בשתי התמונות מופיע אותו הר (וכמובן בשתי התמונות ההר נראה אותו דבר). אנו מניחים שמקסימום התזזה בין שתי התמונות הוא 50 בכל כיוון (כלומר לא ניתן שההර בתמונה הראשונה נמצא בהזזה של 300 ימינה פיקסלים מהתמונה השניה).

שאנו תמיד צריכים לעשות הנחה כלשהי כדי שלא נציג נס麤 לחפש הזרק הכלול בתמונה, שכן זה יהיה חישוב יקר מאוד.

במקרה בו אנו מניחים מקסימום תזזה של 50 פיקסלים בכל כיוון, יש לנו 100 פיקסלים בציר האופקי ($50 \text{ ימין} + 50 \text{ שמאל} - 100 \text{ באני} (50 \text{ למעלה} + 50 \text{ למטה}), \text{ולכן } \text{סה"כ} \text{ נציג} \text{ לחפש}^2 100 \text{ הזרק.}$

הערה אנו מניחים גם שיש איזושי אוזר חיפוי שמשמעותו בתמונה, לדוגמה החצי הימני של התמונה, ואז בסכום $\sum_x \sum_y$ אנו סוכמים רק על ערכים $x-y$, כלומר הזרה שמשמעותו אותנו.

דבר זה שימושי בתמונה פנורמה לדוגמה, שם אנו יודעים שאמור להיות אוזר חיפוי גדול כי היד שלנו לא זהה הרבה כדי לצלם את החלק הבא בתמונה הפנורמה.

nocell לרשום

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

כאשר I_1^2 כמעט קבועים, שכן הם תלויים באוזר החיפוי. לכן כדי לקבל מינימום $E(u, v)$ נרצה מקסימום I_2^2 , $\sum_x \sum_y I_2^2$ (מקסימום ולא מינימום בغالל המינוס) של

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

את מה זה מזכיר? זה מזכיר *cross – correlation* של I_1, I_2 (זהו כמעט כמו קונבולוציה, רק בלי המינוס). לכן אפשר לחשב את הקروس-קורלציה בצורה מהירה, כמו שאפשר לחשב קונבולוציה במהירות.

בעיה. אם לתמונה שהזינו נוסיף 10 לכל פיקסל, מה יקרה? העין שלנו לא תבחן בזה, אבל החישוב של $C(u, v)$ ישפע מאוד. זו בעיה.

כדי לעשות זאת נרצה להתגבר על השינויים על ידי הסתכלות על הממוצע. הנוסחה שנשתמש בה היא כדלקמן: נניח כי $I_2 = aI_1 + b$ דהיינו שינוי אפיני, אז עבור

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

אנו נקבל שנפטרנו מרעשים כאלה. מסתבר שגם מחשבים לנו את הממוצע בבחנים כדי שהממוצע יהיה 85. זה אומר שאם בתמונה נוסף איזשהו קבוע, הנרמול יdag שהשגיאה תהיה מייצנת. בנוסחה \hat{I}_2 ה- \hat{I}_1 , I_1 , I_2 בהתחמה.

הערה נחזור על דברינו שоб: NCC-Normalized-Cross-Correlation שראיינו עכשו הוא אינוריאנטי להוספה של קבוע והכפלתו בקבוע. התמונות I_1 ו- $I_2 = aI_1 + b$ יקבלו קורלציה 1. משענות הדבר היא שצבעים מאבדים משמעות בין שתי התמונות ומה שמשנה זה שהטרנספורמציה ביניהם היא אפינית (לינארי פלוס קבוע).

איך אפשר לחסוך את החישוב 10^4 פעמיים? אפשר לבנות פירמידה - טשטוש ודגימה. אנו בונים פירמידה לשתי התמונות, איז אם אנו רוצים לחפש באזור מסוים, אנו יכולים לנשט לרמה מתאימה. צריך רק לוודא שהפירמידה כוללת את הפרטים שיש בתמונה - כולל הפרטים הקטנים. متى אין מידע זה? למשל, צילום שלחן, די מהר נקלט סתם צבע בז' ללא הסבירה שמדובר בשולחן. אנחנו צריכים **שיהיו פרטיים שמאפשרים לנו לעשות את ההתאמה**.

נלק לרמה המתאימה, בדרך כלל הגבואה ביותר ונמצא את ההזזה, אם התוצאה היא x ברמה הקודמת היא הייתה $2x$, רק צריך לשים לב שאנו מעגלים כלפי מעלה או מטה את התוצאה. בכל שלב אנחנו בודקים זאת ומכפילים ב-2 את התוצאה שקיבלנו. זה קורה בغالל שברוזולוציה נמוכה אנחנו מפסידים דיווק של פיקסל אחד ולכן ברמה הקודמת נוכל לתקן.

אפשר להשתמש בקורס-קורלציה גם כדי **לעקוב אחרי אובייקטים בסרטון**. אנחנו מצפים שב אדם בסרטון, בין פריים אחד לזה שאחריו, לא יקפו מאזור אחד בתמונה לאזור אחר. לכן נוכל לעשות קروس-קורלציה באזור שמסביב למקומות האחרונים בו מצאנו את האדם. נוח להשתמש בקורס קורלציה כאשר רוצים להתאים סביבה קטנה (למשל 5×5) לסייע אחרית בתמונה מואצת. אם ההזזה קטנה, אין יותר מדי אפשרות שcharיך לבדוק ונוכל לבחור את הערכים השונים מקסימום. זה ישמש אותנו כאשר נדבר על OpticalFlow

Lukas-Kanade 12.3

הבעיה בקורס-קורלציה היא שלוקח זמן רב לחשב אותה: N^2 עבור שני משתנים (כמו בהזזה - שם הפרמטרים הם (v, u)) ו- N^3 עבור 3 (כמו בסיבוב - שם הפרמטרים הם (α, v, u)).

כדי להתמודד עם זה הוא נשתמש בנסיבות - בשיטה שנקראית, ."LK" Lukas-Kanade, מתקיים כי $f(x+u, y+v) \approx f(x, y) + \frac{\partial f}{\partial x} \cdot u + \frac{\partial f}{\partial y} \cdot v$

נזכר שהבעה שלנו היא לסייע את

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

לשם פשוטות נסתכל על פיקסל בודד (בלי הסכימה): $E(u, v) = (I_2(x+u, y+v) - I_1(x, y))^2$. נסיק לפי טילור כי (לאחר נזירה של I_2):

$$E(u, v) = (I_2(x+u, y+v) - I_1(x, y))^2 \approx \left(I_2(x, y) + \frac{\partial I_2}{\partial x} \cdot u + \frac{\partial I_2}{\partial y} \cdot v - I_1(x, y) \right)^2 = (I_x \cdot u + I_y \cdot v + I_t)^2$$

כאשר $I_t = I_2 - I_1$, $I_x = \frac{\partial I_2}{\partial x}$, $I_y = \frac{\partial I_2}{\partial y}$.
כלומר באופן כללי אנחנו מנסים לסייע (ביחס (u, v)) את

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

כדי לקבל מינימיזציה, נוכל לגזר ולהשווות לאפס ולקבל כי

$$\frac{\partial E}{\partial u} = \sum_{x,y} 2 \cdot I_x (I_x \cdot u + I_y \cdot v + I_t) = 0$$

$$\frac{\partial E}{\partial v} = \sum_{x,y} 2 \cdot I_y (I_x \cdot u + I_y \cdot v + I_t) = 0$$

נוכל לרשום משווה זו בצורה מטריציונית:

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

זו מערכת עם 2 משוואות ו-2 נעלמים לא ידועים (u, v) . המערכת זו יש פתרון אם מטריצה יש שני ערכים עצמיים שאינם אפס (המטריצה סימטרית, לכן לפי המשפט הספרטורי היא לכסינה וב吐ו יש לה שני ערכים עצמיים).

כדי לפטור את מערכת המשוואות נוכל להפוך את המטריצה (אם הדרגה שלה מלאה).

הערה באlgorigitms האריס, המטריצה היא אותה מטריצה רק על חלון קטן. כאן הסכום הוא על כל הפיקסלים, لكن בדרך"כ שני הערכים העצמיים יהיו גדולים.

שאלה האם אלגוריתם זה מושפע מהתבניות?

תשובה לא, כי הוא מסתכל על התמונה הגלובלית ולא על אזור ספציפי.

אזורית חיפוי

היתרון המרכזי של לוקאס קאנדה, הוא שהוא לא מושפע מאזור חיפוי. באלגוריתם הקודם סכמנו על $I_2(x+u, y+v)$ ולכן במידה והערכים לא היו בגבולות התמונה, הינו צריכים לחזור ולהסתכל רק על אזור החיפוי עם u, v דבר שהוביל לכך שעבור u, v מסוים גודלים, קיבל אפס בפונקציית השגיאה. כמו, לוקאס קאנדה מניה שההזאות קטנות ומשתמש בקירוב טילור כזכור $u \cdot \frac{\partial I_2}{\partial x} + v \cdot \frac{\partial I_2}{\partial y} = I_2(x, y) + I_2(x+u, y+v) - I_2(x, y)$ וקיבלו למשה שאין שימושו לציאת מהגבולות, מבחןינו מדבר בתמונה אינסופית. למעשה, יציאה מהגבולות מקבלת ערך דומה לגבולות עצם כי u, v קטנים, וכך זה לא מטריד אותנו. מעבר לכך אם u, v גדולים מדי והיציאה מהגבולות גדולה, כל הקירוב לא נכון ולכן האלגוריתם עצמו יכשל.

אם הם באמות קטנים, אנו נקבל זאת בפתרון מערכת המשוואות, ולא נctrיך לדואוג מיציאה מהגבולות.

הרצאה 7

בהרצאה הקודמת נתנו בבעיה הבאה: אם התמונה הוסטה עם u, v גדולים, קירוב הטילור לא טוב, וכן האלגוריתם LK לא יעבוד. נרצה לפתור בעיה זו.

נבחן כי אפשר להתקדם בצורה איטרטיבית עם תוצאות של dv, du קטנים בכל פעם. אם כך, עבור (u, v) **גדולים** אנו יכולים לפעול בשיטה הבאה:

Algorithm 14 LK איטרטיבי (עבור (u, v) גדולים)

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

הערה חשוב לשים לב שבשלב ההזהה, **שלב ה-warping**, אנו לוקחים את התמונה המקורית ומזינים אותה ב- (u, v) , ולא לוקחים את התמונה האחורה שחשבנו ומזינים אותה ב- (du, dv) .

זאת כי אם לדוגמה אנו מזינים את התמונה בחצי פיקסל, אנו בעצם עושים טשטוש לתמונה. אז להזיז את התמונה חצי

פיקסל ועוד חצי פיקסל ועוד חצי פיקסל ועוד חצי פיקסל יטשטש לנו ממש את התמונה, בזמן שיכלנו פשוט להזיז את התמונה שני פיקסלים (ארבע פעמים חצי פיקסל) ואז לא יהיה טשטוש בכלל.

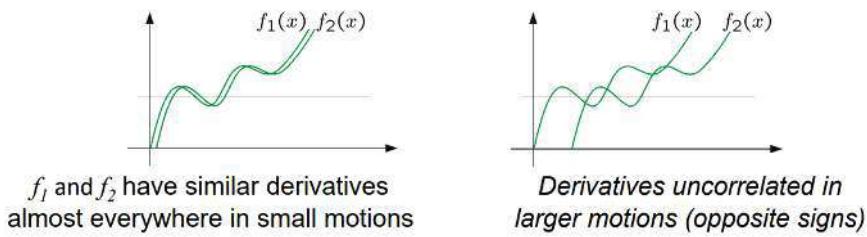
הערה פתרון המשווה, (\mathbf{dv}), הוא אולם לא מדויק אבל עצם זה שאנו חווים על התחילה וכל פעם מתקרבים עוד ועוד לכיוון הנכון, בסוף תביאו אותנו לפתרון טוב.

הערה באלגוריתם המקורי אחד היתרונות היה שלא קיבל 이것이 \mathbf{u} , היוצרת חיפוי אפס. כאן, זה נראה כאילו הבעיה חורה בغالל שמחשבים (\mathbf{v}) $I_t(x, y) - I_1(x + u, y + v) = I_t$. למרות זאת, נבחן כי A היא מטריצה קבועה, לכן אנו תמיד לוקחים בחשבון את כל פיקסלים בשתי התמונות. לכן תוצאות גדולות מדי לא יקבלו התאמה טובה. עתה, 이것이 \mathbf{u} נתן לנו לפחות התרבות לתמונה שהוזה תוך כדי שמירה על הסתכליות גlobלית. נבחר כי החישוב עצמו של I_t נעשה אכן רק על אזור החיפוי.

יחד עם זאת, יש מצבים בהם האיטרציה לא תתכנס, אז נשתמש בשיטה אחרת, המשמשת **בפרמידות**.

שימוש בפרמידות במקרה $\mathbf{u} = \mathbf{v}$ גדולים

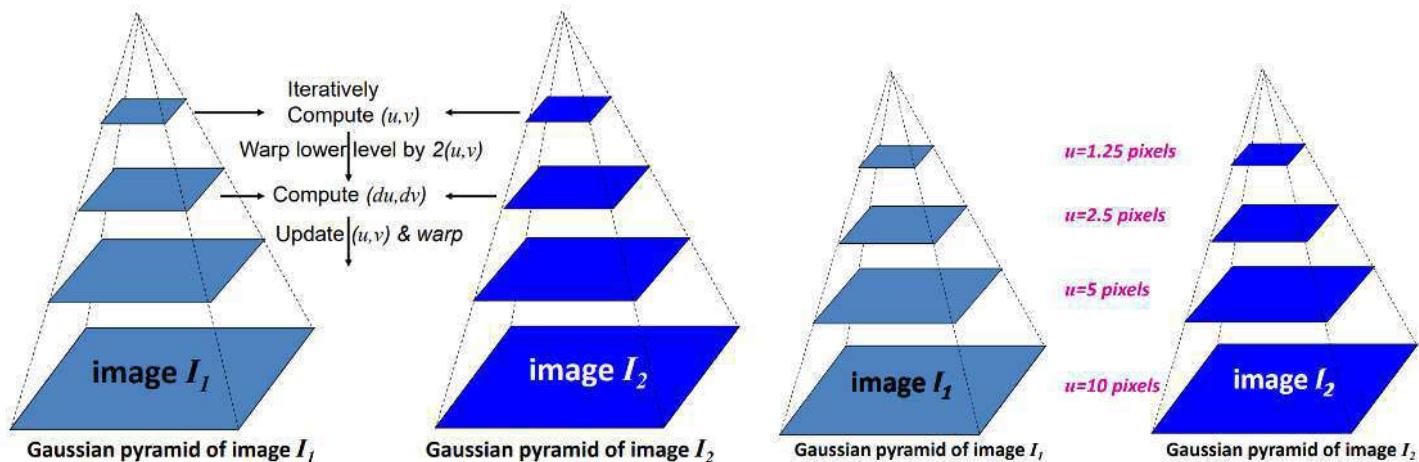
כדי שנוכל לעבוד בתנועות גדולות של (\mathbf{u}, \mathbf{v}), נוכל להשתמש בפרמידות באופן דומה לממה שכבר עשו בעבר. הסיבה ש-LK לא עובד טוב עם הזרות גדולות היא שהנחה שלLK היא שבשתי התמונות, הנגזרת בפיקסלים מתאימים בשתי התמונות זהה לשתי התמונות. הנחה זו סבירה גם כאשר הנזרות דומות, ולא בהכרח זהות. את הסיבה שהוא לא נכנס עבור הזרות גדולות של שתי תמונות ניתן לראות באירור הבא (דוגמה חד-מימדית):



איור 78: בהזזה קטנה (משמאל) הנזרות דומות מאוד בכל מקום. זאת בניגוד להזזה גדולה (ימין), שם לכל ערך x שנבחר הנגזרת ($f'_1(x)$ שונה מאוד מ- $f'_2(x)$, ואפילו הנזרות הן עם סימנים הפוכים).

כדי למנוע את הדבר הזה, אנו **מקטינים את התמונה** (וכמובן לפני הקטנה חובה לטשטש!).

כלומר נבנה פרמידה גאוסינית של תמונות מטווששות, כך שם במקור הייתה 이것이 הזרה של $10 = \mathbf{u}$ פיקסלים מ- I_1 ל- I_2 , ברמה הבאה בפרמידה זה יהיה כבר $5 = \mathbf{u}$ ואו $2.5 = \mathbf{u}$, וכך שלבסוף הזרה \mathbf{u} תהיה קטנה מאוד ונוכל להשתמש באלגוריתם LK בלי בעיה.



איור 79: בניית פירמידה גאוסיאנית לשתי התמונות, כדי להשתמש ב-LK.
באир התיכון מותואר גם אופן פעולה האלגוריתם המורחב של LK לפירמידות.

כלומר לאחר שאנחנו עושים LK ברמה התחתונה, אנו מקבלים ניחוש לערך של (v, u) ואיתו נעה לרמה הבאה בפירמידה.

בגלל שהרמה הבאה בפירמידה גדולה פי 2 באורך וברוחב, נכפיל גם את הניחוש שלנו: $(2u, 2v)$.

עתה, הניחוש ההתחלתי לתמונה שלנו יהיה $(2u, 2v)$ ואנחנו לא נתחל מהתמונות המקוריות, בהן הנזירות מאד שונות, אלא ניקח את I_1 , נזיז אותה ב- $(2u, 2v)$ (ונחשב את (du, dv)).

כך נמשיך בתהליך עד שנגיע לרמה الأخيرة.

LK עם פירמידות (שימושי כהנגזרות בשתי התמונות לא דומות)

- 1 : Compute gaussian pyramids for both images
 - 2 : Compute (u, v) at the highest level using LK
 - 3 : Repeat until we're at the lowest level:
 - Go to the next level in the pyramid - bigger size
 - **Warp** the current level by $(2u, 2v)$
 - Apply another iteration of LK on the current level and update (u, v) by the (du, dv) we've just found.
 - 4 : return (u, v) , as it is the translation.
-

טייף השיטה לא עובדת לכמ?تطשטשו עוד יותר! אםتطשטשו מספיק זה יעבד.

הערה למה לטשטש עוזר? דמיינו שיש לכם תמונה שחייבין שלה לבן וחצי שמאל שחור. בכל מקום הנגזרת היא אפס, מלבד מעבר משחור לבן, שם הנגזרת גבוהה. אם נזיז את התמונה זו ימינה או שמאללה, הנגזרת עדין תהיה אפס בכל מקום חוץ מבנקום בו יש מעבר משחור לבן בתמונה החדשה (כלומר הנגזרת מוזגת גם היא).

לכן, אם נפעיל את LK בغالל שהנגזרת היא אפס בכל מקום אם נזיז קצת ימינה או קצת שמאללה, הנגזרת תישאר אפס

ולא תהיה לנו אינדיקציה אם אנחנו הולכים בכיוון הנכון.
אם נטשטו את התמונה, הנגרת כבר לא תהיה אפס וכן תהיה אינדיקציה.

LK לעומת נקודות אפיון (Feature Points)

- חשיבות: בשני המקורים אנו עוברים על כל התמונה כדי לחשב נזירות חלקיות. סיבוכיות דומה.

- יעילות:

- אם ניתן למצוא נק' אפיון טובות, הם יהיו טובות יותר מ-LK שכן הם יכולות לחשב **הומוגרפיה**.
- בתמונות מוטשות או נק' עם דפוסים, עשוי להיות קשה למצוא נק' אפיון, שכן נדרש את LK.
- בהזזה LK יותר מדויק מנק' אפיון, שכן הוא גלובלי.

LK בהזזה + הגדלה אחידה

נניח שהזינו את התמונה ב- dx בציר ה- x , ב- dy בציר ה- y , והגדלו את כל התמונה באופן אחד בפקטור s .
אם (x_1, y_1) עבר ל- (x_2, y_2) נקבל:

$$x_2 = s \cdot x_1 + dx$$

$$y_2 = s \cdot y_1 + dy$$

נוסחת השגיאה שלנו הייתה

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

כאשר (v, u) הם משתנים שמייצגים את השינוי בין התמונות, $(x_1, y_1 - x_2, y_2 - y_1)$. במקור, השינוי היה רק הזזה. אבל כאן הם מייצגים שינוי שהוא גם הגדלה, וכל נקודה ישaza אחרת. על כן

$$u = x_2 - x_1 = (s - 1) \cdot x_1 + dx$$

$$v = y_2 - y_1 = (s - 1) \cdot y_1 + dy$$

כלומר s, u, v הם פונק' של (x, y) .

הערה אם אנחנו עושים הגדלה לתמונה 1000×1000 , ומקבלים תמונה חדשה לדוגמה 1200×1200 , אנו גוזרים את 200 הפיקסלים הקיצוניים (100 מימין, 100 משמאלי, 100 מלמטה ו-100 מלמעלה) כדי לקבל עדין תמונה בגודל 1000×1000 .
לאחר ההגדלה, אזור החיפוי יהיה רק האזור במרכז התמונה המקורי (שכן את הקצוות זרקו).

מהצבת v, u , פונק' השגיאה שלנו תהיה

$$E(dx, dy, s) = \sum_{x,y} (I_x(x, y) \cdot [(s - 1) \cdot x_1 + dx] + I_y(x, y) \cdot [(s - 1) \cdot y_1 + dy] + I_t(x, y))^2$$

גם הפעם נחשב את s באמצעות הנזירות והשוואתן לאפס:

$$\frac{\partial E}{\partial dx} = 0; \quad \frac{\partial E}{\partial dy} = 0; \quad \frac{\partial E}{\partial s} = 0$$

נקבל 3 משוואות לינאריות עם 3 נעלמים, dx, dy, s

$$\begin{aligned} \frac{\partial E}{\partial dx} &= \sum_{x,y} (I_x(x, y) \cdot [(s - 1) \cdot x_1 + dx] + I_y(x, y) \cdot [(s - 1) \cdot y_1 + dy] + I_t(x, y)) \cdot I_x = 0 \\ \frac{\partial E}{\partial dy} &= \sum_{x,y} (I_x(x, y) \cdot [(s - 1) \cdot x_1 + dx] + I_y(x, y) \cdot [(s - 1) \cdot y_1 + dy] + I_t(x, y)) \cdot I_y = 0 \\ \frac{\partial E}{\partial s} &= \sum_{x,y} (I_x(x, y) \cdot [(s - 1) \cdot x_1 + dx] + I_y(x, y) \cdot [(s - 1) \cdot y_1 + dy] + I_t(x, y)) \cdot (x_1 I_x + y_1 I_y) = 0 \end{aligned}$$

הפתרון למערכת זו ימזרע את השגיאה ויתן לנו את הערכים שאנו רוצים.

LK בהזהה + סיבוב

כמו קודם, עלינו לתאר את x_2, y_2 לאחר הטרנספורמציה כתלות בהזהה (dx, dy) .

אם כך,

$$\begin{aligned} x_2 &= x_1 \cos \alpha - y_1 \sin \alpha + dx \\ y_2 &= x_1 \sin \alpha + y_1 \cos \alpha + dy \end{aligned}$$

וכמו קודם,

$$\begin{aligned} u &= x_2 - x_1 \\ v &= y_2 - y_1 \end{aligned}$$

הערות

1. גם במקרה זה נעשה טור טיילור ל- α , $\cos \alpha$ ו- $\sin \alpha$ וזה יתן לנו טרנספורמציה ליניארית, מיד נראה. אבל אם α לא קטן, הטרנס' לא יהיה ליניארי וקירוב הטילור יהיה רע. ככלומר עבור α גדול האלגוריתם לא יעבוד טוב.
 2. אנו משתמשים בהנחה ש- α קטן רק לצורך פתרת המשוואות. ה-*warping* נעשה עם חישוב מדויק של \sin , \cos - ככלומר כשניזי' את התמונות, נשימוש ב- α , $\cos \alpha$, $\sin \alpha$ שמצונו ולא בקירוב.
 3. עולה השאלה, למה שלא נשימוש גם הפעם בפרמיידה כדי להקטין את α ?
- (א) פרמיידה **לא** מקטינה סיבובים, הסיבוב ישאר אותו דבר لكن הטריק הזה לא יעבוד כאן.

על כן, כדי שהמשוואות תישארנה ליניאריות, אנו רוצחים קירוב עבור זווית קטנות של סיבוב ב- α . כאשר α קטן נוכל לומר:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots \quad .1$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots \quad .2$$

לפי הקירוב, קיבל שהמשוואות ה-

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

$$y_2 = x_1 \alpha + y_1 + dy$$

ולכן

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

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

וכן

$$E(dx, dy, \alpha) = \sum_{x,y} (I_x(x, y) \cdot [-\alpha \cdot y_1 + dx] + I_y(x, y) \cdot -\alpha \cdot x_1 + dy + I_t(x, y))^2$$

לכן אם נגזר ונשווה לאפס נקבל:

$$\frac{\partial E}{\partial dx} = \sum_{x,y} (I_x(x,y) \cdot [-\alpha \cdot y_1 + dx] + I_y \cdot [\alpha \cdot x_1 + dy] + I_t(x,y)) \cdot I_x = 0$$

$$\frac{\partial E}{\partial dy} = \sum_{x,y} (I_x(x,y) \cdot [-\alpha \cdot y_1 + dx] + I_y \cdot [\alpha \cdot x_1 + dy] + I_t(x,y)) \cdot I_y = 0$$

$$\frac{\partial E}{\partial \alpha} = \sum_{x,y} (I_x(x,y) \cdot [-\alpha \cdot y_1 + dx] + I_y \cdot [\alpha \cdot x_1 + dy] + I_t(x,y)) \cdot (x_1 I_y - y_1 I_x) = 0$$

או בצורה מטריציונית:

$$\begin{pmatrix} \sum I_x I_x & \sum I_x I_y & \sum I_x (I_y x_1 - I_x y_1) \\ \sum I_x I_y & \sum I_y I_y & \sum I_y (I_y x_1 - I_x y_1) \\ \sum I_x (I_y x_1 - I_x y_1) & \sum I_y (I_y x_1 - I_x y_1) & \sum (I_y x_1 - I_x y_1)^2 \end{pmatrix} \begin{pmatrix} dx \\ dy \\ \alpha \end{pmatrix} = \begin{pmatrix} \sum I_x I_t \\ \sum I_y I_t \\ \sum (I_y x_1 - I_x y_1) I_t \end{pmatrix}$$

הערה אם נפעיל מספיק איטרציות התוצאה צפiosa להתקנס.

הערה באotta שיטה, נוכל לחשב כל טרנס' בעזרת LK כל עוד ההבדלים בין הפיקסלים לא גדולים מדי.

הרכבה של טרנספורמציות

נעיר כי כדי לקבל כמה טרנספורמציות שונות על תמונה, אנחנו מייצגים כל אחת בקוארדינטות הומוגניות, וכופלים אותן - בכפל מטריצות. רק בסוף, לא לאחר כל כפל, אנו מחלקים בקוארדינטה השלישית.

$$\begin{bmatrix} x_2 \\ y_2 \\ w \end{bmatrix} = \text{טרנספורמציה הומוגנית} \text{ אשר אנו משתמשים את התמונה על מישור הצלום שלנו. נקבל כי} =$$

ומהכפל וחלוקת ב-w נקבל כי $x_2 = \frac{ax_1+by_1+c}{gx_1+hy_1+1}, y_2 = \frac{dx_1+ey_1+f}{gx_1+hy_1+1}$. אנו משתמשים ממש חזק בחלוקת באיבר האחרון. שוב את החלוקת מבצעים רק בשלב הסופי.

בහנתנו נקודות השוואה של שתי תמונות כאשר אחת היא הומוגרפיה של השנייה, נסמן את הנקודות בהומוגרפיה

azi השגיאה תהיה

$$E^2 = (\hat{x}_2 - x_2)^2 + (\hat{y}_2 - y_2)^2 = \left(\hat{x}_2 - \frac{ax_1 + by_1 + c}{gx_1 + hy_1 + 1} \right)^2 + \left(\hat{y}_2 - \frac{dx_1 + ey_1 + f}{gx_1 + hy_1 + 1} \right)^2$$

הבעיה היא שזו לא פונקציה לינארית ולכן לא נוח למצער אותה, גם אם נגורר אותה, זה לא יהיה לינארי. כדי לטפל בזה אנו כופלים במכנה וממיצרים ביטוי זה.

זרימה אופטית - Optical-Flow

זרימה אופטית השינוי במקומות של כל פיקסל בין שתי תמונות.

נדמיין שמכונית נוסעת והנаг מצלם שתי תמונות בהפרשים של שניות של מה שקרה מחוץ לחלון. נניח שמדובר יש עץ קרוב ובתים רחוקים.

ברור לנו שהפיקסלים של העץ זו יותר בתמונה מאשר הפיקסלים של הבתים הרחוקים. כמובן, לכל פיקסל הייתה תנוצה מסוימת.

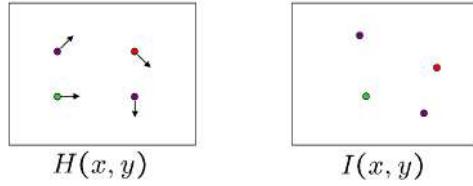
אם נחשב על שתי תמונות הללו תנוצה גלובלית, או של הזזה וסיבוב, או הומוגרפיה וכו', זה לא יעזור לנו כי אין כאן תנוצה גלובלית: לכל פיקסל התנהגות משלו.



איור 80: קשה לראות את ההבדלים, אך אלו שתי תמונות שצולמו במכונית נוסעת. פיקסלים רחוקים בקושי השתו, ופיקסלים קרובים (כמו העץ) זו הרבה. הבדלי ההזאה יוצרים אי רציפות, שכן העץ זו הרבה ועוד הבית בקושי זו.

הערה אם נפעיל על שתי התמונות לעיל LK , נקבל תוצאה לא רעה, שכן LK מנסה למצער שגיאת גלובלית, ובגלל שרוב התמונה מורכב מודחא בתמונה ולדוחא יש קוונרטסט גדול, זה מה שיצומצם ונקבל הזאה שמתאימה יותר להזאה של הדחא (שבמקרה שלנו זה לא רע בכלל). בambilים אחרות, כל עוד ההתנגדות הגלובלית בתמונה השתנתה בצורה אחדה בהזאה, הוא יעבד טוב.

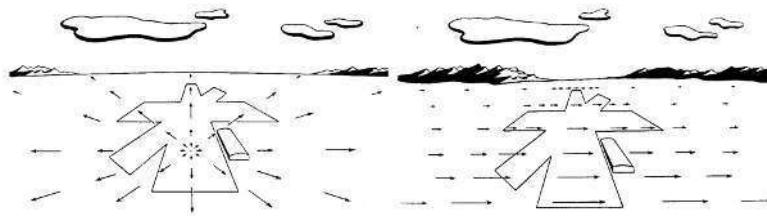
בhinntnu שתי תמונות, I, H , נרצה למצוא את הזראה של כל פיקסל. הזרימה האופטית תחזיר וקטור הזרה לכל פיקסל, המתאר לאיפה הוא עבר מ- I ל- H .



איור 81: שתי תמונות עם 4 פיקסלים. החצים הם וקטורי הاهזה של כל פיקסל, המתאים לאן כל פיקסל עבר מ- H ל- I .

הנחות כאשר אנחנו מחפשים זרימה אופטית, אנו מניחים שתי הנחות מרכזיות:

- (i) **הצבעים לא משתנים** - אם הייתי ירוק פה אני אהיה יрок שם (בתמונות אפור: הבהירות נשארת קבועה).
- (ii) **התנועות קטנות** (כל נקודה לא זהה הרבה).



איור 82: דוגמאות לזרימה אופטית.

דוגמא. באIOR הנ"ל יש שתי דוגמאות לזרימה אופטית.

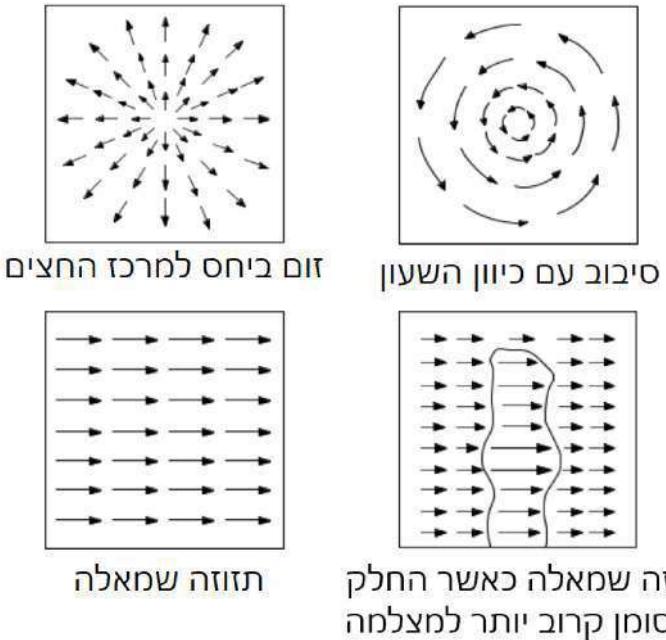
משמאלי: יש זרימה אופטית רדיאלית, לכן אנו מבינים שהטרנס' המתאימה לזרימה היא הגדלה (zoom) סביב נקודות המרכז של החצים או התקדמות לעבר המרכז.

אם היה מדובר בהגדלה (zoom) אז החצים ליד המרכז היו קטנים, וגדלים ככל שהיינו מתרחקים מהמרכז. אכן אין זה המצב: יש באפקח חצים מאד קטנים שרחוקים מהמרכז, אך בין שמדובר בהתקדמות. הסיבה שהם קטנים קטנים בכך שהוא פיקסלים מאד רחוקים.

דוגמא שמתחילה לסתואציה כזו היא מטוס בזמן נחיתה: הגוף נע למטה וקידמה לעבר נקודות המרכז.

מימין: יש זרימה אופטית בכיוון ימין, לכן אנו מבינים שהטרנס' המתאימה לזרימה היא האזהה בכיוון שמאל. שימושו לב CID הasticsים הרחוקים יותר כי אובייקטים רחוקים זוו פחות.

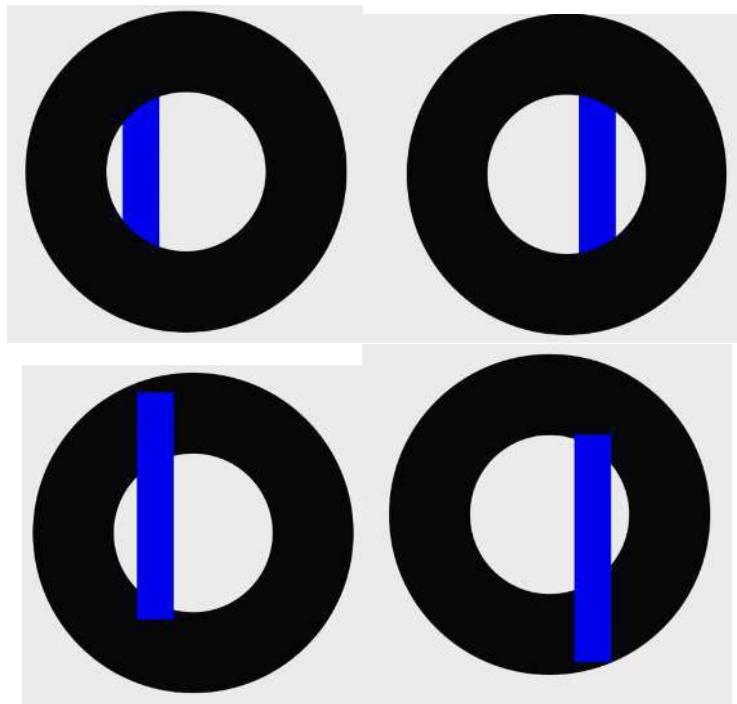
דוגמא. להלן מספר דוגמאות נוספות לזרימה אופטית. הכתוב מתחת כל תמונה מתאר מה התנועה שעשה הצלם כדי לקבל את הזרימה האופטית באIOR.



איור 83: דוגמאות נוספות לזרימה אופטית.

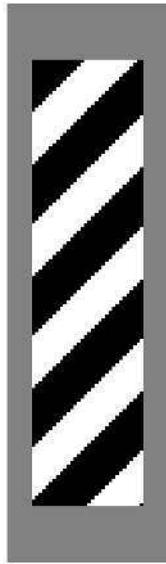
בעיית הצמצם - ApertureProblem

כשיש לנו צמצם של מצלמה, הוא לא עלול לתפוס את כל מה שקרה במציאות (הרי אין לו טווח ראייה אינסופי בכל הכיוונים). זה עלול ליצור בעיה בשם בעיית הצמצם.



אייר 84: זוג תמונות שצולמו אחט אחרי השניה (השMAILית צולמה קודם). התמונות הערלונות הן כפי שהן נראות דרך חירר מצלמה, והתחרתנותן הן כפי שהן נראות במצבות.
למעלה: נדמה שאובייקט נע מימין לשמאלי.
למטה: האובייקט בעצם נע בתנועה אלכסונית.

בתמונה כאן אין לנו מספיק אינפורמציה כדי להבין שהייתה תנועה לא רק אופקית. אם לדוגמה היה איזשהו corner או פרט מסוימת כלשהו, היינו יכולים להבין שהתנועה לא רק אופקית. מסיבה זו, נרצה להשתמש בסיטואציות כאלה באלגוריתם האריש, ודוקא edge-detector הוא לא טוב כאן.



Barberpole Illusion



איור 85: דוגמה נוספת לבנייה החירר. כאשרמוד הספר מסטובב, הוא יוצר אשלה אופטית שגורמת לנו לחשב שהפסים יורדים למיטה.

שאלה איך ניתן לחשב את הזרימה האופטית של זוג תמונהות?

תשובה נחלק את התמונה לאזורים (לדוגמה 5×5 או 7×7) ולכל אזור נחפש את האזור שמתאים לו בתמונה השנייה (קרי, עם הקורלציה הטובה ביותר).

איך ניתן לוודא שאזור α בתמונה א' מתאים לאזור β בתמונה ב'? נבדוק האם הדרך הפוכה מתאימה: ניקח את אזור β בתמונה ב' ונבדוק אם האזור שהכי מתאים לו בתמונה א' הוא α . דרך נוספת היא להסתכל על האзор הבא. רוב הפעמים אזוריים קרובים ימופו למקומות קרובים (זה לא נכון כשלש חוסר רציפות), ואז אם שני אזוריים סמוכים β, α בתמונה א' מומו ל- δ, γ בתמונה ב', כאשר δ, γ סמוכים גם הם, כנראה שההתאמה נכונה.

לבחירות שונות של גודל אזור יש יתרונות וחסרונות:

- **אזור גדול - יותר מיוחד וחד משמעות ✓.** **תנווה מדויקת** - ימנע את בנייה הצמצם, **локאליזציה לא טובה** - יכול להיות שהאזור הגדל שבחרנו מכיל כמה אזוריים קטנים שככל אחד אז באפן אחר, ולא נדע כמו צריך לאן הפיקסל אז.

- **אזור קטן - יהיה יותר התאמות אליו ✗.** **תנווה לא מדויקת, לוקאליזציה טובה.**

כדי להקטין את אזור החיפוש, ניתן להשתמש **בפרמידות**.

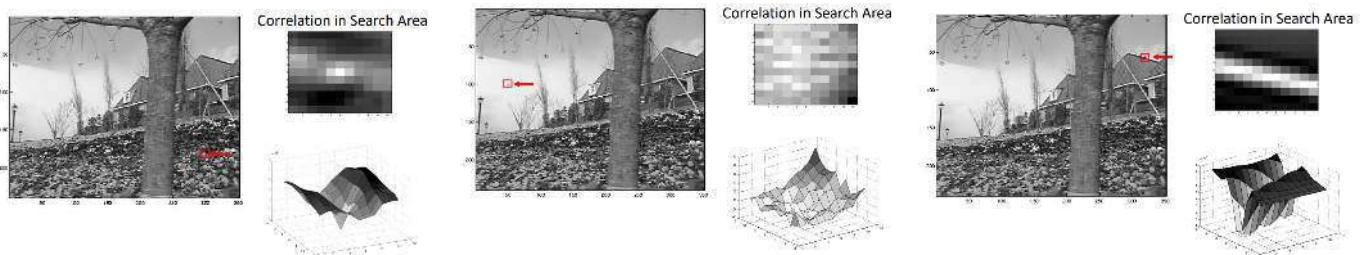
חישוב זרימה אופטית באמצעות פירמידות 16

- 1 : Create two Gaussian pyramids from the two input images
 - 2 : Compute optical flow using “ 5×5 ” regions on smallest pyramid level
 - 3 : Smooth the optical flow, and use it as initial guess for higher resolution
 - 4 : Continue with next level. Search close to guess from higher resolution
-

שאלה (שצריך לענות עליה) למה ההחלה חשובה? **אני לא הבנתי...**

תשובה (כנראה) כדי למצע שנייה ברזולוציה גבוהה.

נשים לב שבעת התאמת של נק' *edge*, שנמצאת על נק' *edge*, אנו מקבלים של נקודות בעלות קורלציה גבוהה ויהי קשה להבין לאיזו נקודה מתאימה. באזורי חלק קשה אף יותר להבין מי הנקודה המתאימה. באזור שיש לו טקסטורה טובה (לא *edge* ולא חלק) נקבל מעט מאוד נקודות בהן הקורלציה מאוד גבוהה, ולכן יהיה קל להתאים את הנקודה.



איור 86: מימין: קורלציה באזור חיפוש עבור פיקסל שנמצא על *edge*.
באמצע: קורלציה באזור חיפוש עבור פיקסל שנמצא באזור חלק.
משמאל: קורלציה באזור חיפוש עבור פיקסל שנמצא באזור עם טקסטורה מסוימת.

אפשר לחשב opticalFlow על ידי LK - המבוססת גרדיאנט:

מחשב (u, v) באמצעות LK לכל שני אזורים קטנים בתמונה. היתרונות והחסרונות לבחירת גודל האזור הם כמו קודם.

בחלק מהבעיות של זרימה אופטית אנו רוצים להניח שנקודות קרובות זאת בערך באותו האופן ולדרשו שתהייה מעין רציפות/חלקו. במצב זה, אנו רוצים למזער את סכום הנגזרות הראשונות (כדי שהשינוי יהיה מינימלי) ולכן בהינתן

$$\nabla I = (I_x, I_y)$$

$$\text{נרצה למצוא את } (u, v) = \text{ שמזער את } \nabla I = (I_x, I_y)$$

$$E^2 = \int \int \left(\underbrace{(\nabla I \cdot v + I_t)}_{(*)} + \underbrace{\alpha^2 \left(\left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2 + \left(\frac{\partial v}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial y} \right)^2 \right)}_{(**)} \right) dx dy$$

כasher (*) הוא הנוסחה שאנו מעריכים ב-LK ואנחנו מוסיף את (**) שהוא חלוקות: סכום הנגזרות התנועה בכל כיוון.

(**) מקבל ערכים קטנים כשהণויים קטנים. כאן α מצין את משקל החלוקת, לא זווית.

כלומר אנחנו מנסים להתאים את עצמנו גם לנוסחה של LK וגם לדאוג שהתנועה תהיה חלקה.

הערה. דרישת החלוקות נcona בהמון מקומות, אך לא בקצבות של עצמים נעים.

שאלה איך מטפלים בזה?

תשובה אי רציפות מופיעות כאשר אובייקט זהה ואובייקטים לידו נשאים במקומם, למשל, מכונית זהה. לכן אנו יכולים לטעות באזור זה. כדי לטפל בזה אנו צריכים להשתמש באלגוריתמים רובסטיים שלא מושפעים ממשמעותי משינויו קיצור - נראה במפורט בתרגול.

אם כך, קיבל את הוריאנט הבא לאלגוריתם:

Algorithm 17 חישוב זרימה אופטית באמצעות פירמידות עם LK

1 : Create two Gaussian pyramids from the two input images

2 : Iterative LK on smallest images

- Estimate velocity at each pixel by solving LK equations in its neighborhood
- **Warp** I_2 towards I_1 using the estimated flow field
- Repeat until converges

3 : Continue to next pyramid level.

נבהיר מה אנו עושיםפה:

• כמו ב-LK על פירמידות, מחשבים את הפירמידות לשתי התמונות.

• נתחל מהרמה הגבוהה ביותר (תמונה הכי קטנה)

▷ נחשב לכל פיקסל את ההזאה שלו (ווקטור ההזאה $v = u$), על ידי פתרת משוואת ה-LK בסביבה שהגדכנו לו (למשל 3×3) אל מול הסביבה בתמונה השנייה, זה לא יקר לחישוב, כי התמונה ברמה גבוהה בפירמידה, ונקבל שדה ההזאה \vec{v} לכל התמונה. את השדה נחשב תוך כדי דרישת חלוקות לפי המשווה שראינו קודם.

▷ נכפיל ב-2 את ההזאה, נרד רמה אחת בפירמידה ונוציא את התמונה לפי ההזאה. נחזור על התהליך עד שתהייה התוכנות.

13 פסיפס - ריצוף תמונות (Mosaicing)

כדי לתפוס שטח מאד גדול, אנחנו יכולים להשתמש במכשיר אחד עם טווח ראייה רחב מאוד, או לצלם המון תמונות עם מצלמה בעלת טווח ראייה פחות ורחב. הדרך הראשונה לא נוחה בכלל לעבודה, ולכן אם אנחנו בוחרים בדרך השנייה, علينا

להבין כיצד צריך לחבר (לרכז) את התמונות כדי שיהו תמונה אחת גדולה.

כדי לבנות תמונה פנורמה, למשל, המורכבת מ-4 תמונות: 4, 1, 2, 3, 4, ננקוט בשלבים הבאים:

- נחשב את התנועה היחסית בין כל שתי תמונות סמוכות.

▷ באמצעות KT או התאמת נקודות אפיון.

- נבחר תמונה אחת שתהווה מערכת ייחוס עבורנו.

- נתאים את כל התמונות לתמונה היחסוס.

▷ הרכבה של כמה תמונות (כפל מטריצות: לדג' ההזהה מ-4 ל-2 היא ההזהה מ-4 ל-3 וזו מ-3 ל-2).

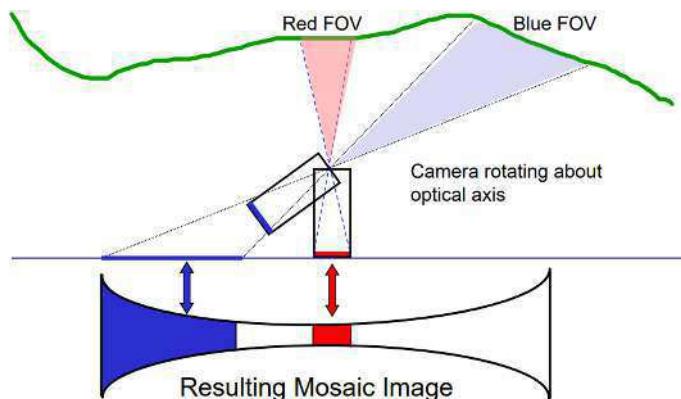
▷ נבצע Warping של התמונות לתמונה היחסוס (עדיף בbackward-warping כפי שלמדנו).

- איחוד התמונות לתוך תמונה פסיפס אחת - נראה עתה איך לעשות זאת.

הערה. אם נבחר תמונה ייחוס שונה, התוצאה תשתנה שכן שאר התמונות יבצעו טרנס' אחרות כדי להתאים אליה וכך גם תמונה היחסוס תמיד תהיה האמצעית.

נניח שאנו מצלמים את המשטח הירוק באיוור. התמונה האדומה שצילמנו תהיה תמונה היחסוס. עתה, כל תמונה נוספת שנצלם, נרצה להטיל על המישור של תמונה היחסוס. לכן, אם נסובב את המצלמה בזווית **围绕着中心点旋转** התמונה המולטית תהיה גדולה יותר.

Extending image plane of reference frame



איור 87: הטלה של תמונה על מישור של תמונה היחסוס, בכלל שחריר התמונה מצלם חלקים שונים מהקן, חלקים עם מרחקים גדולים יותר יקבלו צילום שטח רחב יותר ולכן הגודל היחסי של חלקים אלה יהיה גדול יותר ביחס לחלקים קרובים, لكن בשיטה זו, הקצוות בתמונה הסופית יהיו גדולים יותר.

בזווית שוגדלה מ-90 מעלות תיאורטית נקבל שוגדלו התמונה יהיה אינסופי, כלומר זה לא אפשרי להטיל תמונה בזווית כזו.



Limitation: “Explosion” at 90°

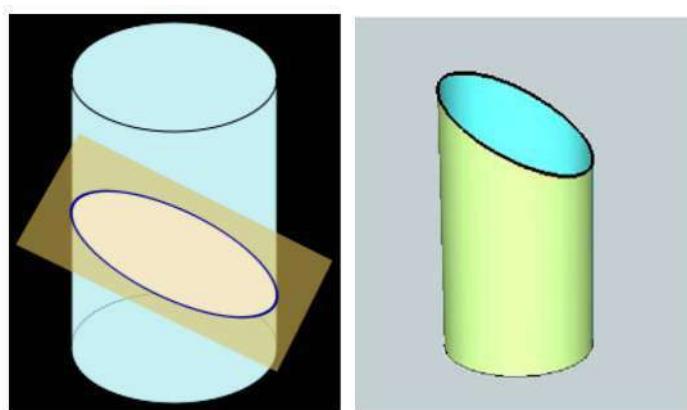
איור 88: חיבור תמונות שצולמו עם שני זווית ביחס לצמצם. שימוש לב כיצד הקצוות ארוכים יותר מהמרכז.

שאלה כיצד ניתן לפטור את בעיית התהפטוצות ב-90 מעלות? מה אם נרצה להסתובב אפלו ב- 360° ?

תשובה במקומות להטיל את התמונה על מישור (להזיכרכם, הבעה הייתה שלא ניתן להטיל ב-90 מעלות על מישור), נטיל את התמונה על צילינדר.

הטלה על צילינדר תוכל לבטל את האפקט של הקצוות עם החתק של מישור הצלום עם הצילינדר אופקי ונוצר מעגל, אחרת החתק הוא אליפסה ויוצרו מרחקים שונים מהחריר ולכן יהיו אפקטים של קצוות.

נשים לב שבhetלה על צילינדר, קווים ישרים לא נשמרים. זאת כי בגין הטלה על מישור, כאשרחנו מטילים על צילינדר אנחנו חותכים את מישור התמונה עם צילינדר ומתקבל מישור עקום, ולכן אנו רואים חותכים של אליפסות בתמונה התוצאה. החיתוך נותן אליפסה כשהוא אלכסוני, וכשהוא ניצב אנו מקבלים תמונה נורמלית.



איור 89: חיתוך של מישור עם צילינדר, החיתוך אליפטי כאשר הוא אלכסוני, כשהוא אופקי נקבל מעגל.

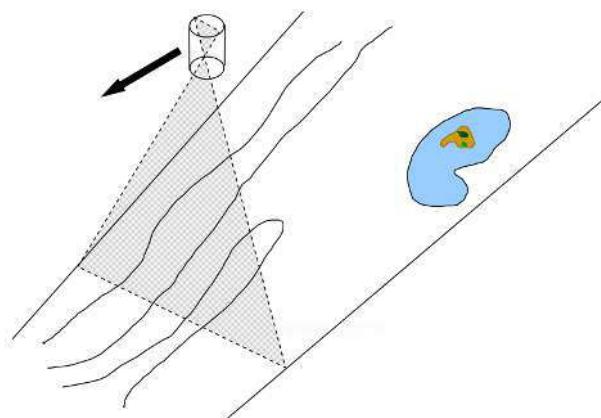
למשל, הקמפוס של מייקרוסופט, צילום על ידי הטלה על צילינדר, נראה כך:



איור 90: בגל שהחטלה על החירר היא אליפטית אנו מקבלים שהקוויים הישרים הופכים לאליפטיים. יש כאן הרבה מהרבה מאוד תמיונות שאוחדרו לכדי תמונה אחת פנורמית. למעשה אפשר לחשב את כמות התמיונות, אך מה שמשמעותן זה שבכל תמונה,ekoויים הישרים נשארים ישרים, האקט של האליפסה נובע מהאחדות של התמיונות כחטלה על צילינדר. יש לנו הרבה קווים ישרים שמאוחדרים לאובייקט אחד וההתוצאה היא קו עקום.

עתה נרצה להבין כיצד יש לחבר את התמיונות כדי לקבל פסיפס אחד. נתחיל מדבר על מצלמת *pushbroom*.

מצלמת *pushbroom* היא מצלמה שנמצאת על לוין או מטוס, שצלמת רק שורה אחת אורך של פיקסלים, היסטורית זה נבע בכך שהיה לה רק 10^4 פיקסלים לצילום ולכן הייתה יכולה ליצור תמונות דו מימדיות של 100×100 , لكن בחרנו לצלם שורה אורך בכל פעם ולהזoor על התהליך כדי לקבל תמונות גדולות. לעומת, המטוס/לוין התקדם ואז המצלמה צילמה עוד שורה אורך בכל פעם ולהזoor על התהליך כדי לקבל תמונות גדולות. לעומת, המטוס/לוין התקדם ואז המצלמה צילמה עוד שורה. כך קיבלנו תמונה דו-מימדית: מימד אחד הוא שורת הפיקסלים והימיד השני הוא התזואה של המטוס. קוראים לה *pushbroom* כי היא באמות מזקירה מטאטא באופן פועלתה. מה היתרון בצלמה זו? אנו יודעים בדיקות את התנועה של הלוין.



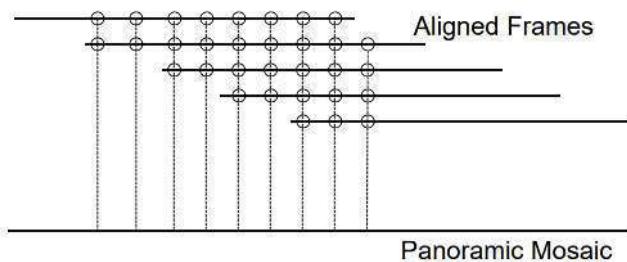
איור 91: המכשלה למצלמת *pushbroom*

היום אין לנו מצלמת *pushbroom*, אבל נרצה לעשות סימולציה שלה. היתרון בצלמת *pushbroom* הוא שהוא יודעים בדיקות מהי התנועה של הלוין ביחס לקרקע ונitin לבנות את הציר בכיוון התנועה לפי המידע שלנו על התנועה.

במצלמה כיום לא ידועה התנועה, אך באמצעות התמונה אנו יכולים לחשב את התנועה! ניקח את התמונות שלנו, שהן הזזה אחת כלפי השניה, ונחשב את התנועה, לדוגמה באמצעות LK. ניקח פסים מהתמונות ונבדיק אותם ביחד. אם הזזה בין התמונות הייתה **גדולה** או **שהיה הרבה שינוי** בין שתי התמונות, העמודה שניקח תהיה **עבה**, ולהפוך. הערת. אם המצלמה זהה 10 פיקסלים ולקחנו פס של פיקסל אחד, התוצאה תהיה מכוכצת. אם המצלמה זהה פיקסל אחד ולקחנו פס של 10 פיקסלים, יהיה לנו שכפולים. על כן, חשוב לנקוט רוחב פס בהתאם לרוחב התזוזה.

איחוד התמונות בפסיפס

כשאנו רוצים לחבר את כל התמונות לרצף אחד, לפני הכל צריך לדאוג שאנו מישרים את כל התמונות כך שיהיו בקו אחד, אז נוכל להשתמש באחת מכמה שיטות:
נוכל לבחור ממוצע או חיצון. אם אנחנו בוחרים באפשרות זו **צריך התאמת מאוד טובה** אחררת התמונה תצא מושטשת.

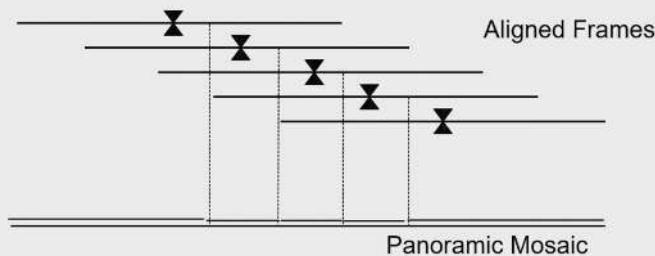


אייר 92: יישור התמונות לקו אחד ולקיחת ממוצע/חיצון

לחופין, אפשר לנקוט פסים מכל תמונה לפי קווי תפ. אם לדוגמה נסתכל על נקודות המרכז של כל התמונות בציר ה- x , נוכל לבחור שקו התפר יהיה בדיק באמצע בין כל שתי מרכזי תמונות. כך, מכל תמונה ניקח את האזור שקרוב למרכז יותר מכל אחד אחר. את הפריים הראשון והאחרון אפשר להעתיק עד הסוף עם קצוות שלهما.

2. Cut & Paste Center Strips:

- Mark centers of frames -
- Cut at midpoints between centers

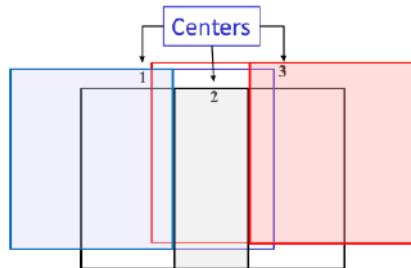


אייר 93: יישור התמונות לקו אחד ושימוש בנק' אמצע לצירוף הפסיפס

כأن צריך התאמת מושלמת רק לאורך קווי התפר (הנקודות של הפסים) – לא צריך שכל הפס יתאים. כדי לבחור איפה לחתור (איפה למקם את קווי התפר) אפשר להשתמש באלגוריתם *Min – Cut* או באlgorigthmi תכנון דינמי כדי למזער את השגיאה ולגרום לחבר בין התמונות להראות טוב כניתן.

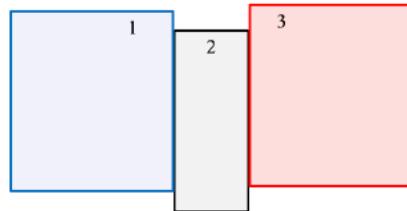
הערה אפשר להשתמש גם ב-*Pyramid Blending* כדי לדאוג שההתפירה תהיה טובה, לא חובה להשתמש ב-*Min – Cut* וככז.

לצורך המוחשה נביט בהדבקה הבאה:



איור 94: בחירת כל חלק בתמונה לפי מקומו בין המרכזים של החלקים האחרים,

דבר זה יביא לתוצאה הבאה:



איור 95: החלקים שנבחרו

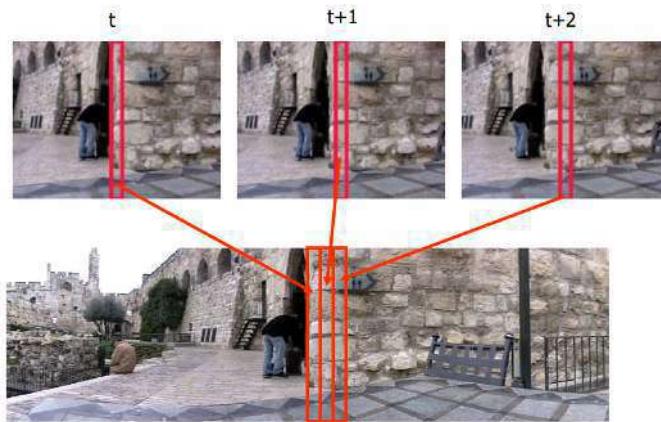
הערה (לודא שזה נכון) הסיבה שלא לקחנו את החלק מעל תמונה 2 (על אף שהוא משותף ל-1, 3) היא שהיברנו כאן את התמונות בזוגות צמודים: תופרים את 2, 1, ואז תומנת התוצאה תופרים עם 3, כך שכשתפרנו את 1, 2 החלק מעל 2 נפל.

הערה ההנחה העיקרית בתפירה כאן היא שהתנווה בין התמונות היא אופקית. אם התנווה הייתה אנכית התפר היה קו אופקי (בניגוד לקרה בו התנווה הייתה אופקית ואז התפר היה קו אנכי).

בהמשך נבין לעומק את השימוש ב-*minCut*, אבל לעת עתה, הכוונה היא שהאחד של התמונות נעשה על ידי מיזור הגרדיאנטים בנקודות, כדי שלא יהיה שינוי חד של הדבקה. למשל, הדבקה לא טובה תתן הבדלי צבעים חדים באיחוד ולא תמונה יחידה. בעצם, יש לנו מטרה דומה ל-*pyramidBlending*, איחוד נקי של התמונות.

הרצאה 8

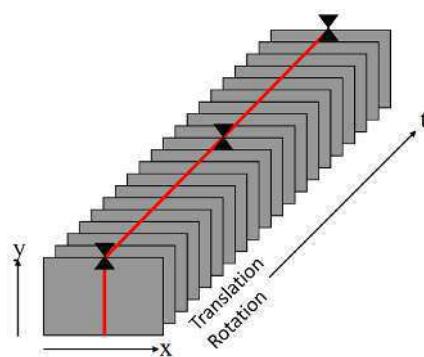
נתחיל בתזוכרת. נזכיר שבתפירת תמונות, כשמדבר בזהה עליה האפשרות פשוט לדמות מצלמת *pushbroom*



איור 96: תפירת תמונות בפסיפס באמצעות שיטה המדממת *pushbroom*

אנו מודדים את התנועה בין פרויימים עוקבים, ולקחimos פסים מכל פרויימים ומדביקים אותם ביחד כמתואר באירור. אם פרויים t היה הפרויקט הראשון היינו לוקחים את כל חלקו השמאלי, ואם $t + 2$ היה האחרון היינו לוקחים את כל חלקו הימני. מפרויימים שאינם הראשון או האחרון ניקח פס בודד שגודלו כפול התנועה: אם אזנו הרבה הפס יהיה עבה, ואם אזנו מעט הפס יהיה צר.

אמרנו גם שניתן להשתמש ב-*Cut – Min* – *Pyramid Blending* כדי לתרוף את התמונה. נניח שאנו מצלמים סרתוון תוך כדי שאנחנו זים או תוק כדי שאנחנו עומדים באותו מקום ומסובבים את המצלמה. כך יש לנו המונח פרויימים של תנועה פשוטה: הזיה (בעיקר בכיוון אחד) או סיבוב. נקבל פונק' $f(x, y, t)$, כך ש- $f(x, y, t)$ הוא ערך הנקודה (x, y) של התמונה צולמה בזמן t . אוסף כל התמונות הללו נקרא *Volume Space-time*. לאחר שהчисלנו את התנועה בין התמונות, אנו יכולים לבדוק את התמונות באמצעות קיוח פס מהאמצע של כל תמונה (הגדלים של הפסים תלויים בגודל התנועה). הטכניקה זו נקראת *VideoBrush*. לאחר שהקחנו את הפסים, כדי שלא יראו תפירה אפשר להשתמש ב-*Pyramid-Blending*.



איור 97: ב-*VideoBrush* אנו לוקחים פסים מהאמצע של כל תמונה ומחברים אותם יחד. עובי הפס תלוי בתזואה.

13.1 ריצוף תמונות עם מספר נקודות מבט

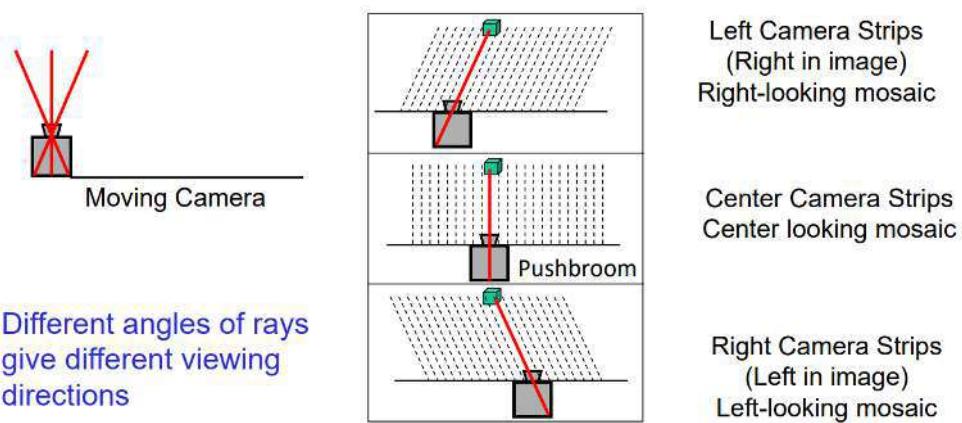
כאשר מצלמה מצלמת תמונה, קרני אור נכנסים אליה באזויות שונות. בתוך הקamera אובסקורה, החלק השמאלי ביותר של האור שנפל על המצלמה, מתאים לחלק הימני של התמונה (אקרו שהתמונה הפוכה), ולהפך.

על כן, אם בעת ריצוף נבחר פס ממוצע כל תמונה וכן נרצף, נקבל שאנו מסתכלים על הסצנה מקדימה. אם נבחר פסים מצד שמאל של המצלמה (זה צד ימין של כל התמונה), נקבל פסיפס שמסתכל לצד ימין של האובייקט (או אIOR).

כל בחירה של המקום ממנו ניקח את הפסים יתן פנורמה אחרת.

אפשר לחשב על הפס השמאלי בצלמה (שוב, ימין בתמונה), כמו שנכנס לעין שמאל שלנו, ועל הפס הימני בצלמה כמו שנכנס לעין ימין שלנו. זה מה שנקרה ה-Stereo-Effect.

הערה אם אנחנו יוצרים שתי פנורמות מפסים במקומות שונים, אפקט הסטריאו יהיה יותר חזק כהה שהפסים שניקח יהיו רחוקים יותר זה מזה, כי אז שתי התמונות מוצגות מזווית ראייה מאוד שונות זו מזו.



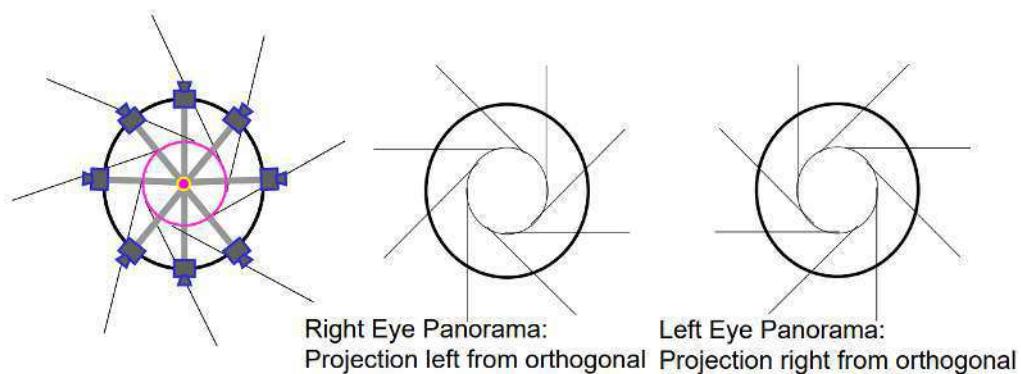
אייר 98: בעת ריצוף, ניתן לקבל תמונות שונות באמצעות בחירת המקום ממנו אנו לוקחים את הפס.

כך לדוגמה, נוכל לקבל את שתי הפנורמות הללו:



איור 99: בתמונה העליונה: אנו לוקחים פסים מצד שמאל של כל תמונה (ימין של המצלמה), כך שאנו רואים את הקיר השמאלי של הבתים.
בתמונה התחתונה: אנו לוקחים פסים מצד ימין של כל תמונה (שמאל של המצלמה), כך שאנו רואים את הקיר הימני של הבתים.

אם במקומות תנואה מכיוון אחד לאחר, היה לנו סיבוב סביב ציר קלשו, קורה דבר קצר אחר.
נניח שמקל תמונה ניקח פס מצד שמאל של התמונה, ימין של המצלמה (כמתואר בחלק השמאלי של האיור המצורף), נקבל של הקרןיהם משיקות לאיזשהו מעגל, שבאיור סומן בלבן ורוד.
הקרןים מצד שמאל מותאמות לעין ימין. הקרןים מצד ימין מותאמות לעין שמאל.



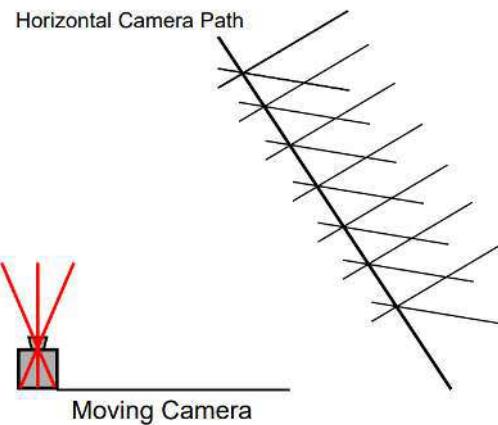
איור 100: פנורמה של פסים מצד שמאל של התמונה מותאמות לעין ימין, ולהפך.

בעיניים שלנו אנחנו רואים ב-*stereo*: אם מסתכלים על בן אדם כל עין רואה חלק אחר מהאדם באופן מאוד דומה למה שתיארנו זה עתה. אחרי 90° אנחנו כבר לא יכולים לראות בעיניים שלנו שום דבר (בטח שלא 180° , כלומר סטריאו זה דבר נורא צר. כדי לראות סטריאו עליו להכניס משהו לעין ימין ומשהו לעין שמאל שייהיו שונים אחד מהשני).
עוד דבר שאנחנו יכולים להסביר - משקפי תלת ממד כחול אדום שמקבלים בכל מיני ספרי תמונות. מה קורה שם בעצם? צבע

אחד מכוון לעין אחת והציבו השני מכובן לעין אחרת, ככה שכל עין מקבלת צד אחד של הבטים בציבו אחר ונוצר אפקט של תלת מימד. למעשה, אם ניקח את משקפי הולונוע הקלאסיות (המשקפיים בהן צד אחד הוא אדום וצד אחר הוא *cyan*) ונשים שתי פנורמות אחת על השנייה, אחת עם הדפס אדום ואחת עם הדפס *cyan*, נקבל אפקט של עומק - תחושה של *stereo*.

אז לסייע, גם בסיבוב ניתן ליצור פנורמות שמסתכלות מזוויות שונות באמצעות חירית המקום ממנו ניקח את הפסים. נסתכל על מצלמה שזזה בקו אופקי. דרך חירר המצלמה, מלבד זה שנכונות קרני אור מימין ומשמאלו (כמתואר בחלק התיכון של האיור) ננכונות קרני אור מלמעלה וממטה (כמתואר בחלק העליון של האיור, בו הפס העבה מתאר את מיקום החירר לאורך הזמן).

בכל רגע בו החירר זו ננכונות קרני אור חדשות מכל הכיוונים: מניפה של קרני אור שיכול עברות דרך החירר.



איור 101: כשהמצלמה זזה מימין לשמאלו, קרני אור חדשות ננכונות דרך החירר מכל הכיוונים.

גיאומטרית, בכיוון x הקרןיהם מקבילות אחת לשנייה, כי אלו איזים בכיוון x . לכן הקואורדינטה x בתמונה שווה לקואורדינטה x בעולם האמיתי.

לעומת זאת, הקואורדינטה בכיוון y בתמונה משתנה: בגלל שהוא עובדים עם חירר ככל שהגוף רחוק יותר הוא נראה קטן יותר (הזכירו בפרק הראשון בספר זה). לכן, אם נסמן את אורך המוקד של המצלמה ב- f נקבל שהמעבר מהעולם אמיתי לתמונה הוא כדלקמן:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} x \\ f\frac{y}{z} \\ f \end{pmatrix}$$

הערה במלצת פרספקטיבה המכפל ב- $\frac{f}{z}$ קורה גם ב- $-x$ וגם ב- $-y$. אולם כאן מדובר במלצת *pushbroom* ובה הקרןיהם בציר x מקבילות.

שאלה איך נראה תמונה בעולם בה הקרןים בציר x מקבילות וב- y הן פרספקטיביות?

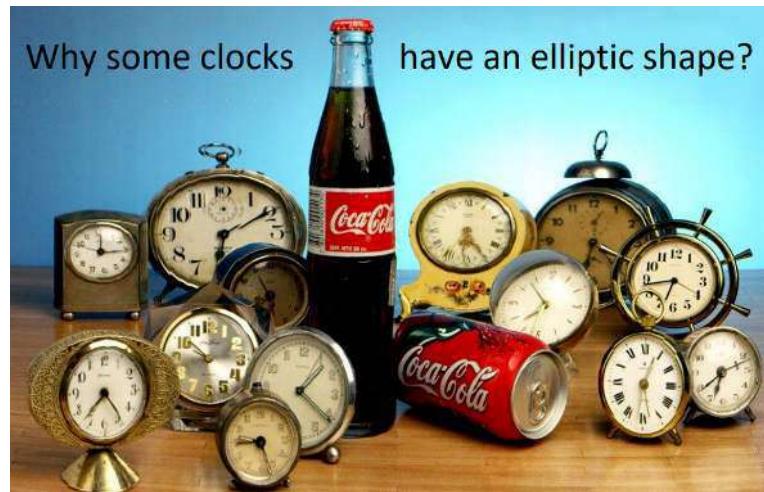
תשובה אובייקטים רוחקים נמוכים ונדחסים, ואובייקטים קרובים נמוכים ונהיים גבוהים ודקים. גופים רוחקים יותר קצט **כשمزיזים את המצלמה בפרשפקטיבת.**

בכיוון x - יש הטלה מקבילתית, כשمزיזים שמאליה יימינה אין הבדל בין גוף קרוב לגוף רחוק - אין תלות במרחב. על כן,

הרוחב של הגוף לא משתנה

בכיוון y - ככל שימושו רוחק יותר הוא נמוך יותר ולהפך.

ניתן לראות זאת באירור הבא.



איור 102: דוגמה לתמונות פנורמה הממחישה את ה-*pushbroom*. לא אני צילמתי את התמונה אך צריך לקרוא לה – *cola*.

13.2 פסיפס דינמי

נניח שאנו מצלמים סרטוון של עץ שהעלים שלו מותנדנים ברוח ועל האדמה רואים את הצל שלהם. הסרטוון מצולם כך שתחילה רואים את האדמה, אז המצלמה מסתכלת למעלה יותר וייתר עד שהיא רואה את העלים.

ונכל ליצור מהסרטוון זהה פסיפס באמצעות ליקית פסים אופקיים (המצלמה זהה למעלה لكن לא ניקח פסים אנכיים). באמצעות מעבר על כל מיקומי הפסים, יוצרת פנורמה בהתאם למיקום הפסים, וחיבור הפנורמות השונות, נקבל סרטוון של כל העץ במלואו (כולל האדמה והעלים) מתנדנד ברוח.

אולם, **בגלל שהאדמה צולמה בזמן שונה מהעלים, הצל על האדמה לא יתאים לתנועה של העלים.** זייפנו את הזמן - דברים נראים שהם קורים סימולטניים על אף שהיא הפרש זמן. לדבר זה אנו קוראים זמן לא כרונולוגי.

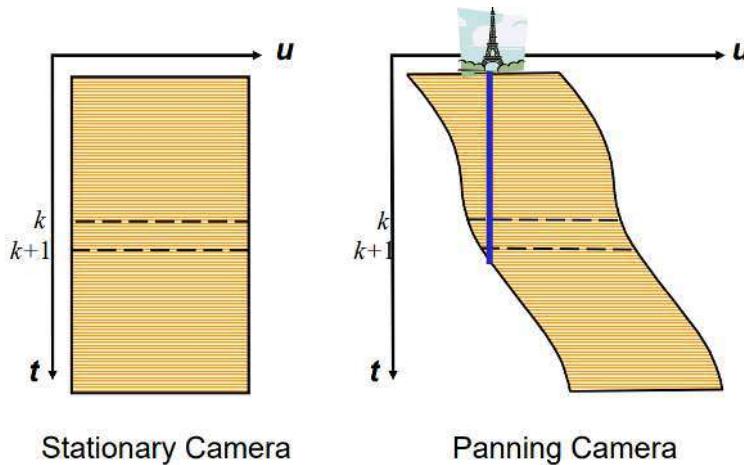
יצירת Dynamic-Mosaic

נעבור לפי השלבים הבאים:

(i) דגם תמיות - ניקח את התמונות שמרכיבות את הסרטוון.

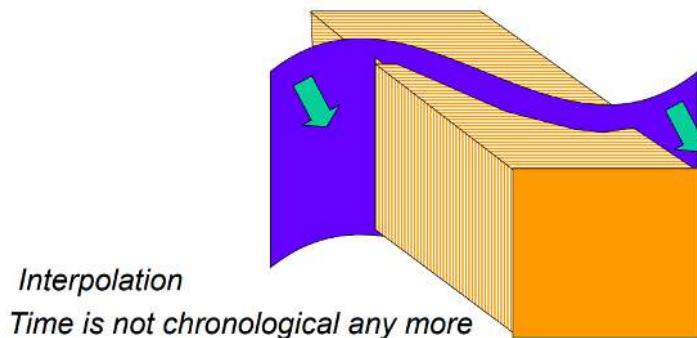
ii) יישור - נייר את התמונה כפי שלמדונו: נמצא טרנס' באמצעות *MOPS* ו-*Harris*, או באמצעות *SIFT*, ונפעיל אותה על התמונה שלנו. כך נקבל התאמה בין כל הפריים.

נשים לב שגם המצלמה לא זהה (צילום על חצובה), כל התמונות מראש יהיו מישוריות. כך אנו יוצרים *Aligned-Space-time-Volume*.

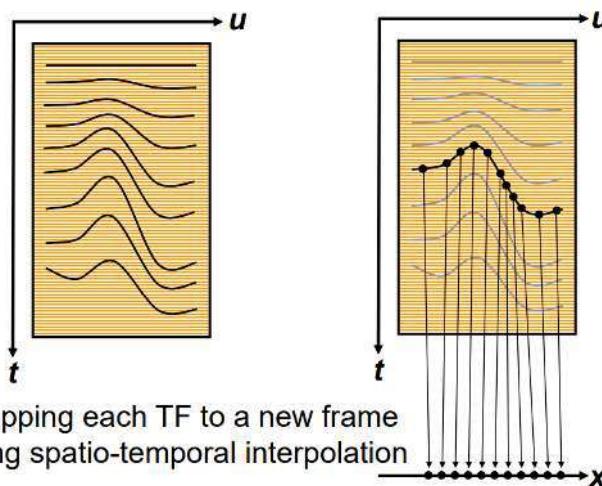


איור 103: בצד שמאל: המצלמה לא זהה לנו אך כל התמונות היו מישוריות.
בצד ימין: המצלמה זהה לנו אך ה-Volume עבר יישור.

iii) התאמת זמינים - כל תמונה ב-Volume צולמה בזמן שונה, לכן כדי ליצור כל מיני אפקטים מגניבים אנו צריכים להחליט בכל מקום מאייה זמן לחתת את ערך הפיקסל. לאחר שנבחרו את הזמנים וניצורו את כל הפריים, נקבל סרטון חדש בו כל חלק בניו מזמינים שונים.

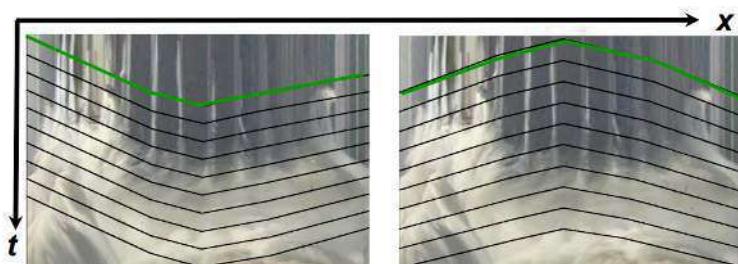


איור 104: בכתום - ה-*Aligned-Space-time-Volume* Shlomi.
בסגול - פריים כלשהו הסרטון החדש שיצרנו. צד ימין של הפריים נלקח מצד מוקדם יותר מאשר צד שמאל.



איור 105: Evolving-Time-Front. הפריים בסרטון (צד ימין למיטה) נוצר מנקודות שונות בזמן.

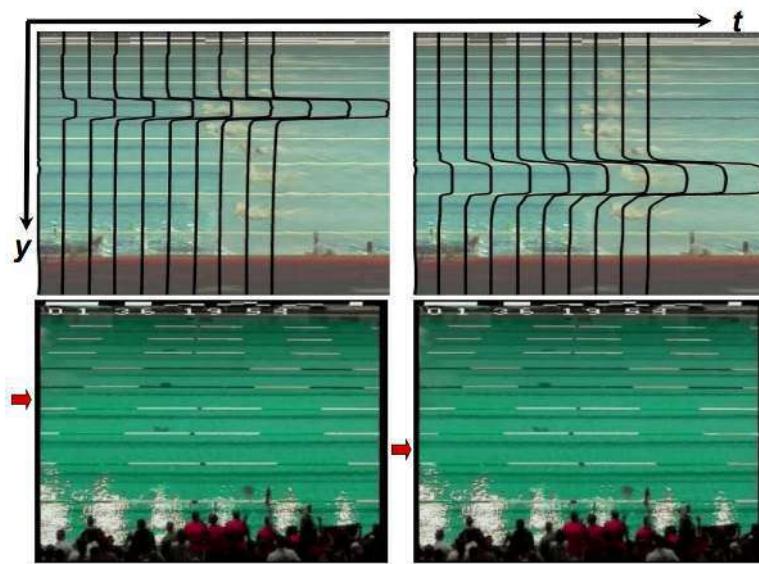
כך לדוגמה אם ניקח סרטון של מצלמה סטטית בה אצטדיון מתפוץ בביטחון אחת, יוכל לגרום לאמצע האדפטיבי להתפוץ ראשון באמצעות Dynamic-Mosaicing. אם ניקח פרייםים כמתואר הצד שמאל של האיר, במרכז יהיו פיקסלים מזמן מוקדם יותר ולכן שם יתרחש הפיצוץ קודם. אם נרצה שהמרכז יתפוץ אחרון, ניקח פרייםים כמו הצד ימין של האיר.



איור 106: ניתן לשנות את זמני הפיצוץ באמצעות בחירה מתאימה של זמינים לכל פריים.

וזכרים להזכיר כמה שקלים בקהל?

כלמו תחרות שחיה ותשימוש ב-Dynamic Mosaicing כדי לעরוך את הסרטון כך ששחיה לבחירתכם יהיה זה שמנצחה. עתה, אתם יכולים למכור את הסרטונים האלה לכל האימהות בקהל, בהתאם ליד המנצה הסרטון הערוך, וכך תרוויחו מיליארדים!



איור 107: חצי ימין של התמונה: עברו המסלול של השחיין עליו מכביע החץ האדום ניקח פרויימי מהעתיד, ואת כל שאר הפרויימים (של השחיינים האחרים) נשאיר מהעבר. כך נdag שאותו השחיין ינצח בפער על האחרים.
חצי שמאל של התמונה: אותו דבר עברו השחיין אחר.

אם נצלם סרטונו של מפל, כאשר הצלם מזיא את המצלמה מימין לשמאלי, נוכל לגרום לסרטונו לווז משמאלי לימיין באמצעות ניגון הסרטונו הפוך. הבעיה עם זה היא שעכשו המים במפל יזרמו למעלה, ולא יפלו למיטה.
ניתן להשתמש ב-Dynamic-Mosaicing ועם זוויתות שונות של *slice*-ים כך שנוכל לשנות את כיוון התנועה וגם לשמר על המים שייזרמו למיטה.

חלק XI

למידת מכונה

נמצא כיום בכל מקום: תרגום שפות, חיפוש לפי תמונות, זיהוי פנים, נהיגה אוטונומית, Deep-Learning Voice-Assistants וכו' וכו'.

זה אחד התחומים החשובים ביותר כיום והוא פוגש אותנו גם בעיבוד תמונה. אולם תחילת עליינו להסביר כיצד עובדות רשתות נוירוניים בסיסיות.

רשתות נוירוניים עוסקות באמצעות למידה מודגמות - Set Data של מידע קודם. עד רשתות נוירוניים חשבנו ש策יך לפתח לכל דבר אלגוריתם, אבל יש בעיות מאוד מסובכות שבני אדם לא יכולים להבין אותן בעצם ובטעות שלא לבנות להם אלגוריתמים טובים שהם Hand-Crafted.

על כן, רשתות נוירוניים באות להצלחה! חשוב לציין עם זאת, שאי אפשר להסתדר עם כל בעיה רק באמצעות רשתות נוירוניים. מלבד זה שלא תמיד יש מספיק דוגמאות ללמידה וכל מיני בעיות דומות, הרבה פעמים אם אנחנו עוסקים בסאונד מכניים, לרשף ספקטוגרמה, בזידאו מכניים *optical flow*. כמובן, אנחנו צריכים עדין לדעת תיאורית, ואיך לחשב *optical flow* ומשם הרשות מבינה יותר טוב.

14 רשתות נוירוניים

דוגמה לבעיה אם אנחנו מעוניינים לבנות פונק' F שה賓תנו תמונה של אדם קלט, היא מספרת לנו כפלט האם האדם הוא זכר או נקבה, המטריה שלנו תהיה ללמד את F .

אנו לא יודעים מהי F (אין אדם בעולם שידוע לבנות ביד פונק' מתמטית ש יודעת לבצע את המשימה הזו), אך יש לנו מגוון דוגמאות למידה (*training samples*). הרשות תוכל להשתמש באותן דוגמאות כדי להבין מה F אמרה להיות, ואז נוכל לבדוק את הרשות על סט של דוגמאות מבחן (*test set*).

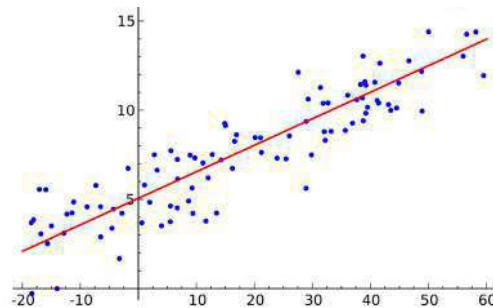
כמויות הדוגמאות ש策יך משתנה בהתאם למשימה. ככל שיש יותר דוגמאות יותר טוב (בדרכ' נctrack אפיילו מיליון דוגמאות).

סיכום אימון - בהינתן סט אימון של דוגמאות מותיוגות $\{(x_1, y_1), \dots, (x_n, y_n)\}$ (התיוג כאן הוא y_i), נרצה להעריך מהי פונק' המטריה שלנו F , באמצעות מזעור שגיאת החיזוי שלנו על סט האימון.

בחינה - נפעיל את F שמצאנו על סט דוגמאות מבחן שימושיים לא ראיינו, ונראה מהי השגיאה.

דוגמה בהינתן מגוון דוגמאות (x_i, y_i) נרצה למצוא זוג פרמטרים (a_0, a_1) שנותנים את הקירוב הlienar הטוב ביותר לדוגמאות הללו. $y(x) = a_0 + a_1 x$

$$\text{אנו נרצה למזער לדוגמה את פונק' העלות } E = \sum_i (y_i - (a_0 + a_1 x_i))^2$$

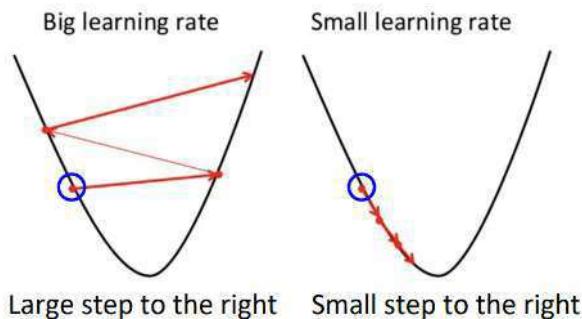


איור 108: נרצה למצוא קו ישר $y = a_0 + a_1x_i$ שמתאר היטב את הדגימות שלנו כדי לסייע פונק' עלות E . לבעה מסוג זו קוראים Linear-Regression.

הערה אף אחד לא הבטיח לנו שהדגימות שלנו מתאימות לפונק' ליניארית, במקרה באյור נדמה שכן זה מותאים.

שאלה כיצד מעזרת פונק' המחר?

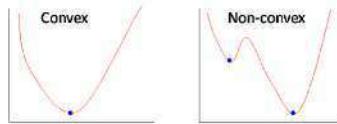
תשובה נרצה שפונק' המחר שלנו תהיה גזירה. במקרה הקודם הפונק' הייתה Mean-Squared-Error (MSE) והיא אכן גזירה. עתה, נרצה להתקדם נגד כיוון הגרדיינט שלו שכן הגרדיינט מצביע בכיוון העלייה/גדילה הגדולה ביותר, ולכן מינוס הגרדיינט יתן לנו את הירידה הגדולה ביותר. כך נתקדם אט-אט עד שנגיע לנקודת מינימום. נשים לב שגם הצעדים בהם אנו מתקדמים גדולים מדי, יתכן שלא נגיע למינימום כי כל הזמן נדלג עליו (גם אם הצעדים קטנים מדי ייקח לנו יותר מדי זמן להגעה למינימום).



איור 109: נתקדם בכיוון מינוס הגרדיינט כדי להגיע לנקודת מינימום. לדבר זה קוראים Gradient-Descent. שימו לב כיצד משמאלי הצעד גדול מדי ואנחנו כל פעם עוברים את המינימום. מימין הצעד קטן לנו לבסוף של דבר נגיע למינימום (אך אם הצעד קטן מדי זה יקח המון זמן) הפרמטר שקבע באיזה קצב להתקדם נקרא Rate Learning

שאלה מה קורה אם הפונק' לא קמורה?

תשובה במקרה זה, המינימום אליו נגיע לא בהכרח יהיה המינימום הגלובלי.



איור 110: פונק' לא קמורה לעומת פונק' קמורה

רוב הדוגמאות שלנו (~ 80%) יהוו את סט האימון, וחלק קטן יותר נלקח לסט המבחן (~ 20%). חשוב לראות ירידה הן בסט האימון והן בסט המבחן!

השגיאה בסט האימון תמיד צריכה לדנדת, כי אנו מבצעים מינימיזציה על זה. השגיאה בסט המבחן תרד אם הפונק' שלמדנו כלליה (*generalize*) כמו שצרכיך מעבר לדוגמאות האימון עליהם היא למדה.

הערה הסיבה שאנו צריכים סט מבחנים היא כדי לוודא שיש הכללה טובה של הפונק' שלמדנו, וכך נדע אם אותה פונק' היא באמת הכללה או שהיא פשוט שינוי את כל הדוגמאות.

לדוגמא, אם אנו מנסים לאחות האם אובייקט כלשהו הוא אקס, ואני עוברים על המון תכונות של אגים בלי כדי לקבל תוצאה טובה המחשב יכול "לשנ" את תכונות האגים שהוא כבר ראה ולהגיד שאלה כן אקס, ועל כל תמונה אחרת להגיד שהיא לא אקס. זו הכללה רעה.

שאלה אמרנו שרשתות נוירוניים לומדות פונק' F , אבל זה מאוד כללי. מה קורה בפועל?

תשובה בהשראת הנוירונים שיש לנו במוח, נוצר מודל מתמטי לנירון בו אנו משתמשים ברשתות נוירוניים, אותו ניתן לראות באירור. מודל זה מורכב ממהרכנים הבאים:

(i) קלט - ערכים ממשיים x_1, \dots, x_n .

(ii) משקלות - w_{1j}, \dots, w_{nj} .

(iii) משקלות bias - θ_j ; זו משקלות מיוחדת שלא כופלת אף קלט. המון פעמים ממדומים שיש קלט נוסף x_0 שערכו תמיד 1, ואז θ_j היא המשקלות שלו.

(iv) פונק' transfer - פונק' שמחשבת ממוצע משוקל לקלט: $\theta_j + \sum_{k=1}^n x_k w_{jk}$

(v) פונק' אקטיבציה - פונק' φ .

(vi) פלט - תוצאה ההפעלה של φ על התוצאה (הסכום) משלב (iv), כלומר $\varphi\left(\theta_j + \sum_{k=1}^n x_k w_{jk}\right)$.

מה שאנו לומדים זה את θ_j - אנו מנסים למצוא ערכים של המשקלות הללו כך שהשגיאה תהיה כמה שיותר.

הערה במקרה של Linear-Regression יש משקלות בודדות a_1 ו- a_0 שהוא $bias$.

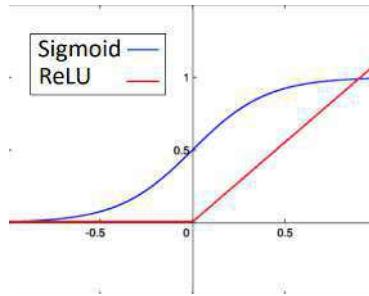
הערה פונק' האקטיבציה φ צריכה להיות לא לינארית. בכלל שפונק' $transfer$ היא לינארית, אם פונק' האקטיבציה הייתה לינארית בסה"כ היינו מקבלים שככל הרשות היא פונק' לינארית - ככלומר כפל במטריצה. אולם, לא כל הביעות בעולמו ניתנות לפתורן באמצעות כפל במטריצה.

בין פונק' האקטיבציה הנפוצות ביותר נמנוט:

$$\varphi(z) = \max(0, z) - \text{ReLU} .1$$

$$\varphi(z) = \frac{1}{1+e^{-z}} - \text{Sigmoid} .2$$

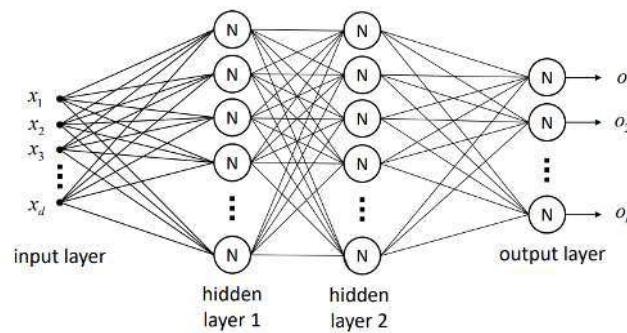
עכשו כשהפונק' לא לינארית, מסתבר שאנו יכולים ללמוד כמעט כל דבר.



.איור 111: פונק' Sigmoid ו ReLU

שאלה אז ראיינו כיצד נראה נוירון בודד. איך נראה רשת נוירונים?

תשובה רשת נוירונים מורכבת מכמה שכבות. השכבה הראשונה היא שכבת הקלט, והשכבה האחורה היא שכבת נוירונים של הפלט הסופי. השכבות באמצע יהיו שכבות שmorphובות מנוירונים גם הן, ולהן קוראים *hidden layers*. כל שכבה מורכבת מכמהות כלשהי של נוירונים (בשכבות שונות יכולות להיות כמותות שונות) ולכל שכבה, הקלט של השכבה הנוכחית הוא הפלט של השכבה הקודמת. כאמור, לכל נוירון יש משקלות (כולל משקלות *bias*) משלו, ואנו נרצה למצוא את המשקלות שימצעו לנו פונק' עלות.



.איור 112: סכמה כללית של רשת נוירונים.

הערה שכבה i ברשת נקראית Fully-Connected אם כל נוירון בשכבה i -ה מוחובר לכל נוירון ברמה $i-1$.

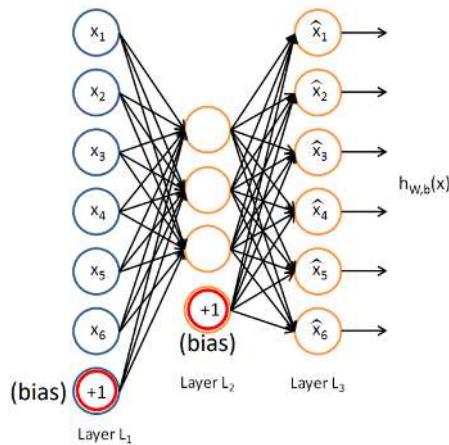
הערה יש כמה סוגים של בעיות שרשתות נוירונים פותרות, ואנו מסוגים אותם כך: בעיית classification (i) - הפלט דיסקרטי - יש *classes* שונים. נרצה לדוגמה להיות אילו אוטיות יש בתמונה,

נקבל כקלט את הפיקסלים של התמונה ונוציא כפלט את האותיות שזיהינו: פלט בינה-ארית של "זיהינו או לא" או מספר בין 0 ל-1 שמצוין כמה אנו בטוחים שיש שם אות (כמו מעין הסתברות).

בעייה (ii) - הפלט רציף. נרצה לדוגמה לקבל כקלט פיקסלים של תמונה ולהוציא כפלט פיקסלים של תמונה מוחודדת של התמונה המקורית.

דוגמה יש רשתות בשם AutoEncoders בהן ה-*input* וה-*output* הוא אותו דבר, אז באמצע יש שכבה עם הרבה פchnות נוירונים מאשר מספר הקלטים. לדוגמה, קלט שהוא תמונה 100×100 והפלט הוא גם תמונה 100×100 , רק שבאמצע יש שכבה עם רק 200 נוירונים.

דבר זה מאפשר קידוד של תמונה (דחיסה) זה יכול גם להסיר רעש.



איור 113: דוגמה ל-AutoEncoder. שימוש לב גם כיצד כאן אנו מייצגים את ה-*bias* באמצעות קלט נוסף $x_0 = 1$, שלו יש משקלות משלו (זה היא ה-*bias*).

פונק' האקטיבציה Softmax

הרצאה 9

נניח אנו מבצעים *classification* כדי להציגו דוגמה מי האובייקט שמצוג בתמונה, בין K מחלקות שונות. השכבה האחורה ברשות שלנו תכיל K נוירונים, כל נוירון מתאים למחלקה אחרת, ונקבל את האקטיבציות שלהן z_1, \dots, z_K . כדי לקבל נוירון יחיד שדולק, המציג את המחלקה הנכונה, נוכל ללקח בסוף הרשות \max על כל הערכים z_K, \dots, z_1 והערך המקסימלי יהיה המחלקה שהרשות בחרה.

הערה כך נוכל להשיג וקטור $v = \begin{pmatrix} c_1 \\ \vdots \\ c_k \end{pmatrix}$ בו i אם $c_i = 1$ ואם $c_i = 0$ אחרת. יציג זה נקרא *one-hot*.

בעיה ללקח מחלקה על \max על z_1, \dots, z_K היא ש- \max לא גירה ובשביל *backpropagation* נזדקק לפונק' גירה.

פתרון משתמש בפונק' *Softmax* המנורמלת את z_1, \dots, z_K ל"הסתברויות" y_1, \dots, y_k :

$$y_i = \frac{e^{z_i}}{\sum_{j \in [K]} e^{z_j}}$$

כך שמתקיים $0 \leq y_i \leq 1$ ו $\sum_{i \in [K]} y_i = 1$.

פונקציות הפסד (Loss Functions)

לשם מדידת הסטטיה שלנו מהתוצאה הנכונה והרצוייה, אנו משתמשים ב-Loss Functions. בעיות שונות משתמשות בפונק' שונות.

רגרסיה נניח כי $y_i = \{y_i\}_{i=1}^N$ הן תוצאות הנכונות ל- N קלטים, ו- $\{z_i\}_{i=1}^N$ הן הפלטים של רשת הנוירונים שלנו בכל עיטה. אחת הפונק' הפופולריות בהן משתמשים לרגרסיה נקראת MSE – Mean Square Error והוא מוגדרת כדלקמן:

$$MSE = \frac{1}{N} \sum_{i=1}^N (z_i - y_i)^2$$

קלסיפיקציה נניח שהפעלו *Softmax* וקיבלו הסתברויות מהצורה $y_i = \frac{e^{z_i}}{\sum_{j \in [K]} e^{z_j}}$ כאשר $i \in [K]$. אם אנחנו יודעים שהמחלקה הנכונה היא r , אז השגיאה תהיה

$$L = -\log z_r$$

במקרה בו אנו לא בטוחים מי המחלקה הנכונה, אך אנו יודעים מה ההסתברויות הנכונות, שנסמנו p_i , לכל מחלקה (לדי *Cross – Entropy* 80% שזה 4-20% שזה 1) נוכל להשתמש בשגיאה בשם

$$L = -\sum p_i \log z_i$$

כמו כן, קיימת פונק' *Loss* נוספת לגמרא:

Perceptual-Loss אם אנו רוצחים לשווות שתי תמונות כדי לראות אם הן דומות או לא, להשתמש ב-*MSE* זה לא פתרון טוב כי אז אפילו שנייה תאורה או זהה או פיקסל יחיד בין שתי התמונות תגרום לשגיאה מאוד גדולה (על אף שהוא רואים בעין שהתמונות מאד דומות).

על כן, רעיון נוכל להכניס את שתי התמונות לרשת שאימנו קודם ונקבל וקטור אקטיביזיות לכל תמונה.

לוקטורים הללו נוכל לעשות MSE .
 נשים לב שה- $Loss$ זהה מאוד תלוי ברשת שאימנו קודם, ונרצה שרשת צזו תוצאה וקטורים דומים לתמונות שנראות דומות, וקטורים שונים לתמונות שנראות שונות.
 דוגמה אם יש לנו רשת שאימנו לאוזות פרצופים, נוכל לנקח Tamuna של אדם בוגרTamuna של אותו אדם בתור תינוק. נרצה ששתי התמונות הללו יפיקו וקטור דומה ברשת שכבר אימנו בגללהadam בתמונה זו אותו אדם (על אף שתתי התמונות שונות למדי מבחינת הערך של כל פיקסל).

אימון רשת

אנו מאמנים רשת באמצעות שינוי המשקלות ברשת, $\{w_{ij}^{(l)}\}$, לפי דוגמאות שכבר יש לנו: כל דוגמה היא זוג (x, y) בה x הוא הקלט לרשת ו- y הוא הפלט הנכון.
 עם משקלות אופטימליות, נרצה שכל x מה שהרשת תחזיר יהיה y .
 לפי המשקלות הנוכחיות, $L_S(w_{ij}^{(l)})$, נשווה את פלט הרשת, (x, o) , ל- y ונקבל שגיאה כלשהי: $L_S(w_{ij}^{(l)})$. את השגיאה זו נרצה למזער.

המוצע נעשא על פי האלגוריתם הבא, הקרויה Stochastic Gradient Descent (SGD).
 כי אנו מגרילים קבוצה דוגמאות)

Algorithm 18 Stochastic Gradient Descent

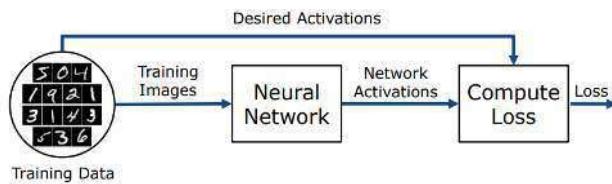
- 1 : **Initialize** weights and biases of network (e.g. random values)
 - 2 : **Randomly Select** training images in a mini-batch
 - 3 : **Compute** Loss for all images in the mini-batch
 - 4 : **Update** weights and biases by back-propagation using derivatives of loss function by each weight (Chain Rule)
 - 5 : **Repeat** from (2).
-

לקבותח הדוגמאות שהגרנו כל פעם אנו קוראים $mini - batch$. את שינוי המשקלות אנו עושים אחרי כל מעבר על $mini - batch$.

לדוגמא, אם בבעיית רגרסיה היו לי ב- $mini - batch$ $(x_1, y_1), \dots, (x_k, y_k)$, אני לא אחשב את השגיאה עבור (x_1, y_1) , אשנה את המשקלות ואז עבור (x_2, y_2) וכן הלאה.
 במקרה זה, אני אחשב את השגיאה MSE עבור $(x_1, y_1), \dots, (x_k, y_k)$ ביחיד, ורק עכשו לפי השגיאה זו אשנה את המשקלות (כלומר, כל הדוגמאות ב- $mini - batch$ חושבו לפי אותן משקלות והמשקלות ישתנו רק ב- $mini - batch$ הבא).

כאמור, ברגע שאנו מגעים לשלב 5 אנו ממשיכים לשלב 2 עם חדש שנבחר מקבותח הדוגמאות שעוד לא בחרנו.
 בשלב מסוים, יסתדרו לנו הדוגמאות ובמקרה זה נאמר Epoch אחד.

לאחר שסימנו *Epoch* נוכל להמשיך ולעשות *Epoch*-ים נוספים: לחזור על כל האלגוריתם שתיארנו להלן, רק שלא נתחל באופן רנדומלי את המשקלות, אלא **נעבוד עם המשקלות מה-*Epoch* הקודם**.



איור 114: תהליך אימון רשת נוירוניים

הערה לא נפרט עתה כיצד *Back – Propagation* פועל לעומק, אולם כאמור מדובר פשוט בכל השרשרת. מלבד זאת יש שימוש בשיטות רבות כדי לגרום לאלגוריתם להיותiesel בכל הנitan (לדוגמא, *Adam Optimizer*) אותן לא נוכל לתאר בפורים זה.

15 רשותות קונבולוציה

הרצאה 10

על סט נתונים שמורכב מדברים כמו תמונות או אוטות (צליל), בהן פועלות הקונבולוציה היא פעולה "משמעותית", גלו שאין צורך ברשות הkonvolוציה עם פילטרים (קרנלים) שונים היא מספקת. Fully-Connected. פועלות הקונבולוציה עם פילטרים (קרנלים) שונים היא מספקת. כך הרשות גם זקופה לפחות כח חישוב כי יש לה פחות משקלות:

דוגמה נניח כי יש לנו פילטר $10 \times 5 \times 5$, ותמונה $32 \times 32 \times 3$ כמה פרמטרים יש לנו ברשות? מהקונבולוציה יש לנו $75 = 5 \times 3 = 75$ פרמטרים. אבל אסור לנו לשכוח את $bias$ שהוא עוד פרמטר אחד וכך זה 76. יש לנו 10 כללה וכך הכל 760 פרמטרים. זה מעט מאוד בהשוואה לרשותות נוירונים עם מיליון משקלות.

הערה ברוב רשותות הקונבולוציה (*CNN*) משתמשים בשכבות konvolוציה של אחריהן, בסוף הרשות, שמים כמה שכבות בודדות .Fully-Connected שהן

תוצאת הקונבולוציה עם הפילטר היא לκייחת מכפלה פנימית בין הפילטר וכל אזור בתמונה בגודל $3 \times 5 \times 5$ (קרי, מכפלה פנימית $3 \cdot 5 \cdot 5 - 75 = 5$ -מימדית $+ bias$): $w^T x + b$. חשוב לשים לב שגム כאן, כמו ב-*NN* רגיל, יש קבוע *bias*. **כלומר**, הקונבולוציה נעשית על כל ערך בתמונה הקלט.

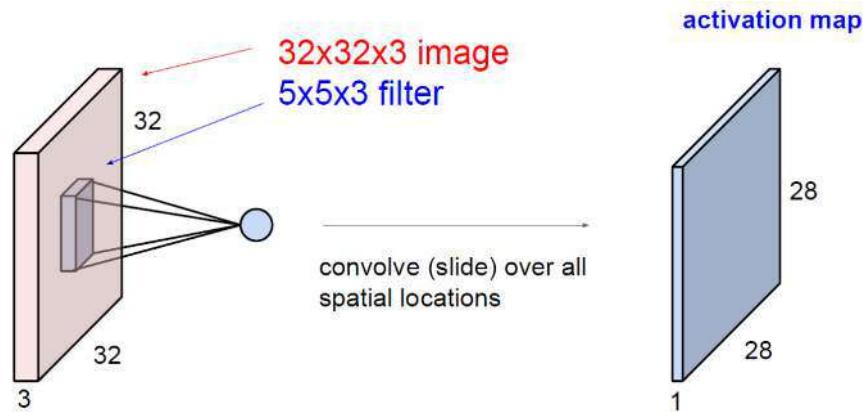
שאלה מה נעשה עם הקצוות?

תשובה רגיל, יש לנו כמה אפשרויות.

(i) אפשר שלא נמצא מהקצוות: כך יוצאה שלאחר הפעלת הקונבולוציה התמונה תקטן.

(ii) נהוג המון פעמים לרדוף את רק את קצה התמונה באפסים ואז להפעיל konvolוציה בלי לצאת מהגבולות החדשים: מתמונה בגודל $M \times N$ קיבל תמונה בגודל $(M+2) \times (N+2)$ כי יש שתי קצוות מימין ומשמאלי ושתי קצוות מלמעלה ומלמטה.

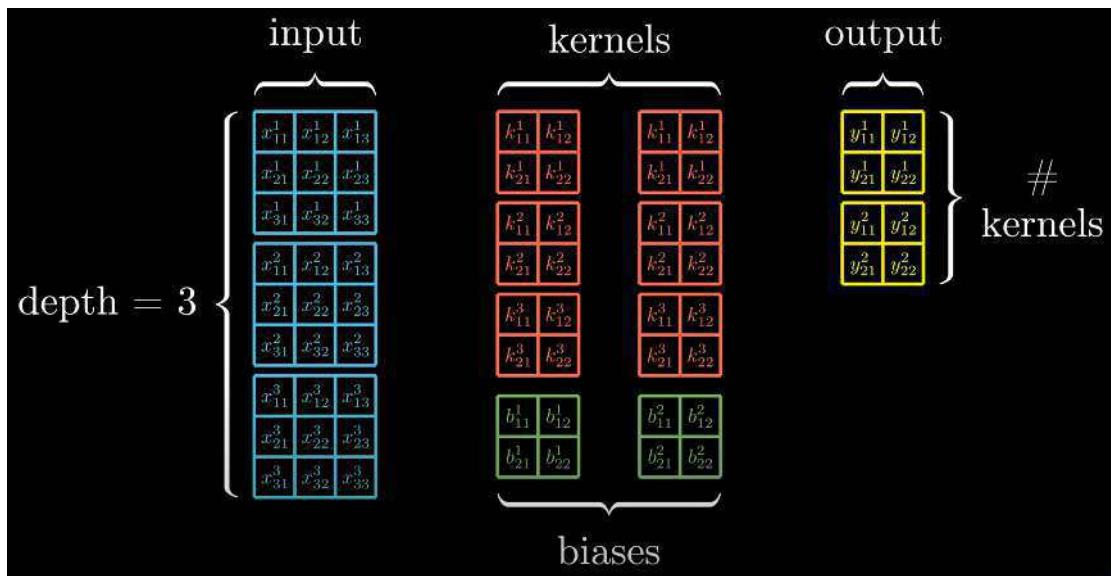
לאחר שנחשב את הקונבולוציה עברו כל ההוצאות האפשרות נקבל מפת אקטיבציה (activation map) שהיא הפלט של השכבה הנוכחית.



איור 115: מפעלים על התמונה קומבולוציה עם פילטר w כלשהו, במקרה זה בגודל $3 \times 5 \times 5$. נשים לב שם עומק התמונה הוא 3 (لتמונה יש 3 ערוצי צבע) אז עומק הפילטר יהיה 3 גם כן. כמובן, הקומבולוציה נעשית על כל ערוץ בתמונה הקלט. הבחןנו כיצד מפת האקטיבציה היא בעלת עומק 1, וכך גובה והרוחב שלה קטןו ביחס לתמונה המקורי.

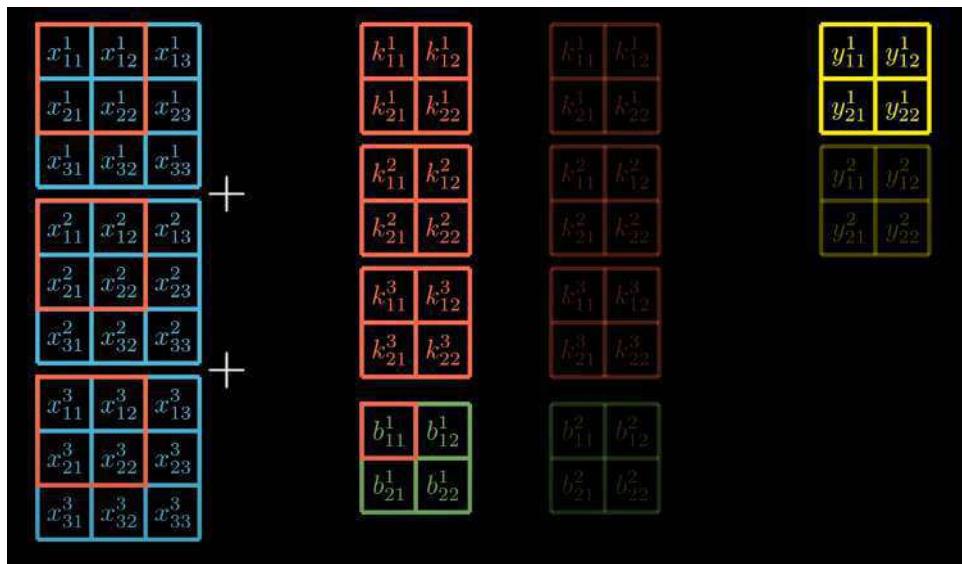
חשוב לשים לב שכל קרנל הוא עם אותו עומק כמו של התמונה, והעומק של הפלט הוא כמות הקרנלים שיש לנו בשכבה. חישוב הקומבולוציה נעשה ע"י סכימת הקומבולוציה של הקרナル בכל רמה בעומק של התמונה, והוספת ה-*bias* המתאים.

ויזואלית, נראה זאת כך:



איור 116: דוגמה לפרמטרים המגדירים שכבה ברשת קומבולוציה

הчисוב נעשה כדלקמן:

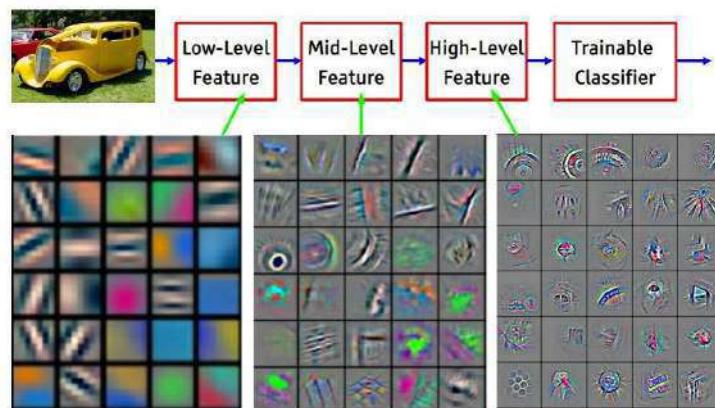


איור 111: חישוב פלט בקונולוציה עבור קרנל אחד נעשה ע"י הפעלת הקרNEL על כל מקום בתמונה, סכימת התוצאות בכל העומקים והוספת ה-*bias* המתאים בכל נקודה.
באյור ניתן לראות זאת עבור פעולת הקונולוציה הראשונית שמתאימה לפיקסל y_{11}^1 בפלט.
יש לחזור על הפעולה עבור כל קרNEL בשכבה כדי לקבל את הפלט המלא.

שאלה מה כל קרNEL ברשת יעשה?

תשובה לא יודעים, הרשות תאמן את המשקولات של הקרNEL כדי שיבצע את המשימה הכי טוב.
מה הקרNEL יכול לעשות? לגלוות פינות, לגלוות צורות, וכו' וכו'.

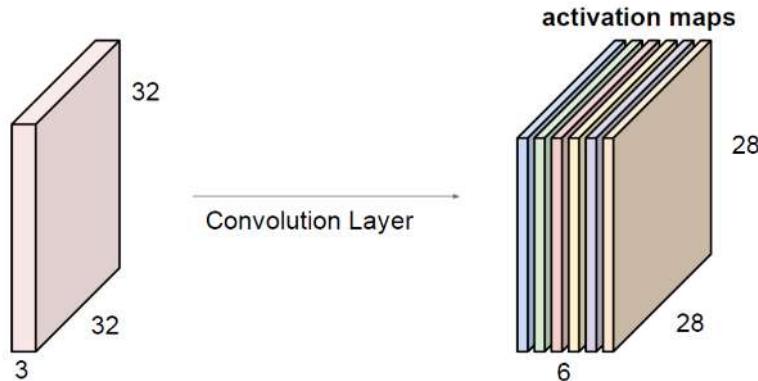
אם אנחנו מאמנים CNN ליזוי אובייקטים בתמונה (כמו מכונית) אז בדר'כ':
הפילטרים הראשוניים יארתו Low – Level Features - קוים אופקיים, אנכיים וכו'.
אחר כך פילטרים שמתארים Mid – Level Features - לדג' צורה של גלגל מכונית.
אחר כך פילטרים שמתארים High – Level Features - לדג' צורה של מכונית.



איור 118: פילטרים שונים מأتרים דברים שונים בתמונה - מתחילה מצורות פשוטות והולכים לדברים יותר ויותר מורכבים

סיכום טוב שנזדקק ליותר מקרナル אחד (לדוגמה, חוץ מ-*edge edge* אופקי אנכי רוחים לגלוות גם *edge edge* אנכי) שכן בכל פעם נפעיל יותר מקרナル אחד באותה שכבה, וכך נקבל כמה מפות אקטיבציה.

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

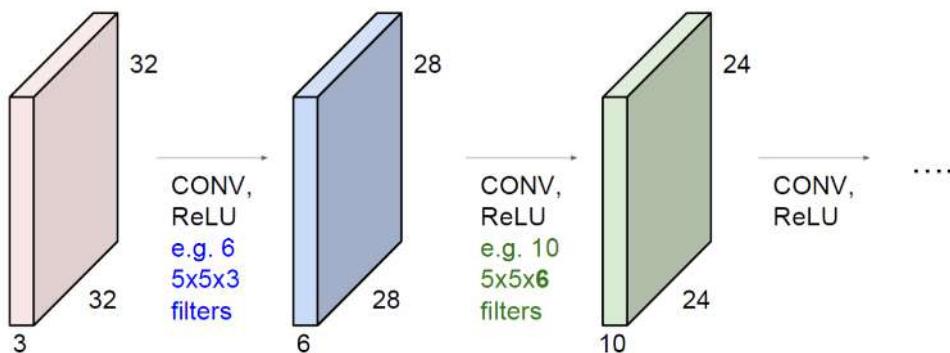


We stack these up to get a “new image” of size 28x28x6!

איור 119: שימוש בפילטרים מרובים תוביל ל渴בלת מספר מפות אקטיבציה, שיהיו תמונה חדשה. באյור יש תמונה חדשה בגודל $6 \times 28 \times 28$. 6 העורצים הללו הם כבר אינם ערויצים כמו בתמונה המקורית, הם העורצים של האקטיבציות השונות של כל קרナル בكونבולוציה.

הערה אחרי הקונבולוציה אנו משתמשים בפונק' אקטיבציה, בדר"כ *ReLU*.

ניתן להבין שלאחר שיצאנו מהתמונה המקורית, בה יש חשיבות לסדר של העורצים בתמונה: *B, R, G, B*, בכל שאר השכבות אין **חשיבות לסדר**. אנו יכולים להחליפ את העורץ הראשון בעורץ השלישי ולקבל רשות שעושה אותו דבר.



איור 120: *CNN* הוא רצף של קונבולוציות על פילטרים ואז שימוש בפונק' אקטיבציה (בדר"כ *ReLU*). קיימת גם שכבת Pooling שלא צוינה כאן ותתואר בהמשך.

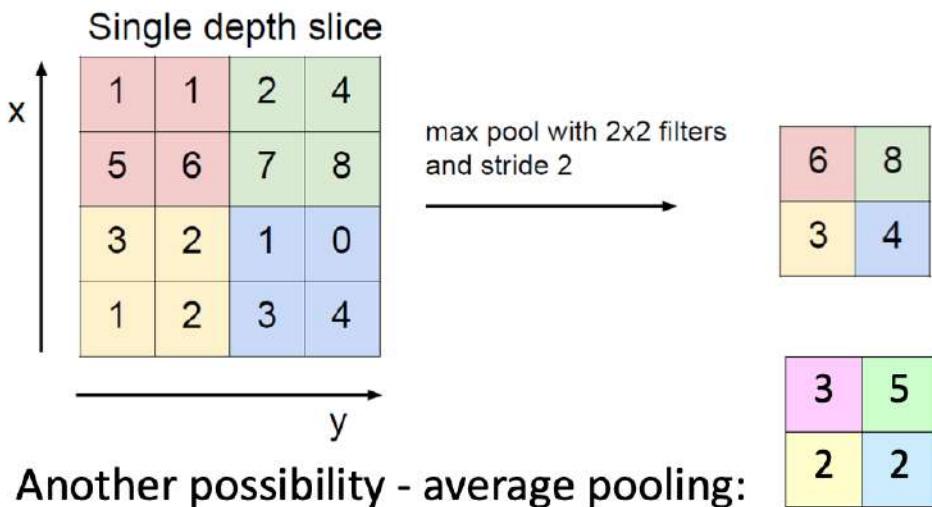
כאמור, הפעלת הקונבולוציה נעשית לרוב ללא ריפוד באפסים ולכן מתקבלים פלט קטן יותר. אנו קוראים לקפיצה בדges מ-*stride = 1*, *Stride* *F* אומר שדוגמינו כל פיקסל כל עד אנו גבולות התמונה, ואם $stride = 2$ מגדלים על כל פיקסל שני. כדי שצינו קודם, לעיתים נרצה כן לריפוד באפסים. במקרה זה, נניח כי N הוא מספר הפיקסלים בשורה, *stride* הקפיצה, *F* גודל הקונבולוציה אז גודל הפלט הוא $\frac{N-F}{stride} + 1$. אי עבור $N = 32$, $F = 3$ לא ריפוד נקבל שהפלט הוא בגודל 28, עם

ריפורד, למשל עם הוספת מסגרת בגודל 1, קיבל פלט בגודל 32×32 .

Pooling 15.1

ברשתות נוירוניים יש שכבה הנקראת שכבת Pooling. בשכבה זו מוחלטים את המידע על ידי הקטנת ה-data. אחת השיטות היא *maxPooling*.

בשיטה זו, אנו בוחרים גודל פילטר, בדרך כלל כל אורך 2×2 ומחליפים כל אורך 2×2 בפיקסל בעל ערך המקסימלי שלו. כמובן, נבצע זאת בנפרד לכל ערז.

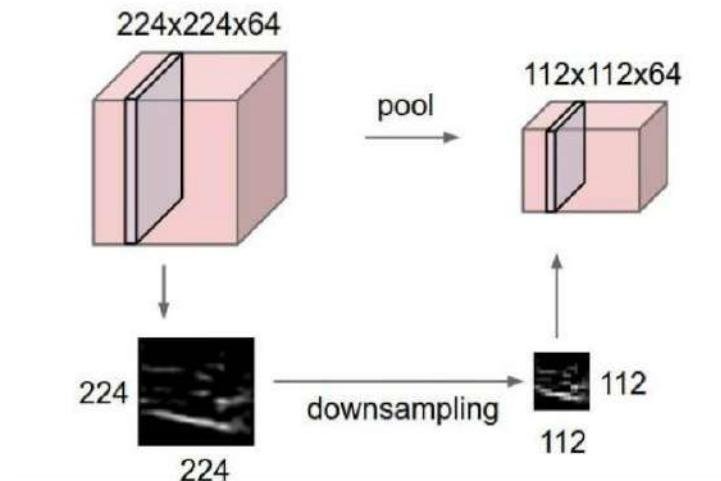


איור 121: המחשה ל-*MaxPooling*

מדוע זה טוב? כי זו פעולה לא לינארית.

אפשרות נוספת היא שימוש ב-*Average Pooling* -לקיחת ממוצע מכל אזור - זה פחות טוב, כי זו פעולה לינארית.

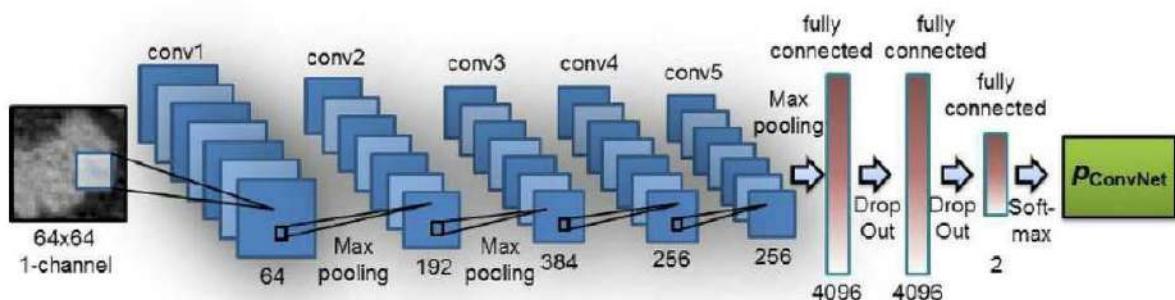
אך יש דרך נוספת להקטין תמונה על ידי שימוש ב-*stride = 2*, אבל ההבדל הוא ש-*MaxPooling* מוסיף את האילינאריות ודוגם למעשה מכל השכבה בצורה לא לינארית.



איור 122: המבנה ל-Pooling

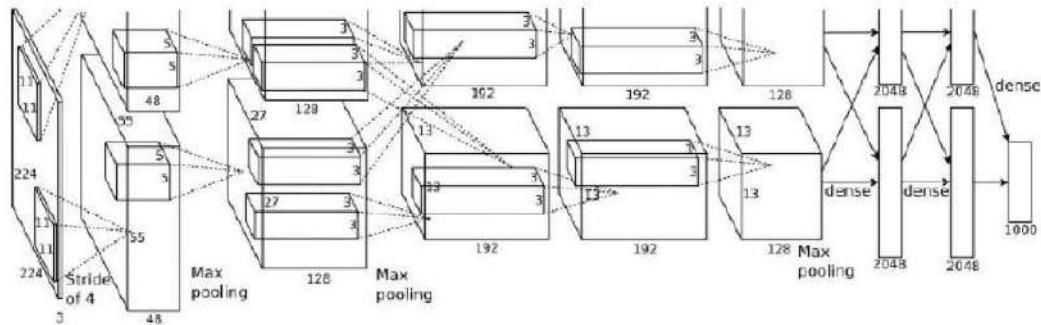
למעשה, השתמשנו בשיטה זו בעבר - בפירמידות. שם הקומבולוציות היו קבועות - במלים בוטות - מישחו חלים בלילה על הפרמטרים הנכונים. ברשותנו נוירונים לעומת זאת אנו מאמנים את הרשת על ידי מציאת הפרמטרים הטובים ביותר.

באיור הבא ניתן לראות דוגמא לשימוש ב-.maxPooling



איור 123: דוגמא לרשת המשמשת ב-Pooling

לאחר שראינו רכיבים בראשת, נראה דוגמא לכטיבת ארכיטקטורה של אחת. להלן גרסה מופשטת של ארכיטקטורת הרשת :AlexNet



Full (simplified) AlexNet architecture:

[227x227x3] INPUT

[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0

[27x27x96] MAX POOL1: 3x3 filters at stride 2

[27x27x96] NORM1: Normalization layer

[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2

[13x13x256] MAX POOL2: 3x3 filters at stride 2

[13x13x256] NORM2: Normalization layer

[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1

[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1

[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1

[6x6x256] MAX POOL3: 3x3 filters at stride 2

[4096] FC6: 4096 neurons

[4096] FC7: 4096 neurons

[1000] FC8: 1000 neurons (class scores)

איור 124: דוגמא לכתיבה ארכיטקטורתית של רשת

הערה. ברשתות קלסיפיקציה, על הפלט של השכבה הראשונה מפעלים SoftMax כדי לקבל את הווקטור ההסתברותי.

HyperParameters

יש פרמטרים רבים שצורך קבוע לפני שמתחילה לחפש משקלות.

- ארכיטקטורה

▫ עומק

▫ רוחב

▫ סוגי השכבות

▫ גודלי החלונות

▫ קפיצות

- אימון

▷ סוג האופטימיזציה

LearningRate ▷

▷ אתחול - הרשות יכול להתבצע בשתי דרכים
- שימוש במידע ראנדומלי. *Random* ▷

- *Transfer Learning* ▷
שימוש בנתוני רשות שכבר אומנה. מסתבר שבאמצעות מידע ראנדומלי מקבלים
לרוב תוצאות יותר טובות.

▷ פונקציית *Loss*

Epoch 15.2

אנחנו רוצים לעבור על ה-*Training Data* *Mini* – פעם אחת בלבד כדי לחסוך בזמן ריצה, אך אנו משתמשים באלגוריתם – *:Batch – SGD*

Algorithm 19 Mini – Batch – Stochastic – Gradient – Descent

1 : **Loop**

- 2 : **Sample** a batch of data
 - 3 : **Forward** prop it through the graph, get loss
 - 4 : **Backpropagation** to calculate the gradients
 - 5 : **Update** the parameters using the gradient
-

אחד הפרמטרים החשובים ביותר הינו ה-*learningRate* הקובע את "קצב ההתקנסות". בחירת ערך גדול מדי תגרום לנו לעקוף את המינימום, בחירת ערך קטן מדי תגרום להתקנסות איטית מדי.

יחד עם זאת, הפונקציות שאנו בודקים הם קמורות בסביבות מסוימות ולא גlobלית לכן אנו עלולים להתפרק במינימום לokaלי, כדי להמנע מזה, אנו מרים את האלגוריתם כמה פעמים עם משקלות ראנדומיות התחלתיות אחרות. אם קיבלנו אותו מינימום כמה פעמים בהסתברות מאוד גבוהה זה המינימום הגלובלי.

Overfitting 15.3

פעמים רבות כשנאמן את הרשות על *DataSet* נגלה שהיא לא עובדת טוב על מידע חדש, אלא רק על ה-*h*. מצב זה נקרא *overfitting*. יש כל מיני דרכים להימנע ממנו.

- למשל, לאחר שאימנו את הרשות במידה מסוימת, נזרוק כמו נוירונים בצורה ראנדומית ונאלץ אותה להיות כמו שיטת כללית.

batch אחד, ונוסף את כל האקטיבציות של הנירון מה-*batch* (*BatchNormalization*) •

הנ"ל. נורמל את הנירון על ידי החסרת הממוצע וחולקה בשונות. נבצע זאת לכל נורון בשכבה.

נרצה לפרט מעט יותר על *:BatchNorm*

נניח שאנו רוצים לשים *BatchNorm* אחרי השכבה ה-*i*. במהלך האימון, אנו מחשבים עבור כל *Batch* את התוחלת (ממוצע) והשונות של האקטיבציות של הנירונים בשכבה ה-*i*, בדר"כ לוקחים את הערכים לפני הפעלת פונק' האקטיבציה.³

כלומר, נחשב את התוחלת והשונות של $Z^{(i)}$

$$\mu = \frac{1}{n} \sum_i Z^{(i)} \quad \sigma^2 = \frac{1}{n} \sum_i (Z^{(i)} - \mu)^2$$

לאחר מכן נורמל את $Z^{(i)}$

$$Z_{norm}^{(i)} = \frac{Z^{(i)} - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

כאשר אנו משתמשים ב- ϵ , שהוא יושם להיות ערך מאד קטן, לציבות נומרית (למנוע חלוקה ב-0).

לבסוף, שכבת ה-*BatchNorm* מכילה גם שני **פרמטרים נבדדים** המסומנים β , γ . הפלט של שכבת ה-*BatchNorm* יהיה אפוא

$$\tilde{Z} = \gamma \cdot Z_{norm}^{(i)} + \beta$$

שים לב ש- γ שולטת על השונות של \tilde{Z} ו- β שולטת על התוחלת של \tilde{Z} .

שאלה בזמן אימון יש לנו *Batch*-ים והתהlik הגיוני. אך מה נעשה אחרי שסיימנו לאמן את הרשת שלנו (ואנו נגיד מעלים אותה לאינטרנט שתפעיל כמוצר)? אם היא מקבלת קלט ייחיד אין לנו נורמל לפיו.

תשובה לטיפול בעיה זו, אנו מחשבים שני ערכים: μ_{pop}, σ_{pop} :

(*test*) μ היא התוחלת של הקבוצה (*population*) אותה למדנו במהלך הבנייה של הרשת (גמ *train* וגם

(*ii*) σ היא סטיות התקן (שורש השונות) של הקבוצה (*population*) אותה למדנו במהלך הבנייה של הרשת

(גמ *test* וגם *train*).

הערכים האלה כאמור מחושבים ע"י כל ה- $\mu_{batch}, \sigma_{batch}$ שהיחסנו בזמן האימון וה-*test*. לסייעם, בערכים האלה

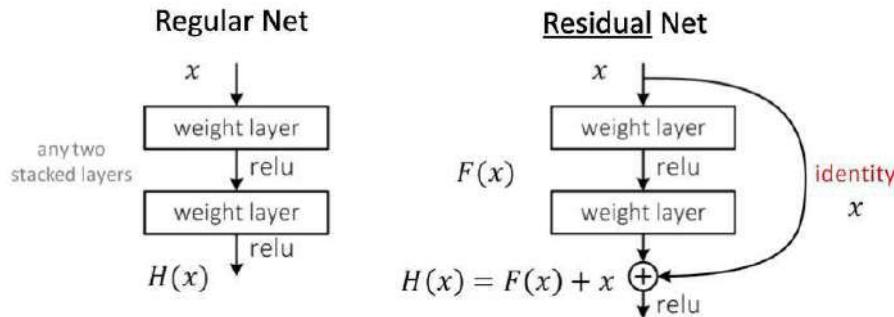
נשתמש בתווך ה- σ, μ של שכבת ה-*BatchNorm* והnochית.

³את הערכים לפני הפעלת פונק' האקטיבציה נהוג לסמן $a^{(i)}$ ואת הערכים אחרי נהוג לסמן $\varphi(z^{(i)})$ כאשר φ היא פונק' האקטיבציה.

ResNet – ReisudalNet

רשות זו היא רשות שבכל שכבה, מוסיפה את הפלט לתוכה הנוכחית ובסוף מוחזירה את התוצאה. כמובן היא מקבלת קלט x , ומוסיפה את x בכל פעם לתוצאה, וכך, מתקיים $H(x) = F(x) + x$ כאשר היא הפונקציה הנוכחית. לכן אנו לומדים למעשה את $H(x) - x = F(x)$

זו רשות מאוד פשוטה, אבל יוצרת מבנה מאוד ארוך - אין כאן .maxPooling, Dropout, Hidden Layers



איור 125: הממחה למבנה של ResNet

Anonymity In Social Media

האם ניתן לאוזוות בין אדם לפי סרטון שהוא צילם? נכון, אבל לפי הדרך שהוא צילם - תוצאות המצלמה וכו' .
למעשה, על ידי חישוב ה-opticalFlow לכל תמונה, באמצעות אימון רשות ניירונים יחסית קטנה אפשר לקבל תוצאות טובות.

Visual Speech Enhancement

דוגמאות נוספות לשימוש ברשות ניירונים היא ניקוי רעשים מצלול. למשל, אם שני אנשים במקביל, ניתן לקבל שמע נקי של ישות אחת מדברת.⁴

GAN – Generative Adversarial Network

מציג את הרעיון בקצרה - נרצה לאמן רשות שתמצא קלטים שגורמים לרשות אחרת להחזיר תוצאות שגויות, וכך נוכל להמשיך לאמן אותה.

AvivGabbay, AsafShamir, ShmuelPeleg, "visual speech enhancement using noise – invariant training"⁴

גאומטריה

מטרתנו היא חישוב סטריאו - חישוב עומק של אובייקט באמצעות תМОונות דו מימדיות, נעשה זאת על ידי מציאות נקודות התאמה בין תמונה שצולמה על ידי מצלמה אחת לבין תמונה שצולמה על ידי מצלמה ממיקום אחר. (3D Stereo with Reconstruction)

Perspective Projection **15.4**

בפרק הראשון, למדנו על הקשר בין קוארדינטה (X, Y, Z) בעולם, ל- (x, y) במצולמה:

$$\begin{aligned} x &= \frac{f}{Z} X \\ y &= \frac{f}{Z} Y \end{aligned}$$

כאשר f הוא אורך המוקד.

את קשר זה נוכל לרשום בצורה הומוגנית

$$\begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

כלומר אנו מקבלים שתי דרכים לרשום את הקשר

$$\begin{array}{l} \text{Regular } \begin{pmatrix} f\frac{X}{Z} \\ f\frac{Y}{Z} \end{pmatrix} \\ \text{Homogenous } \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} \end{array}$$

יחד עם זאת, במצבות, המטריצה איננה כה פשוטה, ולכן יש צורך בעוד הרבה פרמטרים, כלומר בפועל אנו מקבלים את הקשר

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

יש כאן 11 דרגות חופש. עובדה זו איננה טריומאלית, אך לא עמוקה להוכחה, אך לא עמוקה להוכחה. השווון כMOVן הוא שוויון פרויקטיבי - علينا לחלק לאחר הכפל במטריצה בקוארדינטה השלישי.

מטריצה זו נקראת **CameraMatrix**.

עליה השאלת, מה נעשה כדי לחשב את הפרמטרים הנ"ל? נוכל להתאים נקודות בין העולם לבין התמונה ולפתור את מערכת המשוואות שנקבל. ראשית, נזהה את מבנה המטריצה:

- למטריצה יש 11 דרגות חופש. (12 פרמטרים במטריצה אבל יש לנו גם scaling שכן אפשר לקבע פרמטר אחד במטריצה)

- 5 דרגות חופש - Intrinsic Parameters

▷ המרחק בין החירר למשור התמונה בציר ה- x וה- y בהתאם.

▷ מיקום מרכז המצלמה בציר ה- x וה- y בהתאם.

▷ אוריינטציה (skew) - האם הציר האופטי ניצב למשור התמונה או אלכסוני לו.

- 6 דרגות חופש של מיקום המצלמה למרחב (דברים שאנו יוכלים לעשות למלמה) Extrinsic Parameters

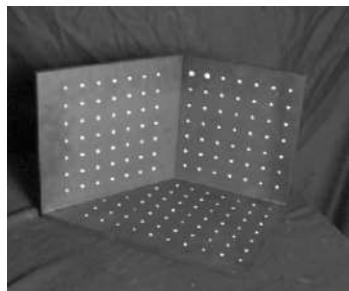
▷ 3 דרגות חופש לסיבוב.

▷ 3 דרגות חופש להזאה.

- כל התאמה בין שתי נקודות תתן לנו שתי מושוואות, לכן צריך 6 מושוואות כדי לחשב את כל הפרמטרים.

נזכיר שבמטריצה כל דרגות החופש שתיארנו לעיל מעורבות בינויו, כלומר לא נוכל לומר ש- m_{01} מותאים לסיבוב ו- m_{12} מותאים ל-skew: המטריצה נוצרת מאיישו שילוב של כל הפרמטרים הללו. המשמעות של דרגות החופש שתיארנו היא שאליה הדברים היחידים המשפיעים על המטריצה.

לتحליק זה, של מציאת הפרמטרים אנו קוראים Camera Calibration. בהינתן n נקודות עם נקודות ידועות בעולם ונקודות ידועות בתמונה, אנו מוצאים את הפרמטרים על ידי פתרת מערכת המשוואות.



איור 126: משתמשים בשטח "קליברטור" בו קל לנו להבין איך נקודת מתאימה לאיזו נקודת למרחב. היום קליברציות נעשות באמצעות חישובים סטטיסטיים: אנו ניתן למלמה לפעול ממש זמן מה כך שייעברו לידי אובייקטים שונים, ואנו ניתן יהיה להעריך את התנהוגות המצלמה: אם עברו לידי המון בני אדם, נוכל להעריך את הcoil כי אנחנו יודעים מה אמרור להיות גובה ממוצע של בן אדם.

נבחן כי מערכת המשוואות שרשמנו ניתנת כתיבה بصورة הבאה:

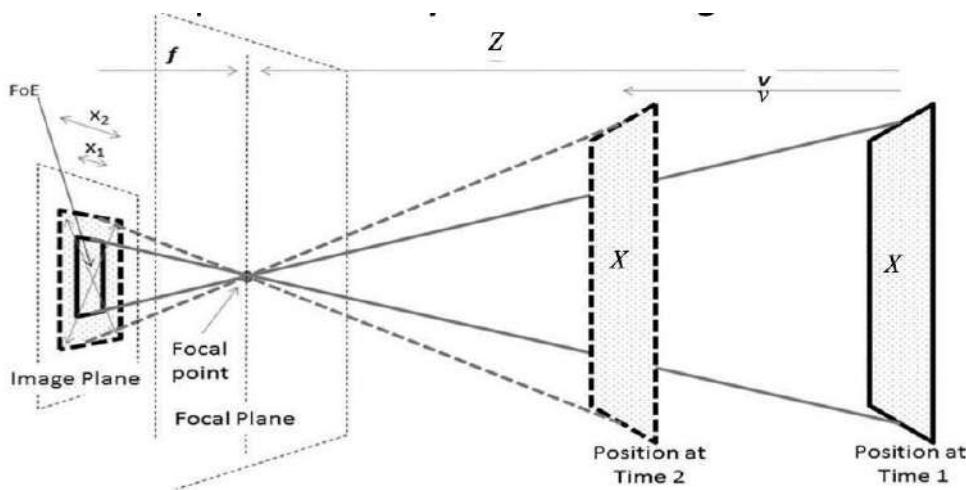
$$u_i = \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$

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

כאשר המיפוי הוא מסת נקודות (X_i, Y_i, Z_i) למקומות על התמונה (u_i, v_i) . המערכת על פניו נראה לא לינארית, אך על ידי כפל ב מכנה ניתן לקבל מערכת לינארית שנייה לפטור כדי לקבל את ערכי $h-m$ השונים.

TCC – Time To Collision 15.5

נרצה לדעת על ידי צילום תמונות את הזמן שיקח לנו להגעה לנקודה מסוימת.



איור 127: המasha לצילום שתי התמונות ותנועת האובייקט

נניח כי אנו מצלמים משאית ברוחב X וממרחק Z שמתקרבת לכיוון המצלמה במהירות v . נוכל לחשב את זמן ההתנגשות?

$$\text{מתקיים } \Delta t = \frac{Z}{v}. \text{ אבל המצלמה שלנו, לא יודעת את:}$$

- מהירות בזורה ישירה.
- גודל המשאית.
- מרחק המשאית.

- מטריצת המצלמה (למרות שלא צריך אותה לצורך החישוב).

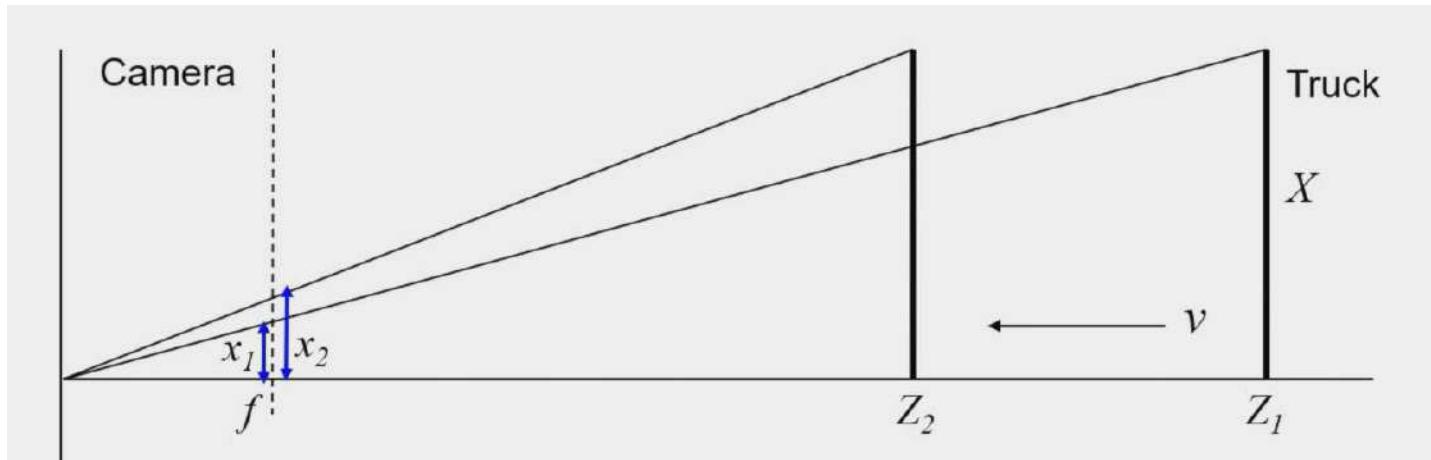
היא צריכה להסיק את זמן ההתנגשות איכשהו. אפשר לעשות זאת ע"י צילום שתי תמונות:

נניח שיחידת הזמן שלנו היא ייחידת הזמן שעוברת בין שני פריים ושבתמונה הראשונה המרחק של המשאית היה Z_1 ובשנייה הוא היה Z_2 . כמו כן, עובי המשאית בתמונה בפרי x_1 ובפרי x_2 הוא היה x_1 .

אנו יודעים כי מהירות המצלמה היא $Z_1 - Z_2$ וכאן זמן ההתנגשות הוא $\frac{Z_1}{Z_1 - Z_2}$. את הפרמטרים Z_1, Z_2 המצלמה לא ידעת, אבל היא כן יודעת את הרכיב במישור x_2, x_1 מתקיים כי

$$\text{TCC} = \frac{x_2}{x_2 - x_1} = \frac{f \frac{X}{Z_2}}{f \frac{X}{Z_2} - f \frac{X}{Z_1}} = \frac{\frac{1}{Z_2}}{\frac{1}{Z_2} - \frac{1}{Z_1}} = \frac{Z_1}{Z_1 - Z_2}$$

ולכן ניתן לחשב זאת בקלות. האם התוצאה הגיונית? אם $x_2 = 2x_1 \Rightarrow \text{TCC} = 2$ שכן המשאית התקדמה ביחס של 2, ואם $x_2 = x_1 \Rightarrow \text{TCC} = \infty$ שכן המשאית לא זזה.



איור 128: שני צילומי התמונות במרווח זמן Δt

15.6 גאומטריה אפיפולרית - Epipolar Geometry

המטרה שלנו תהיה לחשב את הקואורדינטות התלת-מימדיות של נקודות במרחב, באמצעות תמונות סטראיאו (קרי, שתי תמונות משתי מצלמות).

נניח כי יש לנו **שתי מצלמות** (או מצלמה אחת שזזה ושינה את מיקומה). נשאל את השאלות הבאות:

- בהינתן תמונה ונקודה x מהמצלמה הראשונה, כיצד זה מופיע על מיקום הנקודה Correpondence Geometry •

במצלמה השנייה x' ?

- בהינתן קבוצה של נקודות התאמה, בין שתי התמונות, מה מטריצות המצלמה של שתי המצלמות Camera Geomtry •

- בהינתן נקודות התאמה, מה המיקום של נקודה X בעולם Scene Geometry •

Epipolar Line

נסמן ב- C_L את החירר של המצלמה וב- X נקודה בעולם. אז מתאימה לה נקודה x במישור המצלמה שמתאים לה קו ישר מ- L ל- C_L .

נניח יש לנו מצלמה נוספת, נסמן ב- e_L את התמונה של החירר של המצלמה השנייה על המצלמה הראשונה. הישר המחבר בין e_L לבין x נקרא **קו האפיפולרי** והנקודה e_L היא **נקודות האפיפול** (שוב: הנקודה בה אני רואה על התמונה את חירר המצלמה השנייה).

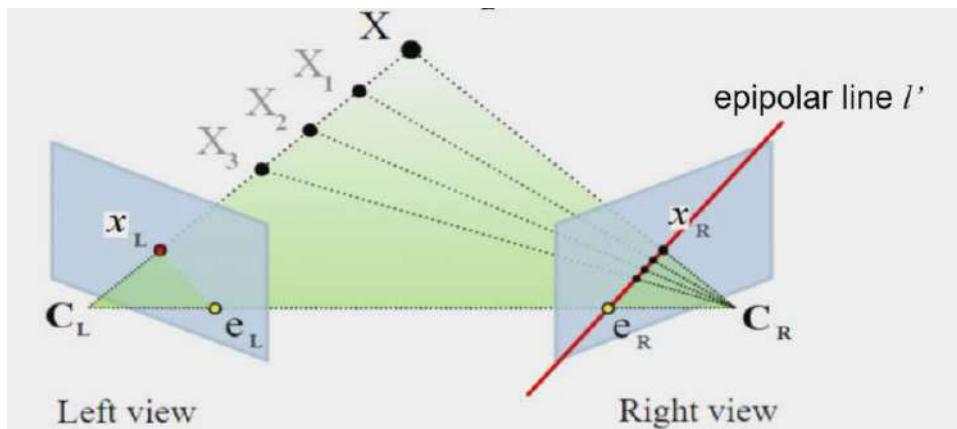
חזרה על הדברים חסיבות הקו האפיפולרי היא שם נקודה X מופיעה ב- x בצלמה השמאלית, הנקודה X חייבת להיות על אותו ישר עליו ישבות הנקודות x_L, e_L , כאשר e_L היא נקודת האפיפול.

ישר במציאות הוא גם ישר בתמונה, אך אם נסתכל על הישר זהה בצלמה הימנית, הנקודה x_R (שמתאימה ל- X בצלמה השמאלית) תהיה אפשרה על אותו ישר. כאמור, לשער הזה בתמונה קראו **קו האפיפולרי**.

מסקנה אנחנו יכולים למצוא את נקודת האפיפול באמצעות מציאת הנקודה בה כמה קווים אפיפולריים נחתכים.

הערה המון פעמים באחת/שתי המצלמות נקודת האפיפול נמצאת מחוץ לתמונה. למרות זאת, אנחנו עדין נהיה מסוגלים מתמטית להמשיך עם החישובים שלנו כאשר נקודת האפיפול תהיה נקודה דמיונית מחוץ לתמונה.

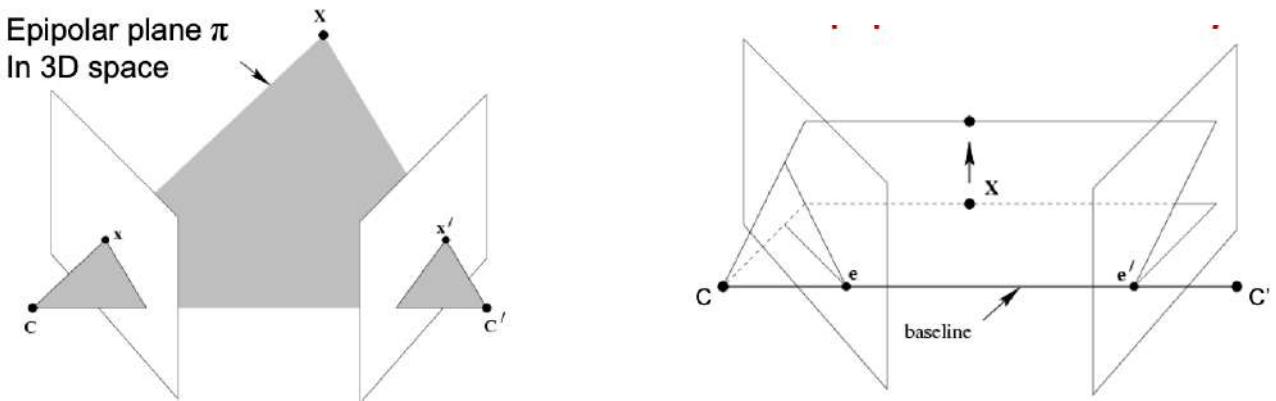
הנקודות X, e_L, e_L' יוצרות מישור, שלו אנו קוראים **מיישור האפיפולרי** והקו המחבר בין C_L, C_L' נקרא **קו הבסיס**. למעשה, כל מישור הכלל את ה-Base Line הוא מיישור אפיפולרי, הוא פשוט מתאים לנקודה X אחרת.



איור 129: הקווים האפיפולרים

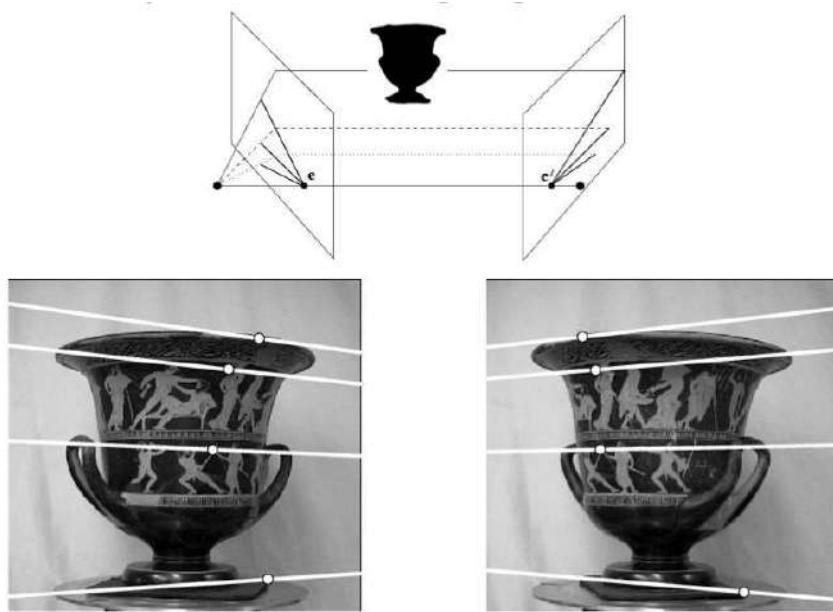
אחד התכונות של מיישור זה, הוא שם נקודה השטנה, והמצלמות נשארו במקומן המיישר יכול רק להשתנות עד כדי סיבוב, כי נשarra לו דרגת חופש אחת.

תכונה נוספת, היא שהחיתוך של המיישר האפיפולרי עם התמונות, הוא הקווים האפיפולרים.



איור 130: חיתוך המישור האפיפולרי עם מישורי התמונות הוא הקווים האפיפולרים, וקבוע המישור עם דרגת חופש אחת.

עתה, בהינתן שתי תמונות עם קווים אפיפולרים, כיצד נוכל לדעת מאייזה מצלמות צילמו אותן? נוכל להזות את נקודת החיתוך בין הקווים ולפי המיקום שלו לדעת לאיזה מצלמה היא שייכת. ניתן לקבל המשנה בדוגמה הבאה:

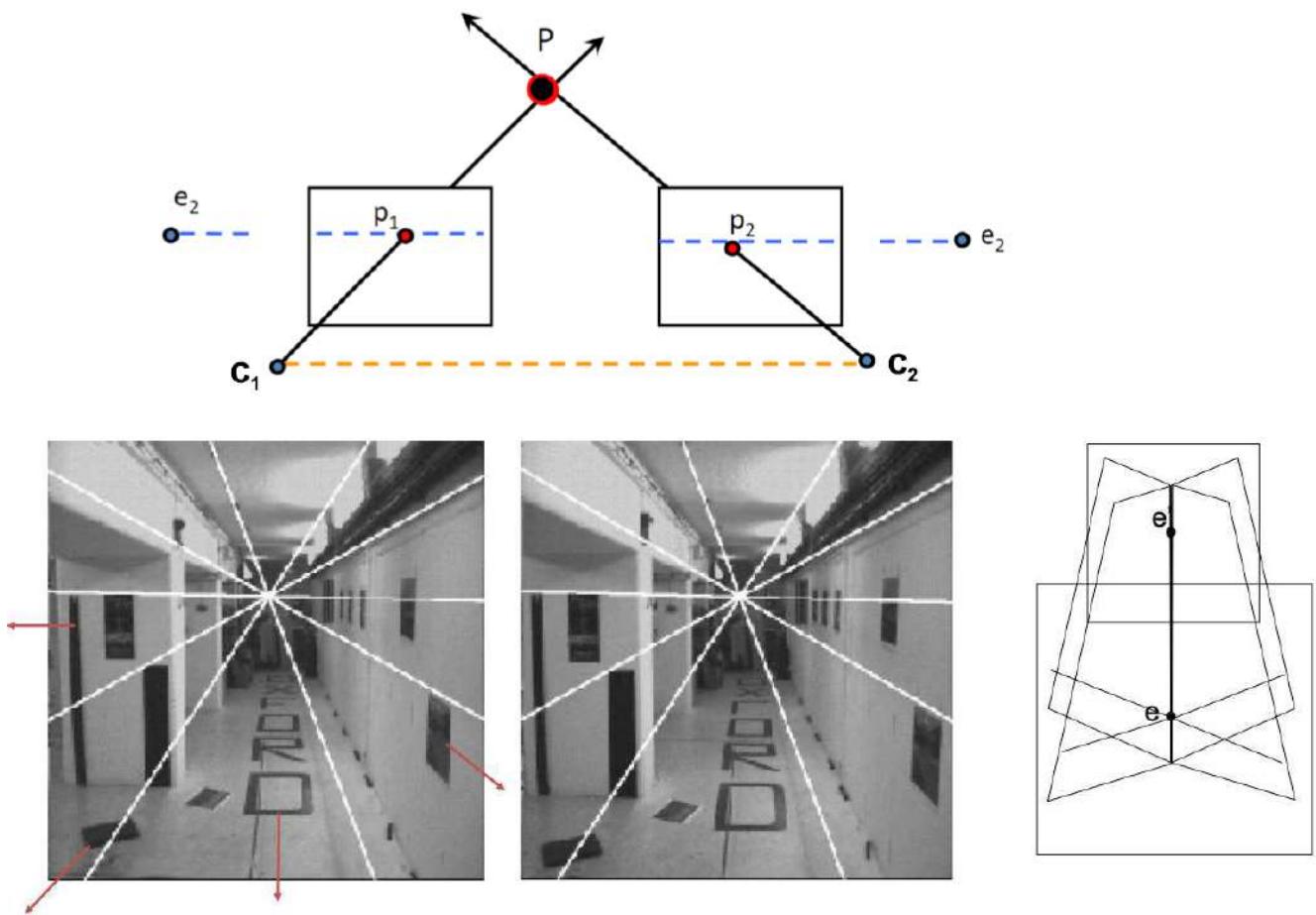


איור 131: זיהוי המצלמות שצלמו את התמונות לפי הקווים האפיפולרים

Parallel Motion

נבחן כי אם המצלמות צוות במקביל, Base Line, פוגש את מישורי התמונות באינסוף. כיצד במקרה זה נראים הקווים האפיפולרים? מכיוון שקו הבסיס פוגש את מישור התמונה באינסוף, **הקווים יהיו מקבילים אליו**, ולא יתכנסו. למעשה, כאשר הקווים הנ"ל מקבילים יש לנו תנופה מקבילה.

הערה כאשר אנו מתקדמיים קדימה, כל האובייקטים בתמונה צוים מלבד נקודת אחת - הנקודה אליה הולכים שאנו מכנים. הקווים האפיפולרים שנתקבלו יוצרו מהרבה מישורים מסובבים שיעברו דרך ה-Base Line. Focus Of Expansion



איור 132: תנוצה מקבילה של שתי התמונות ויצירת קווים אפיפולרים מחיוך מישורים המסובבים סביב ישר הבסיס

Fundamental Matrix - 15.7

עתה נדבר על הדבר החשוב ביותר - כיצד לחשב קווים אפיפולרים? אנו ידעים כי בהינתן נקודה x בתמונה הראשונה, נדע שהנקודה המתאימה לה' x' נמצאת על הקו האפיפולי בתמונה השנייה המתאים. קיימת מטריצה F המקיים $Fx'^T Fx = 0$ לנקודות על הקווים האפיפולרים בלבד.

דבר זה נותן לנו משווהת קו ישר בתמונה השנייה. כיצד נחשב את F ? נוכל למצוא נקודות توאמות (x, x') . המטריצה היא

$$F = \begin{pmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & 1 \end{pmatrix}$$

אבל עבור $x = (u, v, 1)^T, x' = (u', v', 1)^T$ אנו מקבלים את המשוואה

$$(u, v, 1) \begin{pmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{pmatrix} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = 0$$

$$\iff (uu', uv', u, vu', vv', v, u', v', 1) \begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{pmatrix} = 0$$

בה אנו מקבלים דרגת חופש אחת בלבד מכל נקודה. למעשה אנו יכולים לרשום את המערכת בצורה המלאה הבא:

$$\begin{pmatrix} u_1u'_1 & u_1v'_1 & u_1 & v_1u'_1 & v_1v'_1 & v_1 & u'_1 & v'_1 \\ u_2u'_2 & u_2v'_2 & u_2 & v_2u'_2 & v_2v'_2 & v_2 & u'_2 & v'_2 \\ u_3u'_3 & u_3v'_3 & u_3 & v_3u'_3 & v_3v'_3 & v_3 & u'_3 & v'_3 \\ u_4u'_4 & u_4v'_4 & u_4 & v_4u'_4 & v_4v'_4 & v_4 & u'_4 & v'_4 \\ u_5u'_5 & u_5v'_5 & u_5 & v_5u'_5 & v_5v'_5 & v_5 & u'_5 & v'_5 \\ u_6u'_6 & u_6v'_6 & u_6 & v_6u'_6 & v_6v'_6 & v_6 & u'_6 & v'_6 \\ u_7u'_7 & u_7v'_7 & u_7 & v_7u'_7 & v_7v'_7 & v_7 & u'_7 & v'_7 \\ u_8u'_8 & u_8v'_8 & u_8 & v_8u'_8 & v_8v'_8 & v_8 & u'_8 & v'_8 \end{pmatrix} \begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \end{pmatrix} = - \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

מדוע אנו זקוקים רק ל-8 דרגות חופש? זו טרנספורמציה הומוגנית ולכן $F_{33} = 1$.

דרך אחרת לפטור זאת היא על ידי שימוש בהרבה נקודות במטרה לעשות *minimization* לביוטי. כמובן, בדגימת הנקודות השתמש באלגוריתמים רובסטיים כמו ransac.

סיכום באמצעות מטריצה זו נוכל לחשב מדויק את הקו האפיפולרי בתמונה השנייה, שכן על ידי בחירת נקודה x נוכל למצוא

את הנקודה x' בתמונה השנייה שתקיים את המשוואה $x'^T F x = 0$

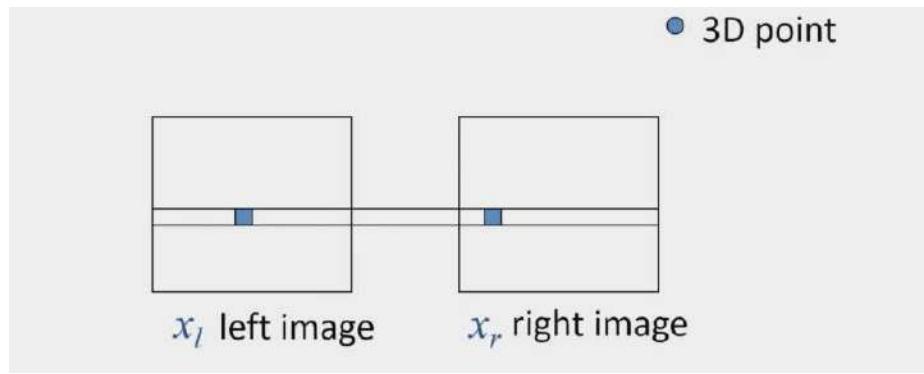
כל נקודה x' המקיים את המשוואה $x'^T F x = 0$ נמצאת על הקו האפיפולרי זה.

כשידועה לנו F קל לנו לחשב עומק באמצעות *triangulation* (גאומטריה פשוטה).

Depth From Stereo 15.8

הרצאה 11

לאחר שהישבנו את המטריצה הנ"ל נוכל לחשב נקודות תואמות על הקו האפיפולרי בלבד, ללא מעבר על כל נקודות התמונה. אם המצלמות זרות במקביל זה מאד פשוט. בנוסף, אם זה המצב, הנקודה בתמונה השנייה תהיה באותו מרחק אנסי ולכן נדרש לחשב אך וرك את ההפרש האופקי $\text{disparity} = d = x_l - x_r$.

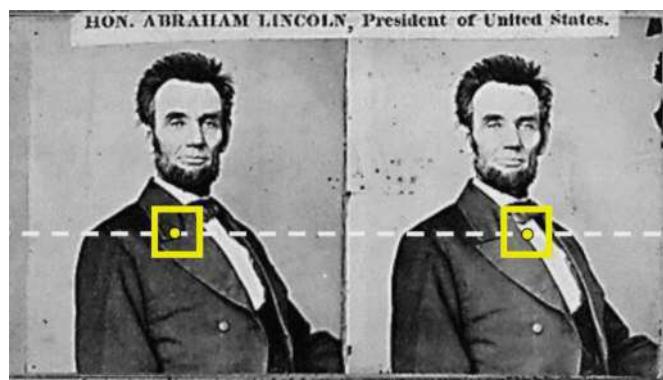


איור 133: שינוי המרחק האופקי בין הנקודות

נבחן כי **disparity קטן** מעיד על **נקודה רחוקה** ו**אילו disparity גדול** מעיד על **נקודה קרובה** - שכן הבדלי המיקום של המצלמות משפיעים הרבה על מיקום הנקודה.

כיצד נמצא את ה-*Disparity*?

נסתכל על זוג התמונות הבאות:

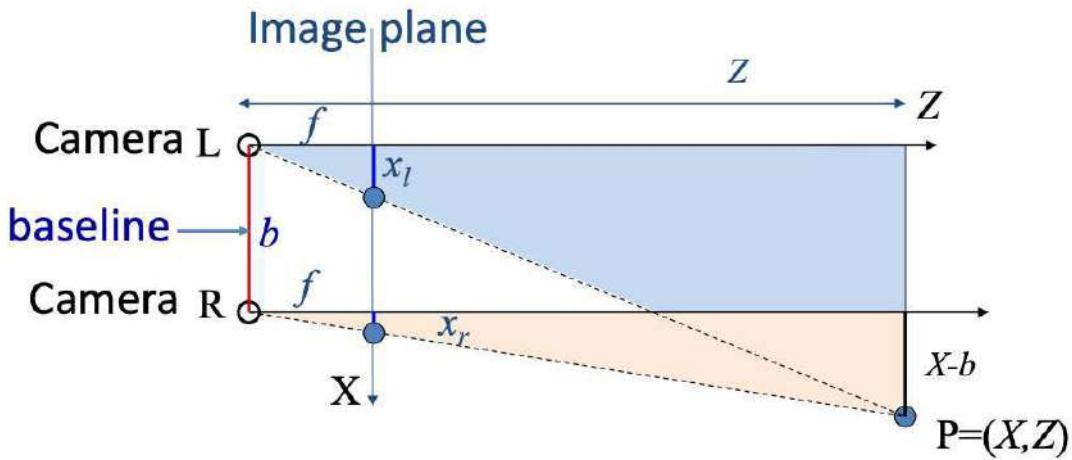


איור 134: שתי תמונות בהזאה אופקית. הקו המקווקו הוא קו אפיפולרי

נשים לב שהישר האפיפולרי המקווקו מקביל לציר x لكن תנועת המצלמה הייתה אופקית. נוכל לעבור על כל פיקסל בתמונה הימנית על הקו האפיפולרי ולמצוא את הפיקסל שהכי מתאים לו על הקו האפיפולרי בתמונה הימנית (באמצעות שיטות כמו optical flow או מזעור SSD וכו', נפרט על כך בהמשך).

חישוב העומק

באמצעות כל הנתונים שמצאנו, נוכל לחשב את העומק של הנקודה:



אייר 135: חישוב עומק הנקודה

יש לנו מושלשים דומים ולכון מתקיים הקשר

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

$\triangle LZP \sim \triangle Lx_l$

$$\frac{X-b}{Z} = \frac{x_r}{f}$$

$\triangle P(X-b)R \sim \triangle Rx_r$

מכאן נסיק כי

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

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

כאשר $x_r < x_l$. נסיק שוב כי d גדול מסמל נקודה קרובה ו- d קטן מסמל נקודה רחוקה.
הערה. אם המצלמות מאוד קרובות $0 \approx d$ ולכון אין לנו סטריאו. ככל ש- d יותר גדול יש רגשות גדולה יותר לסטריואו.
מסקנה. בהנתנו שתי תמונות מצלמות במרחב אופקי ביחס לנקודות העומק, נוכל לחשב את עומק הנקודה.

Block Matching ≈ Optical Flow 15.9

אפשר להתאים בין הנקודות כמו שחשבנו Optical Flow. למעשה, LK המקורי הומצא כדי לחשב סטריאו. על כן השאלה העיקרית היא מה גודל הסביבה שצר.

נותר רק לקבוע, כיצד נקבע את ההתאמות ההתחלתיות? נוכל להשתמש ב-SSD, קלומר

$$E(x, y; d) = \sum_{(x', y') \in N(x, y)} [I_L(x' + d, y') - I_R(x', y')]^2$$

כאשר d הוא ההזזה האופקית (disparity), ו- N זו סבiba של הנקודה y, x . כיצד נבחר את גודל הסביבה? סבiba קטנה מדי עלולה ליצור רעש, אך תשמור על הרבה פרטימס וסבiba גדולה מדי תהיה יקרה מדי לחישוב ותפספס כל מיני פרטימס.

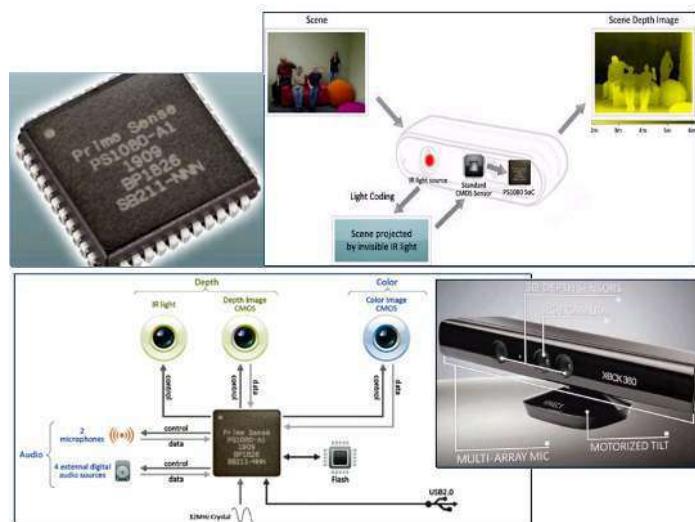
סיכום התהליך לחישוב עומק

- נצלם תמונה מצלמה מס' 1 ומצלמה מס' 2 כך שbezן מושתקות אחת מהשניה.
- נמצא את הנקודה שאנו רוצים לחשב את העומק שלה - בשתי התמונות. למשל, באמצעות המטריצה F וחיפוש על הקו האפיפולרי.
- קיבלנו נקודות (x_l, y_l) , (x_r, y_r) נסיק את העומק לפי הנוסחה $\frac{fb}{d}$ כאשר b קבועים שכן b הוא ה-*baseline* ו- f הוא מרחקי החיררים מミישור המצלמה, וכן $d = x_l - x_r$.

אלטרנטיבת לחישוב עומק Kinect 15.10

ה-*xbox* של Kinect השתמש בטכנולוגיה של חברת ישראלי בשם Prime Sense. במקום להשתמש בשתי מצלמות ולחשב את הנקודות המתאימות, הם השתמשו במרקון והקנו על העולם באינפרא אדום תבנית כלשהי.

הערה מייקרוסופט הפסיקו להעביר את טכנולוגיית Kinect שלחם לטכנולוגיה אחרת ש מבוססת לא על *stereo* ולא על LiDAR. מוכר לאנשים בשם הקרן ליזר, אלא על מדידת הזמן שלוקח ללייזר לחזור לחישון (time of flight).



איור 136: מבנה ה-*kinect*. ה-*kinect* מכיל מקרן אינפרא אדום ומצלמה אינפרא-אדומה שמשמשים יחד לחישוב העומק. מצלמת ה-*RGB* הרגילה נמצאת באמצע. שימושו לב לכו ה-*baseline* שמחבר בין המקרן למצלמת ה-*RGB* (בחלק הימני התיכון באיוור).

המקרה מקרים אוסף נקודות שתבנית שנראית קצר כמו רעש. אולם התבנית הזאת שימושית כי בכל סביבה, בכל בлок 9×9 בתמונה, תבנית הנקודות שמויפה היא ייחודית.

התבנית הזאת דומה ל-Blue Noise (לרעש לבן התמרת הפורייה תהיה איחודית, לרעש כחול התדרים הגבוהים יהיו יותר חזקים).

אנחנו לא צריפים לחפש נקודות בסקלות שונות (להשתמש בפרמידות וכו' כפי שלמדנו) כדי לוודא שמצאנו את התבנית הנכונה כי התבנית משתנה עם המרחק: ההגדלה של המרחק בין נקודות הליאזר שווה להקטנה של התמונה עם המרחק.

נבהיר זאת: הליאזר פוגע בעצם קרוב וגורם להופעה של נקודות למרחק קטן, כשהלייזר פוגע בעצם רחוק הוא גורם להופעה של נקודות למרחק גדול יותר.

עם זאת, כשאנחנו מצלמים עצם רחוק, הוא נראה קטן יותר.

הגדילה של הליאזר מתקזצת עם הקטנה של הצלום כך שאין צורך בחיפוש בסקלות שונות.

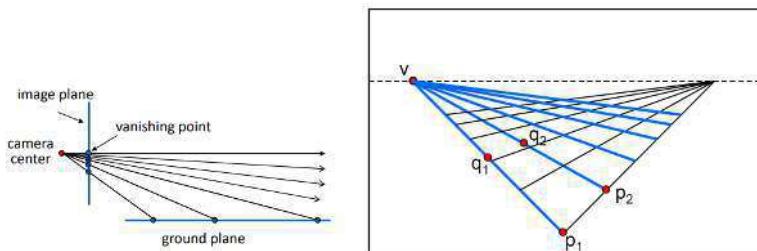
הבעיה היחידה עם Kinect הוא שהוא לא עובד בחוץ איפה שיש אור שמש חזק כי אז הליאזר לא בולט מספיק.

15.11 נקודות נעלמות - Vanishing Points

נניח שצלמה מצלמת את הרצפה מלמעלה. נשים לב שנקודות רחוקות מופיעות יותר ויותר גבוהה בתמונה, כפי שניתן לראות באירוע. הגובה המקסימלית יהיה עboro הנקודה באינסוף: האופק.

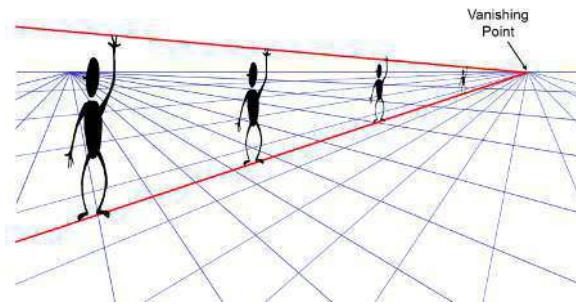
הנקודה בה מתקבל הגובה המקסימלית נקראת ה-Vanishing Point. נשים לב שלקוויים מקבילים יש את אותו ה-Vanishing Point. נפגשים באותו נקודה באינסוף.

הערה: לקוים אופקיים ה-Vanishing Point הוא תמיד באופק.



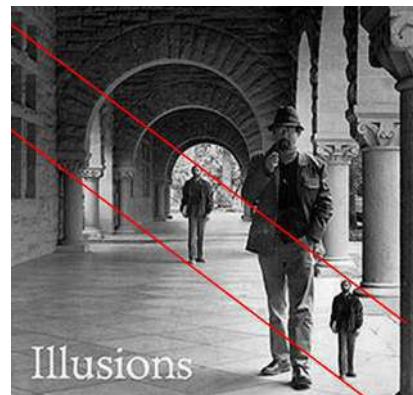
איור 137: ככל שהרכפה מתרחקת כך היא מופיעה יותר גבוהה בתמונה. נשים לב שלקוויים מקבילים יש אותו Vanishing Point.

דוגמה: נסתכל על אנשים באותו גובה עומדים בשורה.eko שמחבר את הידיים שלהם מקביל לeko שמחבר את הרגליים שלהם, לכן שני הכוונים יפגשו באותה נקודה באופק (Vanishing Point). מכאן שככל שהאנשים יהיו יותר רחוקים, הם יראו יותר קטנים.



אייר 138: נקודות העלמות של הקו שמחבר את הידיים של האנשים הוא אותו קו שמחבר את הרגליים שלהם. הקווים האלה מקבילים ולמעשה, לכל קבוצה של קווים מקבילים יש אותו Vanishing Point.

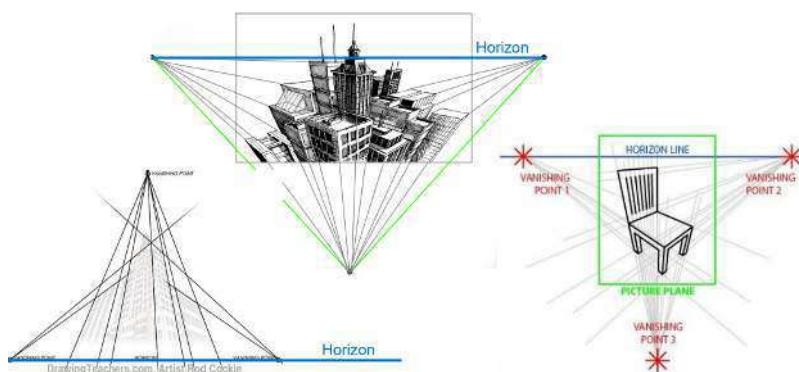
המוח שלנו רגיל לkoncept זהה, אך בתמונה הבאה האיש השמאלי והאיש הימני ביותר ("הקטן") נראים לנו בגבהים שונים, על אף שהם באותו גובה:



אייר 139: אשליה אופטית שגורמת לשני האנשים בתמונה להראות בגבהים שונים

מלבד נקודות העלמות אופקיות, יש גם נקודות העלמות אנכיות.

אוהבים להשתמש בנקודות העלמות בציור כדי ליצור פרספקטיבה טובה.



אייר 140: נקודות העלמות לא חייבות להיות אופקיות.

15.12 רמזים נוספים לעומק

יש דברים נוספים שמאפשרים לעיניים שלנו לאזהות עמוק.

הסתירה

אם חפץ אחד מסתיר חפץ אחר, זה אומר שהוא קרוב יותר אליו.

עראפל

אם האוויר לא צלול, ככל שאובייקט יהיה רחוק יותר, הוא נראה יותר מעורפל.



איור 141: ההרים הרחוקים נראים יותר מעורפלים כי הם יותר רחוקים

צל

נביט בדוגמה הבאה, שמחישה כיצד צל משפיעה על הפרטקטיביה ותפיסת המיקום שלנו:

Texture Gradients

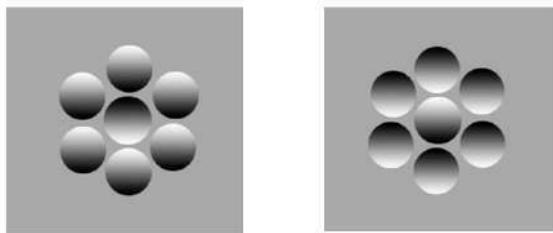
הצורה והגודל של אלמנטים בטקסטורה:



איור 143: משמאל: אבני רחוקות הן כנראה קטנות יותר כי כנראה כל האבניים הן באותו גודל. מימין: נראה כאילו האובייקט הוא גליל והצדדים רחוקים יותר.

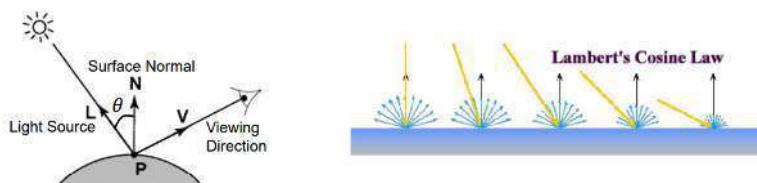
צורה מהצללה

בהתאם למקומות הצל, אנו מסיקים האם אובייקט מסוים בולט החוצה או נכנס פנימה. המוח שלנו מפרש את התמונות הבאות בהתאם לאיפה המוח שלנו חושב שמקור האור נמצא.



איור 144: שתי התמונות הן למעשה אותה תמונה רק בסיבוב של 180°

קיים משטח בשם משטח Matt ("Lambertian") שעבורו כמות האור שמוחזרת ממוקור אור תלוי אך ורק בזווית בין מקור האור והנורמל (וain תלוות בזווית בין וביין המשטח, לא משנה מאיזה כיוון נסתכל עליו נראה אותו דבר). ככלمر עבור משטח Matt מתתקיים $I \approx \cos \theta$, כאשר I היא עוצמת האור.



איור 145: עוצמת ההחזרה משטח Matt תלוי אך ורק בזווית של הנורמל ומקור האור

משטח Matt הוא הפוך למשטח המבריק ביותר: מראה. במראה יש תלות גדולה בכיוון שהוא מסתכלים ממנו.

חלק XIII

דחסית תמונות

דחסית תמונות משמשו הקטנת כמות הביטים הדרושים לייצוג תמונה. דבר זה חשוב כדי שהיה ניתן לאחסן יותר תמונות ויהיה אפשר להעביר אותן ברשת מהר יותר.

דוגמה. תמונה בודדת ברזולוציה $1K \times 1K \times 24bits = 3MB$.
 תמונה בודדת ברזולוציה $5K \times 5K \times 24bits = 75MB$.
 שנייה אחת של סרטון עם 25 מסרונים ברזולוציה $1K \times 1K : 75MB$. שעה של סרטון כזה: $270GB$. כמובן, בלי דחסית כמות הזכרון הדרושה לנו גדולה מאוד.

אחד הסיבות שאפשר לדחוס תמונות וסרטונים היא שיש תלות במרחב: פיקסלים קרובים נראים דומים, ויש תלות בזמן: מסרונים עוקבים נראים דומים. כמובן סיכוי טוב שהוא כפליות. לכן שעה אחת של סרטון HD יכולה להידחס ל- $1GB$ (במקום $270GB$).

סוגי דחסית

- Lossless - אין אובדן אינפורמציה ונitin לשחזר בדיקות את התמונה המקורי. משמש בעיקר לתמונות רפואיות (מקומות בהם אפילו שינוי של בית יכול להיות מאוד חשוב).
- Lossy - עלולים להיות שינויים קטנים בתמונה, אך שבקושי נבחין בהם בעין.

16 קוד האפמן (Huffman Code)

קוד האפמן הוא קידוד שמחזיר קידוד אופטימלי (הסתברותית אין קידוד שדו-חותם מידע יותר ממנו).⁵ הקידוד פועל באופן הבא:

1. נחשב את התפלגות ערכי הפיקסלים בתמונה - היסטוגרמה מונרמלת. (לדוגמה, עבור המילה AAAABBCD ההתפלגות היא $A : 0.5, B : 0.2, C : 0.125, D : 0.15$).
2. ערכים עם הסתברות גבוהה (שחזרים הרבה פעמים במידע) יקבלו קוד קצר יותר, וערכים עם הסתברות נמוכה (שלא מופיעים הרבה במידע) יקבלו קוד ארוך יותר.

⁵במצבים מסוימים כמוון ניתן למצוא קידודים שיתנו תוצאה יותר טובה, אך מדובר מבחינה הסתברותית.

היצוג של הקוד נעשה ע"י בניית עץ ביןארי כך שלכל ערך (אות או פיקסל) יהיה עלה בעץ. כל סימן מקודד ע"י המסלול אליו בעץ מהוירש: כל פניה שמאלת בעץ היא 0 וכל פניה ימינה היא 1.

האופן שבו נקבע סדר הקודוקדים הוא ע"י חיבור איטרטיבי של שתי התפלגיות הנמכות ביותר להתפלגות אחת, וכך התפלגיות הנמכות ביותר יהיו בתחלת העץ - ככלمر הקוד שלhn יהיה הכי ארוך, ושל התפלגיות גבוהות יהיה הכי קצר.

להלן האלגוריתם המוצע באמצעות Priority Queue:

Algorithm 20 Huffman Encoding

```

1 : Compute: Appearance histogram  $H$  of each character
2 : Build A min heap  $Q$  where each node contains a character and a probability  $\left( \frac{H[\text{char}]}{\sum_{\text{char}} H[\text{char}]} \right)$  as a key
3 : While  $Q$  contains more than one element:
4 :      $a = \text{Extract-Min}(Q)$ 
5 :      $b = \text{Extract-Min}(Q)$ 
6 :     Create new node  $c$  with probability  $a + b$  whose children are  $a, b$ 
7 :     insert  $c$  into  $Q$ 
8 : Make the remained element in  $Q$  the head of the tree  $T$ 
9 : return  $T$ 
```

נבחר את דברינו באמצעות דוגמה:

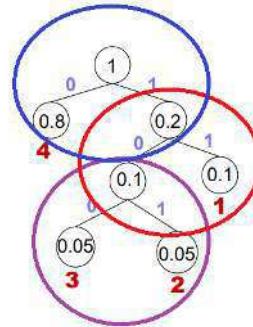
באյור ניתן לראות בטבלה השמאלית העליונה את התפלגיות המקוריות.

באייטרציה הראשונה, התפלגיות הנמכות ביותר הן 0.05, 3 : 0.05, 2 : 0.05 וכאן אנו לחבר אותן לאותו קודקוד, שיחזק את הסכום שלhn: $0.05 + 0.05 = 0.1$. נסיר את התפלגיות הישנות ונוסיף את התפלגות החדשה.

באייטרציה השנייה, התפלגיות הנמכות ביותר הן 0.1, 2 – 3 : 0.1 וכאן אנו לחבר אותן לאותו קודקוד, שיחזק את הסכום שלhn: $0.1 + 0.1 = 0.2$. נסיר את התפלגיות הישנות ונוסיף את התפלגות החדשה.

באייטרציה השלישית, התפלגיות הנמכות ביותר הן 0.2, 4 : 0.8 וכאן אנו לחבר אותן לאותו קודקוד, שיחזק את הסכום שלhn: $0.2 + 0.8 = 1$. נסיר את התפלגיות הישנות ונוסיף את התפלגות החדשה. עכשו הגיענו לקודוד יחיד עם התפלגות 1, לכן נסיים.

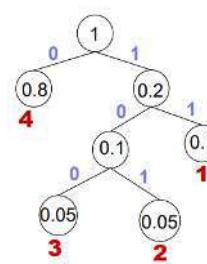
Pixel Value	1	2	3	4
probability	0.1	0.05	0.05	0.8
Pixel Value	1	2	3	4
probability	0.1	0.1		0.8
Pixel Value	123	4		
probability	0.2	0.8		
Pixel Value	1234			
probability	1			



איור 146: בניית קוד האפמן

עכשו הקידוד של כל ערך (צבוע **באדום בולט**), מתקבל כאמור במסלול מהשורש אליו. להלן טבלה שמסכמת את הקידוד לפי מה שאמרנו זה עתה:

Pixel Value	1	2	3	4
probability	0.1	0.05	0.05	0.8
Huffman Code	11	101	100	0
Code Length	2	3	3	1



איור 147: סיכום קוד האפמן עבור הדוגמיה הקודמת

אם נרצה לקודד כך אלף פיקסלים, בתחילת (לפי הסתברות לקל כל ערך בין 1 ל-4), קיבל שכמות הביטים שנזדקק לה היא

$$\underbrace{100 \cdot 2}_{\text{עבור 1}} + \underbrace{50 \cdot 3}_{\text{עבור 2}} + \underbrace{50 \cdot 3}_{\text{עבור 3}} + \underbrace{800 \cdot 1}_{\text{עבור 4}} = 1300$$

אם היינו עושים זאת בלי קידוד, היינו זקוקים ל-2000 ביטים (צריך 2 ביטים כדי לייצג 4 ערכים, ויש 1000 פיקסלים), כלומר **עם האפמן חסכנו המון מוקום!**

הערה הדוגמיה זו הייתה עבור 4 ערכים. ככל שיש יותר ערכים שונים, החיסכון יותר גדול.

הערה אם יש הסתברות שווה לקבל כל ערך, אותו הדבר יהיה נכון דחיסה. הדחיסה בקוד האפמן באה מכך שהוא נוטנים קוד קצר לערכים שכיחים.

הערה הקוד בניו כך שלא יהיה ספק לגבי איך אמרוים לקרוא אותן: לא יתכן שאות אחת תקודד ל-100 ואות אחרת ל-10, כי אז יהיה ספק לגבי האם הרץ הוא 100 או 10 שלאخرو מופיע.

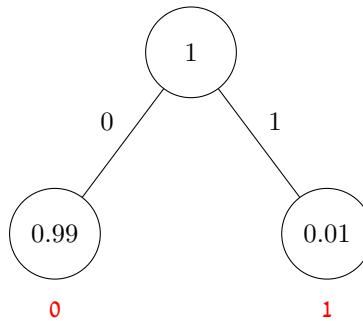
ככה כשהיו קוראים ביטים, אפשר להתחיל לлечט להתקדם מஹשור בהתאם לביטים שאנו קוראים, וברגע שנגיעה לעלה נדע שישינוו לקרווא ערך, יוכל לחזור לשורש ולהמשיך באוטו תהליך לפיה הביטים שנותרו.

שאלה נניח שיש לנו תמונה בינהרית, כך שכל פיקסל הוא בית אחד, עם התפלגות קיזומית של $0.01 : 1 : 0.99 : 0$. בהתפלגות כזו קיזומית, נדמה שהאפמן יכול להגיד.

אולם כפי שניתנו לראותה לפי העץ הבינאי שבנוו לפי הקידוד, צריך עדין בית אחד לכל פיקסל - הרי צריך לפחות בית

אחד כדי ליצוג בית אחד, וביצוג הבנאי המקורי ממילא כל בית תפס בית אחד (באופן טריוויאלי).

מהו ניתן לעשות כדי לאפשר קידוד עיל?



תשובה במקומות לקובוד כל בית בנפרד (שזה לא תורם לנו דבר), נחלק את זרם הביטים למקטעים, למשל קבוצות בגודל 8 ביטים. בעת במקומות שני ערכיים (0 ו-1), נקבל מספר גדול יותר של ערכים (0 עד 255) שנוכל לדוחס לפיו.

... **111111111** | **111111111** | **111111101** | **11111110** | ...

Entropy – Encoding - **דוחיסת אנטרופיה** - 16.1

העשרה - מוטיבציה ואינטואיציה

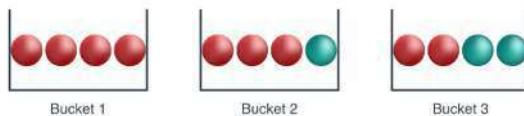
⁶ נתחיל בהעשרה, שמציגת את האינטואיציה של משווהת שאנו לאנתרופיה.

⁶ חלק זה לקוח מסיכומו של Halsadi Assaf. הוא מצרף את **זהה הסרטון** כהסבר נוספת.

שאלה בכללי, מה זה אנטרופיה (Entropy)?

תשובה אנטרופיה היא קונספט מדעי המתאר את "כמות האי-סדר" של משהו. למשל - בפיזיקה אנטרופיה מוגדרת את קצב התנועה של חלקיקים באובייקט (למשל - לקרה יש אנטרופיה נמוכה, לעומת זאת יש אנטרופיה טיפה יותר גבואה ולאדים יש אנטרופיה גבואה). אבל, מה שמעניין אותנו הוא הביטוי של אנטרופיה בעולם הסתברות - מدد לגודל האפקטיבי של מרחב הסתברות. אפשר לחשב על אנטרופיה בהסתברות כ"כמה קשה יהיה לנו לחזות את התוצאה", או "כמה מידע יש לנו על בחירת תוצאה ממרחבי הסתברות כלשהו". ננסה להבהיר את הרעיון הזה בשאלות הבאות.

שאלה נגידר 3 שקים כך שבכל שק יש 4 כדורים אדומים, השק השלישי 3 כדורים אדומים ואחד יrox, ובsek השלישי 2 כדורים אדומים ו-2 יroxים (ראו איור להמחשה). אינטואיטיבית (כי עוד לא הגדרנו כלום), מי מהשקים יש אנטרופיה גבואה ולמי מהם יש אנטרופיה נמוכה?



איור 148: המוחשת השקים

תשובה אמרנו שאנטרופיה היא דרך למדוד כמה קשה יהיה לנו לחזות תוצאה מרחב הסתברותי כלשהו. כל השק מתואר מרחב הסתברותי, לדוגמה - השק הראשון מתואר מרחבו לאדום יש הסתברות 1 ולירוק 0. במקרה שלנו, אפשר לחוש על זה כמה קשה יהיה לנו לנחש איזה כדור אני הולך לשולוף מכל השק. אם קשה לי לנחש, אז האנטרופיה גבואה - ואם קל לי לנחש, האנטרופיה נמוכה.

לפי ההגיון הזה: השק הראשון יהיה אנטרופיה נמוכה מאוד כי תמיד אוכל לנחש אדום ולהיות צודק. השק השני יהיה אנטרופיה בינונית: אם לנחש אדום, ברוב החלטה צודק. השק השלישי יהיה אנטרופיה גבואה, כי לא משנה מה לנחש - בממוצע אטעה בחצי מהפעמים.

אינטואיציה זה טוב ויפה, אבל נרצה למצוא דרך מתמטית למציאת האנטרופיה של מרחב הסתברות כלשהו. לשם כך, נתחיל מהгадיר משחק.

מהלך המשחק:

1. נבחר את אחד השקים.

2. לאורך 4 תורות:

(א) נשלו כדור באקראי מהשק.

(ב) נרשום מה הצביע של הכדור.

(ג) נחזיר את הכדור השק.

3. אם הcodors שנסלפו היו לפי הסדר בציור, ניצחנו במשחק. אם לא, הפסדנו.

לדוגמה - אם בחרנו בשק השני (האמצעי), ננצח אם שלושת codors הראשונים שנשלוף היו אדומים והרביעי יהיה ירוק, אחרת נפסיד.

שאלה מי מבין שלושת השקמים, הוא השק שהכי כדאי לבחור כדי לנצח במשחק? למה?

תשובה אינטואיטיבית, ברור לנו שהشك הראשון עדיף, כי לא משנה מה נשלוף ממנו - תמיד נקבל 4 codors אדומים וננצח במשחק. כדי לחשב את הסיכוי לניצחון עבור כל אחד מה硕ים, נחשב מה ההסתברות לשיליפה של codor ב痼ע המתאים בכל אחד מארבעת התורות, ואז נכפיל ביניהם (בגלל שאנו מחזירים את codor שנשלפנו כל פעם, כל שיליפה לא תלויה באחרות ולכן ההסתברות לשיליפה הארבעה הנכונים היא מכפלת ההסתברויות של כל אחד מהם בנפרד). לדוגמה, עבור השק השני - בשלושת התורות הראשונים אנחנו צריכים לשולוף codor אדום. ההסתברות לשיליפת codor אדום מתוך השק היא 0.75. בתור האחרון צריך לשולוף codor ירוק מהשк הוא 0.25. לכן, ההסתברות לניצחון כשבוחרים את השק השני היא $0.75^3 \cdot 0.25 = 0.105$.³ אם נחזור על התהליך עבור שאר השקמים, נקבל את הטבלה באייר הנ"ל. ההסתברות לניצח כשבוחרים את השק הראשון היא 1, כמובן - תמיד מנצח. לכן, מעדיף לבחור אותו.

	P(red)	P(blue)	P(winning)
	1	0	$1 \times 1 \times 1 \times 1 = 1$
	0.75	0.25	$0.75 \times 0.75 \times 0.75 \times 0.25 = 0.105$
	0.5	0.5	$0.5 \times 0.5 \times 0.5 \times 0.5 = 0.0625$

אייר 149: טבלת הסתברויות לניצח במשחק עבור כל שק

שאלה איך המשחק שתיארנו יכול לעזור לנו לתאר אנטרופיה בצורה מתמטית?

תשובה קודם כל, נשים לב שהאנטואומיצה שלנו לאנטרופיה הובילה לבדוק לנצח ההפוכה מוצאת המשחק (כלומר - לשחק עם האנטרופיה הכי נמוכה יש את ההסתברות הכי גבוהה לניצח במשחק ולהפך). הצעה ראשונה תהיה פשוט להפוך את ההסתברויות, זהה באמות יוביל לתוצאה נכונה - אבל נוצרתפה בעיה. בדרך לפתרון השתמשנו במכפלה בין הסתברויות, מה שיכול לגרור מספרים מאד קטנים. כבר בדוגמה שלנו, למרות שיש בה רק 4 codors, הגענו באחת התוצאות למספר הקטן ביותר 0.0625. אם היו לנו 100 או אפילו 1000 codors, ההסתברויות שהיינו מגיעים אליהן היו אפילו קטנות יותר (ובהתאם קשות לייצוג ולשימוש בחישובים).

הפתרון לעביה הזה הוא לוגריתמים (נשתמש ב- \log_2 !). בפרט, שימוש בתכונה הבאה:

$$\log(a \cdot b) = \log a + \log b$$

כדי להפוך את המכפלות שתיארנו לסכום. נפעיל את הטריק זהה לדוגמה על חישוב ההסתברות של השק השני ונקבל

$$\log(0.75^3 \cdot 0.25) = 3\log(0.75) + \log(0.25) = -3.245$$

כעת, לוגריתם של מספרים בין 0 ל-1 הוא שלילי, ונעדייף שהמדד שלנו יהיה אי-שלילי (כלומר - אנטרופיה נמוכה תהיה 0 ואנטרופיה גבוהה תהיה מספר חיבי גדול), אז נכפיל את הביטוי במינוס 1 ונקבל הפעם

$$-\log(0.75^3 \cdot 0.25) = 3.245$$

כמו כן, נעדייף שהמדד לא יתפרש על מספרים עצומים, לכן נורמל אותו ע"י ממוצע (נחלק במספר הבודדים) ונקבל

$$\frac{1}{4}(-\log(0.75^3 \cdot 0.25)) = 0.81125$$

באותה צורה, האנטרופיה של השק הראשון לפי הממדד שלנו תהיה

$$-\frac{1}{4} \cdot 4\log(1) = 0$$

ועל השק השלישי:

$$-\frac{1}{4} \cdot 4\log(0.5) = 1$$

בכללי, הנוסחה שהגענו אליה לחישוב אנטרופיה במשחק של מחלקות לנצח במשחק:

$$Entropy = -\sum_{k=1}^n p(k) \log p(k)$$

כאשר n זו כמות המחלקות (במשחק יש 2 מחלקות - אדום וירוק), $p(k)$ זו ההסתברות של המחלוקת (לדוגמה, בשק השני ההסתברות לבחור כדור אדום היא 0.75 וההסתברות לבחור כדור י록 היא 0.25). אם נציב עבור השק השני נקבל

$$Entropy = -0.75 \cdot \log(0.75) - 0.25 \cdot \log(0.25) \sim 0.81$$

אדם בשם שאנון עבד הרבה בתורת האינפורמציה, והוא אמר את הדבר הבא:

בהתנחת התפלגות p על גבי מאורעות כלשהם, נרצה לדעת כמה אינפורמציה קיבלו אם נאמר לנו שמאורע מסוים k קרה? (יחידת אינפורמציה היא ביט)

יש מאורעות שמכילים יותר אינפורמציה מאחרים - מאורעות שאנו מוצפים להם מכילים פחות מידע, מאורעות נדירים מכילים יותר מידע.

לדוגמא, אם נגיד לכם שהשעה היא 20:19, קיבתם 0 ביטים של אינפורמציה כי ככלנו כבר יש שעון. אם נגיד לכם שהשעה היא 20:27 כי העולם החליף למרכז עם יותר מ-24 שעות), קיבתם המון אינפורמציה.

על כן, שאנו קבע את הנוסחה לאינפורמציה בתור $\boxed{-\log p(k)}$. אם $p(k)$ קטן - המון אינפורמציה, אם $p(k)$ גדול - מעט אינפורמציה.

בגללSCP כל מאורע k מגיע עם הסתברות (k) , האינפורמציה הממווצעת - האנטרופיה - עבור התפלגות זו היא

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

שים לב לתכונות הבאות:

- אנטרופיה מעניקה את האינפורמציה לכל פיקסל עבור התפלגות כלשהי: (**bits-per-pixel**)

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

- התפלגות עם א-ודאות מקסימלית, $\forall k, p(k) = \frac{1}{n}$

$$\text{Entropy} = - \sum_{k=0}^n \frac{1}{n} \log \frac{1}{n} = \log(n)$$

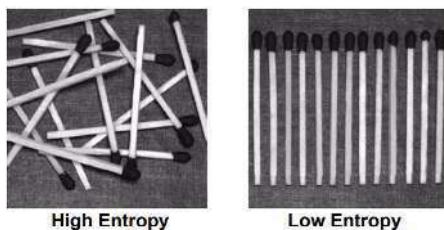
זו האנטרופיה הבי גזולה.

- התפלגות עם ודאות מקסימלית, $p(0) = 1, \forall k \neq 0, p(k) = 0$

$$\text{Entropy} = -1 \cdot \log(1) - \sum_{k=0}^n 0 \log 0 = 0$$

זו האנטרופיה הבי קטנה.

הערה קידוד האפמן נקרא דחיסת אנטרופיה כי במצב בו האנטרופיה אינה מקסימלית הוא מנצל את האנטרופיה כדי להשתמש בפחות ביטים.



איור 150: שתי תמונות של גפרורים, עם היסטוגרמה שווה פחota או יותר (האפקן יחייב קידוד מאוד דומה עבור שניהם) או לאם בתמונה הימנית האנטרופיה נמוכה יותר כי יש תבניות ושורות שדומות זו לזו.

שאלה כיצד ניתן להשתמש בתבניות החוזרות בתמונה הגפרורים הימנית (איור לעיל) כדי להשיג קידודיעיל יותר, ביחס לתמונה השמאלית בה יש פחות תבניות חוזרות?

תשובה נשתמשה בדחיפסה בשם למפל-זיו.

Lempel – Ziv Compression(LZW) 17

דחיפסת למפל-זיו היא הדחיפסה בה משתמשים פורטטי קבצים כמו zip. אלגוריתם זה הוא גאווה ישראלי כי המפתחים שלו הם מהטכניון! יש למפל-זיו כל מיני וריאציות ואנו נתמקד באחת בשם LZW.

- בנגוד, להאפקן שמודע רק להתפלגות הערכים, למפל-זיו מנצח גם תבניות חוזרות בזרם המידע.
- בנגוד להאפקן שמקודד את המידע בשני מעברים (מציאת התפלגות במעבר הראשון וקידוד שני), למפל-זיו מקודד במעבר אחד. אלגוריתם זה נקרא online – algorithm.

- למפל-זיו בונה מילון של מחרוזות (במקרה של תמונות تو במחuzeות הוא ערך פיקסל):

▷ במקומות לשדר את כל המקטע (מחרוזת) שהוא עבר עליו כרגע, הוא יסדר את מיקום המחרוזת במילון (הקוד שמתאים למחרוזת במילון).

▷ אם המחרוזת לא במילון עדין, נבחר לו קוד שייצג אותו ונשמר אותו במילון.

להלן האלגוריתם של LZ7:

Algorithm 21 Lempel – Ziv Compression(LZW)

```

1 : Initialize a dictionary with characters (0 . . . 255)
2 :  $P$  = first input character
3 : While not end of input stream
4 :    $C$  = next input character
5 :   if  $PC$  is in dictionary // concatenation
6 :      $P = PC$  // concatenation
7 :   else //  $PC$  not in dictionary
8 :     output the code for  $P$ 
9 :     add  $PC$  to the dictionary // concatenation
10:     $P = C$ 
11: end While

```

מלבד דחיסה, צריך גם אלגוריתם לפתיחת דחיסה:

Algorithm 22 Lempel – Ziv DeCompression(LZW)

```

1 : Initialize a dictionary with characters (0 . . . 255)
2 :  $O$  = first input code
3 : output  $d(O)$  // dictionary entry of  $O$ 
4 : While not end of input stream
5 :    $N$  = next input code
6 :   if  $N$  not in dictionary // One possibility of this...
7 :      $S = d(O)$  // dictionary entry of  $O$ 
8 :      $S = SC$  // concatenation
9 :   else
10:      $S = d(N)$  // dictionary entry of  $N$ 
11 :   output  $S$ 
12 :    $C$  = first character of  $S$ 
13 :   add  $d(O) C$  to the dictionary // concatenation
14 :    $O = N$ 
15 : end While

```

מה אנחנו עושים כאן למעשה?

- אנחנו רוצים לבנות את הטבלה שבנו בעת הקידוד, אנו מסוגלים לעשות זאת מכיוון שבוודאות התווים הראשוניים בקידוד יהיו תווים באורך אחד. לכן נוכל לחזור על הקידוד שעשינו ולהשים איך נראה הטקסט המקורי.
- אם הקוד הבא נמצא בטבלה כלומר הוא מתאים לרצף שכברמצאנו, נשלוף את הסטרינג המתאים לו ניקח את התו הראשון ונרשיר לסתירינג הקודם שמצאנו, ונוסיף לטבלה. מדוע? מכיוון שבעת הקידוד, כאשרמצאנותו שכבר בטבלה עשינו בדיקת זה. נעדכן את הסטרינג הישן O להיות הסטרינג החדש N .
- אם הקוד הבא לא נמצא בטבלה, **لتקנו**

Run Length Encoding 18

הרעיון ב-Run Length פשטוט למדי: נשלח זוגות (c, r) כאשר c הוא הצבע (*color*) ו- r הוא כמות הפעמים ש- c -הופיע ברצף (*run length*). משתמשים בשיטה זו בדברים כמו פקס (שם הערכיים היחידים הם 1, 0). באופן כללי נרצה להשתמש בשיטה זו כדי לממן אזוריים אחידים בתמונות (בתמונות בינאריות הסיכון לכך גובה).

האלגוריתם הזה נכשל (קרי, לא עיל) כשההנחה שיש סדרות גדולות עם אותו ערך לא נכון.

דוגמה עבור המערך

19 19 19 19 15 15 14 14 14 9 2

נקבל את הזוגות

$(19, 4), (15, 2), (14, 3), (9, 1), (2, 1)$

הערה להשתמש ב-*run length* יותר מפעם אחת לא ידוחס את התמונה יותר ממה שהיא נדחסה קודם. נסתכל לדוגמה על

הסדרה הבאה, שמורכבת מקידוד חוזר לפי run length

```

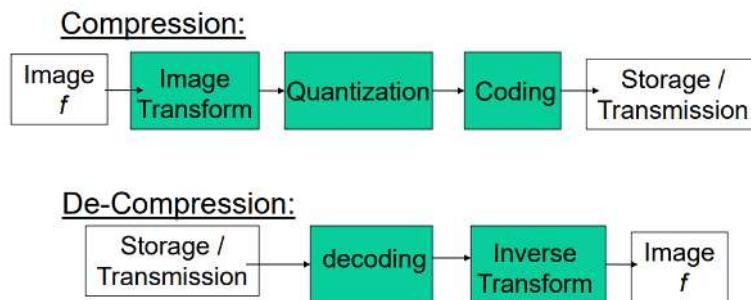
1
1 1
2 1
1 2 1 1
1 1 1 2 2 1
3 1 2 2 1 1
1 3 1 1 2 2 2 1 ...

```

הערה הדבר הזה נכון לכל סוגי הדחיסה! אי אפשר לעשות האפמן ואז על התוצאה לעשות למפל-זיו ואז לעשות עוד איזשהו קידוד, ולצפות לדחיסה מאד טובה. אחרי דחיסה אחת רוב הסיכויים שם נדחוס פעם שנייה הקובץ רק יגדל.

19 מערכות דחיסה

דחיסה כללית (ופתיחה דחיסה) ניתן לתאר כדלקמן



איור 151: תיאור כללי למערכות דחיסה

חשוב נשים לב שהחלק בו נאבד המידע הוא הקוונטיזציה. בתיאור של פתיחה הדחיסה ניתן לראות שיש טרנס' הפוכה שמתאימה לחלק של הטרנספורמציה, ו-*decoding*-*coding* שמתאים לחלק של ה-*coding*. רק לקוונטיזציה אין חלק מותאים. קוונטיזציה היא המקור העיקרי לדחיסה, אך גם המקור העיקרי לשגיאות. את השגיאות הללו לא קל לאמוד.

שאלת למה צריך קוונטיזציה וטרנספורמציה?

תשובה הקוונטיזציה מקטינה את כמות הערכים האפשריים ומקטינה את האנתרופופיה של התמונה. כך גם הטרנספורמציה באה כדי להקטין את האנתרופופיה. מי שגורמת לאובדן המידע היא כאמור הקוונטיזציה

הערה כדי לאמוד את השגיאה אפשר לנסות לדוחס את התמונה, לפתח את הדחיסה ולהשוות לתמונה המקורית באמצעות חישור pixel-wise וסכימת ריבועי כל החיסורים. הבדיקה הזאת לא שווה כי אם ניקח תמונה ונוסיף 2 לערך של כל פיקסל, לא נראה הבדל בעין, אבל השגיאה שנקבל לפי המדריך זהה תהיה ענקית.

דוגמה לדחיסה

- **טרנספורמציה:** קונבולוציה עם $(1, -1, -1)$.
 - **קונטיזציה:** בלי.
 - **קידוד:** קידוד האפסון.
 - **אובדן מידע:** אין, לא הייתה קונטיזציה.
- זכור שקונבולוציה עם $(1, -1)$ היא נזורת. אם רוב הפיקסלים השכנים דומים זה לזה, רוב ההיסטוריה תהיה סביר 0 (כמובןiento) גם edge-edge'ים בהם הנזרת לא תהיה אפס). עקב כך, האנתרופופיה תרד והדחיסה תשתרף. יחד עם זאת - לא איבדנו מידע כי לא הייתה קונטיזציה!

הערה כדי לשחזר את התמונה, מלבד הקידוד שתיארנו **צריך** לשולח גם את העמודה **הראשונה** של התמונה, כי ממנה נוכל לשחזר באמצעות הנזרת את השורה השנייה, ואז את השלישית וכן הלאה.

דוגמה לדחיסה

- **טרנספורמציה:** בלי.
- **קונטיזציה:** אופטימלית
- **קידוד:** בלי.
- **אובדן מידע:** יש אובדן מידע כי עשינו קונטיזציה, אבל בקונטיזציה אופטימלית המון פעמים קיבל תמונה שנראית טוב מאוד לעין שלנו.

Predictive Encoding 20

הרצאה 12

הweeney ב-Predictive Encoding הוא להשתמש בפונק' כלשהי שמנסה לחזות מה אמר או להיות ערך הפיקסל. הקידוד הזה יעבד במעבר raster על הפיקסלים - ככלומר ניבור על הפיקסלים משמאלי לימיון שורה, כך שבקידוד ה-Predictive Encoding כל פיקסל יוכל להשתמש בערכיהם של הפיקסלים שמשמאלו ומעליו, כפי שתואר עכשו.

הקידוד יעבד כדלקמן:

1. נגידר פונק' חיזוי f שמנסה לחזות את הערך של כל פיקסל לפי הפיקסלים השכנים שמשמאלו ומעליו.⁷
2. אם x הוא הערך האמתי של הפיקסל ו- x' הוא הערך ש- f חצתה, מה שנשדר בקידוד הוא $x' - x$: השגיאה של f .

הערה נשים לב שבשלב 2, אם f באמת פונק' חיזוי טובה, רוב השגיאות יהיו קרובות ל-0 ואז כמו בדוגמה עם הקונבולוציה עם $(-1, 1)$ שימוש בהպמן יתן תוצאה טובה.

דוגמה

בחר פונק' לינארית $y_1 = h_1 y_1 + h_2 y_2 + h_3 y_3$. שימו לב שאנו עדין צריכים לבחור את ערכי המשקלות h_1, h_2, h_3 . בחרית הערכים מותבצעת לפי הנחות על התמונה.

y_2	y_3	
y_1	$x=?$	

איור 152: בגלל שאנו עוברים ב-raster על הערכים, אנו כבר יודעים מיהם y_1, y_2, y_3 . בעצם נשתמש בערכים הללו כדי לחזות את הערך של x .

אם אנו חושבים שהתמונה יחסית אחידה, בחרה טובה של משקלות תהיה $\frac{1}{3}, h_1 = h_2 = h_3 = \frac{1}{3}$, כלומר נבצע מיצוע של השכנים של x (שוב, בלי השכנים שמתחתיו ומימינו, כי בזמן פתיחת הדחיסה לא נדע את ערכם כשנחשב את הערך של x). אם אנחנו חושבים שהתמונה גדלה לינארית, אז הגיוני לבחור $h_1 = h_3 = 0 < 0 < h_2$. הסיבה לבחירה זו היא שם התמונה גדלתה לינארית, היא מתנהגת כמו משור "עליה", ולכן ככל ש- y_2 יהיה קטן יותר x יגדל יותר.

אם נבחר $0, h_1 = 1, h_2 = h_3 = 0$, נקבל את אותה תוצאה שקיבנו כמו בקידוד עם קונבולוציה עם $(1, -1)$. זאת כי אנחנו נשלח בקידוד את $y - x$ שהוא בדיקת מה ששלחנו במקרה של הקונבולוציה.

Pyramid Compression 21

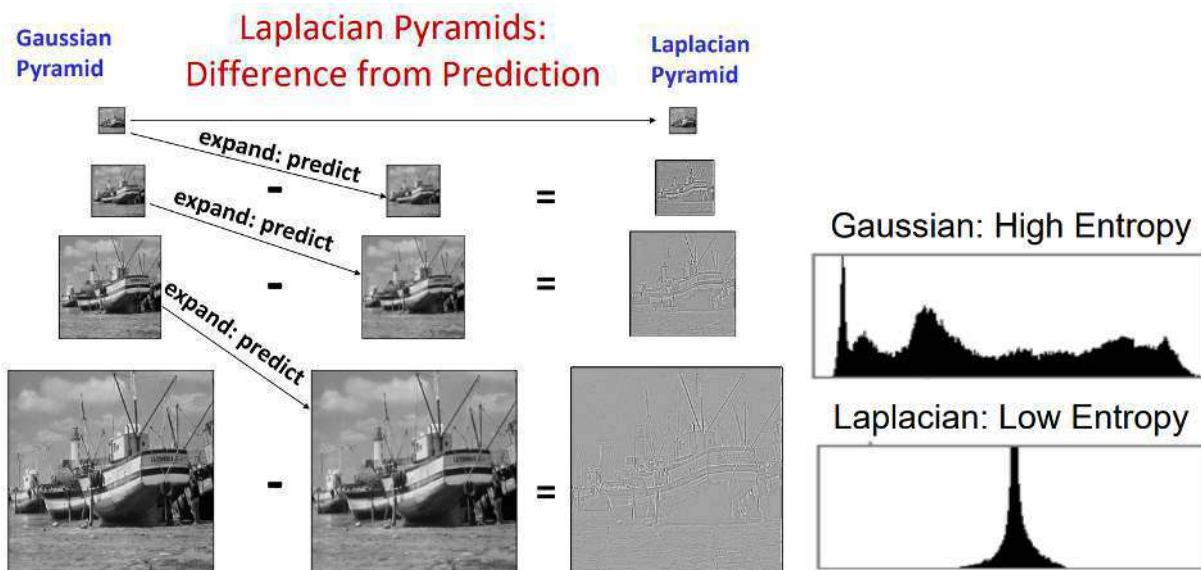
בדומה ל-Predictive Encoding, נשתמש בפרמידות כדי לאמוד את ערכי הפיקסלים. בפרמידת פלסיין אנו יורדים למטה, אומדים את ערכי הפיקסלים בפרמידה בעזרת התמונה שיותר קטנה ומוסיפים את הלפלסיין (שמוסיף את ההפרשים). אם כך, נוכל לבצע דחיסה לפי השלבים הבאים:

⁷אי אפשר להשתמש בפיקסלים שמייננו ומתחתיו כי אנחנו נ עבור על הפיקסלים בסריקת raster.

- **בנייה פרמידת פלסיין**
 - ▷ רמות גבהות יותר בפרמידה יחו אט הרמות הנמוכות יותר
 - **מבצע קוונטיציה אופטימלית של הערכים בפרמידה 5 – 3 ערכים**
 - **מבצע קידוד באמצעות דחיסת אנטרופיה**
 - ▷ לדוגמה האפמן או למפל-זיו - מכיוון שכל תמונה היא תמונה לפלייאן, זה יעבוד טוב.
- כדי לפתח את הדחיסה ולשזר את התמונה, נקרא את הקידוד ונקבל פרמידת פלסיין. נסכם את כל רמות הפרמידה כדי שלמדנו ונקבל את התמונה המקורית.

שאלה למה זה עובד ואיפה החלק של חיזוי הערכים?

תשובה אנו משתמשים בקוונטיציה שמקטינה את כמות הערכים. כמו כן, בזמן בניית הפרמידה אנו משתמשים בפעולות expand, שהוא מנסה לחזות איך נראה התמונה שיוטר גדולה. אנו מחסרים את התמונה האמיתית מהתמונה שחזינו באמצעות expand ומתקבלים את פרמידת הפלסיין.



אייר 153: משמאל: בזמן בניית פרמידת הפלסיין, אנו משתמשים בחיזויי ערכי הפיקסלים ברמות של הפרמידה. מימין: ההיסטוגרמה של פרמידת פלסיין צפופה מאוד ולכן הקידוד יהיה יעיל. ההיסטוגרמות הללו הן אף לפני קוונטיציה. (אחרי קוונטיציה ההיסטוגרמה הייתה נראית כמו מסרק)

בעיה בפרמידת פלסיין יש $\frac{4}{3}N^2$ פיקסלים, לעומת N^2 פיקסלים בתמונה המקורית. כלומר, אנחנו קודם מנפחים את כמות הביטים ואז דוחסים ומקטינים את הכמות.

Transform Compression 22

היא דחיסה כללית הפעלתה לפי השלבים הבאים:

1. נפצל את התמונה לבלוקים (לא-חוופים) בגודל $N \times N$.
2. נפעיל טרנספורמציה על כל בלוק כזה בגודל $N \times N$ ונקבל תוצאה בגודל $N \times N$.
3. על כל תוצאה טרנספורמציה נפעיל קוונטיזציה וקידוד

הפורמט JPEG, לדוגמה, משתמש ב-Transform Compression לפי הפרמטרים הבאים:

- גודל בלוק: 8×8

הגודל הזה נבחר בגלל עניין של עיצוב צ'יפים: אנחנו רוצים חומרה במכשיר שדוחסת את התמונה ב-realtime, אבל צ'יפ שעבוד על יותר מ-64 כניסה (פיקסלים) זה דבר יקר. אם היי עושים את זה, היי בוחרים בלוקים יותר גדולים כדי לשפר את היעילות.

- טרנספורמציה: Discrete Cosine Transform (DCT)

- קידוד: האפסן.

22.1 התמרת קוסינוס - DCT

בדומה להתרמת פורייה, בעולם של DCT כל התמונות משוכפלות באופן אינסופי רק באופן אחר: במקום לשים את התמונות אחת ליד השנייה בראף אינסופי, כל שכפול יהיה שכפול ראי, מתאים באյור הבא.

אחד היתרונות של DCT הוא שהערכים תמיד יהיו ממשיים, ואף פעם לא מורכבים כמו ב-DFT.



איור 4: משמאל: שכפול ב-*DCT* (תמונות מראה בכל כיוון). מימין: שכפול ב-*DFT* (שכפול בלי שינוי).

בגלל ש-*DCT* משתמש בשכפול ראי, כשבור מתמונה אחת לשכפול שלידה, המעבר יהיה רציף: לא תהיה קפיצה גדולה כמו ב-*DFT*, אין שינוי חד. ב-*DFT* אנו קופצים מקצת אחד של התמונה לказה השני, ואין בהכרח קשר בין הערכים בשני הקצוות.

לאורך ההסביר שימו לב לדמיון בין *DFT* ו-*DCT*.

בא בהרבה טעמים, זו אחת הנוסחות שלו:

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

כאשר

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

התמורה ההפוכה ל-*DCT*, קרי *IDCT*, היא

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

מסקנה $f(x)$ הוא צירוף לינארי של תדרי קוסינוס מהצורה $C(u) \cos\left(\frac{(2x+1)u\pi}{2N}\right)$, עם כפל במקדמי ההתמורה $C(u)$ ונרמול $\alpha(u)$.

בדו-מימד הנוסחה היא

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

בדו-מימד *IDCT-1*:

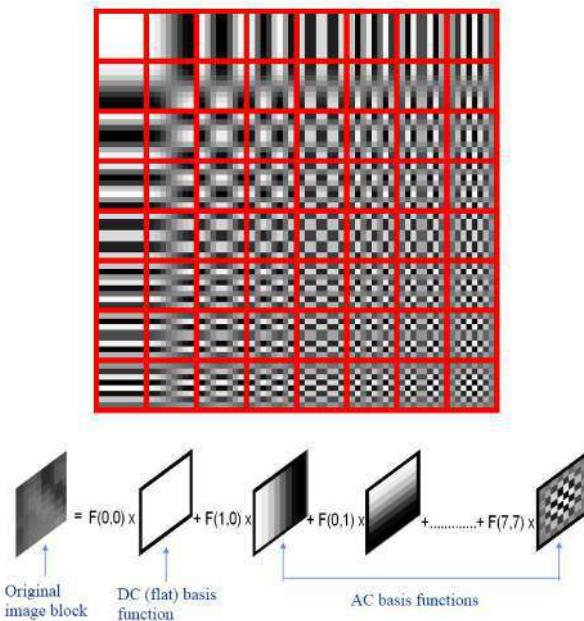
$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u, v) \alpha(u) \alpha(v) \cos\left(\frac{(2x+1)u\pi}{2N}\right) \cos\left(\frac{(2y+1)v\pi}{2N}\right)$$

פונק' הבסיס של *DCT* בדו-מימד הן:

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

דוגמה לבסיסי DCT בגודל 8×8

ב-JPEG השתמש בבלוקים בגודל 8×8 , אך ה- DCT בו נשתמש יהיה עם $N = 8$. כך נראה פונק' הבסיס עבור מקרה זה:



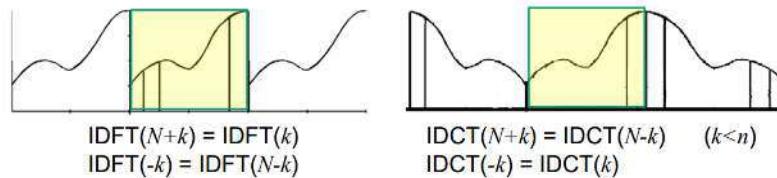
אייר 155: פונק' הבסיס של DCT דו-מימדי עם $N = 8$. תדרים נמוכים יותר נמצאים מצד שמאל-למעלה, ככל שיורדים באירוע ומתקדמיים ימינה התדרים גדלים בכל כיוון.
כאן צבע לבן = 1 וצבע שחור = -1.

כל תמונה בגודל 8×8 היא סכום משוקלל של 64 התמונות באירוע.

 יתרונות של DCT על FFT

אנו נעדיף להשתמש ב-DCT ביחס לתמונות מהסיבות הבאות:

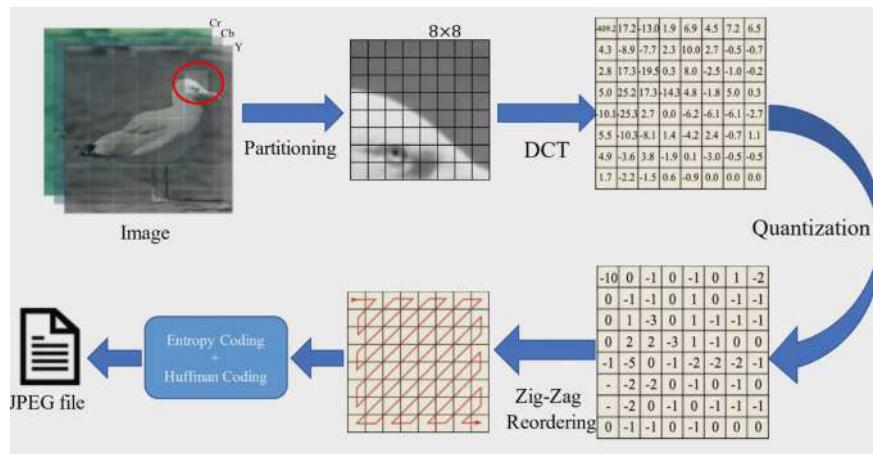
1. מחזוריות רציפה - השיקוף ב-DCT מוריד את האנטרופיה, לעומת FFT בו יש קפיצה לא רציפה, דבר שגדיל את האנטרופיה. אך DCT הוא יתן תוצאה יותר טובה בקידוד.
2. ערכים ממשיים - FFT מוציאה ערכים מרוכבים שקשה לעבוד איתם, לעומת DCT שモוציאה ערכים ממשיים.



איור 156: סימטריה רציפה ב-DCT, ביחס ל-DFT בו יש קפיצות.

JPEG 22.2 דחיסת

להלן השלבים בדחסית JPEG:



איור 157: השלבים בדחסית JPEG

JPEG משתמש בערוצי צבע שונים מ-RGB

Y' הוא עוצמת האור - סכום משוקלל של B

הם ערכי הצבע לפי הפרשי צבעים:

$$Cb = B - (R + G)$$

$$Cr = R - (G + B)$$

המעבר מ-RGB ל-CR, Cb, Y' מתבצע כך:

$$Y' = 0 + (0.299 \cdot R'_D) + (0.587 \cdot G'_D) + (0.114 \cdot B'_D)$$

$$C_B = 128 - (0.168736 \cdot R'_D) - (0.331264 \cdot G'_D) + (0.5 \cdot B'_D)$$

$$C_R = 128 + (0.5 \cdot R'_D) - (0.418688 \cdot G'_D) - (0.081312 \cdot B'_D)$$

ההנחה היא שהעין שלנו יותר רגישה ל-Y ולכן אותו נדחוס בתמונה מלאה. את Cr, Cb אלו הופכים לתמונה מוקטנת יותר כי העין שלנו פחות רגישה בתחום הצבע ויותר בתחום ה-Intensity.

סביר את השלבים הבאים ברורים בדחיסת ה-JPEG: הקוונטיזציה וה-Zig – Zag Recording.

קוונטיזציה

אחד הפרמטרים בדחיסת JPEG הוא אחוז הדחיסה - האחוז הזה מתייחס לקוונטיזציה. ככל שהאחוז יותר נמוך, הקוונטיזציה יותר חזקה. הקוונטיזציה זו תהיה למספרים שלמים (לדוגמה 1.1 יופיע ל-1).

הקוונטיזציה מתבצעת על הפלט של ה-DCT כך שלא נרגיש את הקוונטיזציה בתמונה הסופית. זו הרי כל המטריה של הטרנספורמציה בתחילת הדחיסה: גם ליעיל את הקוונטיזציה וגם לדאוג שהקוונטיזציה לא תפריע בתמונה הסופית.

בהתאם לפרמטר הדחיסה שבחרנו, תיווצר עבורנו טבלה של מספרים. מה נעשה עם הטבלה זו? ניקח כל ערך שקיבלו מהתוצאה של ה-DCT, את הערך הזה נחלק ונכפיל חלוקה של מספרים שלמים בערך במקום המתאים לו בטבלה. המשמעות של זה היא שנקבל קוונטיזציה לכפולות של הערך בטבלה.

לדוגמה, אם במקומות (2,2) קיבלנו ב-DCT 100, ובטבלה במקומות (2,2) היה 11, אז נחלק את 100 ב-11 בלי שארית, ונקבל 9, את זה נכפול ב-11 ונקבל 99. כך כל מספר יעוגל כלפי מטה לכפולה הקרויה ביותר של 11: 11, 22, 33,

כל שהמספר בטבלה יותר גדול, יהיו פחות ערכים אפשריים (יותר גס).

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Better quality (and less compression) with smaller numbers

איור 158: טבלה קוונטיזציה ב-JPEG. כל ערך ב-DCT מחולק (בלוי שארית) ומוכפל בערך המתאים לו בטבלה כדי לקבל קוונטיזציה. האзор הכהול מצין איפה אנחנו עדיין משתמשים בערכים יחסית נמוכים. ככל שהאזור הכהול יותר גדול, פרמטר הדחיסה יותר גבואה והקוונטיזציה קטנה יותר. בטבלות הללו "האזור הכהול" תמיד יתחייב המהדרים הנמוכים ביותר, להם נרצה לעשות קוונטיזציה صغيرة גסה.

הערה JPEG נראה חci גרוע כשייש קפיצות גדולות: תדרים גבוהים. אם נסתכל על אות ב-JPEG בזוכחת מגדלת, נראה שם אפקט של גלים.

Zig – Zag Recording

נרצה לסדר את הערכים בשורה אחת (החלון 8×8 הופך לשורה אחת חד מימדיות 64×1 !) ונעשה זאת זה באמצעות זיג-זג. כפי שניתן לראות באיור של השלבים בדחיסת ה-JPEG. כך תדרים נמוכים ישארו ליד תדרים נמוכים וגובהים ליד גובהים.

ככה גם אפשר להחליט לאפס את כל התדרים הגבוהים החל מתדר מסוים ולשדר את תחילתית הוקטור שנשארה.

הערה יש שיטה בשם JPEG 2000 שמנסה לעשות טרנספורמציה גלובאלית (בלי חלוקה לבלוקים בגודל 8×8) באמצעות wavelet. זה גורם לכך שתמונה התוצאה נראה טוב כי היא לא מורכבת מריבועים (שבמקור נגרמו מהחלוקת לבלוקים).

שאלה למה לא משתמשים ב-2000 JPEG?

תשובה משתי סיבות:

- (i) אנשים רשמו פטנט על JPEG 2000 אך אם היו משתמשים בו היה צריך לשלם תמלוגים.
- (ii) עברו מספיק שנים כדי שלא נצרך לשלם אבל JPEG כל כך נפוץ ויש כל כך הרבה תמונות JPEG וכאן המעביר מ-JPEG 2000 הוא לא קל.

wavelets 22.3

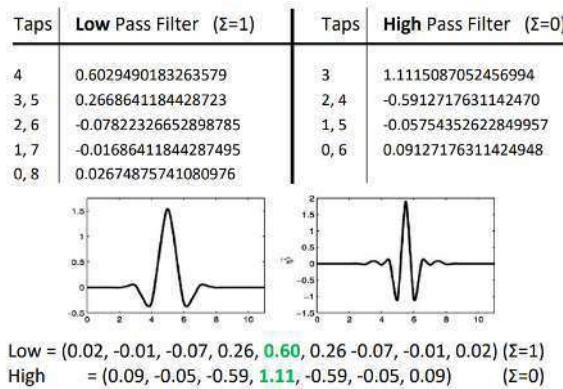
היא משפחת פונק' שבאות הזוגות: low-pass, high-pass. נזכיר כי low-pass, high-pass שנוטן לתדרים נמוכים לעבורו��-pass低-pass נוטן לתדרים הגבוהים לעבורו.

את ה-wavelets מפעילים בכל מקום על התמונה (קונבולוציה) בכל רזולוציה: כלומר בכל רמה בפרמידה שנבנה. בפרמידות רגילים שראינו עד עכšíין, high-pass low-pass שהוא הטשטוש הגאוסייני וה-high-pass שהוא ההפרש (הפלסיאן), מופעלים בו זמנית בציר ה- x וציר ה- y .

לעומת זאת, כל זוג wavelets מופעל בכל ציר לחוד. המטרה שלנו עם wavelets תהיה למצוא פונקציות high-pass שיאפשרו דחיסה טובה כניתן. אנו נרצה ליצור פרמידה שלא תופסת יותר מקום מהתמונה המקורית!

דוגמה

wavelet tap – 9/7 שנראה כך (בדחיסה משתמשים ב-wavelets שנראים כמוות):



Daubechies 9/7 – tap Wavelet Filter : איור 159

ניתן לראות בعين שהפילטר השמאלי הוא low-pass Ci נדמה שהאינטגרל שלו הוא 1, והימני הוא high-pass Ci נדמה שהאינטגרל שלו הוא 0 (אזורים שמבטלים אחד את השני).

נשים לב שהפעלת קונבולוציה עם פילטר שהוא high-pass (פילטר שהאינטגרל שלו הוא אפס) על תמונה קבועה תיתן לנו אפס. הפעלת קונבולוציה עם פילטר שהוא low-pass (פילטר שהאינטגרל שלו הוא אחד) על תמונה קבועה תיתן לנו את אותה התמונה.

הפעלת wavelet

תחילה ניקח הסיגנל לחוד ב- x ולחוד ב- y , כלומר הסיגנל חד-ממדי.

nbצע קונבולוציה של הסיגנל המקורי עם high-pass ונדגום כל פיקסל שני - נקבל תוצאה y_0 .

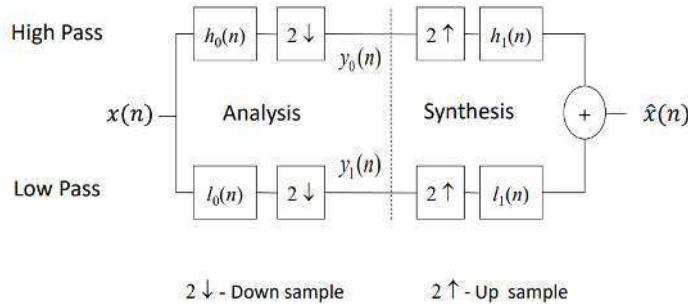
nbצע קונבולוציה של הסיגנל המקורי עם low-pass ונדגום כל פיקסל שני - נקבל תוצאה y_1 .

הערה נשים לב שבשלב זה כמות הפיקסלים לא השתנתה: אם אורך הסיגנל היה 100, אז מהקונבולוציה עם high-pass והדגימה נקבל 50 וכן עבור high-pass-low - סה"כ נותרנו עדין עם 100.

עתה ניקח את התוצאות, nbצע לכל אחת up-sample (הגדלה במקומ דגימה) ונפעיל שוב את high-pass על y_0 ואת low-pass על y_1 .

נסכום את התוצאות וזה יהיה הפלט הסופי.

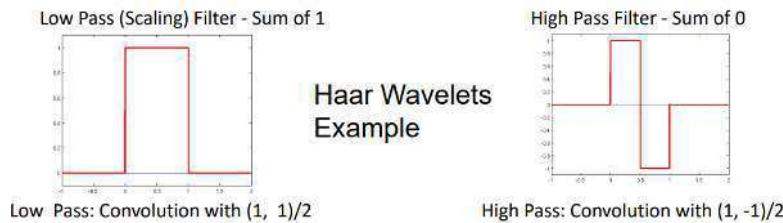
התרשימים הבא מסכם את מה שאמרנו:



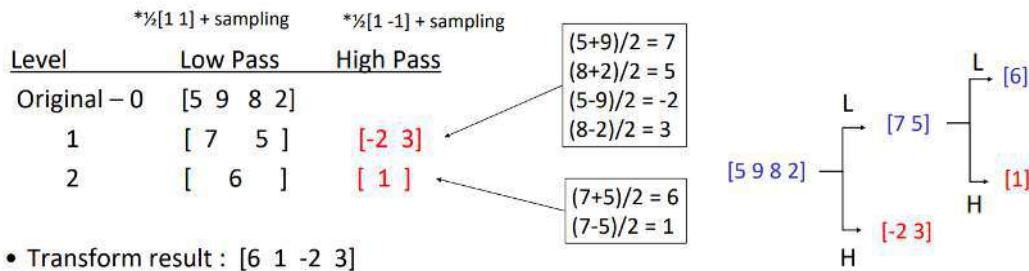
איור 160: הפעלת wavelet

דוגמאות

הו דוגמה פשוטה מאוד הנראית כך:



איור 161: Wavelets Haar

אם נפעיל את התהליך שתיארנו באמצעות $[5, 9, 8, 2]$ התוצאה שנקבל תהיה:

איור 162: הפעלת wavelets על $[5, 9, 8, 2]$. שתי התמונות מייצגות את אותו תהליך, בויזואלייזציה שונה. נבחן כי האיבר הבא בפירמידת HighPass הוא הפולט של פילטר HighPass-LowPass, ניתן לראות זאת לפי העז באיר, שכן אנו רוצים להיות מסוגלים לחשב מחדש הקודמת בפירמידה.

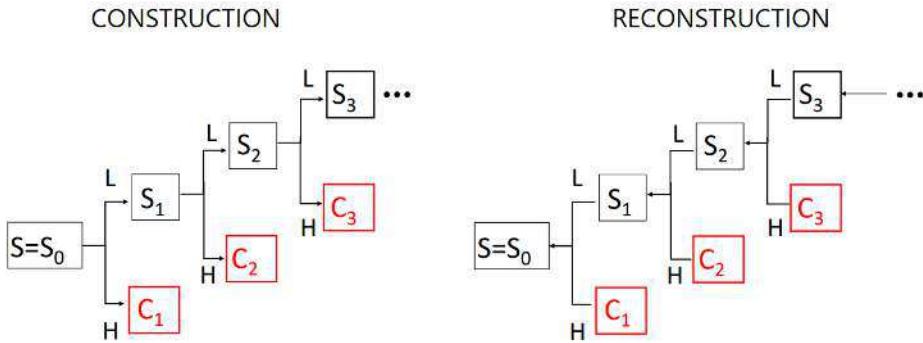
כשלמדנו על הפרמידות, הפרמידה של low-pass הייתה פרמידת גaussin וה-pass high היה פרמידת לפלייאן.

הערה בדומה לפרמידות, באמצעות הרמה האחרונה בפרמידת high-pass וכל הרמות בפרמידת high-pass ניתן לשחזור את הסיגנל המקורי.

על כן, תוצאות ה-pass high עבור הדוגמה זו תהיה $[6 \ 1 \ -2 \ 3]$ כי כפי שאמרנו עכשו זה כל המידע

שצריך כדי לשחזר את הסיגנל המקורי. באופן כללי נצטרך לשמור את האיבר האחרון בפירמידת LowPass ואת כל האיברים בפירמידת HighPass.

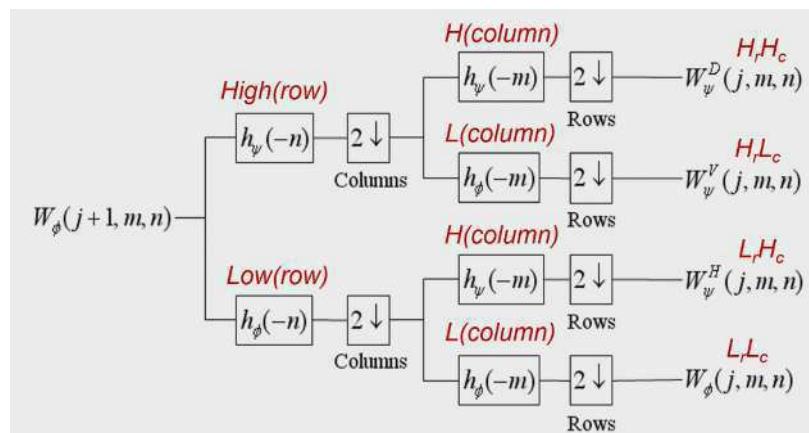
באופן כללי תהליך הבנייה וה恢חזר מותואר באיור הבא. נשים לב שמספר המקדמים שנקלט יהיה זהה לכמות המקדמים בסיג널 המקורי (כמו בדוגמה הקודמת), אך האנטרופיה קטנה.



איור 163: הפעלת wavelet ו恢חזר עבור המקרה הכללי.

תמונה בwavelet

תמונה היא אובייקט דו-מימדי, לכן בו נפעיל את ה-wavelets על השורות בנפרד ועל העמודות בנפרד (ψ מצביע high-pass ו- ϕ מצביע low-pass):

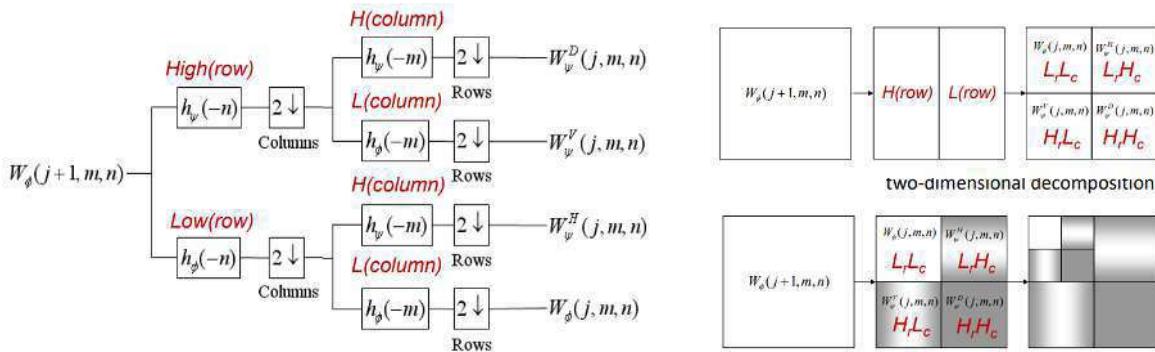


איור 164: הפעלת wavelet על תמונה

נקבל ש- $W_\phi(j+1, m, n)$ מרכיב מ-4 תוציאות של $W_\phi(j+1, m, n)$ - גודל כל אחת מהן היא רבע מהגודל של $(j+1, m, n)$ (בגלל שדגמנו פעמי אחדות בעמודות ופעמי אחדות בשורות).

הערה אם נזכיר בפירמידת גאוסיין, $W_\phi(j, m, n)$ (הוא $L_rL_c(j, m, n)$) מתאים להפעלת הטשטוש ודגימה של כל פיקסל שני בשורות והעמודות - הרמה הבאה בפירמידת גאוסיין (תמונה קטנה יותר).

את ארבעת התוצאות נשים אחת ליד השניה כדי לקבל את התוצאה הסופית.



איור 165: ייצור תמונה wavelet. לאחר שהפעלנו פילטר על השורות אפשר לדגום כל עמודה שנייה, שכן משתי הפירמידות נוכל לחתן את סכום הפיקסלים ואת ההפרש ולהסיק כל אחד מהם. כאשר נעשה פילטר על כל עמודה נוכל לדגום כל שורה שנייה באותו האופן.

נסכם את השלבים.

בכל שלב L_r מציין שהפעלנו low-pass על השורות, H_r מציין שהפעלנו low-pass על העמודות, L_c מציין שהפעלנו high-pass על השורות ו- H_c מציין high-pass על העמודות.

1. נפעיל על התמונה המקורית שני פילטרים בנפרד: פילטר low-pass על השורות ופילטר high-pass על השורות.

2. עברו כל תוצאה של הפעלת פילטר מהשלב הקודם, נדגום כל פיקסל שני בכל שורה - קיבל שתי תוצאות L_r, H_r , H_r, L_r

בתאמה (באיור: $(H(\text{row}), L(\text{row}))$).

3. ניקח את L_r, H_r וונבצע על כל אחד מהם את השלבים הבאים:

(א) נפעיל שני פילטרים בנפרד: פילטר low-pass על העמודות ופילטר high-pass על העמודות.

(ב) עברו כל תוצאה של הפעלת פילטר משלב 3 – (a), נדגום כל פיקסל שני בכל עמודה.

4. תוצאות שלב הקודם תהיה 4 תמונות (כל תמונה בגודל רביע מהתמונה המקורית):

$$L_r L_c, L_r H_c, H_r L_c, H_r H_c$$

כאשר כשאנחנו כתובים $L_r L_c$ אנו מתכוונים שקודם הפעלנו L_r ואז L_c

5. נסדר את 4 התמונות הקטנות כדי לקבל תמונה אחת בגודל המקורי:

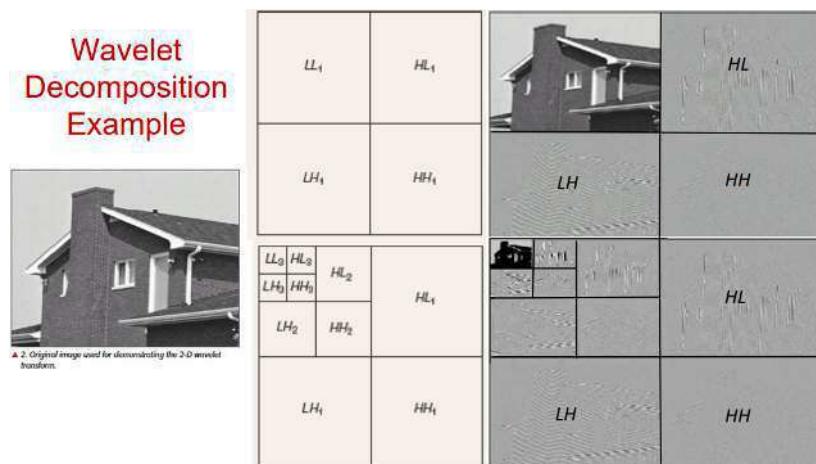
$$\begin{bmatrix} L_r L_c & L_r H_c \\ H_r L_c & H_r H_c \end{bmatrix}$$

6. זו רמה אחת בפרמידה.

(א) נוכל להמשיך לבנות עוד בפרמידה באמצעות כך שנתייחס ל- $L_r L_c$ כ"תמונה המקורית" וביצע עלייה את כל השלבים שטיירנו עכšíו.

זה בדיק כמו שבפרמידת גאוסין אנו לוקחים את $L_r L_c$ ועליו מפעלים טשטוש כדי לקבל $L_r L_c$ ברמה יותר גבוהה בפרמידה (תמונה יותר קטנה) וככה ממשיכים בתהlik.

כך דוגמה להפעלת wavelet decomposition עם 3 רמות על תמונה כלשהי:



איור 166: הפעלת wavelet decomposition עם 3 רמות על תמונה של בית

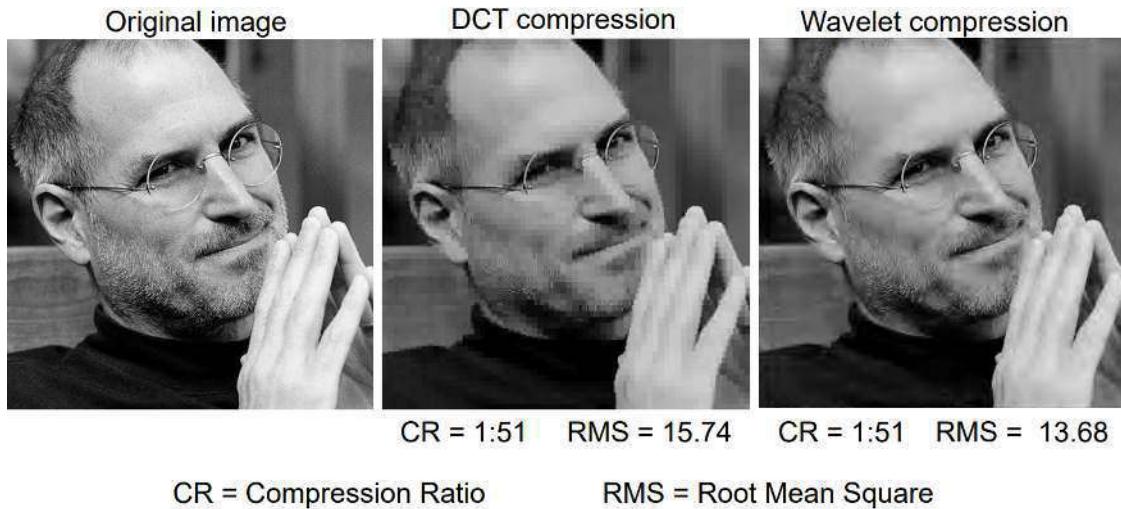
שאלה איך נוכל לאחות ש-*HL* זה high-pass בשורות ו-low-pass בעמודות?

תשובה התת-תמונה זו מורכבת מCKERים אנכיאים, לכן נבין שה-*HL* קלומר ביצעו פילטר *HighPass* על השורות ולאחר מכן פילטר *LowPass* על העמודות ואז נשארנו עם edge אנכיאים (כי הבלטו הבדלים בין שורות). פילטר *LowPass* יהיה קווים (edge) אופקיים. ב-*LH* רואים בדר"כ פינוט (מליטים הבדלים בשני הקיונים).

ב-*HH* רואים בדר"כ פינוט (מליטים הבדלים בשני הקיונים).

רוב תומנת ה-wavelet היא 0. לכן הדחיסה כאן צפואה להיות טובה. בנוסף, אין לנו כאן חלוקה לריבועים ולכן התוצאה צפואה להיות עד יותר נעימה לעין.

נזכיר שוב שקיבלנו כאן תוצאה שמצוירה פרמידה, רק שאין לנו כאן $\frac{4}{3}N^2$ ערכים, אלא רק N^2 ערכים, כמו התמונה המקורית.



איור 167: השוואה בין דחיסות שונות. שימוש לב כיצד ב-*jpeg* ו-*JPEG2000* רוגיל (DCT) רואים שהתמונה חולקה לריבועים, בניגוד ל-*wavelet* (wavelet) שנראה הרבה יותר טוב וחלק (עם שגיאת RMS נמוכה יותר עברו אותה עיילות דחיסה). כמובן ש-*wavelet* אינו מושלם ונימנע לראות שחקן מהזקן של סטיב ג'יבס נעלם בדחיסה.

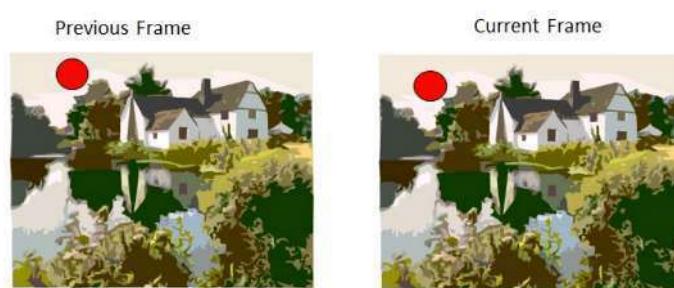
חלק XIV

דחיסת סרטונים

כשיש לנו תמונה, ניתן לנצל רק spatial redundancy - מידע מיותר "במרחב של התמונה". כך לדוגמה, עם הערך 57 הופיעו 90 פעמים ברצף, יכולנו להשתמש ב-run length ולחישום (57, 90).

בסרטון יש לנו לא רק את המקום אלא גם את הזמן: ניתן לדוחס סרטונים אף יותר תחת ההנחה שפריטים חופפים יהיו דומים.

דוגמא בשני הפריטים הבאים, שתי התמונות זהות כמעט לחלוטין. רק השימוש שינוי את מיקומה, וכך יוכל לחסוך המון מקום בדחסיה.



איור 168: כמו שנצפה שפיקסלים צמודים יהיו בעלי ערכאים דומים, כך גם נצפה שפריטים צמודים יהיו דומים

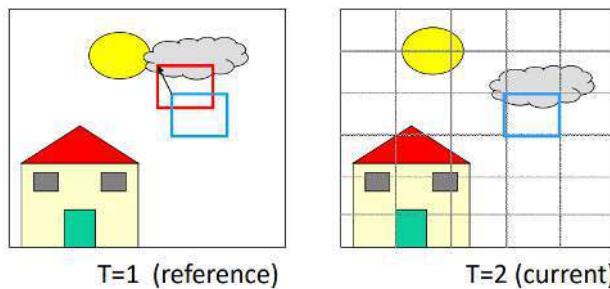
נוכל, אפוא, לנ��וט בגישה פשוטה בה נחסר כל שני פרימיטים עוקבים ונקבל תמונה הפרשיות שבתקווה יהיו מלאות ב-0 ונוחות לדחיסה ייילה.

גישה זו פחות טובה כשאובייקטים זרים, המצלמה זהה, התאורה משתנה יש רושץ חזק וכו'. מה שיפריע לנו זה בעיקר תנועה, לכן ננסה לפצות עליה באמצעות Motion Vectors.

23 וקטורי תנועה - Motion Vectors

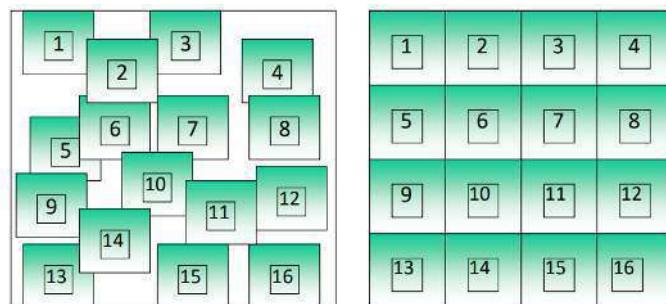
פורמט הקובץ המשמש בשיטה שנציג, של וקטורי תנועה, נקראית *MPEG*.

נחלק כל פרימיטים לבLOCKים, לדוגמה בגודל 8×8 (כי *JPEG* עובד עם 8×8). וקטור תנועה הוא וקטור המתאר את ההיסטט בין מיקום הבלוק שאנו מקודדים בפריים הנוכחי, למיקום הבלוק שמתאים ביותר בתמונה ה-reference לבלוק הנוכחי - כמו optical flow.



איור 169: וקטור התנועה מתאר את התזוזה של הריבוע הכחול לריבוע האדום. בעקבות תנועת הענן, וקטור זה אינו $\vec{0}$.

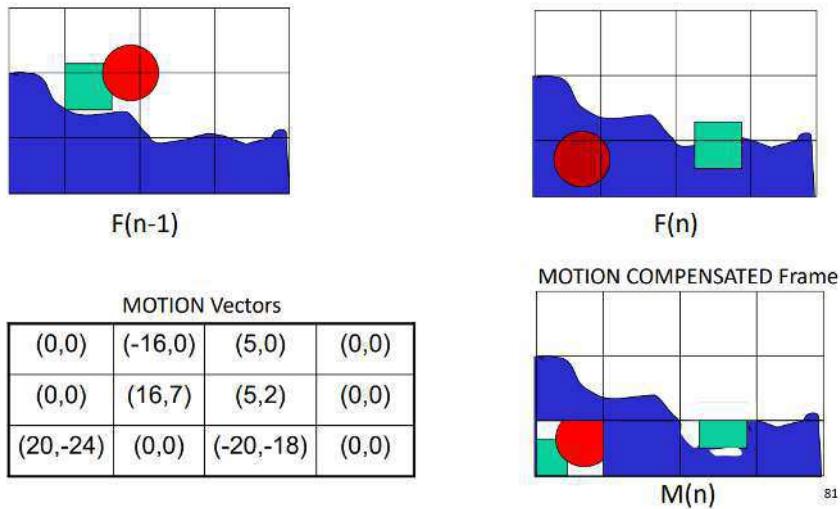
כך נוכל להשתמש בתמונה הרפרנס (יחד עם כל וקטורי התנועה) כדי לבנות את הפריים הנוכחיים.



איור 170: הפריים הנוכחיים (מימין) נבנו באמצעות תמונה הרפרנס (משמאלי).

בדרכ'נו מוגבלים את התנועה כדי שוקטורי התנועה לא יהיו גדולים מדי ונitin יהיה לקודד אותם כמעט ביטים. (**שאלת שלא הבנתי: במה זה מתבטא בפועל?**)

אם נסתכל על תוצאה של בניית תמונה מוקטורי תנועה, התוצאה יכולה להיות עם שגיאות גסות:



איור 171: בניית פריים באמצעות וקטורי תנועה. תמונה הרפרנס היא הפריים הקודם: $F(n-1)$ והתמונה הנוכחיות היא $F(n)$. ניתן לראות משמאל למטה את טבלת וקטורי התנועה, ומימין למטה את התמונה שנבנתה באמצעות הווקטורים הללו - $M(n)$ כלומר העברנו את F_{n-1} ל- F_n באמצעות *Motion Vectors*.

רוב התמונה $M(n)$ נראה טוב, למעט בלוקים בודדים בהם התוצאה לא טובה.

כדי להתחמך עם טעויות כמו שיש באյור, מלבד תמונה הרפרנס ומלבד טבלת וקטורי התמונה, נשלח גם את תמונה ההפרש בין הפריים, שנראית כך:

:MCD – Motion Compensated Diff

$$MCD(n) = F(n) - M(n)$$

כך אמםנו אנו שולחים יותר דברים, אך הדחיסה צפופה להיות יعلاה יותר:

(i) תקווותנו היא ש- MCD תכיל בעיקר אפסים.

(ii) אנו שולחים וקטור אחד עם 2 קואורדינטות לכל בלוק שמכיל 64 פיקסלים.

(iii) אנחנו נשדר יחסית מעט תמונות רפרנס, וכמה פריים ישמשו באותה תמונה רפרנס. לחופין כשאחננו נבנה את

$F(n-1), F(n)$ יהיה כבר בניו ולכן יוכל להחליט להשתמש בו כתמונה רפרנס.

סיה"כ הדחיסה מתבצע כך:

1. לכל פריים, נחשב את וקטורי התנועה לכל בלוק ואת ה- MCD .

2. לכל פריים, נדוחס את ה- MCD ואת וקטורי התנועה.

כדי לפתח את הדחיסה:

1. נחשב עבור כל פריים $M(n) = MCD(n) + M(n)$. כאשר $M(n)$ מחושב באמצעות טבלת וקטורי התנועה עם תמונה

רפרנס כלשי שנשלחה.

אם הפריים העוקבים דומים, נדרש מעט ביטים כדי לשЛОוח את וקטורי התנועה ואת ה-*MCD*.

ב-MPEG יש לנו כמה סוגי של פרייםים:

1. **פריים I** - תמונה JPEG דחוסה (Independent).
2. **פריים P** - פריים שאנו חוזים (Predicted) באמצעות וקטורי התנועה מה-*I* פריים הקודם.
3. **פריים B** - פריים דו-כיווני (Bidirectional). אם אנחנו מתרחקים מאוד מה-*I* פריים האחרון, ההפרש צפויים גדולים, שכן סיכוי טוב שניהה יותר דומים ל-*I* פריים הבא ולא קודם. לכן נדרש להסתכל קדימה ל-*I*-פריים הבא.
(א) לוקח יותר זמן לפענה פריים *B* כי צריך להעלות גם את ה-*I* פריים הבא כדי לפענה את ה-*B* פרייםים שלפניו.

הערה קורה לפעמים בטלוויזיה שפותאים הסרטון הופך כלו לריבועים ואחרי קצר זמן הכל מסתדר מחדש. הריבועים קורים כי משחו ב-MPEG נדפק, ומסתדר ברגע שמניע ה-*I*-פריים הבא.

23.1 הרחבות ל-MPEG

אמרנו שתנועה בסרטון יכולה להגרם מאובייקטים שונים ומתנווה של המצלמה. אם לדוגמה המצלמה זהה בתנועה אופקית, נוכל לתאר את התנועה באמצעות הומוגרפיה בין התמונה הנוכחית לתמונה הרפרנס. לכן, ההצעה היא לחשב את הhomography שהכי מתאימה לתמונה (לפי התהליך שלמדנו), ואז את וקטורי התנועה נחישב יחסית להומוגרפיה. התיקווה היא שכך וקטורי התנועה יהיו יותר קטנים, עם התפלגות יותר קרובה ל-0, ואז הדחיסה שלהם תהיה יותר גבוהה. כמו כן, נדרש לשדר את הhomography. נזכיר שזו מטריצה עם 8 פרמטרים (הר' האיבר הימני התחthon במטריצה הוא 1 אך לא נדרש לשדר אותו). במקומות לשדר את המטריצה עצמה, משדרים את 4 הפינוט של התמונה (הן של המקורית והן של תמונה הרפרנס), מ-4 זוגות הנקודות הללו, ניתן לשחזר את הhomography, וכן נוכל לשדר את המידע על homography בלי לשדר את המטריצה עצמה (אנחנו לא יודעים שום דבר על הערכים עצם במטריצה, ולשלוח מספר ממשי יכול להיות יקר). אם את 4 הנקודות נשלח עם דיקוק של חצי פיקסל (שוב, כדי שלא נשלח סתם מספר ממשי), בערכים שלמים, סביר שבכל התמונה הדיקוק שלנו יהיה של חצי פיקסל (הhomography לא משתוללת יותר מדי).

תמונהות בינהיות

תמונות בינהיות הן תמונות בעלי שני ערכים בלבד: לדוגמה 0/1 או 0/255. כמובן, או שהפיקסל מסומן או שהוא לא מסומן.

יש כמה דרכים לייצג תמונות בינהיות:

- מטריצה (כמו תמונה רגילה).

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

- רשימה של פיקסלים (או של הפיקסלים המסומנים או של הלא-מסומנים)

$$\{(3, 3), (3, 4), (4, 3), (4, 4)\}$$

הטיפול בתמונות בינהיות נקרא גאומטריה דיסקרטית.

23.1.1 שימושים

בתמונות בינהיות ניתן להשתמש בסגמנטציה: כ Schnitzel לצביע אזורים מסוימים בצבע אחד. לדוגמה, נרצה לסמן את כל הפירות בתמונה.

שימוש נוסף הוא Half-toning: זהה שיטה משתמשים בה במקומות כמו הדפסה, בה אנו משתמשים בשחור ולבן בלבד כדי לייצג גווני אפור.

יש שתי שיטות עיקריות ל-Half-toning:

1. - התמונה מחולקת לסדריג קבוע (בכל התמונות יהיה אותה סריג). בכל תא בסדריג נשים עיגול בגודל משותנה בהתאם לצורך.

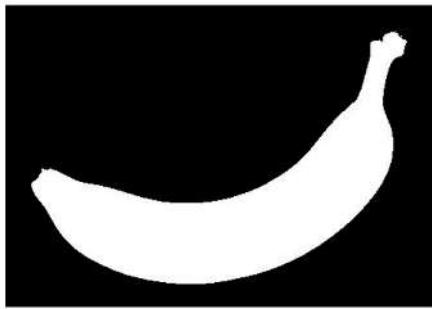
(א) עוצמת האור מגיעה מגודל הנקודה בכל תא. אורך כהה יותר יכול נקודות גדולות יותר.

(ב) משמש בהדפסות ספריים.

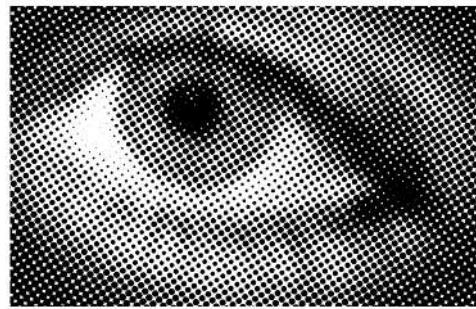
2. - הפעם הנקודות בהן נשימוש יהיו בגודל קבוע, ובמקום הגודל מה שישתנה זה המיקום: כבר לא נשימוש בסדריג בגודל קבוע ונוכל לשים נקודות איפה שנרצה.

(א) עוצמת האור מגיעה מהתפלגות הנקודות. אזורים כהים יותר יהיו צפופים יותר (יותר נקודות שחורות).

(ב) משמש במדפסות Inkjet



Segmentation
Each region different color

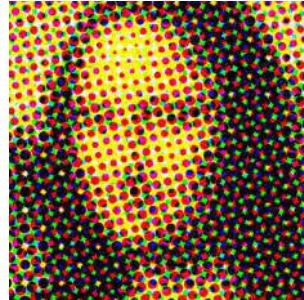


Printing: Half-Tone
Represent grey with B/W

אייר 172: שימושים לתמונות בינהיות

יש אפשרות ל-Half-toning צבעוני. בהדפסות אנו משתמשים בעורçi הצבע CMYK لكن נוכל בשיטת Screening ליצור תמונה לכל ערך, אז לשים את התווים זה על זה (דוגמה באור הבא).

הבעיה עם זה היא שברגע שאנו שמים סריגים זה על זה נוצר לנו aliasing.



אייר 173: אפקט moire. לא נפרט על התופעה אך יש דרכי להימנע ממנה.
[להلن קישור לסרטון חסר פואנטה של Tom Scott על שימושים לתופעה:](https://www.youtube.com/watch?v=d99_30swtM)

שאלה איך נוכל ליצור תמונות בינהיות?

תשובה יש כמה שיטות:

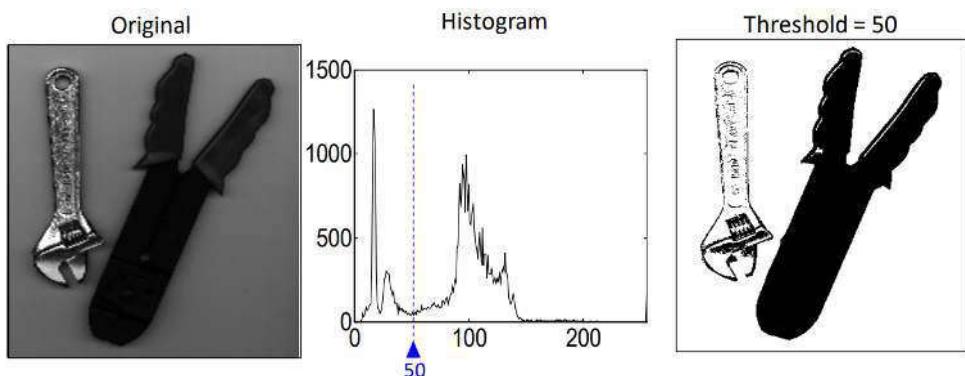
(?) בחירה עמידה: נוכל ליצור היסטוגרמה לתמונה ולבחר מקום במרכז ההיסטוגרמה שהערך בו נמור: "עומק".

עומק זהה ניקח threshold.

למה לבחר דוקא עומק? כי אין שם הרבה פיקסלים ולכן אם נזיז קצת את ה-threshold לא נשנה את התוצאה יותר מדי - זה מקום יציב. אולם קשה לבחור לבדוק את העומק (איזה עומק? כמה עמוק עליי להיות? כמה רחב? וכו')
בדוגמה הבאה ניתן לראות שהשיטה זו עובדת טוב על הכליל הימני, אך לא על הכליל השמאלי, עבورو יש אזורים בהירים מאוד וכחולים מאוד ביחס לרקע, כך שאף סף לא יעבד עבورو.

(ii) שימוש בגרדיאנטים: הנזורת קיבל ערכים גבוהים בשפות של הגוף, וערכים נמוכים במקומות חלקיים. על כן, נוכל להשתמש בממוצע הנזורות כערך הסף שלנו.

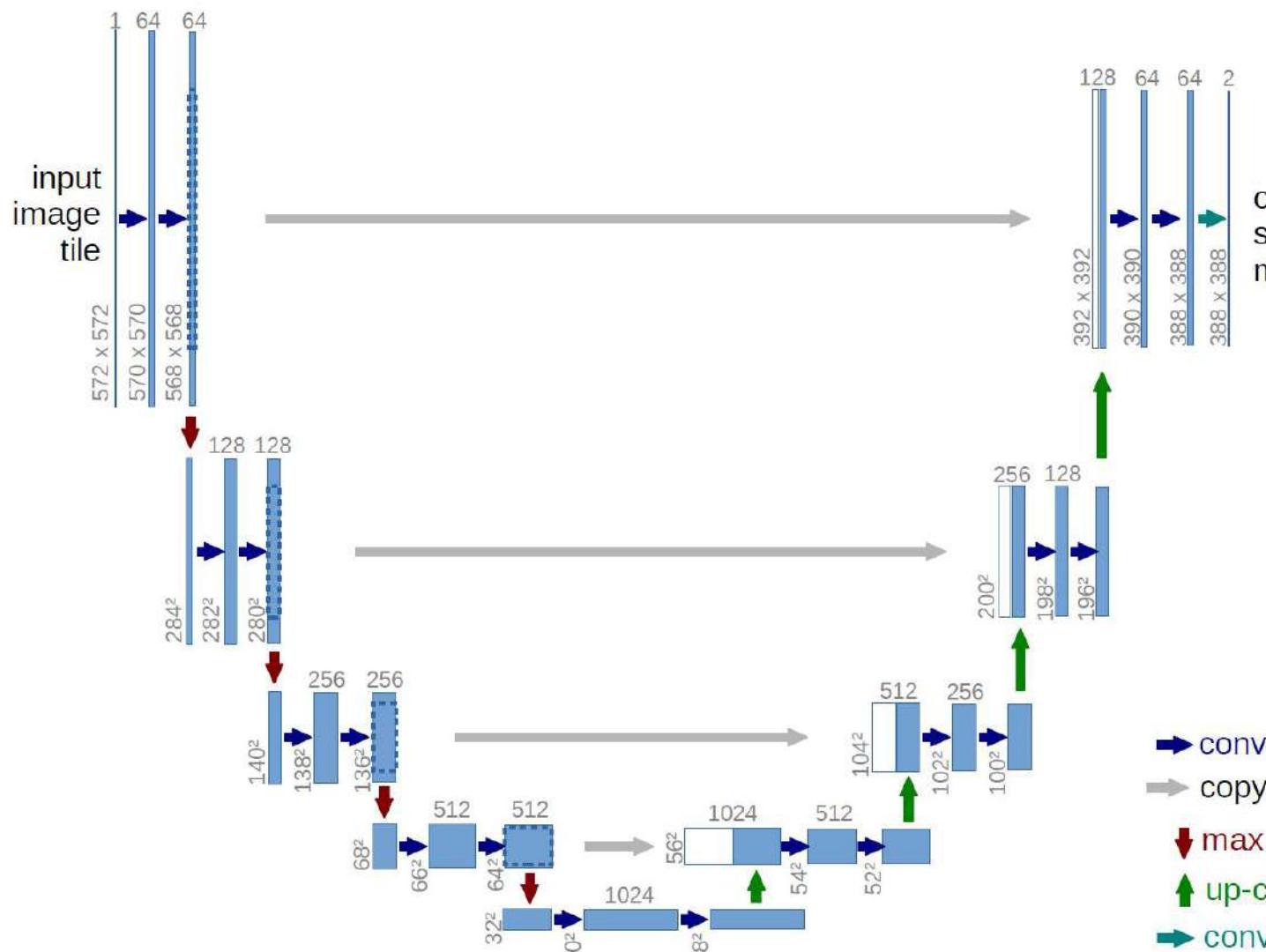
- נסה כמה ערכי סף שונים ונבחר את הסף שהכי מותאים לשפות בתמונה. Threshold Scanning (iii)



איור 174: ייצרת תמונה בינהית בשיטת thresholding עם בחירת עمق. התוצאה סבירה עבור הכליל הראשון אך לא טובה עבור הכליל השני.

דרך נוספת לבצע סגמנטציה היא באמצעות רשותות נירונום (דיברנו על כך בפרק על רשותות). ניתן לאמנו רשותות כך שלא יבצעו סגמנטציה רק למחלקות שונות, אלא גם כל מופיע של המחלקה תיבצע במצב אחר. קרי, אם יש לנו שתי פירות בתמונה, כל פירה תיבצע במצב אחר. בדוגמה זו המחלקה היא "פרה" ושני המופעים הם "פרא א'" ו-"פרא ב'".

אחת הארכיטקטורות בהן משתמשים לצורך משימה זו נקראת Net-U. רשת זו מבצעת קונבולוציות שמקטינוט את התמונה המקורית יותר ויותר - רזולוציה קטנה יותר אך עם יותר ערוצים - ואז החל משלב מסוים היא מתחילה להגדיל את התוצאה עד שהיא חוזרת לתמונה בגודל המקורי. כשדיברנו על רשותות שקטנות ואז גודלות קראנו Autoencoders. כאן יש גם קשר ישיר בין הרמות (בחצים האפורים האופקיים באירוע).

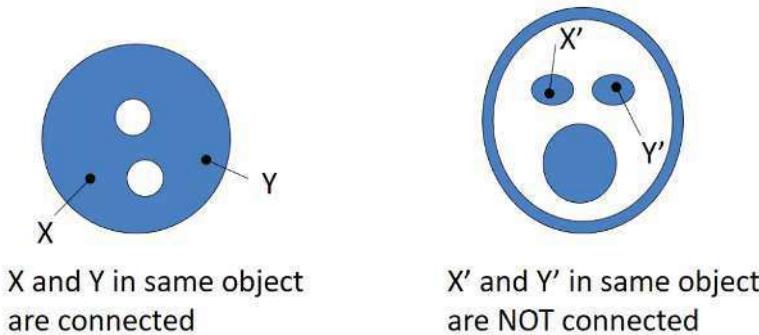


איור 175: ארכיטקטורת Net-U

24 קשירות - Connectivity

הגדרה אם שני פיקסלים y, x נמצאים באותו אובייקט, הם נקראים מחוברים/קשירים. במקרה זה נסמן, $y \approx x$.

שים לב, אמרנו שבתמונה בינהרית יש פיקסלים מסומנים ופיקסלים לא מסומנים. הקשירות היא רק בין פיקסלים מסומנים, פיקסלים לא מסומנים (גמ אם הם "באותו אובייקט") לא יחשבו כמחוברים.



איור 176: מימין: X, Y נמצאים באותו גוף כחול, אך הם מחוברים. משמאלי: X', Y' נמצאים בשני גופים שונים ו�קן אינם מחוברים.

את ההגדרה זו נצורך לתאר באופן פורמלית, בסביבה דיסקרטית של פיקסלים: אם x, y מחוברים יש מסלול שמחבר ביניהם בתוך האובייקט, אך איך יראה מסלול זהה בתוך פיקסלים דיסקרטיים? התשובה היא שהיא תהיה מסלול שכנות.

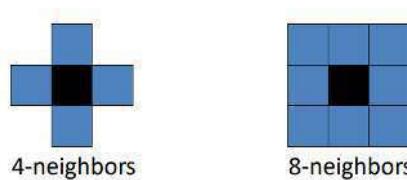
הגדרה הפיקסל (x, y) הוא 4-שכן של (x', y') אם מתקיימים אחד מה הבאים:

$$y = y' \text{ וגם } |x - x'| \leq 1 \quad (i)$$

$$x = x' \text{ וגם } |y - y'| \leq 1 \quad (ii)$$

. **הגדרה הפיקסל** (x, y) הוא 8-שכן של (x', y') אם $|x - x'| \leq 1$ או $|y - y'| \leq 1$

הערה 4-שכנים חולקים צלע, ו-8-שכנים חולקים או צלע או קודקוד.



איור 177: פיקסלים 8-שכנים ו-4-שכנים.

תמונה יחס הקשרות בין שני פיקסלים, $y \approx x$ מקיים את התכונות הבאות:

(i) רפלקסיביות - $\forall x, x \approx x$

(ii) סימטריות - $x \approx y \Rightarrow y \approx x$

(iii) טרנזיטיביות - $x \approx y, y \approx z \Rightarrow x \approx z$

כלומר, יחס הקשרות הוא יחס שכילות.

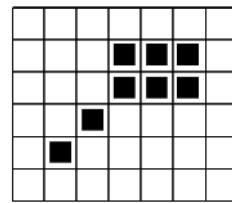
הגדרה קבוצת פיקסלים S נקראת רכיב קשריות אם $(x_1, y_1), (x_2, y_2) \in S$ \forall יש מסלול ביניהם כך שכל שני פיקסלים

עוקבים במסלול, $(x'_1, y'_1), (x'_2, y'_2)$, מקיימים:

- $(x'_1, y'_1), (x'_2, y'_2) \in S$ (i)
 $(x'_1, y'_1), (x'_2, y'_2)$ הם 4- שכנים. (ii)

שאלה לא צריך לדרוש מקסימליות או משווה כדי לוודא שאנו לא לוקחים תת-קובוצה של רכיב קשריות מלא?

הערה ההגדירה הקודמת מכילה למעשה מעשה שתי הגדרות: אחת עבור 4-שכנים ואחת עבור 8-שכנים.



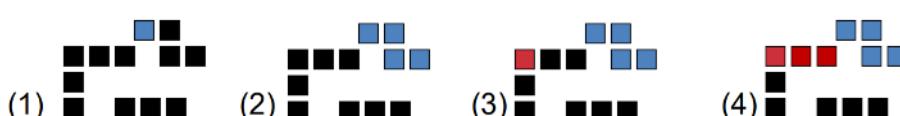
איור 178: עבור 8-קשריות, יש לנו כאן רכיב קשריות אחד גדול. עבור 4-קשריות, יש לנו כאן 3 רכיבי קשריות.

בשביל למצוא רכיבי קשריות נוכל לעבוד לפי האלגוריתם הבא:

Algorithm 23 One Pass Scanning For Connected Components

- 1 : **Scan** the image until reaching first uncolored pixel, and set his color.
 - 2 : **Continue Scan**, color all 4/8-neighbors of colored pixels.
 - 3 : **Start** a new color on an uncolored pixel that is not a neighbor of an existing color
 - 4 : **Note** two colors as equivalent when becoming neighbors
 - 5 : **Mark** all equivalent colors as one component (union-find)
-

כלומר, אנו עוברים על הפיקסלים וברגע שמנגעים לפיקסל חסר שכנים צובעים אותו בצבע חדש. כך מתקדמים וצובעים את השכניםים שלו באותו צבע. אם אנחנו מוצאים שיש לפיקסל בצבע אחד שכן בצבע אחר, אנו קובעים שני הצבעים שלהם זרים. ייזואלית נראה זאת כך:



איור 179: המראה לריצת האלגוריתם

הערה בסוף ריצת האלגוריתם כל רכיב יצביע בצבע אחר. נוכל לדעת כמה רכיבים יש ומיהו כל רכיב. שימוש לב שוחץ כאן אלגוריתם אחד ל-4-שכנות ואחד ל-8-שכנות.

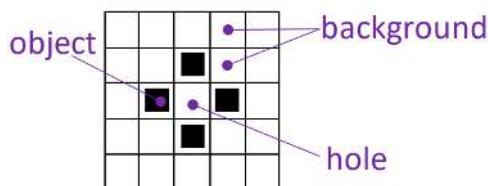
משפט (העוקם של ז'ורדן) כל עקומה סגורה מחלקת את המרחב לחלק "פנימי" וחלק "חיצוני". בנוסף, כל מסלול המחבר נקודת חלק "פנימי" לנקודת חלק "חיצוני" חותך את העקומה.

הוכחה בקורס טופולוגיה.

נדמה שהמשפט זה לא מתקיים עבור תमונות בינאריות, כדוגמה לכך נבitem באירור הבא:

ביחס ל-8-קשריות - יש אובייקט אחד ורקע גם נמצא בחלק הפנימי וגם בחלק החיצוני. אם משפט העוקם היה מתקיים, האובייקט השחור היה אמור לחסום את החור מהיה מחובר לרקע החיצוני.

ביחס ל-4-קשריות - יש 4 אובייקטים 1-2 "רקע" לבנים. אם משפט העוקם היה מתקיים, אז החור הלבן היה אמור להיות מחובר לרקע החיצוני.



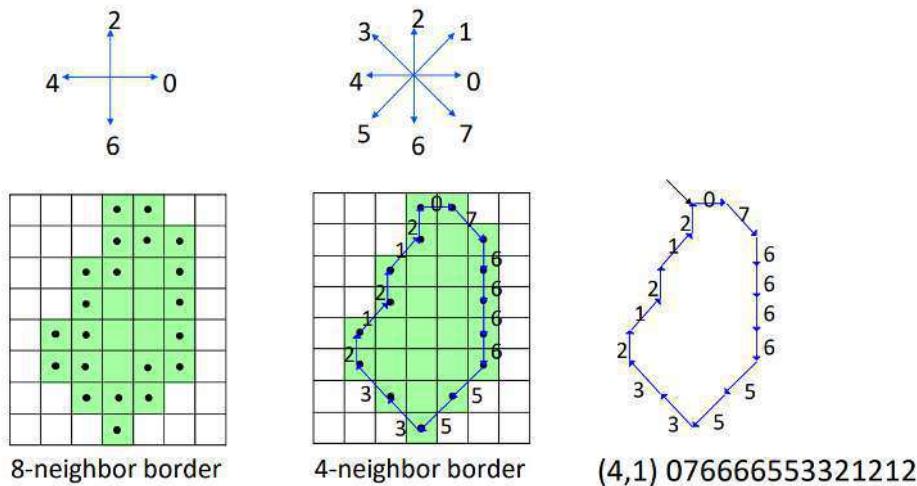
איור 180: משפט העוקם של ז'ורדן לא מתקיים עבור מקרה זה

אי-קיום המשפט עלול לגרום לביעות רבות באנליזה שלנו. על כן, כדי לפתור את הבעיה אם נדבר על 4-קשריות לאובייקטים, נשתמש ב-8-קשריות לרקע ולהפוך. שימוש שבאמצעות השינוי הזה המשפט כן מתקיים.

24.1 גבולות

הגדרה גבול הוא קבוצת כל הפיקסלים באובייקט C שיש להם שכן ברקע (פיקסל לא מסומן).
בנוגע למילה "שכן", אנו משתמשים ב-8-שכנות אם C -קשר, ומשתמשים ב-4-שכנות אם C -קשר.

תיאור גבולות (או עקומות) ניתן לבצע באמצעות Chain Codes. הרעיון בקידוד זה הוא שיכשאנו רוצים לשלווח מסלול, אפשר לשלווח רק את מיקום הפיקסל הראשון ואז את הכיוון אליו הלאנו כל פעם (כדי ליצור את המסלול של הגבול החל מאותו פיקסל). לכל כיוון נתאים מספר (בהתאם לסוג הקשרות, 4 או 8).



איור 181: למעלה: קידוד הכיוונים (כמויות הנקודות היא בהתאם לסוג הקשרות).
 למטרה: בחלק השמאלי נמצא גבול 8-שכנים, ה-chain Code שלו יתואר באמצעות 4-שכנים (כתבוב בהגדלה).
 במרכז נמצא גבול 4-שכנים ועליו מציר מסלול ה-chain Code המתואר באמצעות 8-שכנים.
 מימין למטרה נמצא הקידוד המלא, (4,1) היא קווארדיינטת התחלה (מסומנת בחז) ושאר הערכים הם הכיוונים בהם התקדםנו במסלול לפי הסדר.

הערה קוד השרשרת כפי שהוא אינוריאנטי להזזה - לא כולל נקודת התחלה. אולם, הוא אינו אינוריאנטי לסיבוב - אם נסובב את האובייקט ב- 90° קוד השרשרת נראה אחרת.

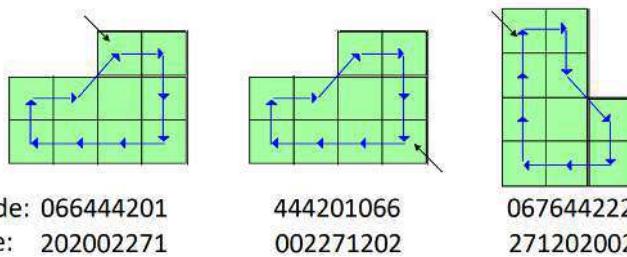
שאלה כיצד נוכל להשיג אינוריאנטיות לנקודת התחלה?

תשובה נשים לב שלא משנה מהי נקודת התחלה, הקוד שנתקבל יהיה אותו קוד רק בהזזה. לדוגמה, 064235 יכול להפוך ל-423506. ככלומר אלה פרמוטציות אחד של השני. על כן, נוכל לחתות את הסיבוב הציקלי שנוטן את המספר הcy קטן (לדוגמה, המספר 064235 קטן יותר מ-423506).

שאלה כיצד נוכל להשיג אינוריאנטיות לסיבוב?

תשובה נdag שהכוון כל פעם יהיה ביחס לכיוון האחרון שהלכנו בו. ככלומר, אם הפעולה ההתחלתייה הייתה להתקדם למטרה ↓, ואז ביחס לתמונה התקדמתי למטרה ↓ ואז פניתי שמאלה ←, אז אני ארשום זאת כך:
 תחילת התקדמתי למטרה ↓ ואז ביחס לפעולה האחורה התקדמתי למטרה ↑ ואז ביחס לפעולה האחורה (התקדמות למעלה) פניתי ימינה →.

מתמטית דבר זה מtoaר באמצעות Curvature - ההפרש בין הקודים $8 \bmod 8$. כך, לדוגמה, 0 מציין שאנו הולכים ישר, 2 מציין שאנו הולכים ימינה 90° , 7 - שמאלה 45° וכן הלאה.



איור 182: אינוריאנטיות לסיבוב מושגת באמצעות שימוש ב-*Curvature*. כל מספר ב-*Curvature* הוא חישור $8 \bmod 8$ של שני מספרים עוקבים בקוד המקורי (כאשר המספר האחרון הוא חישור הקוד המקורי עם הראשון בקוד המקורי).

שאלות ותשובות

שאלה מה היתרונו של Chain Codes על פני *raster*?

תשובה יותר קומפקטי מאשר לשדר את כל הפיקסלים ב-*raster*.

שאלה בהינתן תמונה בינהרית, איך ניציר Chain Code לקצוות של האובייקטים השחורים?

תשובה מילולית, נבוד כמוה במובן - נשים את יד שמאל על הקיר ונתקדים כך (כאשר הקיר הוא הקצה). כלומר נבוד לפי האלגוריתם הבא:

(i) נעבור ב-*raster* עד שנמצא פיקסל שחור כלשהו.

(ii) נעבור על כל אחד מ-8 הכוונים (מלבד הכוון ממנו הגענו) ואם בכיוון זהה הלבן נמצא ממשאל והשחור מימין ("היד שלנו עדיין על קיר המבוקש"), נתקדים בכיוון זהה ונוסיף אותו לקוד.

(iii) נמשיך לבצע את שלב (ii) ונסיים ברגע שנחזור لنקודת ההתחלה.

שאלה בהינתן Chain Code, איך אנחנו יודעים שהלואה סגורה, בלי לצייר את הצורה?

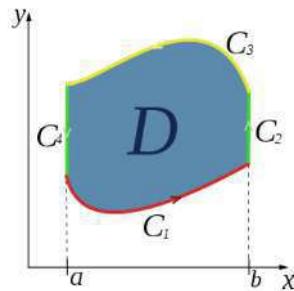
תשובה אנו יודעים עם כל צעד כמה התקדמנו ב-*x* וכמה ב-*y* (התקדמות אחרת תוסיף 1), אז אם סכום כל ההתקדימות יהיה 0 בשני היצירים, נדע שחרפנו לנקודת ההתחלה והלואה סגורה.

שאלה בהינתן Chain Code, איך נוכל לדעת מה השטח של הצורה הכלואה בו, בלי לצייר את הצורה?

תשובה השתמש במשפט גרין:

$$\text{Area} = \oint_C x dy = \boxed{- \oint_C y dx} = \frac{1}{2} \oint_C x dy - y dx$$

באופן דיסקרטי, בתמונה שלנו, נלק לאורך המסלול וכל פעם שנפנה ימינה נוסיף את ערך ה-*y* בנקודתה ובכל צעד שמאלה נחסיר את ערך ה-*y*. ניקח ערך מוחלט של התוצאה ונסיים.



איור 183: המראה למשפט גריין: השטח הכלוא בעקומה D הוא גם השטח מתחת ל- C_3 -פחות השטח מתחת ל- C_1 .

24.2 מרחקים

בhinintן שתי נקודות $P = (x, y)$, $Q = (u, v)$ בין המרחק האוקלידי $d_e(P, Q) = \sqrt{(x-u)^2 + (y-v)^2}$

$$d_e(P, Q) = \sqrt{(x-u)^2 + (y-v)^2}$$

את המרחק האוקלידי קשה לחשב וגם עבר השכנוויות שהגדכנו עדיף להשתמש במרחקים אחרים:
- "距離 מנהטן": d_4

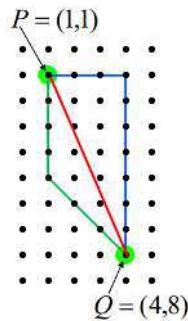
$$d_4(P, Q) = |x - u| + |y - v|$$

モותר להתקדם רק בקווים אופקיים ואנכיים, לא אלכסוניים.

- "거리 לוח השחמט": d_8

$$d_8(P, Q) = \max \{|x - u|, |y - v|\}$$

כלומר הוא בוחר את המרחק המינימלי מבין שתי הקואורדינטות.



איור 184: מטריקות שונות אפשריות. המסלול "הימני" הוא אוקלידי "והשמאלי" הוא d_8 .

למעשה, קל לראות כי

$$d_8 \leq d_e \leq d_4$$

אבל עולה השאלה, עד כמה הם שונים? נבחן כי

$$\begin{aligned} d_8 \cdot \sqrt{2} &= \max \{|x - u|, |y - v|\} \cdot \sqrt{2} \\ &\geq \sqrt{2} \cdot \frac{|x - u| + |y - v|}{2} = \frac{d_4}{\sqrt{2}} \end{aligned}$$

ובנוסף

$$\begin{aligned} d_e(P, Q) &= \sqrt{(x - u)^2 + (y - v)^2} \\ &\leq \sqrt{2 \max(|x - u|^2, |y - v|^2)} \\ &= \sqrt{2} \max(|x - u|, |y - v|) \\ &= \sqrt{2} d_8 \end{aligned}$$

ולכן מתקיים

$$\sqrt{2} \cdot d_8 \geq d_e \geq \frac{1}{\sqrt{2}} d_4$$

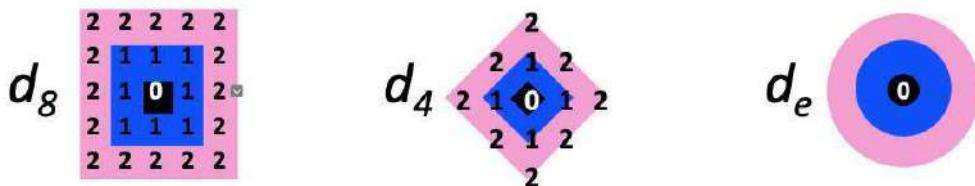
ולכן המרחקים לא מאד שונים אחד מהשני. יתר על כן, כל המרחקים הנילם הם מטריקות, כולם מקיימים את התכונות הבאות:

1. **חיוביות:** $P = Q$ אם $d(P, Q) = 0$ ו- $d(P, Q) \geq 0$

2. **סימטריות:** $d(P, Q) = d(Q, P)$

3. אי שוויון המשולש: $d(P, Q) \leq d(P, R) + d(R, Q)$

נוכל להביט על מעגל לפי המרחקים הנ"ל. d_e יתן לנו מעגל רגיל, ואילו d_8 יתן לנו פשטוט ריבוע. מה יתן d_4 ? הוא יתן לנו את כל הנקודות עם סכום מרחוקים אופקיים ואנכיים שווים, וכך נקבל מעין אלכסוני:



איור 185: הממחשה לIALIZEDים לפי המרחוקים השונים.

את המרחוקים של כל נקודה מהמרכז ניתן לראות באיוור הבא:

3	3	3	3	3	3	3	3	3	3	$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$	$\sqrt{8}$
3	2	3	3	2	2	2	2	2	3	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$	$\sqrt{5}$
3	2	1	2	3	3	2	1	1	1	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$	$\sqrt{5}$
3	2	1	0	1	2	3	3	2	1	0	1	2	3	3
3	2	1	2	3	3	2	1	1	1	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$	$\sqrt{5}$
3	2	3	3	2	2	2	2	2	3	$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$	$\sqrt{8}$
3					3	3	3	3	3	3				

City Block d_4

Chessboard d_8

Euclidean d_e

איור 186: הממחשה למרחקים. כמוון, עלינו לעגל בצורה כלשהי את הערכים כדי לקבל צורה גאומטרית הגיונית.

שאלה עולה השאלה, מדוע אנו משתמשים בפונקציות מרחק כל כך מוזרות ולא מעגלים למיטה את המרחק האוקלידי? היינו

מודיע אנו לא משתמשים במרחב $[d_e(P, Q)]$.

תשובה נביט בנקודות $R = (3, 3)$ ו- $Q = (2, 2)$, $P = (0, 0)$

$$d(P, R) = [\sqrt{9+9}] = [\sqrt{18}] = 4$$

אבל

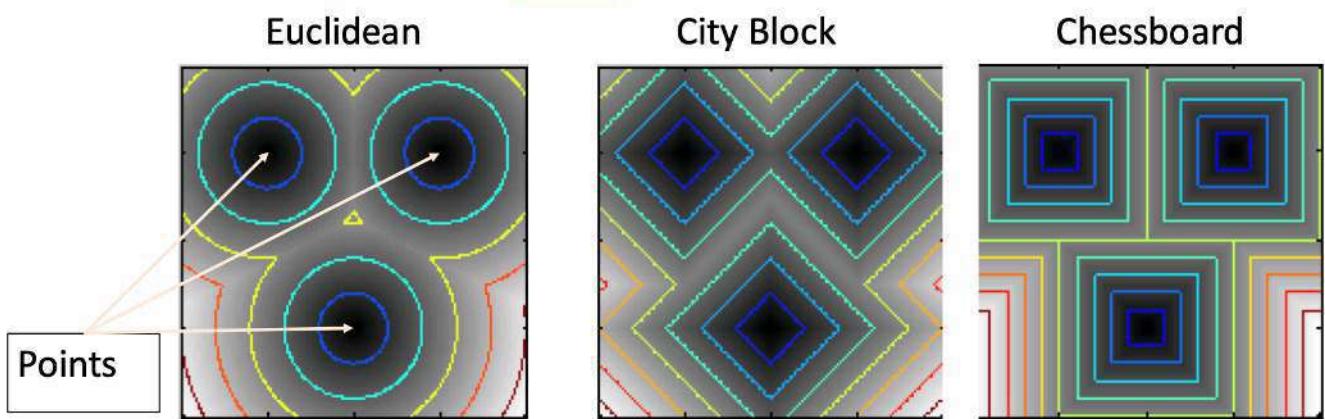
$$d(P, Q) + d(Q, R) = [\sqrt{4}] + [\sqrt{1+1}] = 2 + 1 = 3 < d(P, R)$$

אי שווון המשולש לא מתקיים ולכן זו לא מטריקה. אינטואיטיבית, הסתכלנו על טבלת המרחקים שהצנו קודם, ועיגלנו את הכל, אם נביט בצלע התחתונה ביותר (איפה שכותב 3) נבחן כי הפינות הן במרחב 4 בשונה מאשר הנקודות על המסגרת הזאת, שבמרחב 3, בשונה מהרכיבע של השט.

24.3 טרנספורמציה מרחק

בהתאם לתמונה וקבוצת נקודות אלו מגדירים את טרנספורמציה המרחק עם מטריקה d להיות תמונה חדשה, שכל פיקסל בה הוא המרחק הקצר ביותר של הפיקסל מקבוצת הנקודות, כלומר המינימום בין כל המרחקים לנקודות בקבוצה.

נציג איך זה נראה עם מטריות שונות:



איור 187: ניתן לראות את ההבדל בין הטרנספורמציה לפי המטריות

יש הרבה אלגוריתמים ייעילים לחישוב טרנספורמציה זו, עם זאת, המקרה של מטריקה אוקלידית הוא פשוט כל כך.

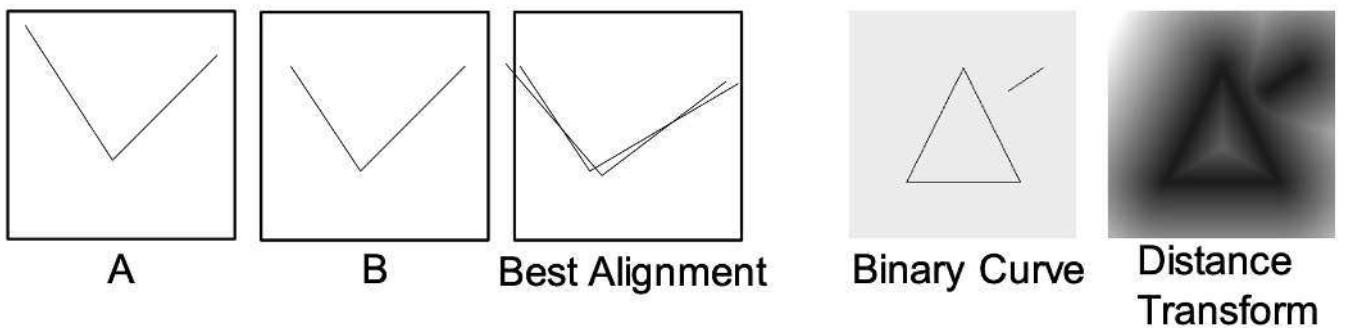
בדוגמא הבאה, נראה טרנספורמציה מרחק מקצועות התמונה:



איור 188: עולה השאלה, מה המטריקה שבחרנו? נבחן כי אם הולכים בצורה אנכית או אופקית לכיוון הקצוות, המרחק משתנה בהתאם צורה עבור כל המטריות שראינו. מה קורה כשהולכים באலכסון? אז זה לא אותו שינוי, אך כל המטריות שראינו מקיימות התנהגות זו ולכן ככל מתאימות במקרה זה.

דוגמה. התאמת תמונות באמצעות טרנספורמציה מרחק:

נניח כי נתנו לנו תמונות עם מעט נקודות התאמה:

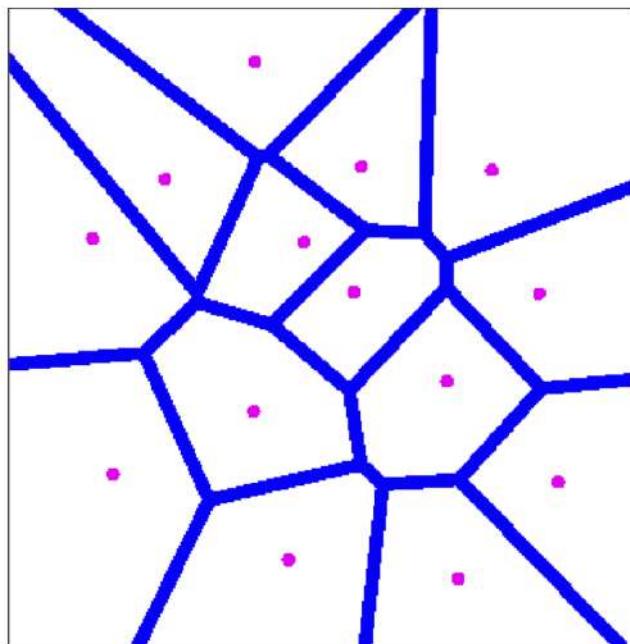


אייר 189: נקודות ההתאמה בין A, B הן הפינות. וההתאמה שנקבל היא עם מעט מאוד חיתוכים, דבר שיוביל לשגיאה דומה לכל ההתאמה אחרת, עם אותו מספר חיתוכים, שכן בעת חישוב השגיאה אנו עיוררים לצורת החיתוך. אבל, אם נחשב נקובל ההתאמה טוביה יותר ונוכל להשתמש ב-LK. למשל, עבור המשולש הימני, טרנספורמציה המרחק תיצור לנו תמונה חדשה שאוותה נוכל להתאים בצורה טוביה יותר להזאה של המשולש.

24.4 דיאגרמת Voronoi - טרייאנגולציה Delaunay

את קבוצת הנקודות שבחרנו קודם נוכל לתחום באזוריים מיוחדים. כל הנקודות שיופיעו בתחום של נקודה יהיו נקודות שהמרחב שלן ממנה הוא קטן יותר מהמרחב שלן לכל נקודה אחרת.

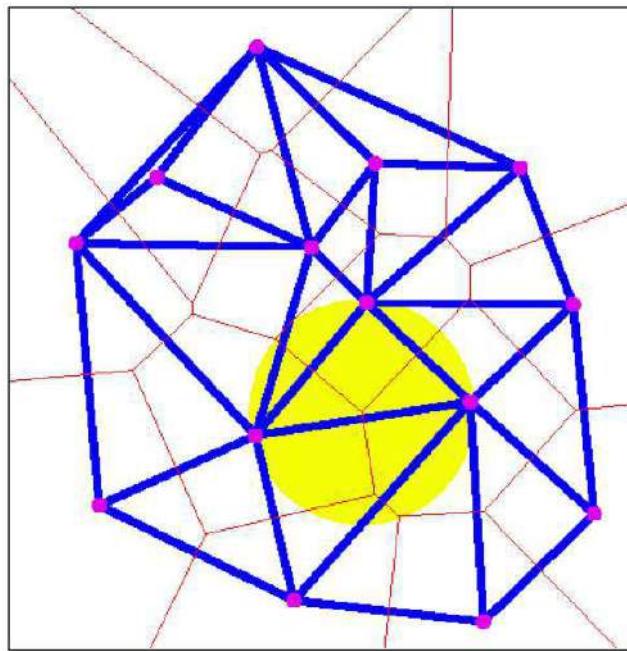
הקו המפריד בין תחומי שתי נקודות הוא למעשה האנך המרכזי העובר דרך הקו המחבר אותן.



אייר 190: הפרדת האזוריים של הנקודות היא לפי הקווים הכחולים.

את הקווים הסמכים שלא נפגשים בדיאגרמה נהוג להמשיך הלאה עד שהם נפגשים כך שמתקובלים תחומים מלאים לכל הנקודות.

נקודות מתחומים סמוכים אלו יכולים לחבר בקו ישיר וככה לקבל דיאגרמת משולשים, לתחילה זה אנו קוראים טריאנגולציה Delaunay



איור 191: טריאנגולציה על דיאגרמת Voroni

24.5 תכונות של אובייקטים בינהירים

הגדרה לכל n, m יש לאובייקט הבינהרי מומנט $\int x^n y^m b(x, y) dx dy$ כאשר $b(x, y)$ הוא 1 כאשר אנו על האובייקט ואפס אחרת.

נבחן כי עבור $0 = m = n$ מקבלים את השטח של האובייקט A

ובעבור מומנט $(1, 0), (0, 1)$ מקבלים את מרכז המסה של האובייקט כלומר

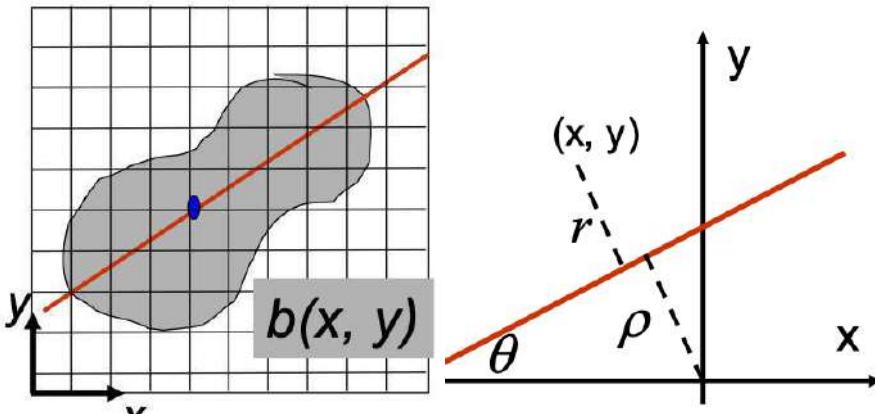
$$\bar{x} = \frac{1}{A} \sum \sum x b(x, y)$$

$$\bar{y} = \frac{1}{A} \sum \sum y b(x, y)$$

לאובייקטים בינהירים אנו מגדירים ציר מרכזי, הציר המרכזי הוא הציר שעבורו קיבל מומנט אינרציה קטן ביותר עבור האובייקט שלנו. מה מומנט האינרציה של נקודה (x, y) ביחס לציר? הוא $x^2 + y^2$. לכן הציר המרכזי הוא ציר שמזען את הערך

של מומנט האינרציה של כל הנקודות על האובייקט:

$$E = \int \int r^2 b(x, y) \, dx \, dy$$



איור 192: הציר האדום הוא הציר המרכזי של האובייקט ו- r הוא מרחקה של הנקודה (x, y) ממנו

ונכל לחשב את הציר המרכזי בצורה ישירה. הפיזיקה אומרת לנו שהוא בהכרח עובר דרך מרכז המסה (\bar{x}, \bar{y}) אותו אנו יודעים לחשב, בנוסף, עבור מומנטים מסדר שני

$$a' = \sum \sum x^2 b(x, y)$$

$$b' = 2 \sum \sum xy b(x, y)$$

$$c' = \sum \sum y^2 b(x, y)$$

$$\text{מתקיים כי } \tan 2\theta = \frac{b'}{a' - c'}$$

אחד השימושים לדבר זה הוא נमול האוריינטציה, אנו יכולים לאוחות רכיבי קשריות של אובייקטים כלומר את (y, x) , ולהציג את האובייקטים לפי זוויות הציר המרכזי שלהם. לדוגמה, אובייקטים הנראים כך

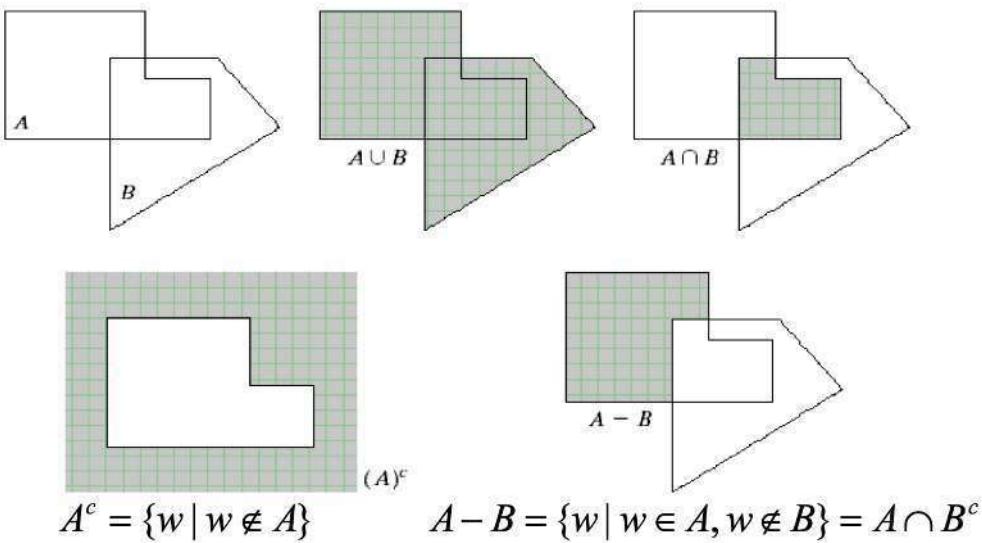


יראו עכשו כך



24.6 מורפולוגיה מתמטית

נזכיר כמה מושגים בסיסיים מהתורת הקבוצת, באמצעות דיאגרמה:



איור 193: המראה לחיתוך, איחוד, משלים וחיסור קבוצות

עבור קבוצת נקודות נוכל להגיד גם את ההזהה ב- \hat{B} על פי $z = (z_1, z_2)$
 קלומר אנו מזינים את הגוף בהיסט z .

נוכל להגיד חיתוך על ידי שיקוף פשוט קלומר $\hat{B} = \{w \mid w = -b, b \in B\}$

Structuring Element 24.7

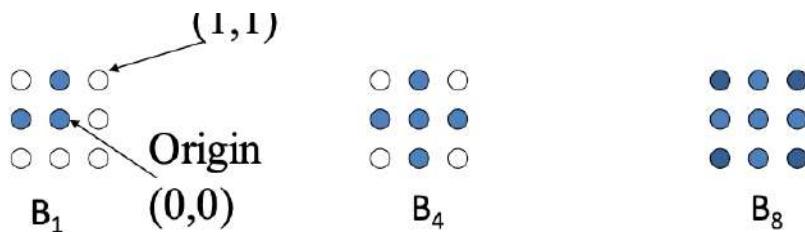
נדיר שלושה גופי מבנה:

$$B_1 = \{(0,0), (-1,0), (0,1)\}$$

$$B_2 = \{(0,0), (-1,0), (0,1), (1,0), (0,-1)\}$$

$$B_3 = \{(0,0), (-1,0), (0,1), (1,0), (0,-1), (-1,1), (1,1), (1,-1), (-1,-1)\}$$

אנו משתמש בעיקר ב- B_8 . שלושתם למשה מייצגים את האזוריים הבאים:



איור 194: נקודת המרכז היא נקודת ייחוס שביחס אליה מוגדר הגוף. אם קבענו את הנקודה להיות בנקודת $(3, 3)$ מבחרינו $(3, 3)$ היא נקודת הייחוס ולכן היא עצמה $(0, 0)$ עבורנו.

באמצעות גופים אלה ניתן להגדיר פעולות על גופים.

הרחבה - Dilation

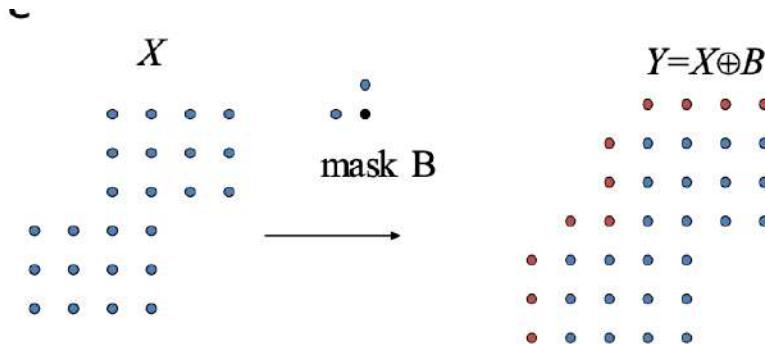
נ取 גוף מבני, למשל B_8 . עבור גוף כלשהו, לדוגמה, הגוף הכחול באירור, נגדיר את ההרחבה של הגוף באופן הבא. נסיע את הגוף המבני על הגוף, ככה שהמרכזו של הגוף המבני בתוך הגוף. ההרחבה היא איחוד כל הנקודות של הגוף המבני בהסעות אלו.

במילים אחרות, עבור גוף מבני וגוף X אנו מגדירים את ההרחבה להיות

$$Y = X \oplus B = \left\{ z \mid \underbrace{(\hat{B})}_\text{שיקוף והזאה}^z \cap X \neq \emptyset \right\}$$

או באופן שקול

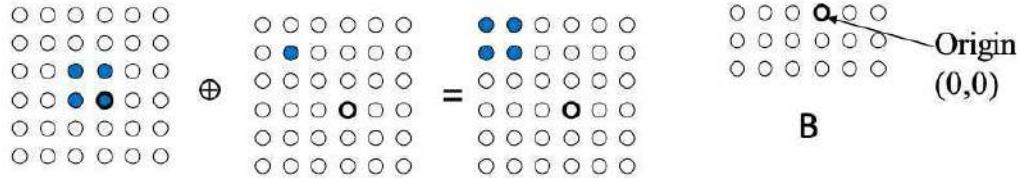
$$Y = X \oplus B = \bigcup_{b \in B} (X)_b = \bigcup_{x \in X} (B)_x = B \oplus X$$



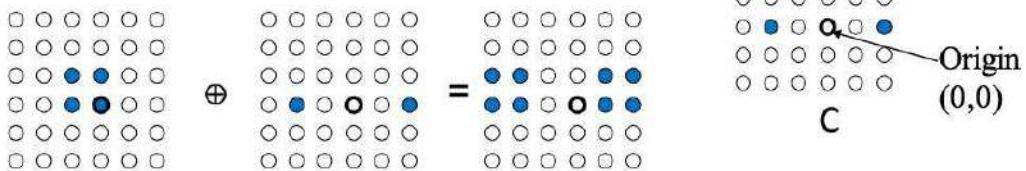
איור 195: הממחשה להרחבה פשוטה

באמצעות ההרחבה נוכל לבצע הפעולות של גופים ואף שכפולים, עם גופים מבנים שונים:

- Shift by $S \oplus B$



- Duplicate by $S \oplus C$



איור 196: באמצעות בחירת גוף מבני מתאים ניתן לקבל הצלחה ואף שכפול, כMOVEDן עליינו לבחור את נקודת הראשית בהתאם. בחירה של גופים מבנים אחרים תתן תוצאות מעניינות אחרות.

חיסור - Erosion

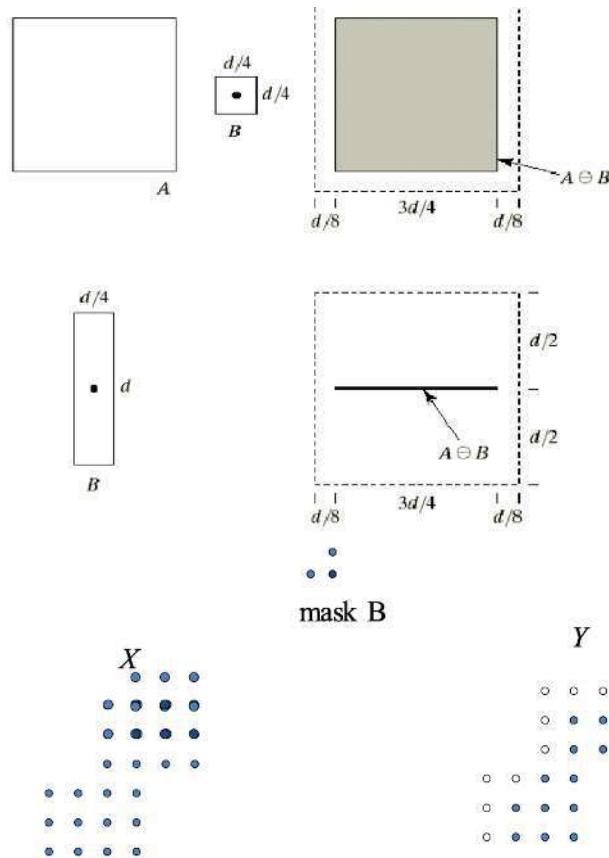
מלבד פעולה ההרחבה, אנו יכולים גם להגדיר פעולה חיסור.

עבור גוף מבני B וגוף X נסיע את נקודת המרכז של B על X ונכפיל אך ורק את הנקודות שהן B נכללו באופן מלא ב- X .

כלומר

$$Y = X \ominus B = \{x : B_x \subseteq X\} = \{x : B_x \cap X^c = \emptyset\}$$

באיור הבא ניתן לקבל המ חשה | ל פעולה:



איור 197: נסיע את המסיכה B על X ונבחר נקודות בהן B מוכל לממרי ב- Y וכן אנו מקבלים צורה קטנה יותר. בתמונה הימנית, ניתן לראות שיחסור עם ריבוע, מצמצם את המסגרת ואילו חיסור עם מלבן עם אורך כמו צורתה המקורית, משאיר לנו קו במרכז הריבוע.

פעולות פתיחה - Opening

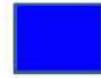
$$X \circ B = (X \ominus B) \oplus B$$

פעולות הפתיחה מוגדרת כחיסור ואז הרחבה כלומר בפיעלה זו אנו בעצם מחסרים נקודות שאין להן סביבה של B ואז מאחדים הכל חזרה, דבר זה למעשה מעלים את הנקודות ללא סביבת B ובעצם מעלים נקודות קטנות בין סביבות של B :

Example

$$\begin{array}{c}
 X = \begin{matrix} \circ & \circ & \circ & & \circ \\ \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ \end{matrix} \\
 \downarrow \ominus \\
 \begin{matrix} & & & \circ \\ & & & \circ \end{matrix} \\
 \text{mask } B = \begin{matrix} \circ & \circ & \circ \\ \circ & \circ & \circ \\ \circ & \circ & \circ \end{matrix} \\
 \downarrow \oplus \\
 X \circ B = \begin{matrix} \circ & \circ & \circ & & \circ \\ \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ \end{matrix}
 \end{array}$$

- Before:



- After:



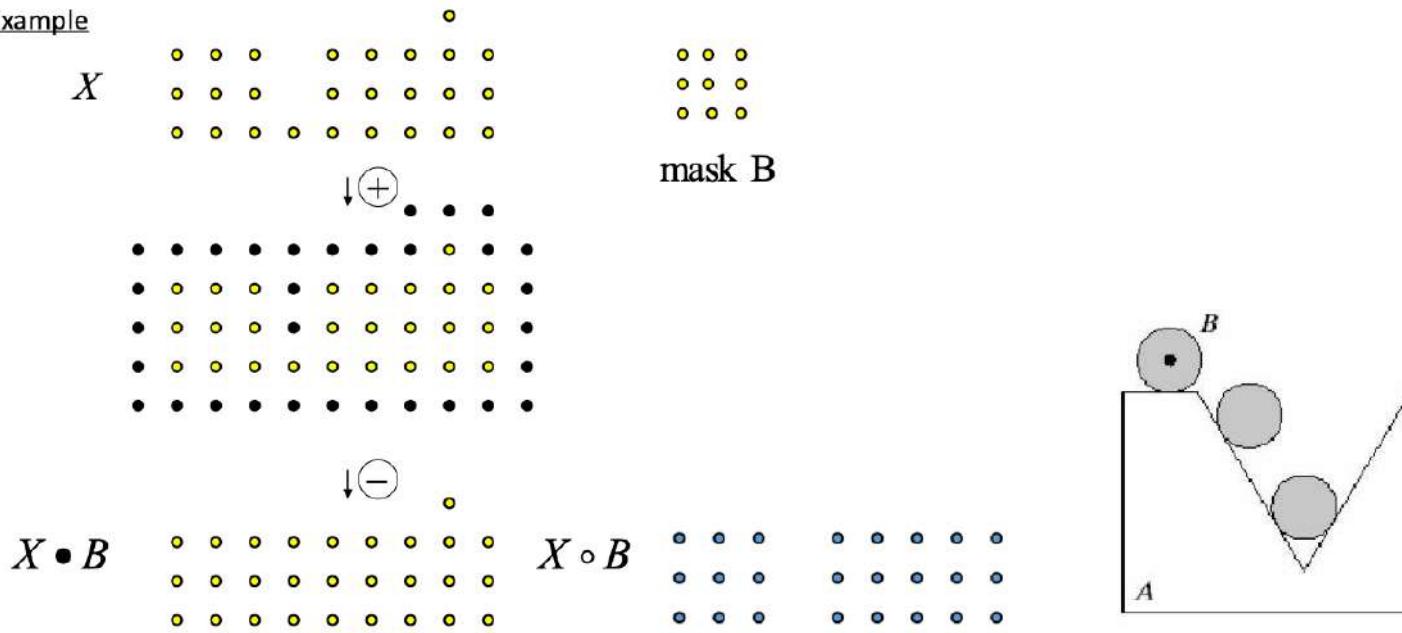
איור 198: הנקודות בין הרכיבים ומעליהם נעלמו ונשארנו עם בלוקים קשריים של נקודות. עבור התמונה הימנית: נניח כי תחילת היה לנו ריבוע כחול מלא. ניקח גוף מבני עגול עם נקודת המרכז האדומה ונבצע תהליך של חיסור. תהליך החיסור יתנו את כל הנקודות שם נשים עליהן את מרכז העיגול נשאר בתוך הריבוע הכחול. לכן אנו מקבלים ריבוע אדום המוכל בתחום הריבוע הכחול, אך קטן ממנו. לאחר מכן נרצה לבצע הרחבה, נשים את העיגולים על הריבוע האדום. אך נבחין כי לא נוכל להציג חזרה את הפינות, שכן העיגול לא מגיע אליה מהריבוע האדום ולכן הפסדנו את הפינות.

האם יתכן כי לאחר פתיחה קיבלנו יותר נקודות? לא. מותקיים כי $X \circ B \subseteq X \circ \circ X$.

פעולות סגירה - Closing

נדיר את פעולות הסגירה להיות הרחבה או חיסור, לעומת $X \bullet B = (X \oplus B) \ominus B$, כלומר

פעולה זו מספחת נקודות שלא עם סביבה B ואז מחסרת נקודות שנוטפו שאין להן סביבה, ובכך אנו מקבלים אחד של רכיבי קשריות קרובים שמה שחייב בינהם קודם הוא נקודה בודדת. במלים אחרות, אנו סוגרים את הפתחים בין הרכיבים:

Example

איור 199: ניתן לראות בתמונה השמאלית כי סגרנו את הרוחה בין שני רכיבי הקשרות כל עוד B הצליח להגיע לנקודות האלה. לבסוף לאחר החסרה, קיבלנו אותו גוף רק עם סגירת הרוחים.

בתמונה הימנית, יש לנו גוף A שמייצג מדרון וגוף מבני B שהוא מעשה עיגול. בפועל ההרחבה אנו מספחים את כל הרוחים בין רכיבי הקשרות על ידי מעבר של העיגול במדרון, ובמקומות בהם הוא לא מצליח לגעת, הוא מעשה סוגר אותם לפי קווי המותאר שלו, ולכן בתחום המדרון במקומם קיבל שפי' אנו מקבלים אורך חלק, יש לה אפקט של ניקוי רעים.

באמצעות פועלות הסגירה, כאמור, נוכל לסגור רכיבי קשרות:



איור 200: לאחר פועלות הסגירה על התמונה השמאלית, קיבלנו חיבור של חלקים הבקטריוט. אך בו זמנית קיבלנו גם איחוד של בקטריוט קרובות. דרך אגב, איך חידקים אנו רואים כאן? חידקי "איקולי".

תכונות של פתיחה וסגירה

מתכימות התכונות הבאות:

עבור הרחבה:

$$X \circ B \subseteq X$$

$$X \subseteq Y \Rightarrow X \circ B \subseteq Y \circ B$$

$$(X \circ B) \circ B = X \circ B$$

עבור סגירה:

$$X \subseteq X \bullet B$$

$$X \subseteq Y \Rightarrow X \bullet B \subseteq Y \bullet B$$

$$(X \bullet B) \bullet B = X \bullet B$$

וכן מתקיימים הקשרים הבאים בין שתי הפעולות:

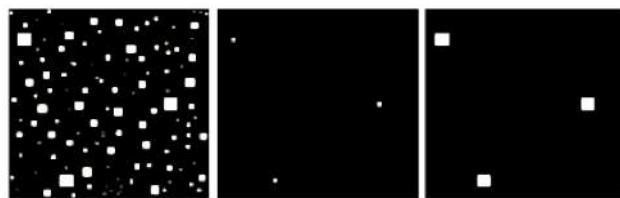
$$(X \bullet B)^C = X^C \circ \hat{B}$$

$$(X \circ B)^C = X^C \bullet \hat{B}$$

שימושים למורפולוגיה

ኖכל באמצעות פעולות אלה להעלים אובייקטים קטנים מהתמונה.

למשל עבור התמונה הבאה:



איור 201: נרצה להעלים את האובייקטים הקטנים

כיצד נעשה זאת? נרצה לשמר אז וرك על גופים בגודל שאנו חווים ווצים, אך נבצע פתיחה עם גוף מבני בגודל הגוף שנרצה לשומר, ככה כל הגוףים הקטנים מהגוף המבני יעלמו.

קיימות פעולה נוספת על הגוף שהיא פועלות שפה $(A \ominus B) = A - (A \Theta B)$ כאשר \ominus הוא החיסור של A עם הגוף המבני B ולכן $(A \ominus B)$ יתן לנו את כל הפיקסלים ב- A שיש להם שכן במשלים - אacen זו שפה.

הערה הפעולה $A \oplus B$ – תשאיר הילה ביןארית מסביב לאובייקטים ב- A (בלי האובייקטים עצם).

מורפולוגיה בדרגות אפור

מלבד זאת, נוכל גם לבחור עבור כל סביבת פיקסלים את הערך המקסימלי, או המינימלי.

עבור הפעולה

$$Y = X \oplus B = \max_{b \in B} \{X_b\}$$

אנו מקבלים שכל פיקסל מוחלף בפיקסל המקסימלי בסביבה של הגוף המבני המותאם עליו.

כלומר

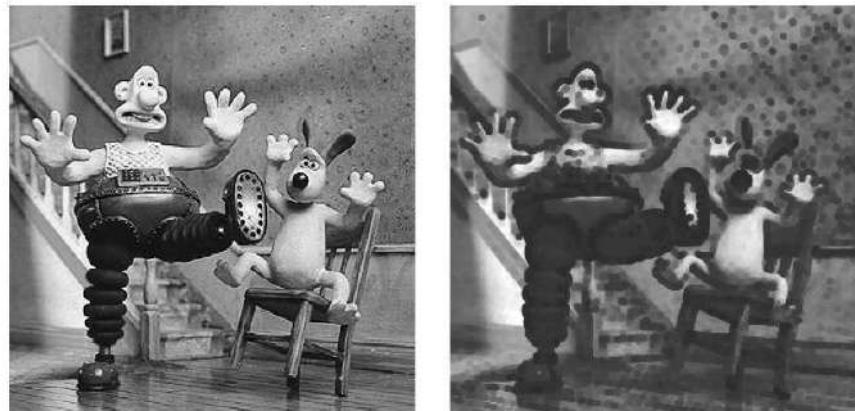
$$Y(x, y) = \max_{b \in B} \{X(x + b_x, y + b_y)\}$$

הערה X היא תמונה דרגות-אפור אбел B היא מסיכה ביןארית.



איור 202: סביבות אפורות הפכו ללבנות בגל שהכilio פיקסל לבן אחד. כאן הלבן מתרפש.

באוטו אופן נוכל לבחור את המינימום, על פי החישור כלומר $\{X_b\}$ וכאן $Y = X\Theta B = \min_{b \in B} \{X_b\}$
ולכן אנו מקבלים את הטרנספורמציה הבאה:



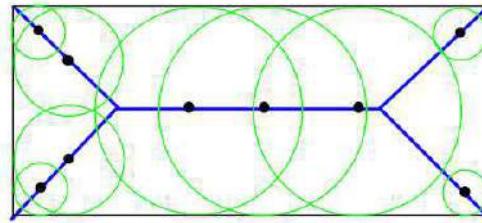
איור 203: כאן סביבות שלמות הפכו לשחורות בגל שהכilio נקודה שחורה אחת. כלומר השחור מתרפש בתמונה.

24.8 מקסימום מקומי בטרנספורמציה מרחק - Skeleton – Medial Axis

נקודת Skeleton היא נקודת מקסימום מקומי בטרנספורמציה מרחק, הסבירה יכולה להיות 8-שכנים או 4-שכנים.

ניתן לקבל המuschiese באילור הבא:

1	1	1	1	1	1	1	1	1	1	1
1	2	2	2	2	2	2	2	2	2	1
1	2	3	3	3	3	3	3	3	2	1
1	2	2	2	2	2	2	2	2	2	1
1	1	1	1	1	1	1	1	1	1	1



איור 204: באյור העליון ניתן לראות טרנספורמציה מרחק מהकצוות עם המטריקות d_e, d_4, d_8 , ראיינו שהן מתאימות לטרנספורמציה זו. האם הנקודות האדומות במרחיק 2 יכולות להיות מקסימום מקומי? متى כי? נניח שלקחנו סביבה של 8-שכנים, אז 3 הוא המקסימום ולכון במקרה זה נקבל שהתחובה היא לא. אם ניקח סביבה של 4-שכנים נקבל שככל הנקודות האדומות באյור הן מקסימום מקומי.

במרחיק אוקלידי, נקבל שמקסימום מקומי הוא מרכז העיגול הנחטס על ידי הסביבה בקווים החולמים. אם מדובר בקו האופקי החול אנו מקבלים נקודות על הקו האופקי ואם מדובר בקווים האלכסוניים אנו מקבלים נקודות עליהם, כמרכזי הבדורים החסומים הנוגעים בשתי קצוות.

נבחן כי בחירת נקודות ה-Skeleton לא משמרת קשרות!

24.9 מיציאת פינות

נבחן כי עבור B_4 אנו יכולים להחסיר פינות מאובייקט על ידי $B \circ A$ ככלומר היא תהיה תמונה ללא הפינות ולכון $A - B \circ A$ נתן לנו לבדוק את הפינות. נביט על הפינות של התמונה המכווצת ועל פינות היפוי שלה וכן הלאה...

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

$$S_1(A) = (A \Theta B) - ([A \Theta B] \circ B)$$

$$S_k(A) = (A \Theta_k B) - ([A \Theta_k B] \circ B)$$

נביט באיחוד כל הפינות הנ"ל ככלומר ב- $S(A) = \bigcup_{k=0}^K S_k(A)$. נזכיר באյור המצורף לעיל, עם נקודות השילד האדומות. תחילה אנו מקבלים את הפינות החיצונית, לאחר מכן את הפינות הפנימיות שנוצרו מהיפוי ולבסוף נקבל את הקו האמצעי האדום. כך למעשה מצאנו את כל ה-SkeletonPoints- B_4 עם $.B_4$.

חלק XVI**נושאים נוספים****25 שאלות**

1. איך מוצאים את הזוויות המרכזיות ב-SIFT הسطורמה של כל חלק או הسطורמה של הכל (א) או הسطורמה על הכל, או מוצע משקלל בין כל הזוויות.
2. מה הוא חישוב השגיאה ב-Sift- מהسطורמות?
3. איפה משתמשים ב-Wavelet ב-MOPS?
4. איחוד התמונות בגבולות מותבצע באמצעות פירמידה או באמצעות MinCut - או שניהם
 - (א) תלוי, אם ההתאמה טובה מספיק מחדדים מיד. אחרת משתמשים בפירמידה. צריך לשים לב שהאובייקט צריך להיות קטן ביחס לרוחב הפס, אחרת נקבל חתכים.

26 העשרה**26.1 חישוב התמרת פורייה דו-מימדית עם כפל מטריצות**

נתחיל בתזכורת:

$$\text{מתוקים בחד-מימד } : (\varphi = e^{-\frac{2\pi i}{N}} u) \quad F(u) = \sum_{x=0}^{N-1} f(x) e^{-\frac{2\pi i u x}{N}}$$

$$(\star) \vec{F} = \begin{pmatrix} \varphi^0 & \varphi^0 & \varphi^0 & \dots & \varphi^0 \\ \varphi^0 & \varphi^1 & \varphi^2 & \dots & \varphi^{N-1} \\ \varphi^0 & \varphi^2 & \varphi^4 & \dots & \varphi^{2(N-1)} \\ \vdots & & \ddots & & \\ \varphi^0 & \varphi^{N-1} & \varphi^{2(N-1)} & \dots & \varphi^{(N-1)^2} \end{pmatrix} \begin{pmatrix} f(0) \\ \vdots \\ f(N-1) \end{pmatrix}$$

נניח כי $f(x, y)$ היא מטריצה $M \times N$. רצ' לאורך שורה (0 עד $N - 1$). מתקיים על כן

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

כלומר

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

כאשר

$$(\star\star) F_x(v) = \sum_{y=0}^{M-1} f(x, y) e^{\frac{-2\pi i vy}{M}}$$

מכאן מתקיים כי

$$\vec{F}_x = \begin{pmatrix} \varphi_M^0 & \varphi_M^0 & \varphi_M^0 & \cdots & \varphi_M^0 \\ \varphi_M^0 & \varphi_M^1 & \varphi_M^2 & \cdots & \varphi_M^{M-1} \\ \varphi_M^0 & \varphi^2 & \varphi^4 & \cdots & \varphi_M^{2(M-1)} \\ \vdots & & \ddots & & \\ \varphi_M^0 & \varphi_M^{M-1} & \varphi_M^{2(M-1)} & \cdots & \varphi_M^{(M-1)^2} \end{pmatrix} \begin{pmatrix} f(0,0) & & f(N-1,0) \\ \vdots & \cdots & \vdots \\ f(0,M-1) & \cdots & f(N-1,M-1) \end{pmatrix} = \begin{pmatrix} F_0(0) & & F_{N-1}(0) \\ \vdots & \cdots & \vdots \\ F_0(M-1) & \cdots & F_{N-1}(M-1) \end{pmatrix}$$

כאשר $\varphi_M = e^{-\frac{2\pi i}{M}}$.

נשים המימדים במכפלה הם $(M \times M) (M \times N) (M \times M)$ שכן המכפלה תקינה ומימדי \vec{F}_x הם $M \times N$.

הסיבה לכך נכוון היא שמכפלת המטריצות $C = A \cdot B$ ב- B , ולשים את התוצאות כעומדה ב- B . יחד עם העובדת הזו, הנוסחה ב- (\star) והנוסחה ל- F_x ב- $(\star\star)$ קיבל שאנו זה נכון.

נזכיר שוב שמתקדים $F(u, v) = \sum_{x=0}^{N-1} e^{\frac{-2\pi i ux}{N}} F_x(v)$ ואת v נקבע

$$\vec{F}(u, v) = \begin{pmatrix} \varphi_N^0 & \varphi_N^0 & \varphi_N^0 & \cdots & \varphi_N^{N-1} \\ \varphi_N^0 & \varphi_N^1 & \varphi_N^2 & \cdots & \varphi_N^{2(N-1)} \\ \varphi_N^0 & \varphi_N^2 & \varphi_N^4 & \cdots & \varphi_N^{2(N-1)^2} \\ \vdots & & \ddots & & \\ \varphi_N^0 & \varphi_N^{N-1} & \varphi_N^{2(N-1)} & \cdots & \varphi_N^{(N-1)^2} \end{pmatrix} \begin{pmatrix} F_0(v) \\ \vdots \\ F_{N-1}(v) \end{pmatrix} = \begin{pmatrix} F(0, v) \\ \vdots \\ F(N-1, v) \end{pmatrix}$$

כאשר $\varphi_N = e^{-\frac{2\pi i}{N}}$

לכן בגלל אותו הגיון כמו קודם (שימוש לב גם ל-(*transpose*-�ב גם ל-

$$\vec{F} = \begin{pmatrix} \varphi_N^0 & \varphi_N^0 & \varphi_N^0 & \cdots & \varphi_N^{N-1} \\ \varphi_N^0 & \varphi_N^1 & \varphi_N^2 & \cdots & \varphi_N^{2(N-1)} \\ \varphi_N^0 & \varphi_N^2 & \varphi_N^4 & \cdots & \varphi_N^{2(N-1)} \\ \vdots & & \ddots & & \\ \varphi_N^0 & \varphi_N^{N-1} & \varphi_N^{2(N-1)} & \cdots & \varphi_N^{(N-1)^2} \end{pmatrix} \begin{pmatrix} F_0(0) & F_{N-1}(0) \\ \vdots & \vdots \\ F_0(M-1) & F_{N-1}(M-1) \end{pmatrix}^t = \begin{pmatrix} F(0,0) & F(0,M-1) \\ \vdots & \vdots \\ F(N-1,0) & F(N-1,M-1) \end{pmatrix}$$

נשים המימדים במכפלה הם $(N \times N) \cdot (N \times M)$ ($N \times M$) מכפלה תקינה ומימדי \vec{F} הם

כלומר לסטיקום נקבל שזוויות $:F(u, v)$

$$\vec{F} = \begin{pmatrix} \varphi_N^0 & \varphi_N^0 & \varphi_N^0 & \cdots & \varphi_N^{N-1} \\ \varphi_N^0 & \varphi_N^1 & \varphi_N^2 & \cdots & \varphi_N^{2(N-1)} \\ \varphi_N^0 & \varphi_N^2 & \varphi_N^4 & \cdots & \varphi_N^{2(N-1)} \\ \vdots & & \ddots & & \\ \varphi_N^0 & \varphi_N^{N-1} & \varphi_N^{2(N-1)} & \cdots & \varphi_N^{(N-1)^2} \end{pmatrix} \left(\vec{F}_x\right)^t =$$

$$\begin{pmatrix} \varphi_N^0 & \varphi_N^0 & \varphi_N^0 & \cdots & \varphi_N^{N-1} \\ \varphi_N^0 & \varphi_N^1 & \varphi_N^2 & \cdots & \varphi_N^{N-1} \\ \varphi_N^0 & \varphi_N^2 & \varphi_N^4 & \cdots & \varphi_N^{2(N-1)} \\ \vdots & & \ddots & & \\ \varphi_N^0 & \varphi_N^{N-1} & \varphi_N^{2(N-1)} & \cdots & \varphi_N^{(N-1)^2} \end{pmatrix} \begin{pmatrix} f(0,0) & f(N-1,0) \\ \vdots & \vdots \\ f(0,M-1) & f(N-1,M-1) \end{pmatrix}^t \begin{pmatrix} \varphi_M^0 & \varphi_M^0 & \varphi_M^0 & \cdots & \varphi_M^{M-1} \\ \varphi_M^0 & \varphi_M^1 & \varphi_M^2 & \cdots & \varphi_M^{M-1} \\ \varphi_M^0 & \varphi^2 & \varphi^4 & \cdots & \varphi_M^{2(M-1)} \\ \vdots & & \ddots & & \\ \varphi_M^0 & \varphi_M^{M-1} & \varphi_M^{2(M-1)} & \cdots & \varphi_M^{(M-1)^2} \end{pmatrix}$$

כדי לקבל מטריצה $N \times N$ ניקח transpose של התוצאה, או החלופין לפי חוקי transpose $(AB)^t = B^t A^t$, הגדש

יהיה:

$$\vec{F} = \begin{pmatrix} \varphi_M^0 & \varphi_M^0 & \varphi_M^0 & \cdots & \varphi_M^{M-1} \\ \varphi_M^0 & \varphi_M^1 & \varphi_M^2 & \cdots & \varphi_M^{M-1} \\ \varphi_M^0 & \varphi^2 & \varphi^4 & \cdots & \varphi_M^{2(M-1)} \\ \vdots & & \ddots & & \\ \varphi_M^0 & \varphi_M^{M-1} & \varphi_M^{2(M-1)} & \cdots & \varphi_M^{(M-1)^2} \end{pmatrix} \begin{pmatrix} f(0,0) & f(N-1,0) \\ \vdots & \vdots \\ f(0,M-1) & f(N-1,M-1) \end{pmatrix} \begin{pmatrix} \varphi_N^0 & \varphi_N^0 & \varphi_N^0 & \cdots & \varphi_N^{N-1} \\ \varphi_N^0 & \varphi_N^1 & \varphi_N^2 & \cdots & \varphi_N^{N-1} \\ \varphi_N^0 & \varphi_N^2 & \varphi_N^4 & \cdots & \varphi_N^{2(N-1)} \\ \vdots & & \ddots & & \\ \varphi_N^0 & \varphi_N^{N-1} & \varphi_N^{2(N-1)} & \cdots & \varphi_N^{(N-1)^2} \end{pmatrix}^t$$

ולכן גם עברו הטרנספורמציה הההופוכה:

$$f = \begin{pmatrix} \varphi_M^0 & \varphi_M^0 & \varphi_M^0 & \cdots & \varphi_M^{M-1} \\ \varphi_M^0 & \varphi_M^1 & \varphi_M^2 & \cdots & \varphi_M^{M-1} \\ \varphi_M^0 & \varphi^2 & \varphi^4 & \cdots & \varphi_M^{2(M-1)} \\ \vdots & & \ddots & & \\ \varphi_M^0 & \varphi_M^{M-1} & \varphi_M^{2(M-1)} & \cdots & \varphi_M^{(M-1)^2} \end{pmatrix}^{-1} \vec{F} \left(\begin{pmatrix} \varphi_N^0 & \varphi_N^0 & \varphi_N^0 & \cdots & \varphi_N^{N-1} \\ \varphi_N^0 & \varphi_N^1 & \varphi_N^2 & \cdots & \varphi_N^{N-1} \\ \varphi_N^0 & \varphi_N^2 & \varphi_N^4 & \cdots & \varphi_N^{2(N-1)} \\ \vdots & & \ddots & & \\ \varphi_N^0 & \varphi_N^{N-1} & \varphi_N^{2(N-1)} & \cdots & \varphi_N^{(N-1)^2} \end{pmatrix}^t \right)^{-1}$$

26.2 קוואורדייניות הומוגניות

נראה כיצד ניתן להרחיב את מערכת הקוואורדייניות שלנו כדי להפוך העתקה אפנית (שאינה לינארית) להעתקה לינארית. אנחנו נמצאים במרחב תלת-ממדי כלשהו (או דו-ממדי, זה לא משנה איזה). יש לנו נקודות למרחב שהם המיקומים שלנו (*positions*). אנחנו יכולים לבצע העתקה (הזהה/*displacement*) לכל נקודה. לדוגמה להזיז את הנקודה $\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$ בהעתקה של $\begin{pmatrix} 5 \\ 10 \\ 3 \end{pmatrix}$ ולקבל מיקום חדש: $\begin{pmatrix} 4 \\ 9 \\ 3 \end{pmatrix}$

כך באופן כללי, ריעונית, ניתן לבצע את הפעולות הבאות:

$$\text{position} + \text{position} = \text{displacement} .1$$

$$\text{position} + \text{displacement} = \text{position} .2$$

$$\text{displacement} + \text{displacement} = \text{displacement} .3$$

$$\text{position} + \text{position} = \text{position} + \text{position} .4$$

אובייקט נוסף שיש לנו מלבד מיקום הוא גם כיוון (*displacement*) הוא כיוון במובן מסוים). לדוגמה, בהינתן מבלן עם ארבעה קודקודים בתורת המיקומים, ו-4 נורמלים שיוצאים מהקודקודים, אם נבצע סיבוב אז גם המיקומים (הקודקודים) וגם הcyions (הנורמלים) ישתנו, אך אם נבצע רק הזהה (*translation*) רק המיקומים ישתנו.
לסיכום:

- מיקומים וכיוונים מסתובבים שניים.

- מיקומים ניתנים להזיז, אך לא כיוונים.

איך נתמודד מתמטית עם הדבר הזה? איך נציג שוקטור המציין נקודה, לדוגמה $\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$, יהיה שונה במהותו מוקטור המציין כיוון, לדוגמה $\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$?

נוסיף קוואורדיינטה חדשה, בשם "w". זו הקוואורדיינטה ההומוגנית.

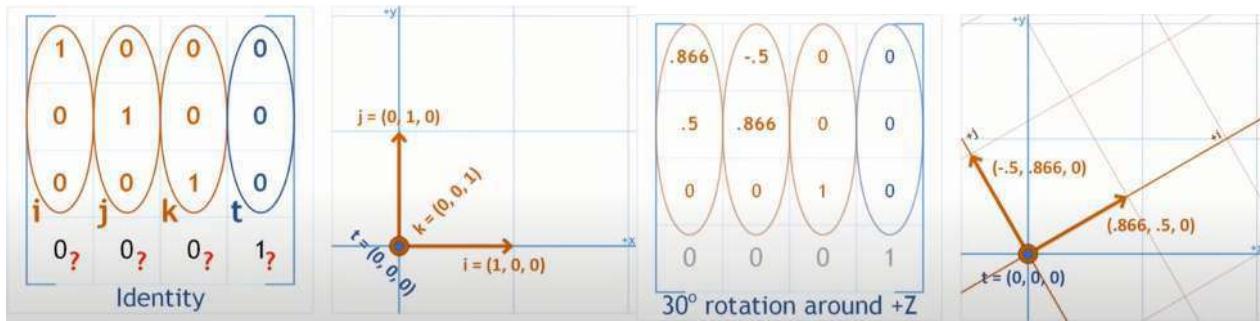
- עבור כיוונים $w = 0$

- עבור מיקומים $w = 1$

ככה אם נרצה לבצע *translation* נוכל ליצג זאת באמצעות המטריצה $\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$. המטריצה היא 4×4 כדי להתחשב ב-w כך שהמטריצה ריבועית.

שלשות העמודות הראשונות (לא כולל הקוואורדיינטה התחתונה), $\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$ מתאימות לקטורי הבסיס למרחב הוקטורוני שלו.

העמודה الأخيرة $\begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$ מתארת את הזהה, וה-1 בפינה התחתונה הוא ה-w שמתאר שאנו עובדים עם מיקומים.



אייר 205: משמאל: i, j, k הם הבסיס הסטנדרטי لكن ההעתקה לא מבצעת شيئا. t גם הוא יכול מלא באפסים שכן אין זה זהה. מימין: i, j, k הם לא הבסיס הסטנדרטי, אך הבסיס עדיין אורתונורמלי, כך שההתוצאה היא סיבוב. t מלא באפסים שכן אין זה זהה.
הערה: אם i, j, k אינם הבסיס הסטנדרטי, ו- t אינו $0, 0, 0$, לא בהכרח נוכל להציג על התנוגות אינטואטיבית פשוטה של ההעתקה. נשתדל שתמיד רק k, j, i ישתנו או רק t ישתנה (בハーצאות וברגולים נראה זהה לא תמיד המצב).

אם נרצה לבצע סיבוב של המרחב, נוכל להציג את הבסיס הסטנדרטי לבסיס כללי, i, j, k , ואז מטריצת הסיבוב תראה כך:

$$\begin{pmatrix} i_x & j_x & k_x & 0 \\ i_y & j_y & k_y & 0 \\ i_z & j_z & k_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

דוגמה סיבוב ב- 30° סביב ציר ה-z יראה כך:

$$\begin{pmatrix} 0.866 & -0.5 & 0 & 0 \\ 0.5 & 0.866 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}_{30^\circ \text{ rotation around } z} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}_{position} = \begin{pmatrix} 0.866x - 0.5y \\ 0.5x + 0.866y \\ z \\ 1 \end{pmatrix}_{position}$$

$$\begin{pmatrix} 0.866 & -0.5 & 0 & 0 \\ 0.5 & 0.866 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}_{30^\circ \text{ rotation around } z} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}_{direction} = \begin{pmatrix} 0.866x - 0.5y \\ 0.5x + 0.866y \\ z \\ 1 \end{pmatrix}_{direction}$$

אכן המיקומים נשארו מיקומיים אחרי הטרנספורמציה, והכיוונים נשארו כיוונים.

דוגמה הזהה של המיקום ב-(0.7, 0.2):

$$\begin{pmatrix} 1 & 0 & 0 & 0.7 \\ 0 & 1 & 0 & 0.2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}_{30^\circ \text{ rotation around } z} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}_{position} = \begin{pmatrix} x + 0.7 \\ y + 0.2 \\ z \\ 1 \end{pmatrix}_{position}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0.7 \\ 0 & 1 & 0 & 0.2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}_{30^\circ \text{ rotation around } z} \cdot \begin{pmatrix} x \\ y \\ z \\ 0 \end{pmatrix}_{direction} = \begin{pmatrix} x \\ y \\ z \\ 0 \end{pmatrix}_{direction}$$

ואכן כפי שאמרנו קודם, המיקומים השתנו בעקבות הטרנספורמציה, אולם הכוונים לא!

כלומר השימוש ב- w כדי שהגדינו נתן לנו את התוצאות שאנו רוצים!

נזכיר למטריצה המקורית: $\begin{pmatrix} i_x & j_x & k_x & t_x \\ i_y & j_y & k_y & t_y \\ i_z & j_z & k_z & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$.
הוקטורים i, j, k מסוימים ב- w , שכן הם כיוונים.

הוקטור t מסתומים ב- $w = 1$, שכן הוא מיקום, שזה מוזר כי הינו חושבים *sh* *translation* הוא *displacement*. איז בתוצאה שלושת העמודות הראשונות ישארו עם 0 אם נכפול את $\begin{pmatrix} 1 & 0 & 0 & t'_x \\ 0 & 1 & 0 & t'_y \\ 0 & 0 & 1 & t'_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$ במטריצת הזאת מהצורה $\begin{pmatrix} i_x & j_x & k_x & t_x \\ i_y & j_y & k_y & t_y \\ i_z & j_z & k_z & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$ בסוף, ורק העמודה האחרונה (שיש בה $1 = w$) תשתנה. לכן נוכל לומר את הדבר הבא:

$w = 0$ עבור "דברים שאמורים להיות מושפעים מ-*translation*". •

$w = 1$ עבור "דברים שכן אמורים להיות מושפעים מ-*translation*". •

נזכיר שהתחלה אמרנו שלא ניתן לחבר שני מיקומים. אכן אם נחבר שני מיקומים נקבל תוצאה עם $2 = w$, שזה לא משווה שראינו עד כה. אך אם ניקח ממוצע של שני מיקומים, אז בಗל' שאנחנו מחלקים ב- $2 = w$ ואז הכל הגיוני. ככלומר כל עוד אנחנו מחלקים את התוצאה בסקלר כדי לקבל w תקין, הפעולות האלה הן בסדר. לדוגמה, ממוצע של N מיקומים גם הוא בסדר כל עוד נחלק בסוף ב- N .

אם לדוגמה נרצה לשנות פרספקטיבה, לגרום לאובייקט להראות רחוק או קרוב יותר, אנחנו פשוט מחלקים את קוואורדיניות x, y, z (כלל שאנחנו מתרחקים מאובייקט הוא יתכנס למרכז המסלך, $(0, 0)$). לסיום, כדי לגרום לאובייקט להראות קטן יותר ויתור קרוב למרכז המסלך, משתמש במערכת צירים עם $(0, 0, 0)$ במרכז המסלך (*Clip Space*) ואנחנו מחלקים את y, z ב- $-z$

מתמטית זה יתבטא באופן הבא:

כדי לקבל מה שנקרא **פרספקטיבה "טהורה"** נdag שה- w של ציר $-z$ יהיה 1. אז

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 0 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ z \end{pmatrix}$$

כלומר השינוי היחיד שהטרנספורמציה עשתה היא שעכשיו בסוף יש לנו 1! זה נקרא *Clip Space* אם נחלק עכשיו $-z = w$. נקבל $\begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \\ \frac{z}{z} \\ 1 \end{pmatrix}$. לסיום,

ניתן לחשב על w כמו אובייקט הוא "מקורע" במרחב האמיטי, שזה אומר ש- w הוא מdad למידה של "האם אנחנו יכולים להציג את האובייקט".

מכאן ש- $0 = w$ לכיוונים, $1 = w$ למיקומים, אבל $1 = w$ למטריצת הזאת. אובייקטים בהם $1, 0 \neq w$ יש לחלק ב- w כדי לחזור "זרה למציאות".

אינטואיטיבית גם ניתן לחשב על w כאלמנט של עומק בתמונה.

26.3 משפט ריאלי ועקרון המקסימום לערכים עצמיים (בקשר של אלגוריתם האריס)

בדיברנו על אלגוריתם האריס, אמרנו שהוא של M הם אלה שמשמעותם את $(u \ v) M (u \ v)$ (תזכורת: זה היה קירוב ל- $(u \ v) M (u \ v)$) (תזכורת: זה היה קירוב ל- $(u \ v) M (u \ v)$) כאשר M היה

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

עתה נוכח שבאמת הדבר מתקיים, באמצעות כלים מילינארית 2. לשם האנלוגיה חשוב לשים לב- M היא מטריצה סימטרית!

תזכורת (הגדרה) תהי $A \in M_n(\mathbb{F})$. A תקרא צמודה לעצמה כאשר

אם A צמודה לעצמה אז:

(i) כאשר $A, \mathbb{F} = \mathbb{R}$ נקראת סימטרית.

(ii) כאשר $A, \mathbb{F} = \mathbb{C}$ נקראת הרמיטית.

תזכורת (המשפט הספקטורי מעל \mathbb{R}) יהיו $(V, \langle \cdot | \cdot \rangle)$ ממ"פ נ"ס ו- $f \in End(V)$. אזי f לכסין בבסיס אורתונ' אם"מ f צמוד לעצמו.

תזכורת (המשפט הספקטורי מעל \mathbb{C}) יהיו $(V, \langle \cdot | \cdot \rangle)$ ממ"פ נ"ס מעל \mathbb{C} ו- $f \in End(V)$. אזי f לכסין בbasis אורתונ' אם"מ f נורמלי.

תזכורת יהו $x \cdot y = \bar{x}_1 y_1 + \dots + \bar{x}_n y_n = \bar{x}^t y$, כאשר באג' שמאל יש לנו מכפלה $x \cdot y = x^t y$ סקלרית ובאג' ימ' יש כפל מטריצות. מעל \mathbb{R} יתקיים פשטוט.

הגדרה תהי $A \in M_n(\mathbb{C})$ מטריצה הרמיטית. נגדיר את שארית ריאלי להיות

טענה תהי $A \in M_n(\mathbb{C})$ מטריצה הרמיטית. אזי $\min_{v \in \mathbb{C}^n} R(v) = \min_{v \in \mathbb{C}^n: \|v\|=1} R(v)$

הערה טענה זו מסתמכת על כך שהמינימום קיים. לעומת זאת שתהא אcn קיים ובמשפט הבא נוכחים זאת ואף נראה מיהו המינימום.

בנוסף, ניסוח דומה (עם הוכחה אנלוגית) קיים עבור $\max_{v \in \mathbb{C}^n} R(v)$.

הוכחה יהיו $w \in \mathbb{C}^n$ עבורו $R(w) = \min_{v \in \mathbb{C}^n} R(v)$. נביט ב- $\frac{w}{\|w\|}$. מתקיים כי $\frac{w}{\|w\|} \cdot \frac{w}{\|w\|} = 1$. יותר על כן,

$$R(v) = R\left(\frac{w}{\|w\|}\right) = \frac{\left\langle \frac{w}{\|w\|} \mid A \frac{w}{\|w\|} \right\rangle}{\left\langle \frac{w}{\|w\|} \mid \frac{w}{\|w\|} \right\rangle} = \frac{\frac{1}{\|w\|^2} \langle w \mid Aw \rangle}{\frac{1}{\|w\|^2} \langle w \mid w \rangle} = \frac{\langle w \mid Aw \rangle}{\langle w \mid w \rangle} = R(w)$$

לכן $R(v) = E(w)$, כלומר v מינימלי. לכן, בגלל ש- $1 = \|w\|$ נסיק כי $\|v\| = 1$.

משפט (ריילי - Rayleigh) תהי $A \in M_n(\mathbb{C})$ כך ש- $A^* = A$. אז:

(i) הערכים העצמיים של A כולם ממשיים.

(ii) יהיו $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ הע"ע של A . אז

$$\begin{aligned}\lambda_1 &= \min \left\{ \frac{\langle v | Av \rangle}{\langle v | v \rangle} \mid \vec{0} \neq v \in \mathbb{C}^n \right\} \\ \lambda_n &= \max \left\{ \frac{\langle v | Av \rangle}{\langle v | v \rangle} \mid \vec{0} \neq v \in \mathbb{C}^n \right\}\end{aligned}$$

הוכחה (i) לפי המשפט הספקטורי, יש מטריצה אלכסונית D הדומה אוניטרית ל- A . כלומר, יש מטריצה אוניטרית כך $D = UAU^{-1} = UAU^*$. אז

$$D^* = (UAU^*)^* = U^{**}A^*U^* = UAU^* = D$$

כלומר לערכי האלכסון של D , שהם הע"ע של A (ככה עובד לכsoon) מתקאים $D_{ii} = \bar{D}_{ii}$, ומספר שווה לצמוד של עצמו רק אם חלקו המדומה הוא 0, דהיינו אם הוא ממשי. יתר על כן, נציין שבגלו ש- A - v לכסינה יש לה n ע"ע (שאינם לאו דווקא שונים, אך הוא המתאים להם הם בת"ל).

(ii) עבור v ו"ע של λ_1 מתקיים

$$\frac{\langle v | Av \rangle}{\langle v | v \rangle} = \frac{\langle v | \lambda_1 v \rangle}{\langle v | v \rangle} = \frac{\lambda_1 \langle v | v \rangle}{\langle v | v \rangle} = \lambda_1$$

ולכן $\lambda_1 \geq \min \left\{ \frac{\langle v | Av \rangle}{\langle v | v \rangle} \mid \vec{0} \neq v \in \mathbb{C}^n \right\}$

מצד שני, בಗלו ש- A - v לכסינה אורתוגונלית לפי המשפט הספקטורי, אז יש ל- V בסיס אורתונורמלי של ו"ע v, v_1, \dots, v_n .

יהי $v \in V$, אז לפי שיוויון בסל

$$v = \sum_{i=1}^n \langle v_i | v \rangle v_i \Rightarrow \|v\|^2 = \sum_{i=1}^n |\langle v_i | v \rangle|^2$$

ואז

$$\begin{aligned}\frac{\langle v | Av \rangle}{\langle v | v \rangle} &= \frac{\left\langle v \mid \sum_{i=1}^n \langle v_i | v \rangle Av_i \right\rangle}{\langle v | v \rangle} = \frac{\sum_{i=1}^n \langle v_i | v \rangle \lambda_i \langle v | v_i \rangle}{\sum_{i=1}^n |\langle v_i | v \rangle|^2} = \frac{\sum_{i=1}^n \langle v_i | v \rangle \lambda_i \overline{\langle v_i | v \rangle}}{\sum_{i=1}^n |\langle v_i | v \rangle|^2} \\ &= \frac{\sum_{i=1}^n |\langle v_i | v \rangle|^2 \lambda_i}{\sum_{i=1}^n |\langle v_i | v \rangle|^2} \geq \frac{\sum_{i=1}^n |\langle v_i | v \rangle|^2 \lambda_1}{\sum_{i=1}^n |\langle v_i | v \rangle|^2} = \lambda_1 \frac{\sum_{i=1}^n |\langle v_i | v \rangle|^2}{\sum_{i=1}^n |\langle v_i | v \rangle|^2} = \lambda_1\end{aligned}$$

כשהשתמשנו בכך ש- λ_i מתקיים $\lambda_1 \geq \lambda_i \forall r \in [n]$, וכן גם המינימום

מקיים זאת - וזה הא"ש הנגיד.

הוכחה עבור המקרים זהה, רק עם אי-שוויונים הפוכים.

הערה הוכחנו זה עתה כי R מקבלת מינימום ומקסימום עבור v של A , והמינימום/מקסימום הוא הע"ע הקטן/גדול ביותר של A בהתקאה.

עקרון רילי תהי $A \in M_n(\mathbb{C})$ מטריצה הרמיטית. יהיו $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ הע"ע של A . אזי λ_1, λ_n הם ערכי המינימום

$$R(v) = \frac{\langle v | Av \rangle}{\langle v | v \rangle} = \frac{\bar{v}^t A v}{\bar{v}^t v}$$

יתר על כן, הע"ע λ_1, λ_n הם אלה שננותנים את ערכי המינימום והמקסימום.

הוכחה ראיינו שמתקיים $\min_{v \in \mathbb{C}^n} R(v) = \min_{v \in \mathbb{C}^n : \|v\|=1} R(v)$ וראינו שהו"ע המתאים ל- λ_1 נותן את המינימום. לכן, לפי ההוכחה של הטענה הראשונה, הע"ע v_1 המתאים ל- λ_1 עברו $1 = \|v_1\|$ (תמיד ניתן לנормל את הוקטור) מקיים

$$\lambda_1 = \frac{\langle v_1 | Av_1 \rangle}{\langle v_1 | v_1 \rangle} = \frac{\langle v_1 | Av_1 \rangle}{\|v_1\|^2} \stackrel{\|v_1\|=1}{=} \langle v_1 | Av_1 \rangle$$

כלומר זהו המינימום של הפונק' $R_{norm}(v) = \langle v | Av \rangle = \bar{v}^t A v$ הוכחה זהה קיימת עבור \max .

הערה נשים לב שכאשר $A \in M_n(\mathbb{R})$ סימטרית, לפי המשפט הספקטורי A לכסינה בסיס אורותוני' ולכן יש לה n ממשיים $\lambda_1, \dots, \lambda_n$ ובתל' שכולם עם מקדמים ממשיים. לכן אם נסתכל על A כמטריצה מעל \mathbb{C} , יתקיים שככל המשפטים שהוכחנו תקפים עבור A .

לכן אם $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ הם הע"ע של A , אזי

$$\begin{aligned} \lambda_1 &= \min \left\{ \langle v | Av \rangle \mid \vec{0} \neq v \in \mathbb{R}^n \right\} = \min \left\{ v^t A v \mid \vec{0} \neq v \in \mathbb{R}^n \right\} \\ \lambda_n &= \max \left\{ \langle v | Av \rangle \mid \vec{0} \neq v \in \mathbb{R}^n \right\} = \max \left\{ v^t A v \mid \vec{0} \neq v \in \mathbb{R}^n \right\} \end{aligned}$$

27 סיכום בניית הפונרמיה

1. ממציאות *FeaturePoints*

(א) באמצעות *SIFT*-ב-*DogPyramids* או באמצעות *MOBS*-*Harris*

2. צירפת *Descriptors*

(א) *MOBS* - בחירת אזור 7×7 בrama השלישית בפירמידת הגאוסיין

(ב) *SIFT* - יצירה וקטור בגודל 128 של זוויות מישוריות. כל זה נוצר מסביבה בגודל 16×16 וחלוקת לאזורים

$.4 \times 4$

3. התאמת בין אזוריים

(א) *MOBS* - מרחק אוקלידי, ולקיחת התאמות עם יחס טוב בין ההתאמת הטובה ביותר להתאמת האחת פחות טובה.

(ב) *SIFT* - אותו דבר.

4. יישור התמונות ביחס למערכת קווארדינטות

(א) הפעלת הטרנספורמציה ההפוכה על כל תמונה בפונרמיה כך שנתקבל יישור ביחס לרצף שנרצתה. בדרך כלל ביחס לתמונה האמצעית.

5. איחוד התמונות

(א) אם יש הבדלי תaura אבל אין אובייקטים שזו בתמונות - נשתמש ב-*PyramidBlending*

(ב) אם אין הבדלי תaura - נשתמש ב-*minCut*.

(ג) אם יש הבדלי תaura ויש אובייקטים שזו, נשתמש בהיריסטיקות אחרות.

28 סיכום קצר LukasKanade

1. מזעור פונקציית שגיאה על ידי בדיקת הערכים s, u האופטימליים - מעבר על כל האפשרויות.
 - (א) לא יעיל לחישוב.
 - (ב) יוצר בעיה של אפס חפיפה.
2. שימוש ב-*LukasKanade* על ידי מציאת המינימום שלה פונקציה באמצעות גזירה.
 - (א) החפיפה לא תהפוך לאפס.
 - (ב) טוב להזאות קטנות.
 - (ג) מניח שהנגזרות בין נקודות מתאימות בין שתי התמונות הן זהות.
 - (ד) טוב כאשר החפיפה יחסית גדולה.
3. שימוש ב-*LK* איטרטיבי - תזואה קטנה בכל פעם על ידי שינוי הנגזרת לפי הזמן - ככה מקבלים התוכנות אבל גם הסתכלות גלובלית על כל התמונה.
 - (א) גלובי.
 - (ב) מניח שיש חפיפה כלשהי.
 - (ג) לא תמיד מתכנס.
4. שימוש ב-*LK* עם פירמידות
 - (א) לכל רמה נבצע *LK* רגיל ונמצא s, u אופטימליים. ברמה הגבוהה ביותר ההזאה מאוד קטנה וכך אפשר להשתמש ב-*LK* רגיל.
 - (ב) נכפול ב-2 ונסיט את התמונה ברמה הבאה קדימה כניחס התחלתי, ונחשב שוב את ההזאה הבאה.
 - (ג) אם רוצים לחשב גם זווית סיבוב, חייבים להניח שהיא קטנה, כי הזווית לא משתנה בرمות הפירמידה, ואנו משתמשים בקרוב טילור.
5. חישוב *opticalFlow* באמצעות *LK*.
 - (א) בונים פירמידת גאוסיאן וברמה הנמוכה ביותר מחשבים התאמות בין סביבה בתמונה אחת לסביבה אחרת - ככה לכל פיקסל בתמונה.
 - (ב) נעשה *LK* על כל סביבה ונמצא את התזואה s, u , ונdag למשך אותה על ידי דרישת חלקות (הוספה של התיקון של *LK* שראינו).
 - (ג) נעה רמה, נזע את התמונה לפי שדה התזואה שמצאנו ונמשיך הלאה.
6. עדיף על *FeaturePoints* כאשר החפיפה גדולה אבל האזורים תבניתיים - הוא גלובי.

7. בהשוואה לקרוס קורלציה - הוא יותר יעל לחישוב, אך במקרים בהם אין יותר מדי אפשרויות להזאות נועד יפ' לפעמים להשתמש בקרוס קורלציה מקסימלית.

29 סיכום קצר על פנורמות המורכבות משתי נקודות מבט וזמנים שונים

1. פנורמה אחת מחושבת כשהאור מוקן על הצד השמאלי של המצלמה.
2. פנורמה שנייה מחושבת כשהאור מוקן על הצד ימני של המצלמה.
3. כל אחד נותן צד אחר של העולם, ואם נגרום לכל עין לראות את הפנורמה המתאימה לה (עין ימין לאו שצולמה מצד שמאל ועין שמאל לאו שצולמה מצד ימין), נקבל אפקט תלת ממדי.
4. אם צילמנו כמה תמונות בזמנים שונים, נוכל להרכיב פנורמה על ידי לקיחת אזוריים מזמן אחד אחרים כך שהזיה יראה טבעי.
 - (א) למשל, אם מדובר בפיקוח בניין, ניקח אזוריים מהמרכז זמן גדול ונקבל שהמרכז קרס קודם.
 - (ב) אם מדובר בשחיה, ניקח נקודות רוחקות בזמן מאזור הצילום שהוא נמצא בו ונקבל שהוא מקדים את כל השחקנים האחרים.

חלק XVII**תרגולים****30 תרגול 1 - טרנספורמציות על תמונות, שיווי היסטוגרמה וקוונטיזציה****תמונות**

מהי תמונה? מטריצה היא בסך הכל מטריצה של מספרים. כל תא במטריצה נקרא פיקסל והערך שלו הוא הצבע של המיקום זהה בתמונה. בקורס זהו אנו נשנה את ערכיהם אלה במצבות שונות.

הערה פיקסל הוא היחידה הבסיסית ביותר בתמונה.

הגדרה. רזולוציה של תמונה היא מספר הפיקסלים של התמונה.

ברוב המקרים אנו משתמשים ב-8 ביטים לתיאור של פיקסל, כלומר ערכו של פיקסל נמצא בתחום [0, 255] עם 256 אפשרויות סך הכל.

הגדרה. מספר הביטים פר פיקסל מסומן ב-*bpp* = *bits per pixel*.

תמונה צבעונית

בתמונה צבעונית יש שלושה ערוצי צבע לכל פיקסל כולל שלושה ערכים $(R, G, B) = (Red, Green, Black)$. כל אחד מהערכים הוא 8 ביטים ומיצג בעצם ערך *ערוץ* *grayScale* שלם مثل עצמו. נshallת השאלה, מאיפה מגיע הצבע האדום, הירוק, והכחול, הרי מדובר ברמת אפור.

למעשה, כל ערך צהה בין 0 – 255 מייצג "כמה אור פגע במלמה בנקודה זו", כאשר 0 יתנו שחור, שכן אין אור, ו-255 לבן. אז למעשה, (R, G, B) מסמל כמה אור פגע ב:

(Sensor Red, Sensor Green, Sensor Blue)

יחד אנו מחברים את התוצאות ומקבלים את הצבע שצרכי להיות. למעשה, אנו מקבלים תמונה *GrayScale* תלת מימדית שככל מימד מסמן צבע אחר.

מרחבי צבע

ראינו את תמונה ה-*GrayScale*, שנשמרת כמטריצה עם ערכים 0 – 255. אילו עוד אפשרויות יש?

- ניתן לשמר את המידע באמצעות מספרי *floatingPoint*, ככה נוכל לבצע מניפולציות בלי לעגל את המידע ולשמור על דיק מסויים, ורק בסוף לעגל חזרה ל-*GrayScale*.

▷ זה נותן יותר אפשרויות, ונוכל להמיר תמונה *GrayScale* למצב זה, על ידי חלוקה של כל פיקסל ב-256. כך נקבל תמונה עם ערכים ב-[0, 1].

▷ בסוף כשנסים נכפיל ב-256 ונעגל לפי הצורך.

- **תמונה בינהיות** - כאן הכל באמת שחור לבן 1 או 0. נקבל מטריצה של ערכי 1, 0.

▷ נשתמש זהה בשביל מסכות (*Masking*) - סימול האזור עליו נרצה לבצע משהו.

- **תמונות RGB** - שרשת תמונות *RGB*.

יש עוד מרחבי צבע רבים, ונדבר עליהם, אך לעת עתה, נתעסק במרחב מיוחד - *YIQ*.

מרחב צבע *YIQ*

מרחב זה היה מרחב הצבע לטלוויזיות ישנות:

- **הערוץ הראשון Y** מסמל את ה-*GrayScale* ולמעשה מכיל את כל מידע ה-*Intesity*.

▷ כך טלוויזיות בשחור לבן יכולו לקלוט שידור למרות שעדרו בצבע.

- **הערוצים I, Q** מחזיקים את המידע על הצבע.

מרחב זה מוגדר כך

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

ביצוע טרנספורמציה על תמונה צבע זה דבר לא פשוט שכן יש שלושה ערוצי צבע והתלות ביניהם בין *YIQ* אינה לינארית. ככלומר שינוי זהה של *B*, *R*, *G*, *B* לא יתנו את השינוי הרצוי, ובכלל, מרחב זה אינו מפריד בין בהירות הצבע לבין הצבע עצמו מה שמקשה מאוד לעבוד עם התמונות.

לכן כדאי לעבוד עם עצמת הצבע ולא עם הצבע עצמו. ככלומר, לבצע טרנספורמציה על *Y* בלבד ולאחר מכן נמיר חזרה ל-*Y, I, Q*. רק אחראים על הצבע עצמוומי שקובע איך הם יראו הוא *Y*.

- מכאן כאשר נרצה לעבוד עם תמונה צבעונית:

↳ נוכל לעבור ל- YIQ

↳ להמיר לערכים בטווח $[0, 1]$

↳ לבצע את הטרנספורמציה הרצויה

↳ לחזור ל- R, G, B ובזאת לסייע.

מעתה, כל התמונות שנדבר עליהן הן *GrayScale* עם ערכים בטווח $[0, 1]$.

טרנספורמציה הגברת (Inensity)

פעמים רבות נרצה לבצע טרנספורמציה על הפיקסלים בתמונה. את הטרנספורמציה ניתן לעשות באמצעות *LUT* שתיקח פיקסל r ותחזיר את הערך החדש שלו כולם נבצע $(r) \leftarrow T(r)$.

תמונות

טרנספורמציה היפוך לכל פיקסל r נחליף אותו ב- $r = 1 - s$. אם זו הייתה תמונה צבעונית, היינו עוברים ל- YIQ וביצעים את השינוי על Y .

טרנספורמציה לוגריתמית לכל פיקסל r נחליף אותו ב- $r = c \cdot \log(1 + r)$. מה יקרה? פונקציית ה- \log משתנה מאוד לאט עם ערכים גדולים אך מאוד מהר עם ערכים קטנים ולכן נקבל פרישה של הצבעים הכהים ודוחה של הצבעים הבהירים. למה שנעשה את זה? העין האנושית אינה רואה בצורה לינארית, אלא בצורה לוגריתמית, מטעמי אבולוציה.

טרנספורמציה חזקה לכל פיקסל r נחליף אותו ב- $r^c = s$. נבחן כי c קטנה מ-1 תתן אפקט דומה לטרנספורמציה \log - מתייחס כהים וכיום בהירים, ואילו c גדולה מ-1 נותנת אפקט הפוך - וכיום כהים ומתייחס בהירים. משתמשים בה במכשורים שימושיים מażה מס' התמונה תוצג - אולי המשך לא טוב בהציג צבעים כהים? צבעים בהירים? לכן נרצה לבצע טרנספורמציה כזו עבור מסכים שונים, כדי שנוכל להציג את התמונה בצורה ברורה.

תיקון γ

כאמור, אנחנו לא רואים בצורה לינארית, ולכן אם אנו רואים נוף מרהיב ומצלמים אותו, אנו עלולים להתאכזב קשות מההתוצאה, שכן התמונה תשמר בצורה שונה מהדרך בה האנושית רואה, אנו עלולים לקבל משחו דהוי, לא כי יש צבעים דהויים, אלא רק בכלל שהמשך מציג את התמונה ככה. לכן אנו מבצעים תיקון γ באמצעות טרנספורמציה γ כדי שנוכל לקבל תמונה שמתואמת את הנוף שראינו.

ההיסטוגרמיה

ኒצור טבלה של *GrayScale* של מספר הפיקסלים עם הערכים בטוחה [0, 255], כלומר הטבלה במקומות ה- i תתן את מספר הפיקסלים עם ערך i . בambilים אחרות

$$h(r_k) = n_k$$

כאשר n_k מספר הפיקסלים עם ערך פיקסל r_k . מכאן נובע כי N

יחד עם זאת, לדעת כמה פיסקלים יש בכל תא, לא מאד שימושי. לעומת זאת, לדעת כמה אחוז מתוך הפיקסלים בעל ערך מסוים זה כן שימושי, שכן נורמל את ההיסטוגרמה על ידי חלוקה של כל הערכים ב- N .

באמצעות ההיסטוגרמה נוכל לדעת אם תמונה תהיה בהירה או כהה, שכן עמודות הצד ימני של הסקלה מציניות צבעים בהירים, ולהפך - צבעים כהים.

יתכנו גם ההיסטוגרמות שמפוזרות באמצע הסקלה, שלא שחורות או לבנות, אלא דהוויות, זו לא תמונה ברורה. לעומת זאת, תמונות עם ההיסטוגרמה עם התפלגות אחידה יותר, יתנו תמונה ברורה יותר שכן הערכים מפוזרים על כל השפקטרום. תמונות כאלה אנו מכנים תמונות עם "ניגודיות גבוהה" או *High Contrast*. תמונות כאלה עושות שימוש חכם בההיסטוגרמה ואנו נרצה לקבל תמונות כאלה, ככלור בהינתן תמונה מסוימת להמיר אותה לתמונה כזו כך שנקבל תמונה ברורה יותר.

שיעור ההיסטוגרמיה

נרצה ליצור קונראסט יותר טוב - פיזור אחד יותר של הצבעים השכיחים.

לא מאד נוח לדבר על ההיסטוגרמה אחידה, אבל על ההיסטוגרמה המctrברת דווקא כן. כלומר בתא ה- i יהיה לנו n_j נרצה להמיר את ההיסטוגרמה המctrברת שלנו לההיסטוגרמה מctrברת לינארית ובכך לקבל את ההיסטוגרמה הרצוי. יחד

עם זאת, הערך של ההיסטוגרמה המctrברת לא נוח לעובדה במיוחד במילוי שכן הוא בין 0 ל- N .

$$n_{r_k} \text{ נורמל את הערך על ידי כפל ב-} \frac{255}{N} \text{ כלומר נמיר כל פיקסל ל-} n_i = \frac{255}{N} y_k = \frac{255}{N} \sum_{i=1}^k n_i$$

מדוע בדיק הקבוע הזה? החלוקת ב- N אומרת לנו את אחוז הפיקסלים ששוכנו עד כה מתוך סך הכל הפיקסלים, למשל 0.3, לאחר מכן אנו מכפילים בערך המקסימלי של הפיקסלים ומתקבלים שהערכים הם בין 0 ל-255. אם ההיסטוגרמה אכן אחידה, נקבל את פונקציית ההזאות $x = f(x)$. ככה הרבה יותר נוח לשאוף למצב זהה.

מכאן, נשאוף לקבל ההיסטוגרמה שההיסטוגרמה המctrברת המורמלת שלה היא פונקציית ההזאות.

از מה בעצם השוויי זהה עושים? נניח שיש לנו חצי מהאיברים עם ערך מסוים, למשל 5000 מתוך 10000 עם ערך 50, לאחר השוויי נקבל $\frac{5000}{10000} = \frac{255}{2} = 127$ כלומר נקבל כי הערך שמננו קטנים בדיק חצי מהפיקסלים הוא 127, כלומר אנו מושכים כל ערך בהסתוגרמה המקורית לערך המתאים לו בישר $x = y$, עד כדי קירוב מסוים, שכן לא נוכל להגיע לדיק מושלם. נרצה כי הערך המינימלי יהיה אפס וכי הערך המקסימלי יהיה 255, כלומר נמתה את ההיסטוגרמה, ולא רק נבצע את

הנורמל, בambilים אחרות נכפול ב- $255 \cdot \frac{C(k)}{C(255)}$. ולא ב- $255 \cdot \frac{C(k)-C(m)}{C(255)-C(m)}$. שכן אם נכפול רק ב- $C(k)$ נתעלם מכל הערכים שלפני הערך המינימלי ואחרי הערך המקסימלי, אם נכפול ב- $\frac{C(k)-C(m)}{C(255)}$, נקבל מתייחה ימינה אבל ללא מתייחה שמאליה. לכן נבצע את הצעדים הבאים:

השווה היסטוגרמי 24

- 1 : Cimpute to 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 ($Z - 1$) which is probably 255
 - 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
-

למעשה אנו מחשבים LUT באמצעות $T(k) = round\left\{\frac{C(k)-C(m)}{C(255)-C(m)} \cdot 255\right\}$ וזו מבצעים לכל פיקסל בתמונה המקורית I את הפעולה $[I][i] = T(i)$. תוצאה זו גם פורשת את המידע על $[0, 255]$ וגם מבטיחה פיזור אחיד של הפיקסלים. שינוי ההistogramma מתבצע על ידי הסתכלות על (k) לכל k בטוחה הזו היא הערך החדש של הפיקסלים עם הערך $.NewH[T[k]] += H[k]$.

נבחן כי לפעולה זו התכונות הבאות:

- ברוב התמונות זה בלתי אפשרי להגיע לרמת דיוק מושלמת של $x = f(x)$, אלא רק קירוב.
- שיווי היסטוגרמה מונוטוני - קלומר היחס בין הבהירות של פיקסלים שונים נשמר (לא יתכן שפיקסל עם בהירות 10 ופיקסל עם בהירות 4 יחליפו 7 ו-9 בהתאם).
- מספר הערכים השונים (*Full Bins*), יכול רק לקטן - או שנאחד עמודות או שנשארו אותן מופרדות, אנו לא נאחד אותן.

שאלת השאלה, מהי זה נכשל?

כאשר יש לנו צבעים שחורים ולבנים בלבד, אנחנו לא רוצחים שיווי היסטוגרמה שכן אנו לא מעוניינים בכל הספקטרום, אלא רק בשחור ובלבן. לדוגמה, בעיתון.

קוונטיזציה

נិיח את כל הגוונים שיש לנו ונמצאים אותם לכמה צבעים בודדים. למשל, במקרים 256 צבעים נשתמש ב-4. כיצד נעשה את זה? נិיח את ההיסטוגרמה ונחלק אותה ל- a קבוצות. איזה צבע נבחר לכל חלק? למשל, הצבע האמצעי של כל חלק. יחד עם זאת, פועלה זו בעייתית שכן יתכן כי אנו דוחסים המון פיקסלים בהירים לצבע כהה.

כדי לסייע את השגיאה, נרצה שכל צבע לא יוזר הרבה. למשל, כל צבע z שיופיע לצבע q_i יקיים כי $(z - q_i)^2$ קטן. אבל גם זה לא מספיק, נרצה למשקל את זה עם מספר הפיקסלים, שכן לא נרצה להתאמץ להקטין מרווח קבוע קטנה של פיקסלים. לכן ננסה לסייע את $(z - p(z))^2$ כאשר $p(z)$ מייצג את מספר הפיקסלים בקבוצה של z מתוך ההיסטוגרמה המנורמלת. כמובן, נרצה לקבל סך הכל

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

כדי לקבל את ה- z וה- q הći טובים נגוזר לפי כל אחת מהקוודיניות שלהם ונקבל כי ה- q האופטימלי הוא

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

וכי ה- z הכי טוב מתקבל בנקודה

$$z_i = \frac{q_i + q_{i+1}}{2}$$

שני הביטויים תלויים אחד בשני. מה נעשה?

נתחיל על חלוקה איחודית, נקבל את q הכי טוב. בהתאם ל- q נז' את z לנקודה שמצוינו. עתה נחשב שוב את q הכי טוב וכן הלאה... כך בצורה איטרטיבית אנו לאט משפרים את החלוקה ואת המיפוי. התחנשות הזו איננה בהכרח מתכנסת למינימום גלובלי, אבל כן מובטח שהיא מתכנסת למינימום מקומי. יחד עם זאת, הבחירה הראשונית של z עלולה להוביל לאי התחנשות, ולקבלת תוצאה שאינה אופטימלית.

31 תרגול 2 - גלים, התמרת פורייה, $SFTF$ וספקטrogramה

גלים

פונק' מחזוריות (גלים) חשובות מאוד לדיוון בעיבוד תמונה, שכן התמרת פורייה אליה אנו עובדים דורשת שהפונק' תהיה מחזוריית.

הגדרה. פונק' f תקרא מחזורית אם קיימים $T \in \mathbb{R}$ ו x עבורו $f(x) = f(x + T)$.

משמעות הדבר היא ש- f פשוט חוזרת על עצמה.

הגדרה. התדריות מוגדרת כ- $\frac{1}{T}$. התדריות נמדדת ב-Hz הרץ.

הערה. כאשר אורך הגל קטן, התדריות גבוהה יותר פעמים רבות שכן הוא נכנס יותר ביחידת מסויימת.

הגדרה. כל פונק' מחזוריית אצלו תוכל להיות מתוארת באמצעות **תדריות, אמפליטודה ופאזה** (כפי שהוגדרו בדיוןנו הקודם על התמורות פורייה).

נזכיר שהזאת פאזה מתארת פשוט הזזה של הפונק' ימינה או שמאליה.

דוגמה. אורך הגל של $\sin(x)$ הוא 2π . התדריות של $\sin(x)$ על כן היא $\frac{1}{2\pi}$.

באופן כללי אורך הגל של $\sin(\alpha x)$ הוא $\frac{2\pi}{\alpha}$.

לכן לשם כתיבה נוחה נתייחס רק לפונק' מהצורה $\sin(2\pi\omega x)$, שכן עבורן **התדריות** היא ω ואורך הגל הינו $\frac{1}{\omega}$.

הערה. אם f היא פונק' סופית (לצורך העניין, מרכיבת נוספת סופית של נקודות או שהתחום שלה אינו כל \mathbb{R}) אבל לא מחזוריית, אנחנו נרחיב אותה כך שתהיה מחזוריית, וכך יוכל להשתמש בהתמרת פורייה.

לדוגמא, אם במקור הפונק' הייתה

$$f(0), \dots, f(k)$$

נרחיב אותה להיות

$$\dots, f(0), \dots, f(k), f(0), \dots, f(k), f(0), \dots, f(k), \dots$$

התמרת פורייה

מוטיבציה

השימוש בפורייה עוזר לנו לנקח את הפונק' שלנו, שמתוארת באמצעות זמן - מהו $f(t)$ לכל זמן t , ולתאר אותה בשפה אחרת: תדרים - קחו סינוסים וкосינוסים בתדר כזה ובתדר כזה, חקרו אותם ותקבלו את f .
אנו מתארים בדיקו אותו אובייקט בדרך שונה. אך מדובר לנו לתאר את f באמצעות תדרים אם שתי הדריכים נוטנות אותה אינפורמציה ומתראות את אותו אובייקט?
זה אפשר לנו להסתכל על הסיגナル שלנו, f , ברזולוציות שונות:

- **תדרים נמוכים** - המבנה הכללי (גס) של הפונק'.

- **תדרים גבוהים** - הפרטים הקטנים והעדינים בפונק'.

אם מעוניין אותנו רק המבנה הכללי של התדר שלנו, נשמר רק את התדרים הנמוכים (רזולוציה נמוכה) או אם נרצה לראות את הפרטים הקטנים נתמך בתדרים הגבוהים (רזולוציה גבוהה).

מבחינה פרקטית, ככה אנחנו יכולים לחסוך במקומות, ולשמר רק את המבנה הכללי של התדר לדוגמה, ככלומר לשמר רק את התדרים הנמוכים. דוגמה נוספת לשימוש היא שם יש לנו סאונד עם צליל גבוה ומעובן או תמונה עם "רעש", נוכל פשוט להיפטר מהתדרים הגבוהים שגורמים לבעה ולשמור את כל השאר.

למעשה, בחירת התדרים לייצוג הפונקציה תעשה לפי ההתנהגות הכללית והמקומית שלה. ניקח כמה שיותר מהתדר הכללי (הגדל) ואז ניקח כמה שנרצה מהתדרים הנמוכים, זו הסיבה שה��צתה הייתה קפיצה בטרנספורם פורייה בערך התדרות של הפונקציה:

התמרת פורייה

כפי שכבר אמרנו, כל פונק' מחזורי $(x) f$ ניתן לפרק לסינוסים וкосינוסים, כך שסכום אותם גלים של \cos , \sin ינתנו לנו את :

$$f(x) = \sum_{\omega} \left(a_{\omega} \cos \left(\frac{2\pi\omega x}{N} \right) + b_{\omega} \sin \left(\frac{2\pi\omega x}{N} \right) \right)$$

כלומר, ניקח a_{ω} מ- \cos ועוד b_{ω} מ- \sin . השאלה שלנו הופכת להיות איזה a_{ω} , b_{ω} לבחור? מי שלא משתמשים בו בכלל - ניקחAPS ממנה. מי משתמשים בו הרבה - ניקח הרבה.

הערה. \cos הוא בעצם \sin מושג, לכן בעצם כל פונק' מחזורי $(x) f$ ניתן לפרק לסכום של סינוסים.

נשים לב שבביטוי ה"יל", הסכימה \sum_{ω} מציינית סכימה של כל התדרים האפשריים.

הערה. עולה השאלה, מיהם כל התדרים האפשריים? מתמטיקאי בשם הארי נייקויסט (Nyquist) גילה כי התדר הגבוה ביותר שניתן לשחזור מאות ספרתי המיצג דוגמה של מידע הוא מחצית מתדר הדגימה. תדר זה נקרא תדר נייקויסט. במילים פשוטות נוכל להסביר זאת כך: אם דגמוני N דוגמאות במרווחי זמן שווים כמוות התדרים שנוכל להפיק (לשזר במדוק) הינו $\frac{N}{2}$, כאשר התדר המקסימלי הוא תדר נייקויסט - מחצית מתדר הדגימה.

זה גם אומר שאנו רוצים לשזר תדר f , אנו נהיה חייבים לדגום בתדירות שהיא לכל הפחות f .
הערה. אם אותן שדגמוני מכיל תדר גובה מתדר נייקויסט, קיבל טעויות דוגמה ועלולים להופיע תדרים נמנעים איפה שהוא אמורין להיות תדרים גבוהים. תופעה זו נקראת *aliasing*.

איןטואיטיבית, התמרת פורייה תהווה מעין מסננת שאומרת לנו לכל תדר ω מיהם a_ω, b_ω , המתארים לנו כמה לקחת מכל תדר.

נניח כי יש לנו N דוגמאות של הפונק' f : $f(0), \dots, f(N-1)$. התמרת פורייה דיסקרטית (DFT), מוגדרת כך:

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

"לכל תדר יש מסננת, נعتبر אותו בסיגナル והוא יגיד לנו כמה אנו צריכים מהתדר זהה". זה יתן לנו את שני המספרים.

הערה. נשים לב כי $F(0) = \frac{1}{N} \sum_{x=0}^{N-1} f(x)$ הוא ממוצע כל הדוגמאות.

נזכיר בנוסחת אoilר: $F = \cos \theta + i \sin \theta$ ונפתח את

$$\begin{aligned} F(\omega) &= \sum_{x=0}^{N-1} f(x) e^{-\frac{2\pi i x \omega}{N}} = \sum_{x=0}^{N-1} f(x) \cdot \left[\cos\left(-\frac{2\pi x \omega}{N}\right) + i \sin\left(-\frac{2\pi x \omega}{N}\right) \right] \\ &= \sum_{x=0}^{N-1} f(x) \cdot \left[\cos\left(\frac{2\pi x \omega}{N}\right) - i \sin\left(\frac{2\pi x \omega}{N}\right) \right] \\ &= \underbrace{\sum_{x=0}^{N-1} f(x) \cos\left(\frac{2\pi x \omega}{N}\right)}_{a_\omega} - i \underbrace{\sum_{x=0}^{N-1} f(x) \sin\left(\frac{2\pi x \omega}{N}\right)}_{b_\omega} \end{aligned}$$

כך קיבלנו את a_ω, b_ω . לא נסביר מדוע אלה הרכזיות שיש לקחת מכל תדר ω ונקבל זאת פשוט כנתון.

לאחר שפירקנו את f לתדרים (זמן לתרד) באמצעות חישוב F , אנחנו יכולים לחזור אחריה (מתדר לזמן) באמצעות התמרת פורייה ההיפוכה (IDFT)

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

למה צריך לכפול ב- $\frac{1}{N}$? זה סתום נרמול שבlundio הפונק' לא חוזרת להיות בבדיקה הפונק' המקורי.

הערה. נשים לב שלוקח (N) לחשב כל ערך בודד של F , אך לחשב N ערכים שונים, עבור התדרים $0, \dots, N-1$ יקח לנו $\mathcal{O}(N^2)$. עם זאת, קיים אלגוריתם בשם התמרת פורייה מהירה (FFT) שמאפשר לחשב את הערכים של F בזמן $\mathcal{O}(N \log N)$.

הערה. אמרנו שעבור N דוגמאות ניתן לייצר לכל היותר $\frac{N}{2}$ תדרים. אם כך, כיצד אנחנו עוברים על N ערכים שונים של ω בביטוי $?f(x) = \frac{1}{N} \sum_{\omega=0}^{N-1} F(\omega) e^{\frac{2\pi i x \omega}{N}}$? הדבר זה הגיוני מכיוון שלפונק' F יש תוכנה מסוימת של סימטריות: $\frac{N}{2}$ הערכים הראשונים של F סימטריים לאחרונים, שכן $\frac{N}{2}$ הערכים האחרונים לא באמות נותנים לנו מידע חדש (זהו כפילות), ואין סתירה למשפט של ניוקויסט.

נשים לב שה- DFT הינו מעבר בין בסיסים. כשהאנו עוברים מתואר של זמן לתיאור של תדרים, אנו עוברים מbasis הסטנדרטי לבסיס פורייה. בגלל שה- DFT היא העתקה לינארית נוכל לתארה באמצעות כפל מטריצה:

$$\begin{aligned} f &= (f(0), f(1), \dots, f(N-1)) \\ &= f(0) \cdot e_0 + f(1) \cdot e_1 + \dots + f(N-1) \cdot e_{N-1} \end{aligned}$$

אבל בסיס פורייה זה נראה אחרת. נרצה לקבל ביטוי כזה: $\vec{F} = M_{n \times n} \cdot \vec{f}$. נבעז זאת כך:

$$\begin{bmatrix} F(0) \\ \vdots \\ F(N-1) \end{bmatrix} = \frac{1}{N} \begin{bmatrix} \varphi^0 & \varphi^0 & \varphi^0 & \cdots & \varphi^0 \\ \varphi^0 & \varphi^1 & \varphi^2 & \cdots & \varphi^{N-1} \\ \varphi^0 & \varphi^2 & \varphi^4 & \cdots & \varphi^{2(N-1)} \\ \vdots & & & \ddots & \\ \varphi^0 & \varphi^{N-1} & \varphi^{2(N-1)} & & \varphi^{(N-1)^2} \end{bmatrix} \begin{bmatrix} f(0) \\ \vdots \\ f(N-1) \end{bmatrix}$$

כאשר $\varphi = e^{-\frac{2\pi i}{N}}$. מדוע זה מסתדר? כי האיבר $\varphi^{\omega \cdot j}$ בוקטור ה- ω בוקטור הוא בדיק (לפי כפל מטריצות):

$$\sum_{j=0}^{N-1} f(j) \varphi^{\omega \cdot j} = \sum_{j=0}^{N-1} f(j) \left(e^{-\frac{2\pi i}{N}}\right)^{\omega \cdot j} = \sum_{j=0}^{N-1} f(j) e^{-\frac{2\pi i \cdot \omega \cdot j}{N}} = F(\omega)$$

כרצוי. ניתן לראות זאת גם בצורה הבאה:

$$\begin{aligned} F(N-1) &= \sum_{x=0}^{N-1} f(x) e^{-\frac{2\pi i x (N-1)}{N}} = \sum_{x=0}^{N-1} f(x) \left(e^{-\frac{2\pi i}{N}}\right)^{x(N-1)} \\ &= \sum_{x=0}^{N-1} f(x) \varphi^{x(N-1)} \end{aligned}$$

כמו קודם, כפל מטריצות.

תכונות של פורייה1. לינאריות:

$$\Phi(f(x) + g(x)) = \Phi(f(x)) + \Phi(g(x)) \quad (\text{א})$$

$$\Phi(c \cdot f(x)) = c \cdot \Phi(f(x)) \quad (\text{ב})$$

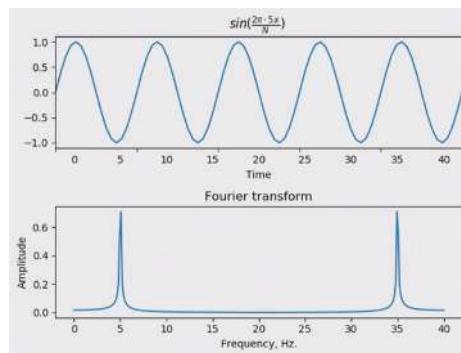
2. מחזוריות: $\forall k \in \mathbb{Z}, F(u) = F(u + kN)$ 3. סימטריה:

$$F(-u) = F^*(u) \quad (\text{א})$$

$$|F(u)| = |F(-u)| \quad (\text{ב})$$

4. הגדלה: $f(x) \xrightarrow[\text{Fourier}]{} F(u) \Rightarrow f(ax) \xrightarrow[\text{Fourier}]{} \frac{1}{|a|} \cdot F\left(\frac{u}{a}\right)$

התמרת פורייה מחזירה לנו מספרים מרוכבים מהצורה $a + bi$, שכן לצורך הייזואלייזציה בגרפים השתמש רק בגודל של $a + bi$, $a + bi$ ו $a^2 + b^2$ ואז נדע לבדוק "כמה לקחנו מהתדר ω ":



איור 206: דוגמה ליזואלייזציה של פורייה. יש עליה ב- $40 - 5 = 35$ מכיוון שהטרנספורמציה סימטרית שכן

$$F(5) = F(-5) = F(-5 + 40) = F(35)$$

באյור הנ"ל, לדוגמה, ניתן לראות כיצד התמרת פורייה, עם כמות דגימות $N = 40$, מותנת ערך מקסימלי בתדר של הגל שלנו. יש נקודת קיצון נוספת שנובעת מתכונת הסימטריה והמחזוריות של F (שימוש לב כיצד היא לא העניקה לנו מידע חדש, וכל המידע מקודד רק ב- $\frac{N}{2}$ הערכים הראשונים).

שאלה מדוע המקסימום של פורייה מתקיים בערך 5, ולא בערך $\frac{5}{40}$?

תשובה זאת כי אנו מתארים תדר באמצעות "כמה מותוך הגל שלנו נכנס ביחס בזמן". בעולם הדיסקרטי אין מושג של ייחידת זמן, שכן נוכל להתייחס ליחידת הזמן בהתאם לכל הדגימות שיש לנו. כדי שניתן לראות בתמונה, בזמן של 40 דגימות גל

הסינוס חזר על עצמו 5 פעמים, ולכן התדר הוא 5.

איפה העובדה זו מתחבطة מתמטית? במציאות טרנספ' פורייה נראה כך: $\int_{t_1}^{t_2} f(t) e^{-2\pi i f t} dt$, כאשר t_1, t_2 הם זמני

התחלת והסיום של הדגימה (הלא-דיסקרטית, מקבל $-x$ אצלנו) ו- f הוא התדר.

במקרה הדיסקרטי של אינטגרל זה קיבל $G(f) = \sum_{x=0}^{N-1} f(x) e^{-2\pi i f x}$. מכאן הרי

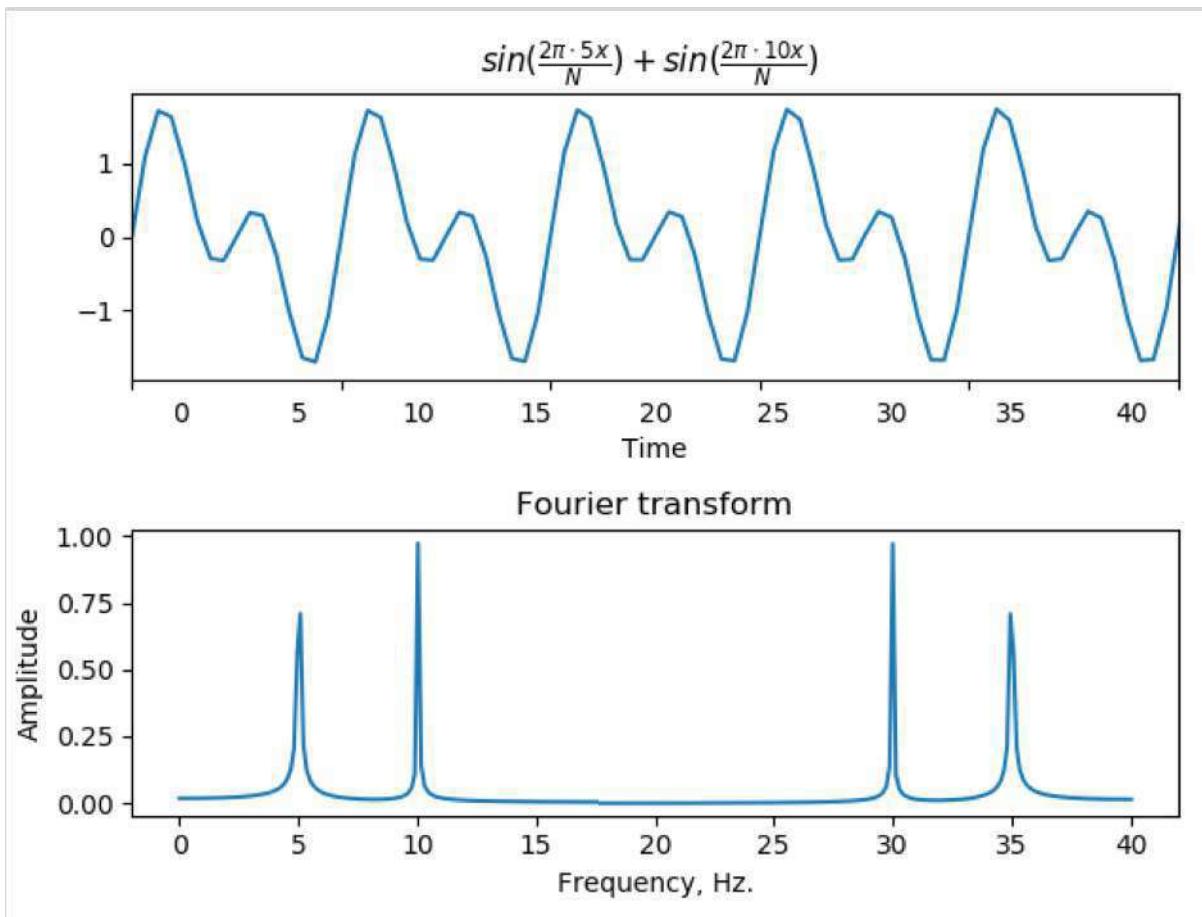
$$F(\omega) = G\left(\frac{\omega}{N}\right) = \sum_{x=0}^{N-1} f(x) e^{\frac{-2\pi i \omega x}{N}}$$

קרי, F היא פשוט מתיחה לינארית של G בפקטור של N .

בדוגמה שלנו $\omega = 5$, $N = 40$ והתדר היה $f = \frac{\omega}{N} = \frac{5}{40}$, לכן ב- G היה ערך קיצוני ב- $\frac{5}{40}$, שכן ערך זה ניתן לו התנהגות כללית של הפונקציה f . בכלל ש- F -הוא מתיחה לינארית בפקטור של $N = 40$, אותו ערך קיצון היה מתקבל רק ב- $5 = 40 \cdot \frac{5}{40}$, ואכן זה מה שאנו רואים באIOR. במקרה אחרות, התדר הופך להיות "כמה פעמים שלמןנו מחזיר בתוך כל הדוגמאות".

הערה. בתאוריה של התמורות פורייה לוקחים $t_1 = -\infty, t_2 = \infty$ (כלומר מסתכלים על הגל לאורך כל הזמן האפשרי). בנוסף, נבחן כי האמפליטודה לא משנה את פורייה (רק את הגובה שלו), שכן התדריות לא משתנות.

דוגמא. נביט בדוגמה הבאה להמחשה, לקחנו לבדוק את התדרים שאנו צריכים:

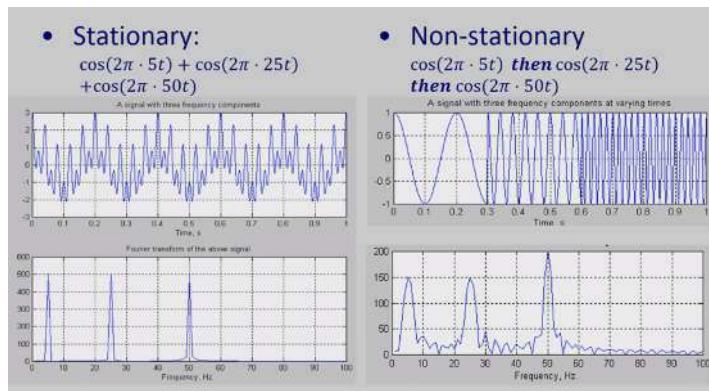


איור 207: אנו נקבע בדיקות את $F(5)$, $F(10)$, $F(30)$ ו- $F(35)$ שכנן אלה התדרויות

סיגנלים לא-סטציונריים

עד כה כל האותות שלנו היו **סטציונריים**: כל התדרים הופיעו לאורך כל הזמן. אך המונח סינגולים הם **אינם סטציונריים**, והדוגמה הבורורה ביותר היא צליל: כשאנחנו מדברים אנחנו משתמשים לעיתים בקול נמוך ולעתים בקול גבוה; אנו לא נשמעות עם קול ייחיד ומונוטוני: הסיגנל **משתנה בזמן**.

עבור סינגולים כאלה, התמרת פורייה אכן תיתן לנו את כל התדרים שהופיעו בסיגנל (אמנם באIOR הנ"ל ניתן לראות גם שיש מעט רעש), אך לא תדע להגיד לנו מתי הופיע כל תדר.



איור 208: התמרת פורייה על סיגנל סטציונרי לעומת לא-סטציונרי

מבחינת יוג, צליל ותמונות אינם כה שונים:

- ממד האיכות

▷ איכות תמונה - נמדדת לפי הרזולוציה; כמות הפיקסלים בתמונה.

▷ איכות הסאונד - נמדדת לפי קצב הדגימה (דgesmotot לשנייה)

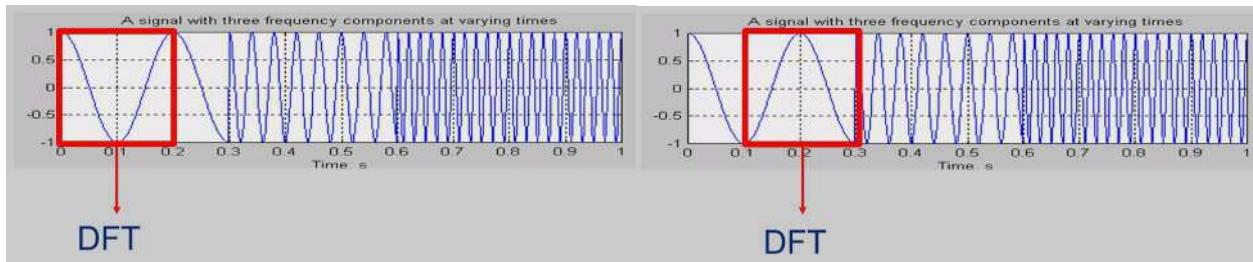
- זכרון

▷ זכרון תמונה - ביטים לכל פיקסל (ערכים בטוח [0, 255])

▷ זיכרון סאונד - depth bit (ביטים לכל דגימה) שמנדרת לפי הרזולוציה של האמפליטודה. מכשירים שונים משתמשים בכמותם ביטים שונות.

Short-Time Fourier Transform (STFT)

ראיינו שיטת DFT לא נותן לנו הבנה לגבי התדר שהופיע בכל נקודת זמן בסיגנל הלא-סטציונרי. לשם כך, נפתח פתרון חדש: $STFT$ נחלק את הסיגנל שלנו לחלקים מאד קטנים, חלונות זמן, ונעבוד תחת ההנחה שכל קטע קטן כזה הוא סטציונרי. כך נוכל לעשות DFT לכל קטע בנפרד ולקבל את התדרים של אותו קטע סטציונרי ולדעת מי התדרים שהופיעו בכל נקודת זמן בסיגnal.



איור 209: המראה של STFT - ביצוע DFT להמוני חלונות זמן קטנים שפזרים את הסיגנל שלנו

להלן שלבי ה-STFT:

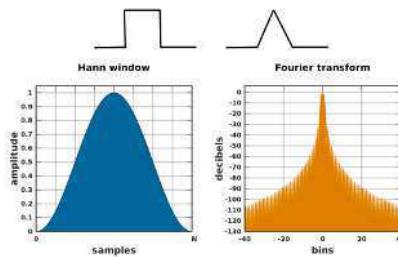
1. נבחר חלון מוגדל סופי (גודל, צורה, פונק' הazeה וכו')
2. נמקם את החלון על תחילת הסיגnal
3. נגזר את הסיגnal באמצעות החלון (קרי, לוקחים את חתיכת הסיגnal באותו חלון זמן)
4. מחשבים DFT על החלון ושמורמים את התוצאה בצד
5. מזיאים את החלון ימינה להמשך הסיגnal
6. חוזרים לשלב 3, עד שהחלון מגיע לסוף הסיגnal

כאמור לחלון שלנו, (t) W , יש מספר תכונות:

- צורה - סימטרי (קבוע), גאוסיין, משולש וכו'.
- אורך (W) - גודל החלון. **חלון ארוך** יוביל לרזולוציית תדר יותר טובה (לפי Nyquist, **חלון קצר** יוביל לרזולוציית זמן יותר טובה (יותר סיכוי שהחלון סטציונירי).
- הazeה (L) - כמה החלון יוזן כל פעם. **הazeה קטנה** תוביל לתוצאה חלקה יותר (יותר איקוטי), **הazeה גדולה** תוביל לכך שנצטרך לחשב פחות דברים (יותר עיל).

נשים לב שהazeה לא חייב להיות שווה לאורך החלון, שכן יהיו דגימות שישתתפו בחישוב של יותר מחלון אחד. בכלל, החפיפה היא בגודל $L - W$. למה אנחנו רוצים אבל שתהיה חפיפה? כדי שההתוצאה תהיה חלקה ולא נקבל אי-רציפות בלתי-טבעית בין חלון לחלון.

בגלל החפיפה זו, בדרכ' לא נשמש החלון בצורה מרובעת, אלא בפונק' שיותר דומה לגאוסיין (באופן כללי: פונק' בה הקצוות מקבלים פחות חשיבות).



איור 210: 4 דוגמאות לפונק' חלון

לסיום ה-STFT שלנו לכל זמן t ותדר ω יהיה

$$F(\omega, t) = \sum_{x=-\infty}^{\infty} f(x) W(t-x) e^{-\frac{2\pi i x \omega}{N}}$$

הערה ראשית, זה בסדר לסקום מ- $-\infty$ עד ∞ כי רוב הזמן $W(t-x) = 0$ (או לפחות שואף לאפס בגבולות $-\infty \pm$). שנית, מציין שאנחנו ממרכזים את פונק' החלון סביב t , כך שב恕בה של $x = t$ קיבל $(0)W(t-x)$, שהוא הערך המרכז שבסכל הדוגמאות שהראינו קיבל את הערך הכי גבוה.

כמו בעבר DFT, יש גם פונק' הפוכה ל-STFT והוא $:ISTFT$

$$f(x) = K \sum_{p=-\infty}^{\infty} \sum_{u=0}^{N-1} F(u, pL) e^{\frac{2\pi i ux}{N}}$$

כאשר K הוא קבוע נרמול.

שימו לב שהפעם כדי להרכיב את הסיגנל המקורי אנו צריכים לזרוץ לא רק על כל התדרים שלנו, אלא גם על כל חלונות הזמן שלנו.

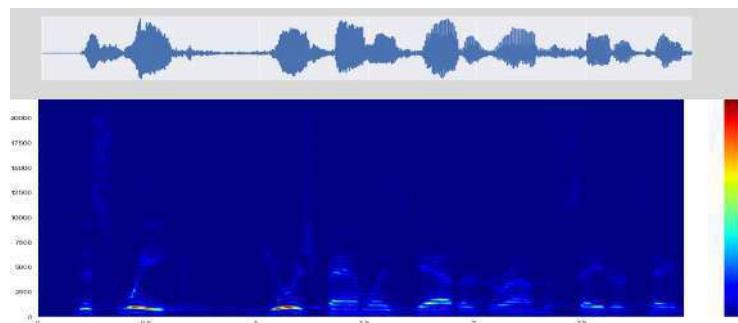
ספקטrogramma

איך נציג, ויזואלית, את ה-STFT? ב-DFT רגיל היו לנו תדרים ומagnitude (עוצמה, כמו לקחת מכל תדר) - זאת ניתן להציג בגרף דו-מימדי.

עם STFT יש לנו נקודות זמן, לכל נקודת זמן תדרים ולכל תדר מגנטודה - זאת ניתן להציג רק בגרף תלת-מימדי! דרך נוחה להציג את המידע התלת-מימדי זהה הוא באמצעות ספקטrogramma בעלת 3 צירים:

- ציר x - מייצג זמן
- ציר y - מייצג תדרות

- הצלעים בכל נקודה - מייצגים את האמפלידותה/מגנטיותה



איור 211: דוגמה לספקטrogramma

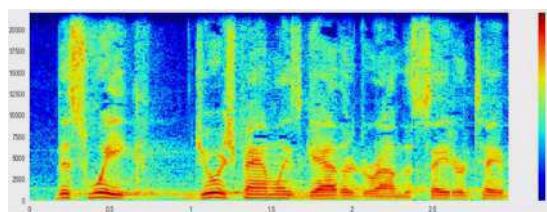
לפעמים הוייזואלייזציה, כמו בדוגמה לעיל, לא מאד ברורה בגלל הטווח הרחב של הערכים, לכן נרצה לדוחזס את הטווח וنعשה

זאת באמצעות טרנספורמצית \log , בשלבים הבאים

(i) נחשב את $\log(|F(u)| + 1)$.

(ii) נמתה את התוצאה כך שתפרוש את מלאו טווח גווני האפור.

התוצאה הרבה יותר אינפורמטיבית.



איור 212: הדוגמה הקודמת לאחר דחיסה לוגריתמית

32 תרגול 3 - טרנספורם פורייה דו מימדי, נגורת של תמונה, פילטרים

דוגמאות לטרנספורם פורייה חד מימדי

נראה כמה דוגמאות בסיסיות לטרנספורם חד מימדי.

$$\cdot f(x) = 1 \bullet$$

$$\cdot F = \delta(u) \Leftrightarrow \text{נקבל } (u)$$

$$\cdot \sin(ax) \bullet$$

$$\cdot F = \frac{1}{2} (\delta(u - \frac{a}{2\pi}) + \delta(u + \frac{a}{2\pi})) \Leftrightarrow \text{נקבל } (u)$$

$$\cdot \text{rect}(ax) \bullet$$

$$\Leftrightarrow \text{נקבל } (\frac{u}{a}) \text{sinc}(\frac{u}{a}), \text{ שהיא מעין גאוסיין בעל קפיצה בראשית, שכן התדר של rect הוא אפסי.}$$

טרנספורם פורייה דו-מימדי

במקומות, נקבל מטריצה של $N \times N$ דגימות, ובמקומות להתחשב בתדר של שורה בלבד, נתחשב גם בתדר של העמודה הרלוננטית, וכן נגדיר בהתאם

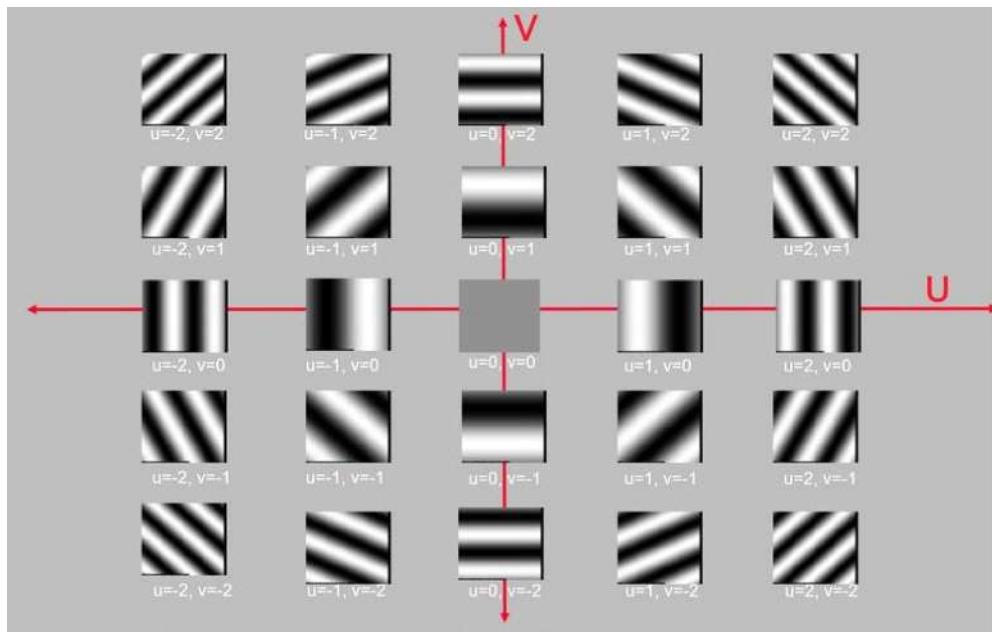
$$F(u, v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) e^{-\frac{2\pi i (ux+vy)}{N}}$$

והטרנספורמציה ההופוכה היא

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

חשוב לשים לב שסך הכל חילקו נ- N^2 , שכן יש לנו N^2 דגימות. כשייברנו על טרנספורם פורייה חד מימדי היה לנו בסיס חד מימדי $e^{-\frac{2\pi i}{N} u}$, עתה יהיה לנו בסיס דומה רק שנותחן גם ב- $(ux + vy)$.

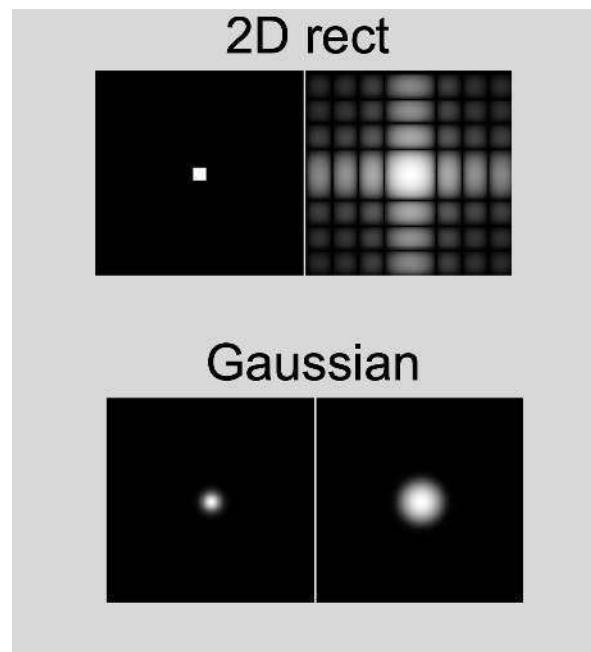
אם נציג כל איבר בסיס במישור (u, v) נקבל את הדבר הבא:



איור 3:213: כאן כל תדר יכול להיות אלכסון, אנכי, אופקי. המפה היא $(white, gray, black) = (1, 0, -1)$

ומתקיים כי תדר חלש, קיבל שתי קפיצות בקרבת אפס, שתאים מסימטריה. תדר חזק, קיבל שתי קפיצות במרחב גדול מאפס בצורה סימטרית.

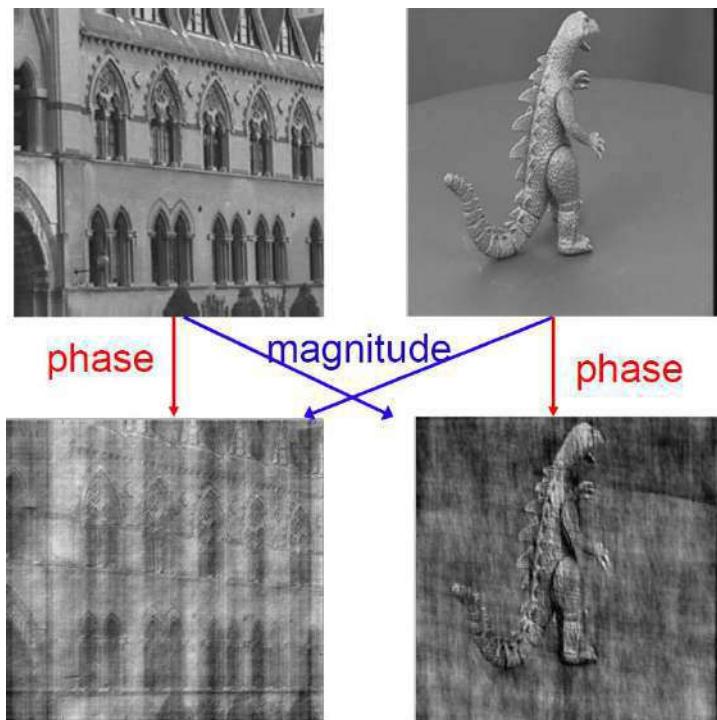
ובאופן קונקרטי, עבור תמונה *Rect* ותמונה גאוסיאן נקבל את הדבר הבא:



איור 3:214: תמונה *Rect*-*Gaussian* מוצגת בטרנספורם פורייה כ- sinc דו מימדי בכל עמודה ושורה. תמונה הגaussיאן סימטרית ולכן התוצאה עצמה גם סימטרית. נבחן כי המרכז, $(0, 0)$ הוא למעשה ממוצע התמונה, כמו שראינו בטרנספורם החד המימדי.

בעיה. בכל התמונות שהצגנו כאן, הצגנו למשה את המגנטיות של הטרנספורם, כולל את $\sqrt{R^2(u) + I^2(u)}$, אבל מספר מרכיבים איננו מרכיב רק מגנטיתודה, אלא גם מפואה, או פשוט חלק ממשי וחלק מדומה. למעשה יש לנו שתי אפשרויות לייצוג הטרנספורם והציגו:

- חלק מדומה וחלק ממשי - שתי תמונות שמצוות משחו לא מאד ויזואלי.
- מגנטיתודה ופואה - המגנטיות מוצגת בצורה מאד ויזואלית ואומرت לנו לבדוק כמה מכל תדר אנו לוקחים. יחד עם זאת, היא אינה מספיק לייצוג התמונה, שכן אנו צריכים לדעת מה הפואה של כל תדר. ביחד הן מספיקות. נבחן אבל כי הפואה כתמונה איננה נותנת ויזואלית טובה, אבל כן נותנת המן מידע. נראה את הדוגמה הבאה להמחשה:



אייר 215: התמונה השנייה שמורכבת משילוש של הפואה שלה אבל מגנטיתודה של תמונה אחרת עדין מובנת, למורות שהרבה פחות. התמונה השנייה באוטו האופן התמונה השנייה גם ייחסת ברורה. זה רק מדגיש כמה הפואה חשובה בייצוג התמונה, למורות שאינה ויזואלית.

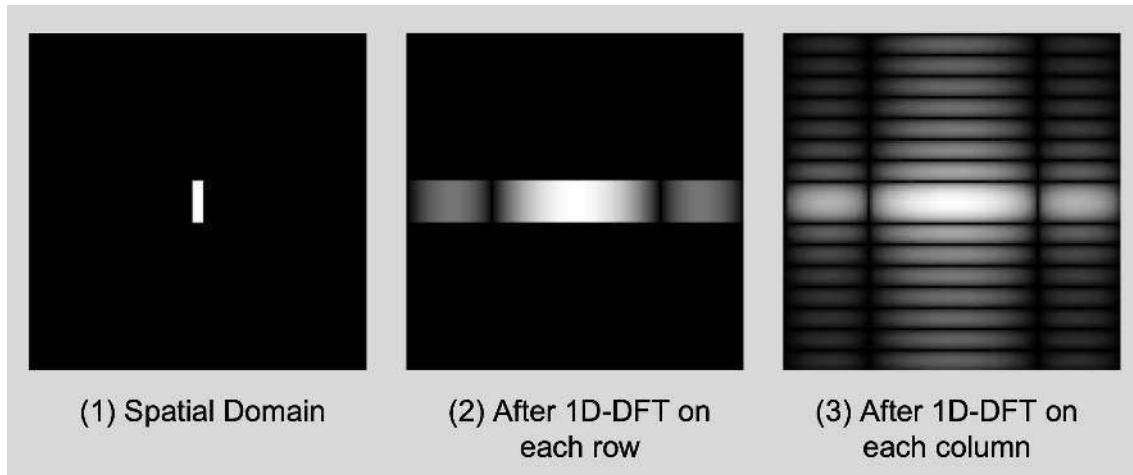
יתר על כן, הצגת המגנטיות בלבד חוסכת מאייתנו המן מידע, ולמעשה מהмагנטיתודה לא ניתן להסיק את התמונה. למשל, אם נסיט את התמונה מעט $f \equiv \sum F(u, v) e^{\frac{2\pi i(ux_0+vy_0)}{N}}$ עם הheiיטס (x_0, y_0) נקבל את הטרנספורם $f(x - x_0, y - y_0)$ וניתן לראות כי המגנטיות איננה מכילה את heiיט, אלא רק הפואה.

עליה השאלה, כיצד לחשב טרנספורם פורייה דו מימדי באמצעות טרנספורם פורייה חד מימדי, נרשום אותה بصورة שונה:

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

כלומר מספיק לחשב טרנספורם פורייה על כל שורה ועל התוצאה לחשב טרנספורם פורייה. כלומר עבור טרנספורם N מימי
ונכל לחשב טרנספורם פורייה חד מימי N פעמיים.

ניתן לקבל המ חשה | באיזור הבא:



איור 216: חישבנו טרנספורם פורייה על כל שורה שהיא בעצם rect ולכן קיבלנו sinc אופקי. לאחר מכן ביצענו על התוצאה טרנספורם פורייה. נבחן כי כל עמודה בתוצאה היא בעצם rect ולכן התוצאה היא sinc אנכי, סך הכל קיבלנו sinc דו מימי.

נבהיר כי הטרנספורמציה עצמה אינה לokaלית, ומיצגה את כל תדריות התמונה בצורה גלובלית. באופן כללי, תדריות נמוכות ייצגו התנהגות כללית של התמונה, תדריות גבוהות ייצגו רמות דיווק ורעשי רקע.

כדי לקבל המ חשה |, נביט בתמונה הבאה:



אייר 32.1: אנו מתחילהים בתדיירויות נמוכות בעייר, כדי לקבל התנאות כללית, וכך כל התוצאה שחורה, כי יש יותר תדרים נמוכים. ניתן לראות שהתדיירויות הגבוהות דוקא נראות "כרעיש רקע". נבחן בסימטריה בין החלק העליון לחלק התחתון שכן כפי שלמדנו הטרנספורם סימטרי. מעבר לכך, הנקודה במרכז התמונה עברו התדיירות $(0, 0)$, היא ממוצעת התמונה.

32.1 נגזרת של תמונה

נוכל לחשב נגזרת של תמונה על ידי שימוש בטרנספורם פורייה. נציג את f עם ההתרמה ההפוכה:

ונחשב

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

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

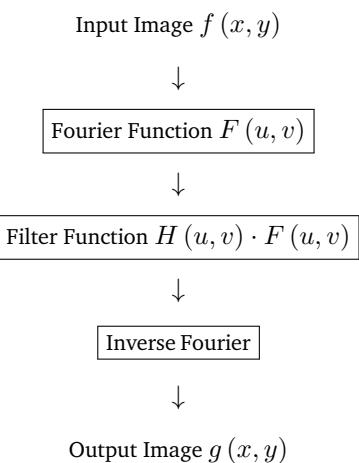
מה קיבלנו? זה דומה מאוד להתרמה ההפוכה! ההבדל הוא ש-מקדמי פורייה נכפלים ב- $-u, -v$. דבר זה יגדיל מאוד תדרים גדולים ובכך יגדיל רעש. נסמן לצורך זה $\hat{F}(u, v) = uF(u, v)$ עד כדי קבוע. ככלומר אפשר לרשום בצורה המתפרשת נכון כך: $\frac{\partial f(x,y)}{\partial x} = \frac{2\pi i}{N} \Phi^{-1}(u \cdot \Phi(f(x, y)))$.

מכיוון שאנחנו אוהבים להסתכל על פורייה כשהאפס במרכז, אנו נדגום מ- $1-N, \dots, -\frac{N}{2}, \dots, 0, \dots, \frac{N}{2}-1$ ולא מ- $1-N, \dots, 0, \dots, \frac{N}{2}$. בכיתה ראיינו מהחשה לכמה הרעשים משפיעים על הנגזרת.

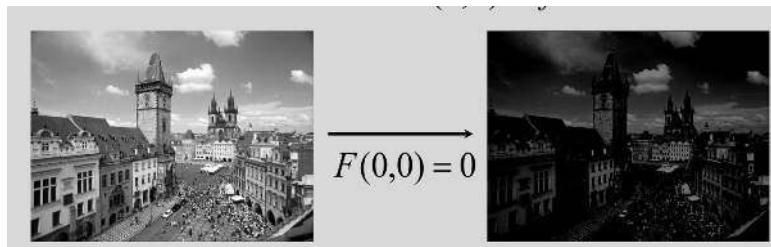
ראו לציין כי תאורטית היינו יכולים להגיד נגזרת תמונה בצורה הקלאסית, עם גבול, אך זה בוודאי יהיה מסובך יותר.

32.2 פילטרים

נרצה להיות מסוגלים להפעיל פילטר על תדרי התמונה, למשל לזרוק תדרים גבוהים או נמוכים. באופן כללי, יהיו פילטר ותמונה $f(x, y)$, אנו נבצע אתה השלבים הבאים:



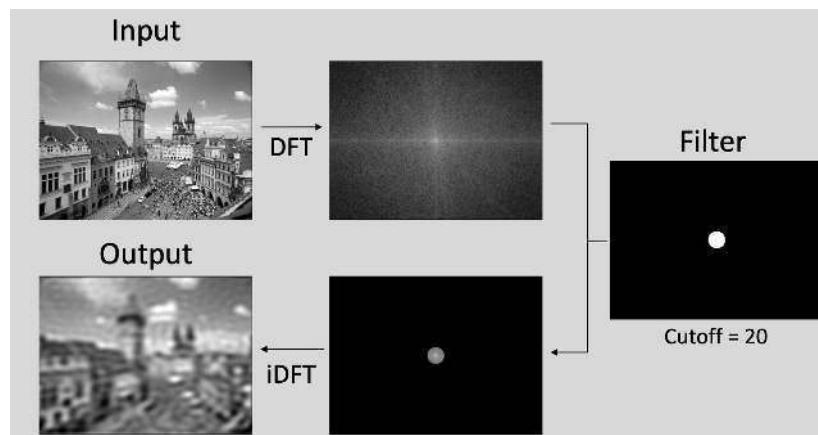
ניקח למשל תמונה ונאפס את הממוצע שלה. מה הדבר יעשה? איפוס הממוצע שקול להחסרת הממוצע מכל פיקסל בתמונה (למה?), דהיינו הורדת התדרים לתדרים נמוכים. לכן נקבל משהו שנראה כך:



איור 218: פילטר פשוט

32.2.1 פילטר Low Pass

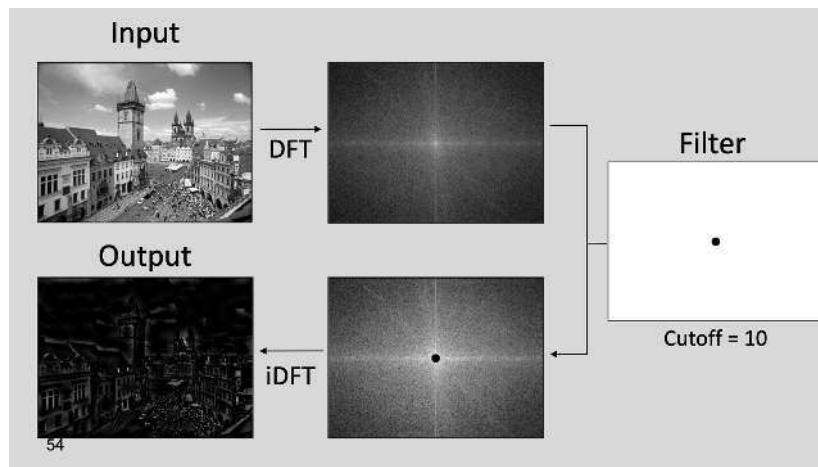
בפילטר זה אנו זורקים תדרים גבוהים בדרך מסוימת (למשל, על ידי הכפלת פילטר גaussini במרכזו, כדי לקבל הורדה הדרגתית ולא הורדה חדה בתדרים). מה שפילטר זה עושה הוא לטשטש בעצם את התמונה, שכן אנו מאבדים חידות עם זריקת התדרים הגבוהים.



אייר 219: איבדנו תדרים גבוהים ולכון איבדנו חדות. סקלת התדרים שלא אפסנו לא השנתנה שכן הממוצע נשאר אותו דבר.

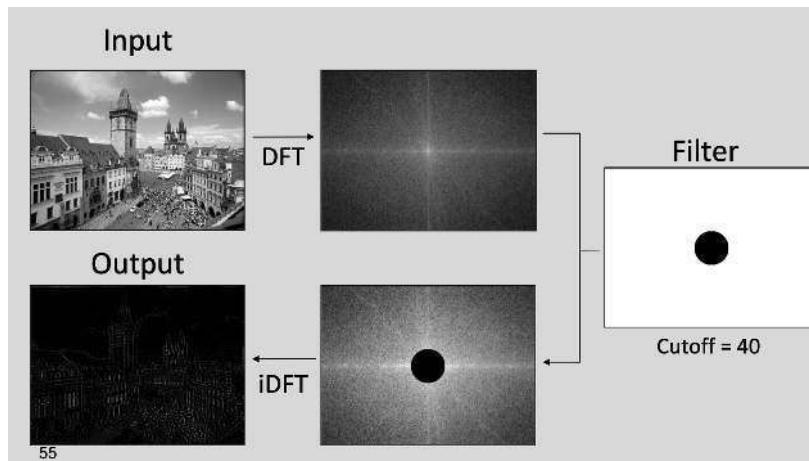
32.2.2 פילטר High Pass

בפילטר זה אנו זורקים תדרים נמוכים, ובכך מקבלים בעיקר קויי מתאר ופחות התחנויות כלליות. רק נבחן כי ככל שנארוך יותר תדרים נמוכים כך נקבל יותר ויוטר קויי מתאר ופחות מתמונה עם רקע, למשל:



אייר 220: כאן זורקו יחסית מעט תדרים נמוכים, אבל בו זמןית גם אפסנו את הממוצע, לכן מצד אחד קיבלנו יותר קויי מתאר ומצד שני קיבלנו הורדה של כל התדרים מטה.

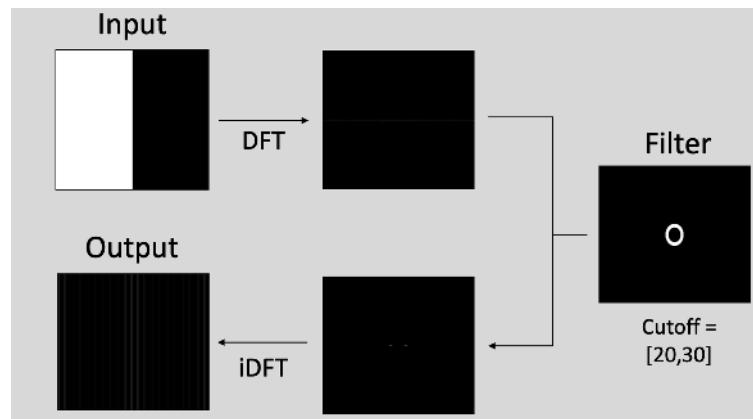
זאת בהשוואה לתמונה הבאה:



איור 221: כאן זרכנו הרבה תדרים נזוכים, אבל בו אפסנו את הממוצע, וכך מצד אחד קיבלנו הרבה יותר קוי מותאר ומצד שני קיבלנו הורדה של כל התדרים מטה. לכן מה שאנו רואים בעיקר אלה קוי מותאר.

32.2.3 פילטר Band Pass

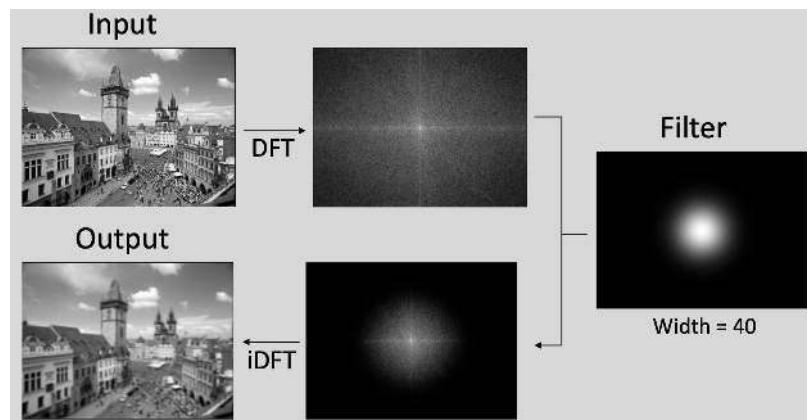
בפילטר זה אנו בוחרים תדריות מסוימות והنتוצאה היא מעין השוואת תלירות התמונה לשמה אחד:



איור 222: גם כאן אפסנו את הממוצע וכן הנטוצאה ירדה לתדריות נזוכות יותר. יתר על כן, בחרנו תדריות מאוד מסוימות וקיבלו תוצאה מיוחדת. כאן למעשה הפילטר לוקח תדריות (u,v) שמקורן מהראשית זהה.

32.2.4 פילטר Gaussian

בפילטר זה אנו בוחרים תדרים (גבויים או נזוכים) בצורה גאוסיינית, כלומר אנו משקלים כל תדר כך שההתוצאה הסופית איננה שונה דרמטית מההתוצאה המקורי:



איור 223: אמנים זרcano תדידיות גבוחות, אבל בצורה הדרגתית מה שומר על קוונטנסט בתמונה.

33 תרגול 4 - קונבולוציות, נגזרות, חיזוק, לפלייאן ו-Edge Detection

קונבולוציה חד-מימדית

הגדירה תהינה f, g זוג פונק' (וקטוריים) חד-מימדיות. הקונבולוציה על f, g מוגדרת להיות

$$h(x) = (f * g)(x) = \sum_{i=0}^{N-1} f(i)g(x-i)$$

נשים לב ש- h גם היא פונק' (וקטור) חד-מימדית.

הערה אינטואיטיבית הקונבולוציה היא לשים את f במקומות כלשהו, לחת את g , להפוך אותו ולהזיז/להסיע אותה על f , כאשר בכל נקודה אנחנו נסכם את המכפלות של החלקים של f, g שמתלכדים. ניתן לחשב שככל מקום בו g, f לא נמצאים, המרחב שלנו מורופד במלא אפסים בכל מקום, למשל $0 = (-1)^{\infty}$ תמיד, אין אינדקס שלילי.

הערה f, g לא חייבים להיות אורך אחד, שכן אנו מreffדים את שאר המרחב באפסים. נשים לב גם ש- $g * f$ לא בהכרח תהיה באותו אורך כמו g, f .

כדי להבהיר את הערה הקודמת נביט בדוגמה באIOR הבא. למרות שקל יותר אינטואיטיבית לחושב ש- g נושעת על f , מיקמו את g מתחת ל- f כדי שייהי ניתן לראות את הערכים של f, g . ניתן ($0, 0, 1, -1, 0$) $f = (0, 0, 1, 0, 0)$, והאינדקסים שלהם רצים משמאלי לيمין החל מ-0 (קרי, $= -1$). שימו לב כיצד בעת חישוב הקונבולוציה אנו הופכים את g ! כאמור, כאשר מעל g אין איבר או מתחחת ל- f אין איבר אנו מתייחסים לזה כאילו יש שם 0, כך ש- $(-1)g$ לדוגמה הוא 0.

$f = (0 \ 0 \ 1 \ 0 \ 0)$ $g = (0 \ -1 \ 1 \ 0 \ 0)$	$h(0) = \sum_{i=0}^{N-1} f(i)g(-i) = f(0) \cdot g(0) + f(1) \cdot g(-1) + f(2) \cdot g(-2) \dots$
$f = (0 \ 0 \ 1 \ 0 \ 0)$ $g = (0 \ -1 \ 1 \ 0 \ 0)$	$h(1) = \sum_{i=0}^{N-1} f(i)g(1-i) = f(0) \cdot g(1) + f(1) \cdot g(0) + f(2) \cdot g(-1) \dots$
$f = (0 \ 0 \ 1 \ 0 \ 0)$ $g = (0 \ -1 \ 1 \ 0 \ 0)$	$h(2) = \sum_{i=0}^{N-1} f(i)g(2-i) = f(0) \cdot g(2) + f(1) \cdot g(1) + f(2) \cdot g(0) \dots$

איור 224: שלושת השלבים הראשונים בחישוב הקונבולוציה $f * g$.

הערה נשים לב שבגלל שבכל שהפכנו את g בעת חישוב הקונבולוציה, $(-1)g$ מופיע הצד ימין של התמונה ולצורך העניין $(-1)f$ מופיע הצד שמאל. מעבר לכך, אנו ממשיכים להניע את g עד שאין יותר הتلכדות בכלל, ככלומר האורך של התוצאה יכולה להיות גדול יותר מהאורך של הקלטים.

לשם נוחות, כפי שהתרגלנו בעיבוד תמונה, נשים את האינדקס 0 במרכז כך שלדוגמה עברו $f = (1, 2, 1)$ ו- $f(0) = 2$ יתקיימם.

תכונות

- (קומוטטיביות) $f * g = g * f$
- (אסוציאטיביות) $f * (g * h) = (f * g) * h$
- (דיסטריבוטיביות) $f * (g + h) = f * g + f * h$
- (לינאריות) קונבולוציה היא אופרטור לינארי שכן ניתן לייצgo באמצעות מטריצה: $f * g \equiv Gf$.

הערה כל שורה ב- G תהיה הזזה של g . עבור $(0, 0, 1, 2) = g$ לדוגמה נקבל שהמטריצה המתאימה היא

$$G = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 2 & 1 \\ 1 & 0 & 0 & 2 \end{pmatrix}$$

لتken אם זו לא המטריצה הנכונה, צריך לשאול מתרגל נגיד.

הערה בחוק הדיסטריבוטיביות רשםנו $h + g$, אך מה אם הגדים של g, h שונים? גם הפעם זה לא משנה, כי אנו מופדים את המרחב באפסים. לדוגמה, $(1, 2, 3, 4, 5) + (1, 1, 1) = (1, 3, 4, 5, 5)$.

קונבולוציה דו-מימדית

הגדרה תהינה f, g זוג פונק' (וקטורים) דו-מימדיות. הקונבולוציה על f, g מוגדרת להיות

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

הערה בהקשר של עיבוד תמונה f נקראית התמונה ו- g נקרא הernal (גרעין - kernel) או הפילטר.

הערה האינטואיציה הייתה לנו לגבי קונבולוציה חד-מימדית תקפה גם כאן, רק שהפעם הזזה היא בדו-מימד, ואנחנו הופכים את הernal גם בדו-מימד (היפוך בציר x ואז בציר y), כלומר מבצעים מעין *flip*.

קונבולוציה דו-מימדית מרגישה מאוד טבעיות בתמונות, כי יש המון מידע בתמונה כאשרנו מדברים על שכנות בין פיקסלים: לדוגמה אם אנחנו לוקחים פיקסל שנמצא בתוך אובייקט, נראה הפיקסלים לידיו יהיו מאוד דומים לו, ואם הוא בגבול האובייקט אז חלק מהפיקסלים לא יהיו דומים לו וחלק כן.

מה עושים בקצוט של תמונה? יש מספר פתרונות: נוכל לרדוף את מה שמחוץ לתמונה באפסים, נוכל להפעיל את הernal רק על פיקסלים בהם הernal נכנס במלואו (ואז התמונה תתכווץ מעט בהתאם לגודל הernal), נוכל לרצף את התמונה באופן מחזורי כמו בפורייה ונוכל לשחק את התמונה באופן מחזורי (שיקוף של התמונה יוצר קשר בין פיקסלים בקצוט, בעוד שבירצוף כמו בפורייה הפיקסלים בקצוט השונים עשויים להיות מאוד שונים זה מזה ואין ביניהם בהכרח קשר). לכל פתרונות יתרונות וחסרונות, ובכל מקום משתמשים בפתרון אחר.

דוגמה אם נפעיל את הernal $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ על תמונה, נקבל תמונה חדשה שהיא טשטוש של התמונה המקורי: בכל מקום我们会 נציג פיקסל ושל שכניו.

נשים לב שהה חשוב לכפול ב- $\frac{1}{9}$ כדי שסכום האיברים במטריצה יהיה 1 (נормול), ולא נשנה את ממוצע הפיקסלים בתמונה (וכך גם קיבלנו את הנוסחה לממוצע של 9 פיקסלים). במצב כז גם דאנו שני שאר בערבי אפור תקיים $(0 - 255)$.

דוגמה הernal $\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$ נראה כמו גאוסיין וגם הוא יוצר טשטוש של התמונה, רק שאויסיין נחשב פחות גס ממוצע. כל פיקסל נשאר בעירובו, עם השפעה פחותה משכניו.

כל שנשתמש בernal טשטוש יותר גדול, הטשטוש יהיה יותר חזק.

דוגמה הernal $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ הואernal הזהות והוא אינו משנה את התמונה.

תזכורת כאמור כשאנחנו מפעיליםernal על תמונה, אנחנו הופכים אותה. המונע פעים ניתן יהיה להניח שהernal הנוכחי הוא כבר הפוך לשם נוחות.

דוגמה הernal $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$ (תחת ההנחה שהוא כבר *flipped*) מיזיאת התמונה **שמאליה!** כל פיקסל לוקח את הערך של הפיקסל מיימינו לנוכח התוצאה היא זהה **שמאליה**.

נגורת

ההגדרה המקורי של נגורת היא $\frac{\partial}{\partial_{\text{rows}}} f(i, j) = \lim_{\epsilon \rightarrow 0} \frac{f(i, j) - f(i - \epsilon, j)}{\epsilon}$. הוויל ואנו עובדים עם מספרים שלמים גרידא (הרי אנחנו קופצים מפיקסל לפיקסל, אין משמעות ללקוף מפיקסל לחצי פיקסל), לא נוכל לקחת את ϵ לבחור מספר קרוב לאפס כרצונו. המספר כי קרוב לאפס שהוא $1 = \epsilon$, אך נקבל שהנגזרת היא (מהצבה של $1 = \epsilon$):

$$\frac{\partial}{\partial_{\text{rows}}} f(i, j) \cong f(i, j) - f(i - 1, j)$$

$$\frac{\partial}{\partial_{\text{cols}}} f(i, j) \cong f(i, j) - f(i, j - 1)$$

כלומר הנגזרת היא ההפרש בין הפיקסל הנוכחי לפיקסל בשורה/עמודה לפני (השורות הולכות מלמעלה למטה במספר והעמודות ממשמאן לימין).

נשים לב שלדברים הבאים:

1. נגזרת לפי שורות זהה בדיק הפעלת קוונטולוציה עם $(\frac{1}{-1})$ (כאן הקונבולוציה לא נתונה כ-*flipped*).

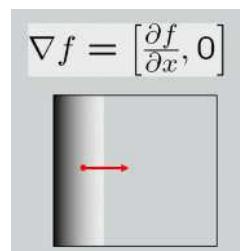
2. נגזרת לפי עמודות זהה בדיק הפעלת קוונטולוציה עם $(-1 \ 1)$ (כאן הקונבולוציה לא נתונה כ-*flipped*).

הערה נשים לב ש- $(\frac{1}{-1})$ אינו מנורמל, ויתכן אולי שימוש בקורסיל מנורמל יהיה מדויק יותר, אך היה שמדובר בקירובים לנגזרת אין זה קריטי מבחינתנו. אין לנו גם בעיה שנקלט ערכים שאינס בטוח האפור, שכן **הנגזרת היא איננה תמונה**. (אם נרצה להציג את הנגזרת האו כתמונה, נוכל לדוגמה לעשות שיפט לערכים או להשתמש בטריקים אחרים).

הערה נבחין כי הפעלת נגזרת על תמונה יוצרת תמונה שחורה עם פיקסלים לבנים במקומות בהם יש שינוי. זה נובע מכך שרוב השכנים דומים אחד לשני, אך כאשר יש מעבר חד, נקבל ערך אפור גבוה.

גרדיאנט

הגרדיאנט מוגדר להיות $(\frac{\partial f}{\partial x} \ \frac{\partial f}{\partial y})$. וקטור זה מצביע בכיוון בו השינוי של הפונק' f הוא הכי גדול. ככל שהשינוי יותר גדול, הגרדיאנט יותר גדול.



איור 225: דוגמה לגרדיאנט. כאן אין שינוי בכיוון y لكن הגרדיאנט הוא רק בכיוון x

הגרדיאנט הוא תאור שלם יותר של "איך התמונה מתנהגת", כי הוא לא מתייחס לציר ה- x ולציר ה- y בנפרד.

תכונות

$$1. \text{ מגניטודה} - |\nabla f| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

- זו העוצמה של השינוי של הפיקסל, ללא מידע לגבי הכיוון.

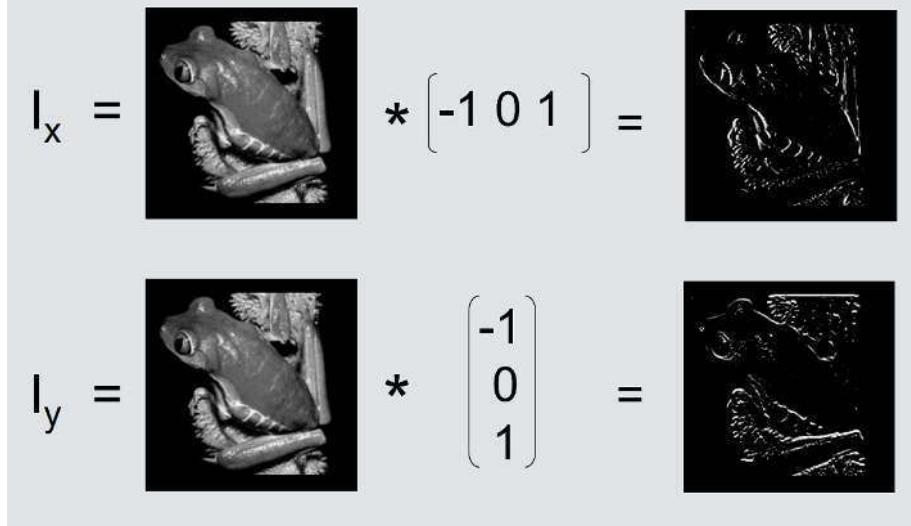
$$2. \text{ כיוון (זווית)} - \alpha = \tan^{-1} \left(\frac{\left(\frac{\partial f}{\partial y}\right)}{\left(\frac{\partial f}{\partial x}\right)} \right)$$

- זה הכיוון של השינוי של הפיקסל, ללא מידע לגבי העוצמה.

$$3. \text{ נגזרת כיוונית} - \frac{\partial f}{\partial x} \cos \alpha + \frac{\partial f}{\partial y} \sin \alpha$$

- אומר לנו מה השינוי לכל כיוון ספציפי (לכל α ספציפי).

הערה המונ פעם נעדיף להשתמש כנגזרת ב- $\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$. זה פשוט קירוב אחר לנגזרת, שונה מהקירוב המקורי שהציגנו.



איור 226: דוגמה לנגזרות על תמונה. שימוש לב כיצד בקטוות בין אובייקטים שונים (כאשר ההבדל בין הפיקסלים גדול), הנגזרת גדולה ובולטת בתמונה.

באיור הנ"ל, ניתן לחשב על הגרדיאנט בטור זוג התמונות לצד ימין של האIRO.

$$\sqrt{|I_x|^2 + |I_y|^2} = \text{[Image of a frog with a black outline showing edge detection results]}.$$

איור 227: המנגניטודה נתנת תיאור מלא יותר של כל השינויים: גם בציר ה- x וגם ב- y .

נגזרת שנייה

נגזרת שנייה היא פשוט הפעלה של הנגזרת על הנגזרת:

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

ניתן למש זאת באמצעות קובולציה עם $\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$ שכך נוחיותכם סימנו ב- \star את אינדקס ה-0 של כל וקטור.

נגזרת לפוי פוריה

ראינו שנגזרת ניתן לחשב באמצעות התמרת פוריה כך:

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

המשמעות של הביטוי היא שנגזרת התמונה היא התמרת הפוריה ההפוכה של התדריות **המשובכות**. תדריות גבוהות יותר משפיעות על התמונה יותר מתחומיים נמוכים, כך שרעש משפייע הרבה יותר בנגזרת מאשר לתמונה המקורי. אם נרצה לדוגמה לראות את השינויים בתמונה, יהיה קשה להשתמש בנגזרת כי הרעשים יפגעו בה מאוד.

שאלה למה שלא נטשטש את התמונה כדי להיפטר מהרעשים?

תשובה אם נטשטש את התמונה, אנחנו אמנים מחליקים את הרעשים, אך תוך כדי אלו מחליקים גם את המעברים בין הפיקסלים, וכך הנגזרות יחרשו.

הפתרון בו נבחר הוא שימוש בקרנל בשם Sobel. הרעיון מאחוריו הוא כדלקמן: נטשטש את התמונה בכיוון אחד, ונחשב את הנגזרת בכיוון הניצב. נחזר על התהליך לכיוון الآخر. פתרון זה עובד כי אם לדוגמה יש לנו רעשים וביניהם יש edge כלשהו שמקביל לציר y , אם היינו מטשטשים באופן אחד אז הכל היה נمرח ואז התוצאה לא הייתה טובה. אך אם נטשטש בכיוון y הרעשים אמנים יטושטו, אך ה-edge ישאר על כנו (אולי הוא רק ימרח מעט למיטה) וכך כשנחשב את הנגזרת בכיוון x נקבל את התוצאה שרצינו (כי כאמור ה-edge נשאר והרעשים נעלמו).

תהליך זה אינו מושלם אך באופן כללי נקבל ניקוי לא-רע של הרעשים בלי פגיעה משמעותית באובייקטים אותם אנו גוזרים. תמיד כשנבצע טשטוש נאבד מידע, אך קרナル Sobel ממצמצם את הפגיעה באופן די טוב.

הקרנלים של Sobel הם:

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

ונשים לב שאכן הם שקולים להפעלת טשטוש גאוסייני לציר אחד ואז נגזרת לציר הניצב:

$$f * \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} * (1 \ 0 \ -1) = f * \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$$

משפט הקונולוציה

במילים פשוטות, משפט הקונולוציה אומר שקונולוציה בזמן שווה לכפל במרחב התדר, ולהפך: מכפלה בזמן שווה לקונולוציה במרחב התדר.

משפט (הkonvolוציה) תהינה f, g פונק' ו- F, G התמורות הפורייה המתאימות להן. אז

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

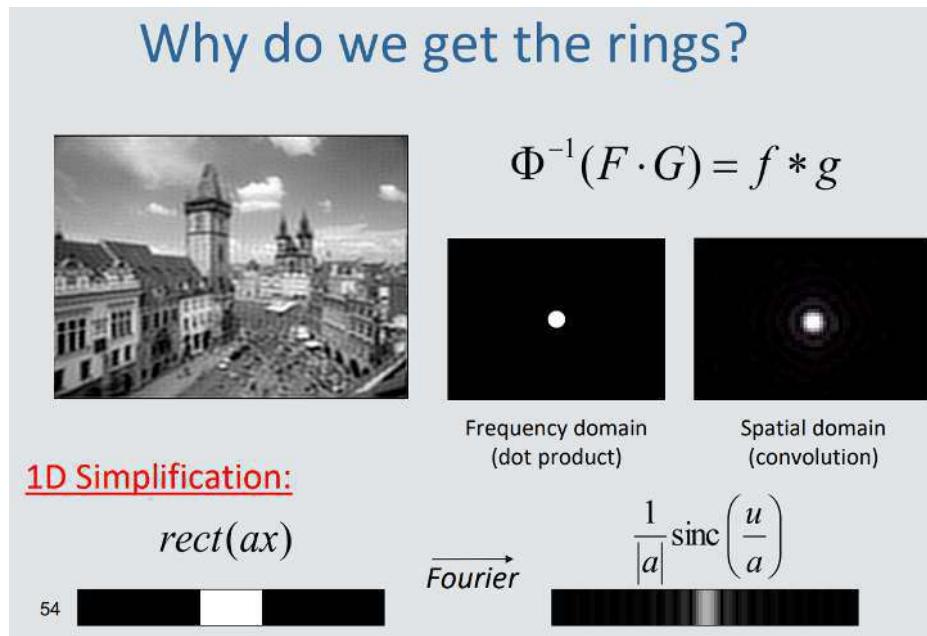
$$\Phi(f \cdot g) = F * G$$

מכאן נובע שמתאים $\Phi(f * g) = \Phi^{-1}(F \cdot G)$. בغالל של חישוב קונולוציה או פעולה מאוד יקרה, עבור קרנלים גדולים מאד ניתן להשתמש בפורייה באמצעות אלגוריתם ה-FFT. היחס בין פורייה לקונולוציה יכול לעזור לנו להבין כל מיני תופעות הקשורות לשני המושגים, ולעזר לנו לתכנן קרנלים. נראה עתה דוגמה לכך:

Fourier Filter	Convolution Filter
$F(u, v) \cdot H(u, v)$	$f(x, y) * g(x, y)$
Low-pass filter	
	$\frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$
High-pass filter	
	$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$

איור 228: Filter High-pass - הפעלת גאוסיין במרחב התדר הוביל לטשטוש, בדיק כמו שקונולוציה עם גאוסיין מוביילה לטשטוש: השניים זהים בגלל משפט הקונולוציה.
Filter Low-pass - נותן לנו את התדרים הגבוהים, שאלה השינויים החזקים בתמונה. דבר זה שקול לפילטר לפלסיאן, אותו נתאר בהמשך.

נבין מדוע אנו מקבלים טבעיות כשאנחנו משתמשים בפילטר *low – pass* – *ideal* (לא גאוסיין). במרחב התדר הפילטר הזה נראה כמו *sinc*. לכן, להפעיל בפורייה פילטר *low – pass* – *ideal* שקוללקונולוציה עם *sinc* על התמונה עצמה – ומשם הגיעו הטעויות בתמונה, כי הczורה של *sinc* אומرت שאנחנו לוקחים פעמיינן יתיר ועמוקים (אלה גלים). ממש גם הגיע הפתרון שלנו להשתמש בפילטר גאוסיין, בו כבר לא קיבלנו טבעיות. כיצד זה מתyiישב עם משפט הקונולוציה? אנו יודעים כי $\Phi(\text{sinc}) = \text{sinc}$ וולכן $\Phi(\text{sinc})$ הוא היפוך של $F \cdot G$. על כן $F \cdot G$ זה המכפלה של F בהיפוך של G כולם הפילטר שרצינו, והפעלת הטרנספורמציה הההפוכה נותנת בדיק $f * \text{sinc}$.



איור 229: נימוק באמצעות קונבולוציה לסיבת שאנו מקבלים טבעות בעת שימוש בפילטר *low-pass* אידאלי

Cross-Correlation

הגדירה תהינה f, g פונק'. נגדיר קרוס-קורלציה ע"י

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

מתמטית קרוס-קורלציה זהה לקונבולוציה, רק בלי *flip* (ברשותות קונבולוציה למשל משתמשים בקרוס-קורלציה במקום קונבולוציה) - זה רץ באותו כיוון ולא בכיוון ההפוך. נשים לב שם הפילטר סימטרי זה לא משנה. קרוס-קורלציה מודדת דמיון בין שני סיגנלים לאורך כל ההזות האפשרות ביניהם. משתמשים בהזה לחיפוש תבניות/דפוסים או התאמות בין תמונות. אם לדוגמה נרצה לחפש איפה יש עץ בתמונה, ניקח את הקרן ופעיל קרוס-קורלציה אותו על התמונה. הערך המקסימלי יתקבל כשהקרן يتלבש בדיק על העץ בתמונה.

חיזוק ולפלסיאן

חיזוד היא הפעולה החופча מטשטוש - הדגשת המעברים (*edges*) מאובייקט לאובייקט. הבעה היא שבקבות חיזוד הרעש מוגדר גם הוא. כדי להתגבר על בעיה זו נציג תחילת את מושג הלפלסיאן.

הגדרה תהי f פונק'. נגדיר את הלפלסיאן של f להיות

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

בזכות לינאריות הקונבולוציה, נוכל לומר שהלפלסיאן הוא מטריצה המיוצגת כך (נזכיר שכל המרחב מרופד באפסים):

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

בעוד שהנגזרת הראשונה נותנת כיוון/גודל של שינוי, הנגזרת השנייה נותנת את השינויים החזקים והקיצוניים ביותר.

כדי להגדיל את התמונה נחסר את הלפלסיאן מהתמונה (נזכר בקונבולוצית הזהות):

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

עתה אם נבצע קונבולוציה עם הkernel הזה על התמונה שלנו נקבל תמונה מחדדת - התמונה פחותה מהלפלסיאן שלו.



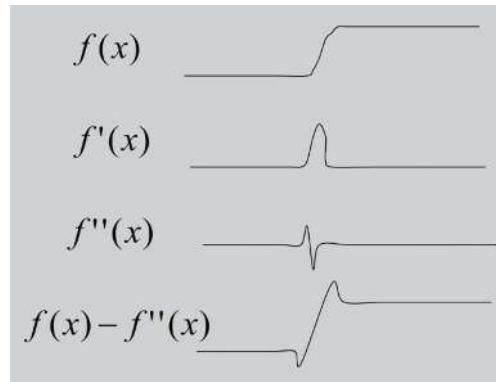
איור 230: לפני ואחרי שהפעלנו עליו קונבולוציה עם הלפלסיאן. אכן התמונה מימין יותר חדה.

כיצד זה עובד?

תהי f תמונה. הנגזרת הראשונה, f' , מסמנת את האזורים בהם היה שינוי בשינוי, כלומר מתי לדוגמה התחליל והסתפים שינוי. באזור בו יש אישוינו אחד שמתפרק בתמונה, הנגזרת תהיה קבועה ולכן הנגזרת השנייה תהיה אפס ולא נשנה את אותם אזורים. הלפלסיאן יהיה חזק כאשר אנו עוסקים ממוקום שהוא קבוע במקום שיש בו שינוי, אותן קצוות אלה המוקומות שאנו מחדדים באמצעות חישור הלפלסיאן.

שים לב כיצד באיזור הנ"ל בערך חישור הלפלסיאן הדבר היחיד שעשינו היה להציג את המוקם בו התחליל השינוי (אותם פיקים קטנים שלא היו ב- f המקורי) - זה בדיקת חידוד תמונה.

לנו בעין האנושית קשה להבדיל בין פיקסל בצבע 50 לפיקסל בצבע 55. מה שקרה בחישור הפלסייאן (התמקדו שוב באирור) הוא שאת 50 משכנו למיטה ל-40 ואת 55 משכנו למעלה ל-65 ואז את ההבדל בין 40 ל-65 אנו רואים יותר טוב. לסיום, ברוב המקומות אנו לא נוגעים אץ באזוריים עם שינוי אנרגני מרחיקים את הצבעים כדי שההבדל ביניהם יבלוט יותר. לעומתנו, **גם הרעש** מוחודד בעקבות כך.

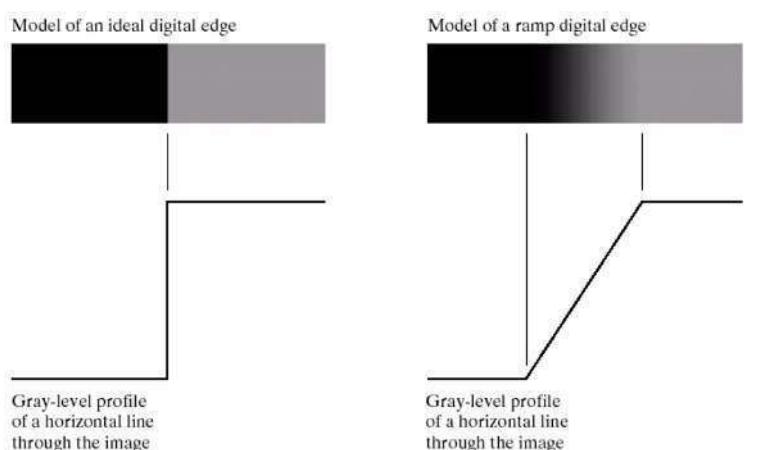


איור 231: אופן הפעולה של חישור לפלייאן

Edge Detection

נרצה לקבל מתמונה כלשהי תמונה בינהרית, בה כל מקום היושב על *Edge* - מעבר מאובייקט לאובייקט - יהיה *True* ואחרת *False*.
קודם כל עליינו להגדיר מהו *Edge*.

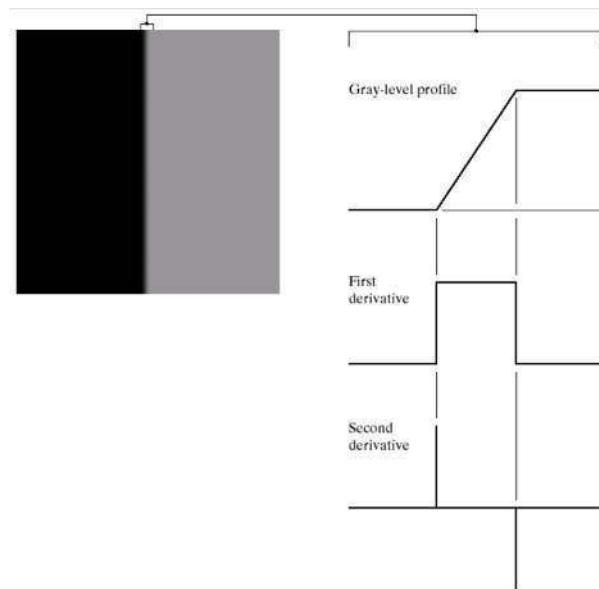
ב-*Edge* אידאלי כבאיור שמשמאלי יש לנו מעבר מובחן מהחלה השחור לחלק האפור. אך במצב שאינו אידאלי, יש לנו מעבר בהדרגה ולא מעבר תוך פיקסל אחד כמו בדוגמה הקודמת. כאן פחותה ברור איפה אנו אמורים לשים את ה-*Edge*, בקו השמאלי ברגע שהשינוי מתחילה? בקו הימני אחרי שהשינוי הסתיים? אפשרו במקרה? זה הרבה פחותה ברור! אנחנו צריכים אלגוריתם קבוע כלשהו ליזיהו *Edges*.



איור 232: משמאל: מקרה אידאלי ופשטוני. מימין: מקרה לא-אידאלי (יותר ראליסטי) ויוטר מורכב

נראה כיצד הנגזרת השנייה מגיבה למעבר כלשהו (לא בהכרח אידאלי) מואובייקט לאובייקט. נשים לב שבנגזרת השנייה ברגע שיש מעבר אנו מקבלים ערך חיובי ואז ערך שלילי. את הנקודה בה אנו חוזים את האפס נגדיר בתור ה-*edge* שלנו. לדבר זהה קוראים Zero-Crossing של הלפלסיאן. **כל נקודה בה יש תהיה edge**.

ההגיון הבא יהיה נכון לכל תמונה: במעבר בין אובייקטים הנגזרת השנייה מקבל ערך חיובי ואז מעבור לערך שלילי (או להפך).



איור 233: תגובות הנגזרת הראשונה והשנייה לתמונה הנ"ל

נסתכל עתה על דוגמה מספרית. באյור הבא נתמקד בשורה בה יש את הנקודה המבודדת משמאלה לקו במרכז. נסתכל על הנגזרת השנייה.

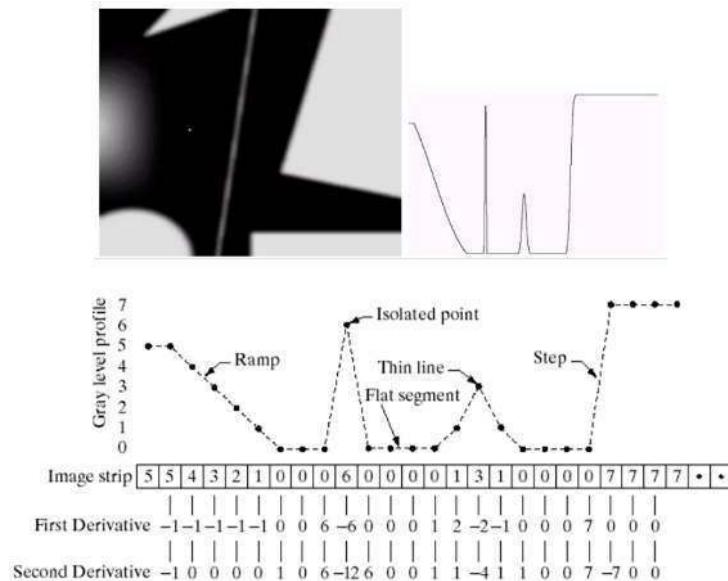
נשים לב שב-6 הערכים הראשונים היה לנו Zero-Crossing מ-1 – ל-1. נגדיר את נקודות ה-Zero-Crossing להיות הנקודה בה חצינו את האפס ל-1, כלומר בין המקום החמיישי לשישי במערך (שם עברנו מהאפס האחרון במעבר ל-1).

לאחר מכן, יש לנו מעבר מ-6 – ל-12 –, מהחובי לשילי, שכן ביןיהם גם יש נקודת Zero-Crossing. לאחר מכן, יש לנו מעבר מ-12 – ל-6, משילי לחובי, שכן ביןיהם גם יש נקודת Zero-Crossing.

אז יש לנו שוב מעבר מ-1 – ל-4 – ואז שוב ל-1, שכן שם יש גם כן שתי נקודות Zero-Crossing – אחת כשכננסו לקו באמצע התמונה ואחת כשייצאנו ממנו.

לבסוף יש לנו מעבר מ-7 – ל-7 –, מכחה לבחיר, ולכן גם שם יש נקודת Zero-Crossing.

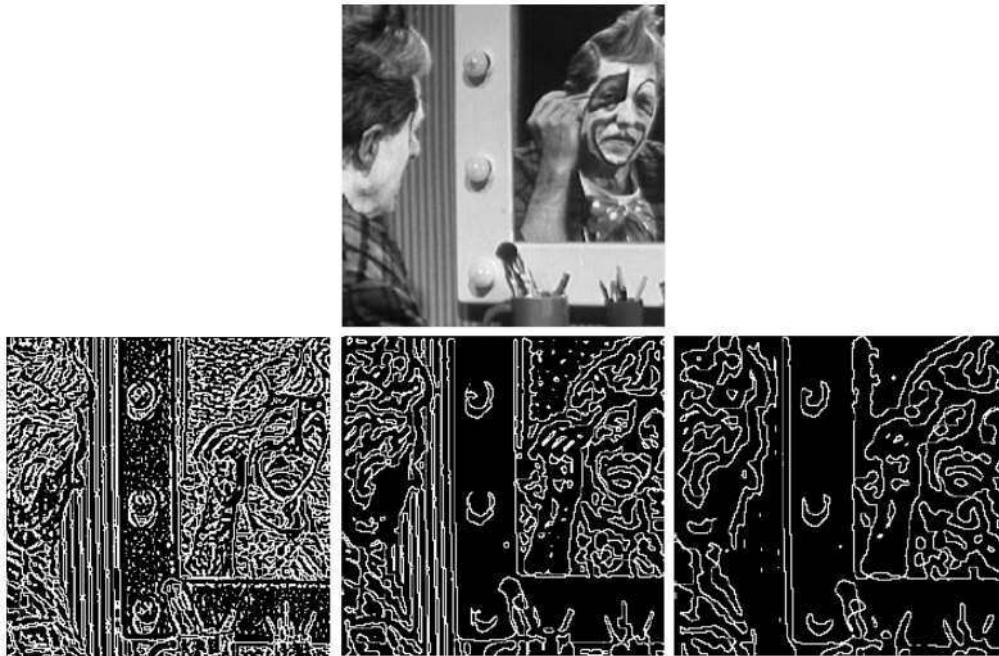
אליה כל נקודות ה-Zero-Crossing בתמונה. אנו מגדירים Edge בכל מקום שבו היה לנו Zero-Crossing.



אייר 234: השורה האמצעית בתמונה (עם הנקודה הלבנה המבודדת), יחד עם הנגזרת הראשונה והשנייה של אותה שורה.

בעיה אלגוריתם ה-*Zero-Crossing* מושפע מרעשים. יש אלגוריתמים שיודעים לטפל בבעיה זו. הפתרון המתבקש בדר"כ הוא טשטוש התמונה.

כל שנטשטש יותר יהיה פחות רעשים, אך ה-*Edges* עלולים לבנות פחות. **כל שמטשטשים יותר - יש פחות נקודות Zero-Crossing**



איור 235: אלגוריתם ה-Zero-Crossing מושפע מרעשים. הבחןו כיצד בתמונה משמאלי במרכזה המטגרת של המראה יש לנו המונ נקודות לבנות שאוთרו כ-*Edge*, על אף שמדובר סתם ברעש. בשתי התמונות מימין אנו מפעילים את אלגוריתם ה-*Zero-Crossing* לאחר טשטוש. התוצאה הרובה הרבה רועשת, אך גם נайдן מידע על ה-*Edges* ככל שמתבששים יותר.

Canny-Edge-Detection

נצע עתה אלגוריתם יותר טוב ל-*Edge-Detection*, שיסיף על רעינו הטשטוש שכבר הצנו, כדי לטפל ברעשים. בעיה נוספת שהאלגוריתם בא לפטור היא שב-*Zero-Crossing* רגלי הרבה פיקסלים יכולים לייצג *edge* שמורכב רק מפיקסל אחד. כלומר באופן אידיאלי, רק פיקסל אחד צריך לייצג כל *edge*, ולא המונ את אותו *edge*.

הנחה עובודה *Edges* הן לא נקודות בודדות, בניגוד לרעשים. בתמונה אמיתית *Edge* הוא בדרך כלל משוחץ או סוף פיקסלים הנוגעים אחד בשני המייצגים קצה/גבול של אובייקט.

הגישה בה ננקוט היא כדלקמן:

1. נטשטו את התמונה כדי להפטר מרעש.
2. נמצא נקודות "חשודות" בגרדיאנט של התמונה - נקודות שהן מועמדות טובות להיות *edge*.
3. נחבר את הנקודות החשודות ונראה אילו מתחברות יפה לקווים ארוכים - אלה יהיו רציף.

לפי הגישה זו ניתן להגדיר את האלגוריתם הבא:

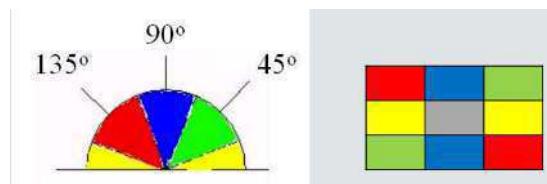
Algorithm 25 Detection Edge Canny

-
- 1 : Smooth the image with a Gaussian
 - Parameter: σ (standard deviation)
 - 2 : Compute the partial derivatives I_x, I_y
 - Use simple derivative kernels
 - 3 : Compute magnitude and direction of the gradient
 - $|G(x, y)| = \sqrt{I_x^2 + I_y^2}$, $\alpha = \tan^{-1} \left(\frac{I_y}{I_x} \right)$
 - 4 : Quantize the gradient directions
 - 5 : Perform non-maximum suppression
 - for each pixel (x, y) trace along its gradient direction
 - if $G(x, y)$ is not a local maximum, set it to zero.
 - 6 : Hysteresis
 - Define two thresholds $T_1 > T_2$
 - Every pixel with $G(x, y) > T_1$ is presumed to be an edge pixel
 - Every pixel which is both:
 - (i) connected to an edge pixel
 - (ii) has $G(x, y) > T_2$
 is also selected as an edge pixel
-

סביר עתה את המשמעות של כל שלב:

- (i) אנו מנקים את הרעשים שקל לנו להיפטר מהם.
- (ii) אנו מחשבים את הנגזרות (אפשר להשתמש ב-Sobel לדוגמה).
- (iii) אנו מחשבים לכל פיקסל את המגנטיות והכיוון של הגרדיינט.
- (iv) נבצע קוונטיzacיה לכיוונים - אם מסתכלים על תמונה ומחשבים את כיוון הגרדיינט, אנחנו מקבלים זווית בטוחה מאוד רחבה. עם זאת, במצבים פיקסל יודע לאז למעט מאוד כיוונים: צפון, דרום, מזרח, מערב, צפון-מזרח, צפון-מערב, דרום-מערב ודרום-מזרח. זו כמובן היריסטיקה בה אנו לא מבדילים לדוגמה בין זווית של 89° ל- 90° . (ראו איור)
- (v) אנחנו לא רוצים ש-4 פיקסלים ייצגו לנו את ה-edge, אלא שرك ייצג את המעבר של ה-edge. לשם כך, לכל פיקסל נצעד בכיוון הגרדיינט שלו ונשאל האם אותו פיקסל מייצג את ה-edge יותר טוב מהשכנים שלו, אם לא, נקבע את הגרדיינט של הפיקסל להיות 0. מי שייצג את ה-edge יהיה הפיקסל שה-magnitude של הגרדיינט שלו הכיל חזקה (מקסימום מקומי).
- (vi) נקבע שני ערכי סף $-T_1 < T_2$. לכל פיקסל שנשאר מהשלב הקודם, שאין אפס, נשאל האם מתקיים עבורו $G(x, y) > T_1$. אם כן, זה אומר שהגרדיינט שלו מאוד עצמתי ולכן זה בטוח edge מב Hindustan. מי שמתוחת ל- T_2 הוא בטוח edge. אחרת, אם גראינט הפיקסל לא עובר את T_1 , כמובן הוא לא עד כדי כך עצמתי, יש לנו ה תלבות אם זה edge או לא (פיקסלים כאלה נקראים weak – edges). לכן נחליט האם הגרדיינט של הפיקסל כן עובר את T_2 ובנוסף הוא מחובר לפיקסל שהוא edge, אז גם הפיקסל הזה הוא edge. כאן נכנסה ההנחה שלנו: edge הוא רצף של פיקסלים אז לקחנו רק את

הפיקסלים בהם יש חיסית עצמתית אבל גם מחוברים ל-edge (למרות שהפיקסל חלש והוא רואים שהוא המשך של edge ולכן ניקח אותו).



איור 236: שלב (iv) - קוונטיזציה. מימין אנו רואים את הכוונים בהם פיקסל יכול להשתרע, כפי שתואר ב-(iv). לכן נבצע קוונטיזציה וכל זווית של הגרדיאנט שתיהה בטוחה האדום (איור משמאלו) תמומפה לאזווית 135° .

כך, כל הרעים שהיו לנו, אלא אם הם היו ממש חזקים, ושרדו את הטשטוש ואת שלב (vi), הם לא יופיעו כ-edge. במקרה cocci גרוע הרעים יפלו בשלב של ה-edges – weak – edges, כי גם אם הגרדיאנט שלהם הוא מעל T_2 , הם ככל הנראה לא יהיו בסביבה של edge, פשוט כי הם רעים ופיקסלים של רעש הם בודדים. יהיו לנו המון שלבים בהם הרעים יכולים ליפול, וזה הסיבה שהאלגוריתם מוצלח למדי.

34 תרגול 5 - פירמידות וטרנספורמציות 2D

בפوريיה הייתה לנו בעיה של חוסר לוקאליות. ידענו מי התדרים אבל לא איפה הם. אם לדוגמה היינו רוצחים לטשטש דשא בתמונה, אבל לא את האדם שבתמונה, לא הייתה לנו דרך לעשות את זה עם פורייה.

נרצה לחבר בין הлокאליות של התמונה (המבנה שלנו) לבין התדרים. הפתרון לבעיה זו היא פירמידה. אנו נראה שכל רמה בפירמידה מאפשר לנו לשנות תדרים עבור חלק ספציפי בתמונה ולא עבור כל התמונה, דבר שעד עכשווי היה בלתי אפשרי בעברנו, בכלל אובדן הлокליות.

פרמידת גאוסיין

פרמידת גאוסיין פירמידה של תמונות, כאשר בסיס הפירמידה הוא התמונה המקורית וכל רמה מעל היא הקטנה של התמונה (באמצעות טשטוש ולקיחת כל דגימה שנייה) עד שנגיע לרמה שיש לנו רק תמונה בגודל 1×1 .

הערה. תמיד לפני שאנו עושים דגימה (לקיחת כל דגימה שנייה) צריך לעשות טשטוש! זאת כדי לא לקבל *aliasing* - התופעה שבה אנו מקטינים את כמות הדגימות ואז התדרים הגבוהים נראים כמו תדרים נמוכים (רק שהם לא אמיתי) ואז אנחנו מקבלים ארטיפיקטים מזויפים בתמונה. ככל פרט אחד או שניים אנו מבצעים - **טשטוש והקטנה כמו שלמדנו** - זה הדבר הכי חשוב בקורס.

ברמה האינטואיטיבית, יש לנו המונח **פרטים עדינים**, אך אם אנחנו דוגמים כל פיקסל שני אנחנו נזורך החוצה המון פרטים. אם היו לנו פרטיים בעובי פיקסל אחד ולא בחרנו את אותו פיקסל אז אנחנו נאבד את האינפומציה מאותו פיקסל. טשטוש לפני הדגימה יdag להעברת האינפומציה (לפחות בצורה חלקיים) לא משנה איפה פיקסל נדגם.

בכל פעם אנו מקטינים את גודל מימי התמונה ב- $\frac{1}{2}$, דהיינו נקבל שסץ כל התמונות הם בגודל $\frac{4}{3}N^2$. על מנת לא לשבור מבנית מקום.

הערה. התמונה לא חייבת להיות ריבועית. במצב זה אנו נקטין בכל רמה את הרוחב פי 2 ואת האורך פי 2, עד שנגיע לנקודה שלא ניתן להקטין יותר.

הערה האם חייבים להקטין את האורך והרוחב פי 2 כל הזמן? התשובה היא לא, אבל בדרך"כ מקטינים פי 2. (לכתובים לא ידוע על תכונות מיוחדות שיש ל-2 בהקשר זה)

האלגוריתם לבניית פירמידה גאוסיאנית (הנקראת כך כי הטשטוש הוא גאוסיאני) היא פשוטה הקטנה חוזרת של התמונה:

הקטנה של תמונה (Reduce)

1 : Blur

- e.g. Convolve with a 3×3 filter $\frac{1}{4} \cdot (1, 2, 1) * \frac{1}{4} (1, 2, 1)^t$

2 : Sub-sample

- Select only every 2^{nd} pixel in every 2^{nd} row

אפשר להשתמש ביטשטווש בקרנל 3×3 או 5×5 אף ייותר, על אף שמקובל להשתמש בקרNELים קטנים. תוצאה האלגוריתם

תהייה

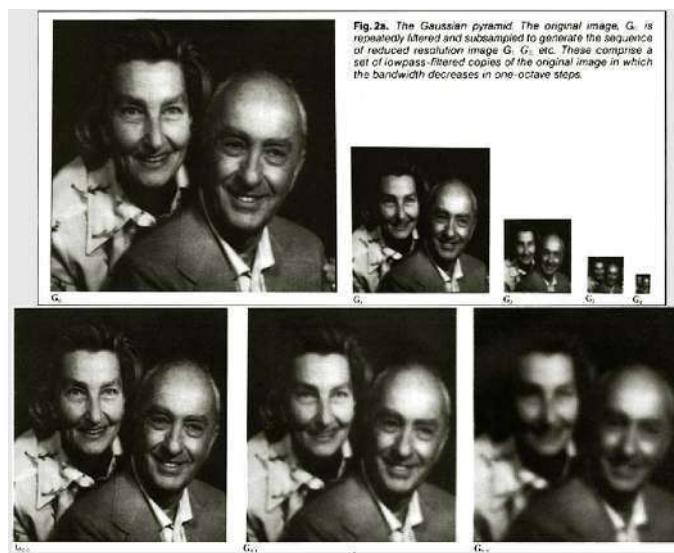
$$G_n = \text{Reduce}(G_{n-1})$$

⋮

$$G_2 = \text{Reduce}(G_1)$$

$$G_1 = \text{Reduce}(G_0)$$

תמונה מקורית



אייר 237: השוואה בין רמות בפרמידה לאחר הגדלתן

אם נגדיל חזרה את הרמות בפרמידה, נראה שככל שהרמה יותר גבוהה (התמונה הייתה יותר קטנה) אז התמונה המוגדלת יותר מטושטשת.

שאלה למה זה קורה?

תשובה נניח שהתמונה טושטשה עם קרナル 3×3 - בرمאה הראשונה לקחנו 3 פיקסלים והפכנו אותם לפיקסל אחד. בرمאה השנייה שוב עשינו אותו דבר - לקחנו 3 פיקסלים והפכנו אותם ל-1. וכל פיקסל בטשטווש זהה היה 3 פיקסלים בתמונה המקורית, כלומר בرمאה השנייה הפכנו 9 פיקסלים ל-1.

כלומר הרמה השנייה שקופה להפעלת קרナル רחב יותר על התמונה המקורית, והשלישית על קרナル יותר גדול וכו' וכו'. אנו מפעילים כל פעם את אותו קרナル אבל הפעלה החזרת מובילה לטשטווש חזק יותר ויותר עם כל רמה.

פרמידת לפלייאן

בנייה פרמידה גאוסיינית עם רמות G_0, \dots, G_n מוגדרת להיות $L_i = G_0, \dots, L_n$. הפרמידה הלפלסיאנית L_0, \dots, L_n מוגדרת להיות $L_n = G_n$ וכן $L_n = G_n - \text{Expand}(G_{n+1})$.

הערה. בגלל שהגודל של G_i שונה משול G_{i+1} (קטנה יותר), אנו נגדיל את G_{i+1} ורק אז נחסר מ- G_i .

לשם בניית פרמידה לפלייאנית (הנראית כך כי התוצאה נראהת כמו לפלייאן - כל רמה בפרמידה מכילה רק קווי מתאר, פרטים עדינים) נדרש להגדיל את G_{i+1} לגודל של G_i . לשם כך נשתמש באלגוריתם הבא:

הגדלה של התמונה (Expand)

1 : Zero Padding

- $(a_1, 0, a_2, 0, a_3, 0 \dots)$

2 : Blur

- Note: Expand blur needs different normalization!

- e.g. Blur with $\frac{1}{2} (1, 2, 1) * \frac{1}{2} (1, 2, 1)^T$

הסיבה שאנו מטישרים גם כאן היא כדי שהמידע "ילוג" לפיקסלים בהם יש 0. לאחר הטשטוש כל פיקסל יהיה 0. שזה ממוצע הפיקסלים לידיו (לכן יהיה חשוב לבצע נורמליזציה אחרת: לכפול ב- $\frac{1}{2}$ את הקרNEL ולא ב- $\frac{1}{4}$).

נשים לב שגם כל פיקסל שאינו 0 ישאר עמו ערך לאחר הטשטוש: הסיבה של כל פיקסל שאינו אפס נראהת כך:

$$0, 0, a_i, 0, a_i, 0, \text{לכן לאחר טשטוש נקבל } a_i = 0 \cdot \frac{1}{2} + 1 \cdot a_i + 0 \cdot \frac{1}{2} = a_i.$$

מתמטית גם בgalל שהריפורד ב-0 מקטין את הממוצע של התמונה, נירמול באמצעות כפל ב- $\frac{1}{4}$ תשאיר את הממוצע נמוך מהמקור, לכן נכפול ב- $\frac{1}{2}$.

$$\text{תמונה } \sum_{i=0}^n L_i = G_0 \text{ בפרט } \sum_{i=0}^n L_i = G_k \text{ שזו התמונה המקורית, זה נובע מכך ש-}$$

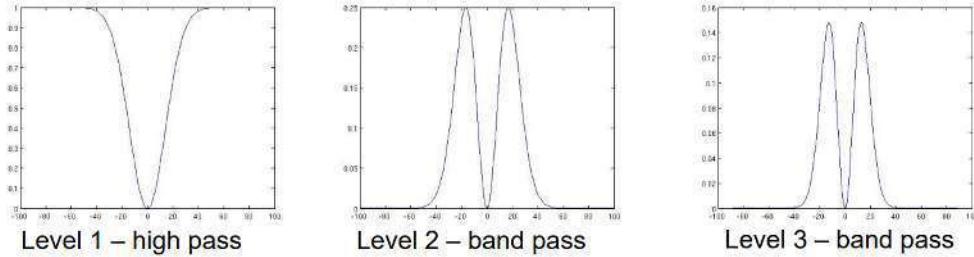
$$L_n + L_{n-1} = \text{Expand}(G_n) + L_{n-1} = \text{Expand}(G_n) + G_{n-1} - \text{Expand}(G_n) = G_{n-1}$$

מה קיבלו בעצם? לקחנו תמונה ופרקנו אותה להמוני ומונה שיחסים שליה ביחיד נותן את התמונה המקורית. אמרנו שככל טשטוש בפרמידת **גאוסיין** הוא טשטוש עם קרNEL הולך וגדל - הולך ומתרחב. במרחב התדר, זה אומר שככל רמה בפרמידת הגאוסיין היא כפל בגאוסיין שניהה יותר ויותר צר.

בגלל שככל רמה בפרמידת לפלייאן היא הפרש בין שתי רמות בגאוסיין (יחד עם הגדלה של הרמה הגבוהה יותר כדי שההתמונות יהיו באותו גודל), נובע שככל רמה בפרמידת הגאוסיין היא (במרחב התדר) כפל בהפרש של שני גאוסיינים שונים - זהו פילטר

band-pass

הרמה الأخيرة מיוחדת, שם אין הפרש של שני גאוסיינים כי שם אנו פשוט לוקחים את G_n , שהוא הפעלה של גאוסיין יחיד ולא חיסור גאוסיינים. כך שהרמה זו תכיל רק את **התדרים הנמנחים ביותר**. גם הרמה האפס (הראשונה) מיוחדת, שם אנו מחסרים את התמונה המקורית, G_0 , מ- G_1 . על כן, אין הפרש של שני גאוסיינים כי בرمה האפס (התמונה המקורית) אנו לא מפעילים שום גאוסיין על התמונה. כך שהרמה זו תכיל רק את **התדרים הגבוהים ביותר**.

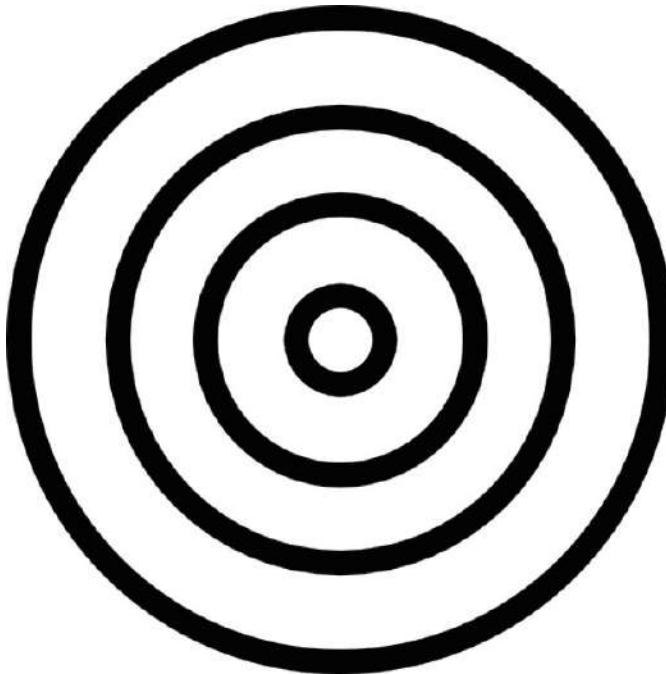


איור 238: הקNELים בפרמידת לפלייאן הם הפרשיים בין שני גאוסיינים עוקבים בהם השתמשנו לחישוב הפרמידה הגאוסיינית

על כן, כל רמה בפרמידת לפלייאן היא תוצאה של הפעלה של פילטר *band – pass* על התמונה (**במרחב התדר**) שזו שונה עבור כל רמה בפרמידה.

כך פרשנו/חילקו את התדרים מהתמונה המקורית לרמות - בכל רמה בלפלסיאן יש רמות שונות של תדרים (והרמות לא חופפות).).

כך פרשנו/חילקו את התדרים מהתמונה המקורית לרמות - בכל רמה בלפלסיאן יש רמות שונות של תדרים (והרמות לא חופפות). יתר על כן, ההחזרה של התמונות זורקת תדרים נמנחים, שכן הפעלת הגאוסיין משאייה תדרים נמנחים והחזרה של זה משאייה תדרים גבוהים, וכך נשאר עם הפרטיטים הדקים בתמונה. בכל שלב אנו מחסרים מהתדרים הנמנחים של התוצאה, שהם גבוהים יותר מהנמנחים של הקודמת וכאן אנו מקבלים מגוון רחב יותר של תדרים, עם פרטיטים דקים שונים בתוצאה - *Band – Pass*. כך אנו למשה מחלקים את התדרים בתמונה לרמות שונות.



איור 239: המחשה לתדרים השונים בפרמידת הלפלסיאן. הטעעות (שיש לחשב עליהם כחסרי עובי) מהוות קווי הפרדה בין רמות תדרים שונות. לכן כאשר אנו במרחב התדר נינן לומר כי:

הרמה ה-0 בפרמידת הלפלסיאן מכילה את התדרים בטבעת החיצונית ביותר.

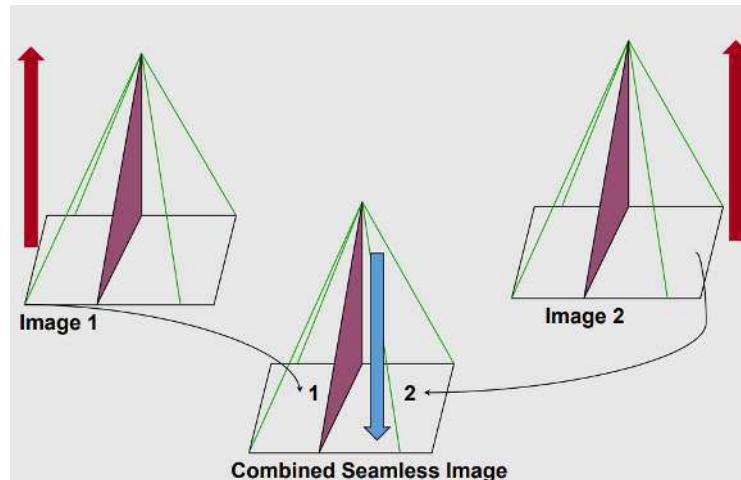
הרמה ה-1 בפרמידת הלפלסיאן מכילה את התדרים מבטבעת האחת לפני האחרונה.

...
הרמה ה-1-a בפרמידת הלפלסיאן מכילה את התדרים בטבעת האחת לפני הפנימית ביותר.
הרמה ה-a בפרמידת הלפלסיאן מכילה את התדרים בטבעת הפנימית ביותר (זהה כבר עיגול).

דבר זה בדוקנו לנו לוקאליות. בכל רמה בפרמידת הלפלסיאן, צד ימין של הרמה או גם צד ימין של התמונה המקורית, צד שמאל ברמה הוא צד שמאל בתמונה המקורית וכו'. אבל כל רמה בפרמידה מכילה רמות שונות של תדרים, לכן אם נשנה ברמה ה-i בפרמידה את צד שמאל של התמונה, שם יש דشا, אז נבנה מחדש את התמונה המקורית באמצעות הפרמידה שלנו, נקבל שינויי התדרים שיביצעו השפייע רק על הדsha.
יש לנו גם את האспект המרוכבי של התמונה - מה נמצא איפה, וגם חלוקה לرمות שונות של תדרים. יתר על כן, אנו מבינים כי סכימת כל הרמות נותנת לנו את כל התדרים במרחב התדר.

תפירת תמונות - *Pyramid Blending*

בhinintן שתי תמונות B , A , נוכל ללקח את פרמידות הלפלסיאן של שתי תמונות, L_a , L_b , וליצור תמונה חדשה C שמחזיקה השמאלי הוא של A ומהציגת הימני הוא של B , בכך שניצור פרמידת הלפלסיאן L_c כך שכל רמה i תיקח את החצי השמאלי של הרמה ה-i ב- L_a ואת החצי הימני ברמה ה-i ב- L_b . (ראו איור). לאחר מכן ניקח את L_c ונקבל תמונה באמצעות סכימה כל הרמות כפי שלמדנו (ראינו שעבור פרמידת הלפלסיאן L מתקיים $\sum_{i=0}^n L_i = G_0$ שזו התמונה המקורית).
דבר זה יגרום לכך שאזור התפירה בין שתי התמונות יהיה חלק ולא יהיה מעבר חד.



איור 240: תפירה של שתי תמונות באמצעות פירמידת לפלייאן

אנו לא חייבים לחלק את התמונה חצי-חצוי, אנו יכולים להשתמש בمسיכה ביןארית כדי להחליט איזה אזור ב- C - A ו- C - B יהיה מ- A . נסמן את המסיכה ב- M .

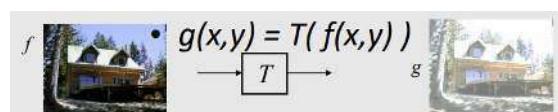
- הרמה הראשונה תהיה הcy חדה - לוקחים מהתמונה אחת ומהשניה בצורה מאוד חדה.
- הסכום הכלול של כל התמונות יטשטש את החודות.
- אם אנו מתחדים בממדים שונה שונה - נרפד את הקטנה באפסים, שכן המסיכה תנתן גם ככה רק את מה שרלוונטי.
- המסיכה לא חייבת להיות רציפה
- תרגיל 3 הוא הcy כי בקורס.

טרנספורמציות 2D

עד כה רק שיחקנו עם צבעים בתמונה, טשטשנו, חידדנו וכו'. המבנה של התמונה נשאר אצלנו אותו מבנה, רק הערכים השתנו, ולא המיקום. טוב לא עוד! היכנו לסוג חדש של משחק בתמונות. לא נשנה רק צבע של פיקסל, אלא גם את המיקום שלו.

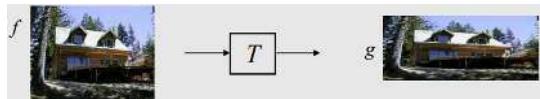
דוגמה. בטרנספורמציה ששינתה רק את הערכים, את **העוצמות**, הייתה לנו תמונה f וקיבלונו תמונה חדשה g באמצעות הפעלה T של

כברior. זה נקרא **Image-Filtering**.



איור 241: 241 - Filtering Image

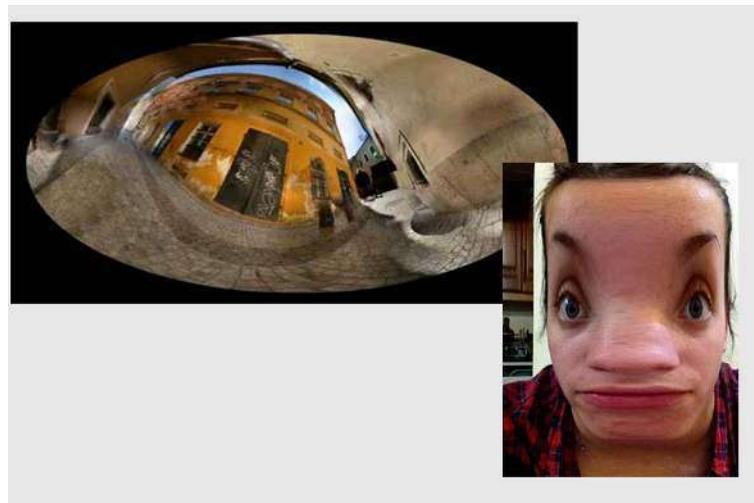
דוגמה. בטרנספורמציה שמשנה את **הקוואורדינטות** של התמונה, T יכולה לפעול כבאיור. זה נקרא Image-Warping.



איור 242 : T - Warping Image.

באיור הבא תוכלו לראותו כל מיני עיוותים של תמונות. עיוותים כאלה נקראים Non-Parametric-Image-Warping ולא נתעסק איתם בקורס שלנו.

עיוות/עיקום כזה משתמש בפונקציית warp יותר מורכבת (*splines, meshes, optical flow (per pixel motion)*)



איור 243: דוגמה ל- Non-Parametric-Warping Image

הטרנספורמציה T על פיקסל במקומ p , פולטת קוואורדינטה חדשה ($T(p)$) שאומרת לאן הפיקסל עובר בתמונה החדשה.

T היא גלובלית - אנו עושים את אותה הטרנספורמציה על אותה התמונה, באותו חוקיות מאוד מסויימת (בניגוד לטרנספורמציה לא-פרמטרית שראינו לפני רגע). דהיינו T עברו מטריצה $\begin{bmatrix} x' \\ y' \end{bmatrix} = T \begin{bmatrix} x \\ y \end{bmatrix}$. T קבוע לפי מספר מצומצם של מספרים (פרמטרים).

להלן כמה דוגמאות:



איור 244: דוגמאות לטרנספורמציות

.1. הזאה של כל התמונה בציר ה- y, x . הזאה של התמונה 3 פיקסלים ימינה לדוגמה.

.2. הגדלה או כיווץ - בכל ציר בנפרד.

.3. סיבוב - *rotation*

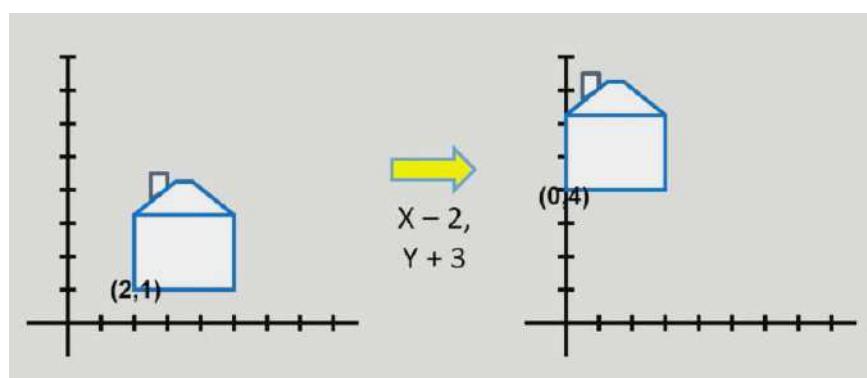
.4. שיקוף, תמונה מראה - *mirror*

.5. מתיחה אלכסונית (להפוך למקבילית) - *shear*

.6. שינוי זווית המשנה את הפרטפקטיבת של איך אני רואה את התמונה. *perspective*

טרנספורמציה *Translation*

הזהה על ציר ה- y, x . מתמטית זו הוספה של סקלר לכל אחד מרכיבי ה- y, x .

איור 245: המבשחה ל-*Translation*

לדוגמה 2 פיקסלים שמאליה ו-3 פיקסלים למעלה. הנוסחה ל-*T* במקרה זה היא $T(x, y) = (x - 2, y + 3)$

באופן כללי:

$$x' = x + t_x$$

$$y' = y + t_y$$

אפשר לדמיין כיילו התמונה נמצאת בחלל אינסופי שחור ו- T מזיהה את התמונה ברקע השחור זהה. כאשרחנו מציגים את התמונה אנו בוחרים איזה חלק להציג מהחלה השחור האינסופי.

לבד $translation$ זה לא מאד שימושי, אבל בשילוב עם המונון תМОנות זה כן יכול להיות שימושי. לדוגמה, כשבננה פונרמות נctrיך להזיז את התמונות למיקום המתאים שלhn כדי ליצור פונרמה אחת גדולה. זו לא טרנספורמציה לינארית $\mathbb{R}^2 \rightarrow \mathbb{R}^2$, היא גם מזיהה את 0 במקומם להשאר אותו במקום. אז אי אפשר ב- 2×2 . האם אפשר ב- 3×3 ? על פניו נראה שלא, כי כל נקודה היא 1×2 , אבל אם נוסיף קוודינטה, נראה שכן.

ב- $translation$, T היא העתקה אפינית. ניתן להפוך אותה להעתקה לינארית באמצעות הוספת קוודינטה: אם הזו היא

ב- t_x, t_y אי

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

הערה. ב글 ש- T היא העתקה אפינית אנחנו צריכים להוסיף קוודינטה נוספת להעתקה לינארית. כאשר מדובר על נקודה מסוימת בקוודינטה الأخيرة, 1, אך ב글 שמדובר על וקטור (הזה) מסוימים 0.

בסוף נשמרות שתי הקואורדינטות הראשונות בוקטור $\begin{pmatrix} x+t_x \\ y+t_y \\ 1 \end{pmatrix}$ כי הן אלה שמשמעותן אותנו.

כפי שמצוין בעשרה, אם נקבל תוצאה שהיא $\begin{pmatrix} x' \\ y' \\ w \end{pmatrix}$, נחלק את הוקטור ב- w כדי לקבל תוצאה בה הקואורדינטה الأخيرة היא $\begin{pmatrix} \frac{x'}{w} \\ \frac{y'}{w} \\ 1 \end{pmatrix}$.

הערה. בהעתקות לינאריות מתקייםות כמה תכונות והן רשומות להלן. בהעתקה של הזו לא הצליחו לייצג אותה כמטריצה 2

על 2 כי לא התקיימה תכונה (i), $0 + t_x \neq 0$. התכונות:

(i) $T(0) = 0$

(ii) קווים ישרים נשארים קווים ישרים.

(iii) קווים מקבילים נשארים מקבילים

(iv) פרופורציות נשמרות - היחס בין שתי נקודות נשאר אותו יחס.

(v) סגורות תחת הרכבה.

(vi) קל למצוא הופכית אם יש צו.

אנו נראה שהתוצאה של הפעלת חלק מהפעולות (כמו $translation, shear$) תגרום לכך שנאבד חלק מהתכונות (לדוגמא

פרופורציות לא ישמרו).

הערה. כדי לשמר על אחידות נציג את כל העתקות בגרסתן הפשוטה כמטריצה 2×2 , אבל גם כמטריצה 3×3 (את (2×2) הרि לא ניתן היה ליצג כ- 3×3 *translation*

טרנספורמציות Scaling

мотichtet הזרים של התמונה. כל ציר ניתן למתחות בפרופורציה אחרת. אם נרצה למתחות את ציר ה- x ב- s_x ואת ציר ה- y ב- s_y מהעתקה T תהיה

$$T \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} s_x \cdot x \\ s_y \cdot y \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

שזו העתקה לינארית.

בגרסה $:3 \times 3$

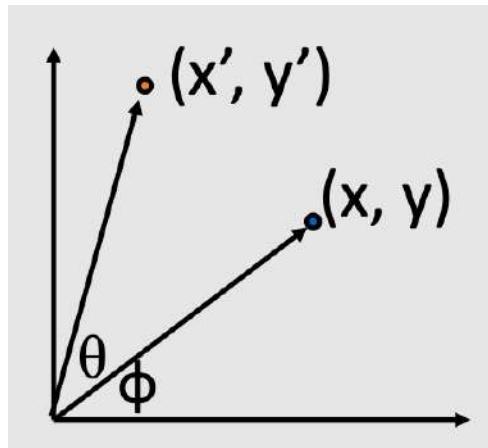
$$T \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} a \cdot x \\ b \cdot y \\ 1 \end{pmatrix} = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

נשים לב שאחרי שביצעו *scaling*, לדוגמה ה- x ב-2 ואת ציר ה- y ב-3, אם פיקסל במקומות $(1, 2)$ עבר ל-(6, 2) והפיקסל $(2, 2)$ עבר ל-(4, 6), נוצר לנו חור: אין פיקסל במקומות $(3, 6)$. בהמשך נראה איך להתמודד עם בעיה זו (נשתמש בمعין אינטרפולציה).

אם ביצעו *scaling* ניתן לחזר חזרה באמצעות כפל ב- $\begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}^{-1}$, ונקבל את התוצאה המקורית עד כדי טעויות שקיבלונו בתהיליך האינטרפולציה.

טרנספורמציות rotation

סיבוב בזווית θ סביב הנקודה $(0, 0)$.

איור 246: המ/change ל-*Rotation*

הנוסחה ל- x', y' היא כדלקמן.

היא תתבצע ע"י ההעתקה הילינארית

$$T \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x \cos \theta - y \sin \theta \\ x \sin \theta + y \cos \theta \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

בגרסה 3

$$T \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x \cos \theta - y \sin \theta \\ x \sin \theta + y \cos \theta \\ 1 \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

הסיבה שנוסחה זו נכונה היא שאם נציג את (x, y) בקואורדינטות פולריות אז יוכל לרשום

$$x = r \cos \phi$$

$$y = r \sin \phi$$

כדי לקבל סיבוב ב- θ נרצה את הנקודות

$$x' = r \cos(\phi + \theta)$$

$$y' = r \sin(\phi + \theta)$$

לכן לפי זהויות הטריגונומטריות הידועות

$$x' = r \cos \phi \cos \theta - r \sin \phi \sin \theta = x \cos \theta - y \sin \theta$$

$$y' = r \cos \phi \sin \theta + r \sin \phi \cos \theta = x \sin \theta + y \cos \theta$$

דיהינו $R^{-1} = R^T$ כי מכיוון $\det R = 1$ יתקיימם כי יתר על כן המטריצה R ולכן המטריצה R -Rotation Matrix

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}}_{R-Rotation Matrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

ההיפכית היא $R^{-1} = R^T = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}$. נבחין כי למרות ש- $\sin \theta, \cos \theta$ אינם לינאריות, הטרנספורמציה עצמה כן. זה נראה כך:

טרנספורמציה shear

המתיחה של *shear* נתונה ע"י הנוסחה (שם היא ה"ל)

$$T \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x + sh_x \cdot y \\ sh_y \cdot x + y \end{pmatrix} = \begin{pmatrix} 1 & sh_x \\ sh_y & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

זו מתיחה מקבילתית של התמונה. בגרסה 3×3 :

$$T \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + sh_x \cdot y \\ sh_y \cdot x + y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

טרנספורמציה mirror

шиקוּף סביב ציר ה- x לדוגמה נתנו ע"י הה"ל

$$T \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -x \\ y \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

ובאופן דומה עבור שיקוף סביב ציר ה- x , או סביב שני הצירים: הערא. ניתן לשלב כמה פעולות לפחות בודדת באמצעות מכפלת המטריצות המתאימות.

לדוגמא:

$$\begin{pmatrix} x' \\ y' \\ w' \\ p' \end{pmatrix} = \left(\begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \right) \begin{pmatrix} x \\ y \\ w \\ p \end{pmatrix}$$

נשים לב ש- (t_x, t_y) היא מטריצה אחת גדולה שנonta לנו את תוצאות הפעולות שרצינו לפי הסדר. הסדר חשוב, כי כפל מטריצות הוא לא קומוטטיבי: אכן $p'' = T \cdot S \cdot R \cdot p$, $p' = T \cdot R \cdot S \cdot p$, הנקודה נקודה אחרת לגמרי.

טרנספורמציה אוקלידית/קשייה (Rigid)

$$\cdot \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta & t_x \\ \sin \theta & \cos \theta & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

דרושים 3 פרמטרים לייצוגה (t_x, t_y, θ) .

נשים לב שבמקרה זה יש חישובות לסדר של מכפלת המטריצות: הזה וזו סיבוב זה לא אותו דבר כמו סיבוב ואז זהה. כל המידע בהעתקה זו נשמר, למעט האוריינטציה (אורכיים, זווית וכוכ) כי סובבנו את התמונה. אנחנו לא מבצעים מתייחא או *scaling*.

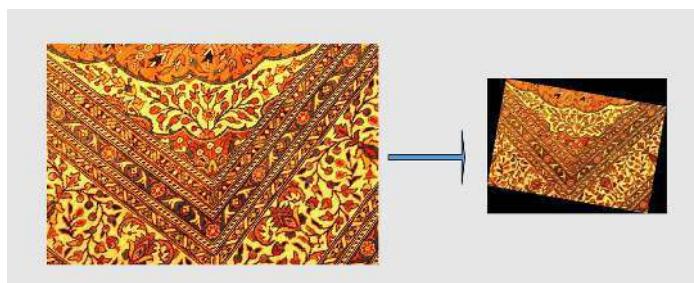
טרנספורמציה דמיון (Similarity)

$$\cdot \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} s \cos \theta & -s \sin \theta & t_x \\ s \sin \theta & s \cos \theta & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

כאן צריך 4 פרמטרים (s, t_x, t_y, θ) .

גם במקרה זה יש חישובות לסדר של מכפלת המטריצות. כל המידע בהעתקה זו נשמר, למעט אורכיים ואוריינטציה (פרופורציות כן נשמרוות אבל). טרנספורמציה דמיון מזכירה לנו במציאות מצלמה בתמונה במקביל לאיישו מישור (משטח כמו ציר לדוגמה). אפשר אז ימינה ושמאליה, להתקרב ולהתרחק, לסובב את המצלמה וכו'. התרחקות והתקרובות זה ה-*scaling*, והסיבוב והתזזה, ביחד יוצרם את הדמיון המלא.

זה נראה כך:



איור 247: המראה לדמיון

טרנספורמציה אפינית

טרנספורמציה אפינית היא שילוב של זהה עם כל העתקה לינארית דו-מימדית:

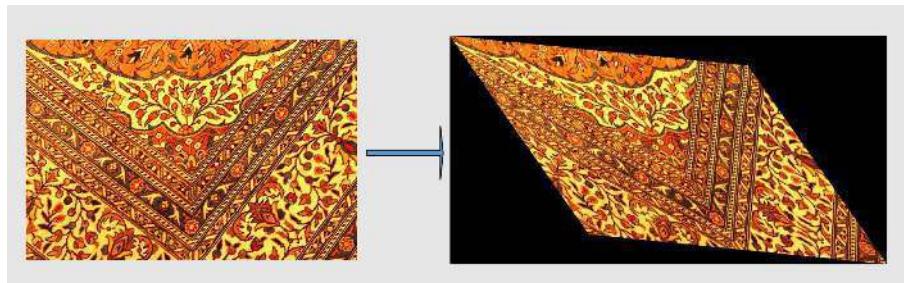
$$T \begin{pmatrix} x \\ y \\ w \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix}$$

כאן צריך 6 פרמטרים.

מה **שאיבדנו** עכשו הוא **זווית** (בגלוּל ה-*shear*), **אוריגינטציה** ו**גודלים** (אותם איבדנו בהעתקות הקודמות שהראינו) וכו'.

פרופורציות נשמרו, קוים מקבילים עדין מקבילים.

זה נראה כך:



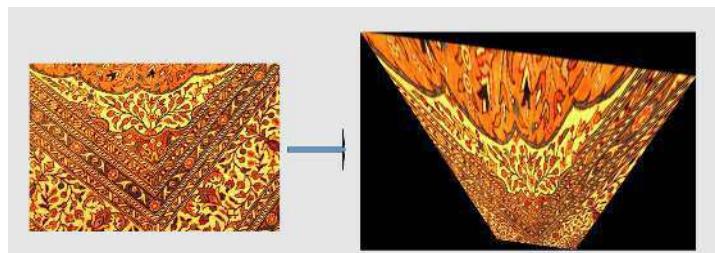
איור 248: המראה *Affine*

טרנספורמציה פרויקטיבית הטלה - *Projective*

עד כה לא התעסקנו בעומק של תמונה. טרנספורמציה זו באה לחתך מענה. זו הטרנספורמציה המורכבת ביותר אליה נגיע והיא נראה כך:

$$T \begin{pmatrix} x \\ y \\ w \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ w' \end{pmatrix}$$

כאן צריכים 8 פרמטרים. זה נראה בערך כך:



איור 249: המראה *Projective*

הטרנס' הזה היא שילוב של טרנס' אפינית עם מה שנקרא *projective warps* - שמשנה את נקודת המבט שלנו. טרנס' זו מיפה נקודות מישור *2D* למישור *2D* אחר. זו הפעם הראשונה שאנחנו מושנים את הקואורדינטה השלישית.

התוצאה של העתקה כזו היא מעין הטלה של התמונה (יש אלמנט של עומק עכשו). אנחנו מושנים את הזרות ממנה אנחנו מסתכלים על האובייקט

עכשו **איבדנו** גם את **קוויים מקבילים**. הדבר היחיד שהעתקה זו בודאות **משמרת** הוא **קוויים ישרים**.

כשאנחנו חוזרים לתמונה אמיתית, אנחנו צריכים לזכור לחלק ב- w' כדי שהקוואורדינטה השלישית תהיה 1.

על כן, לאחר שנפעיל את

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

נקבל שהנקודות החדשות הן, על ידי כפל במטריצה וחזרה למערכת קרטזית:

$$\frac{x'}{w} = \frac{ax + by + c}{xg + yh + 1} \quad \frac{y'}{w} = \frac{dx + ey + f}{xg + yh + 1}$$

ניתן לחשב שהחלוקת ב- w היא מה שמשמעותו לנו כאן אלמנט של עומק.

שאלה אם אנחנו מפעילים כמו טרנס' כאלה ברכף, מתי צריך לחלק ב- w' , אחרי כל כפל מטריצות, בסוף כל השלבים, או זה לא משנה?

תשובה בחישום האמיתיים, כשאנו מזיזים את המצלמה ומוסובבים אותה כדי לחתת תמונה, אנו עושים הכל בתנועה אחת וזה טרנס' אחת.

גם כאן, אולי אנו קופלים כמה מטריצות זו בזו, אך אנו מבצעים רק פעולה אחת ולכן החלוקת תהיה רק בסוף כפל המטריצות - בסוף הפעולה эта.

המקרים בהם נתקלים בטרנספורמציה זו היא כאשר מצלמים אובייקט מצולמות שונות ומטילית למעשה את התמונה על מישור מסוים. כל של בניין הוא הטלה של צילום באמצעות קרטני המשמש.

אנו נלמד בקרוב כיצד, לאחר צילום תמונה של הטלה, נחזיר אותה הפורפורציה כדי לקבל תמונה ישירה כאילו צילמנו ללא סיבוב. כדי לעשות זאת, נסתכל על התמונה שאנו רוצים כאילו ביצענו ממנה טרנספורמציה הטלה וקיבלנו את התמונה שצילמנו. נסתכל על נקודות בהטלה, ונראה איך הן מתאימות לנקודות תמונה שאנו רוצים, ברגע שנמצא מספיק זוגות כאלה, נוכל להשתמש בנוסחה לטרנספורמציה ולקבל מערכת משוואות ולקבל את הפרמטרים הרלוונטיים לטרנספורמציה. מהם נסיק את הטרנספורמציה. למעשה אנו נמצא את הטרנספורמציה באמצעות מציאת הנקודות הנ"ל. שבוע הבא נלמד כיצד למצוא את הנקודות.

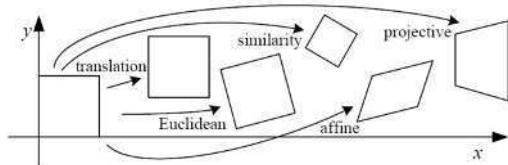
לסיכום, כל מה שראינו מסתכם לתמונה הבאה:



איור 250: המראה לכל הטרנספורמציות שראינו

35 תרגול 6 - Warping ומציאות

נתחילה מ>Showcases קצרה לגבי סוגי הטרנס' שיש לנו:



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

איור 251: סוגים שונים של טרנס' יחד עם דרגות החופש השונות ומה שכל טרנס' משמרת. שימו לב לכך כל הטרנס' הן הכללות של טרנס' אחרות (ראו חיצים באיור). כל העתקות כאן סגורות תחת הרכבה ועם הפיקות

הערה כל טרנספורמציה שומרת חלק מהתכונות אך גם מאבדת כמה. התכונות המדובבות הינן:

- (i) אוריינטציה (נקודות מבט) נשמרות.
- (ii) אורכים נשמרם.
- (iii) זוויות נשמרות.
- (iv) קווים מקבילים נשארים מקבילים.
- (v) קווים ישרים נשארים ישרים.

באյור תחת עמודת ה-"*Preserves*", מופיעות התכונות שנשמרות. כך ניתן להבין שב-*translation* אלו נשמרות על כל התכונות, ב-*rigid* על הכל חז' מאוריינטציה, ב-*similarity*, ב-*Affine* על הכל חז' מאוריינטציה ואורכים, וכן הלאה.

בהתנחת שתי תמונות, כאשר אחת היא טרנס' של השנייה, נרצה לנסות להבין מהי הטרנס' שהופעל על התמונה הראשונה. כדי לעשות זאת, ניקח זוגות של נקודות p', p , כאשר p היא נקודה ב- A - p' היא הנק' ב- B - p . עם מספיק נקודות, יוכל להציגן במערכת משווהות שפתרוניה יספר לנו מהי הטרנס'. כל זוג נקודות נותן לנו שתי משווהות: אחת עבור ערכיו x , ואחת עבור ערכיו y . על כן, עבור כל שתי דרגות חופש של הטרנס' נצטרך זוג נקודות.

בפועל, נרצה לקחת יותר מזוג אחד של נק' לכל שתי דרגות חופש. זאת כדי להתחשב בטעויות מדידה וכו' נshallת השאלה איך אנחנו מוצאים ובוחרים את זוג הנק'. על שאלה זו נענה בהמשך.



איור 252: המראהה למעבר נקודות מתאימות מהתמונה אחת לטרנס' המתאימה שלה

Warping

היא הפעולה של הפעלת טרנספורמציה Image-Warping T , על התמונה. בהינתן תמונה f , והעתקה T , נרצה לחשב את התמונה g שמתאימה להפעלה T על כל פיקסל ב- f . (בעיה זו עולוה לשימוש טריויאלית אך מיד נראה שאין זה המצב) ב כדי לחשב את g נוכל להשתמש בכמה שיטות.

Forward – Warping

שיטה זו פשוטה מאוד. ניקח כל פיקסל (x, y) ונחשב (x', y') בתמונה g באמצעות החישוב הישיר:

שאלת מה נעשה אם פיקסל נוחת "בין" שני פיקסלים?

בגלל שהפעלה T על (y, x) לא בהכרח תביא לנו מספרים שלמים, אנחנו צריכים לחושב מה לעשות אם לדוגמה

$$T(2, 2) = (1.5, 2)$$

תשובה לא נרצה לעגל, כי זה יוצר aliasing וארטיפקטים מכוערים.

אפשרות יותר טובה נקראת splatting - "נפזר" את הפיקסל בין השכנים שלו, בהתאם למרחקו מכל פיקסל (ນctract לנורמל כדי שהערכים לא ישתוללו אך זה הרעינו הכללי).

שאלת מה נעשה אם נוצרים לנו חורים?

אם יש לנו בתמונה g פיקסל (x'_0, y'_0) , אף אחד לא הבטיח שיש (x_0, y_0) ב- f עבורו $T(x_0, y_0) = (x'_0, y'_0)$, לדוגמה, עבור ההעתקה $T(x, y) = (2x, 2y)$ אין אף פיקסל ב- f שעובר ל-(3, 3) (מתקיים $T(\frac{3}{2}, \frac{3}{2}) = (3, 3)$ אך זה לא ערך שלם שמהווה פיקסל ב- f).

תשובה אחרי שנסיים להפעיל את T על כל הערכים ב- f , נctract לעבור על g ולמצוא את כל החורים ולתקן אותם באמצעות איזושהי מניפולציה (אולי טשטוש, אולי משהו אחר). **זה תיקון Dziubek וזו לא אופציה טובה.**

Backward (Inverse) – Warping

בגלל הבעיה של החורים ב- f , **Backward – Warping**, אנו מציגים שיטה אחרת בשם **Foward – Warping**. בغالל שכל הטרנס' שאנו נבדוק איתון, הוא הפיכות (זה לא נכון באופן כללי, רק עם אלה שאנו עובדים איתון), נוכל לכל פיקסל

ליחסב את (x', y') , אז לחתת את הערך ב- $f(x, y) = T^{-1}(x', y')$ ולשים אותו ב- (x', y') .
 ככלומר אנחנו מודמיינים ש- g הוא קובס ריק, וועברים על כל קווארדיינטה ומוחפשים מי הצבע שצורך לשימם במקום ה- (x', y') , בנסיבות מציאות $(x', y') = T^{-1}(x, y)$.

שאלה (שאני לא יודע את התשובה עלייה) איך אנחנו מחליטים מה טווח הפיקסלים (x', y') עליהם אנו עופרים?

תשובה (פוטנציאלית שדניאל אומר לעצמו) אם נפעיל את T על הפינות של f , אנו נקבל שאלה עריכים קיצוניים. האם זה נכון?

שאלה מה קורה אם $T^{-1}(x', y')$ נוחת על פיקסל לא שלם?
 לדוגמה העתקה החפוכה ל- $T(x, y) = (2x, 2y)$ היא $T^{-1}(x', y') = (\frac{1}{2}x', \frac{1}{2}y')$, אז $T^{-1}(3, 3) = (\frac{3}{2}, \frac{3}{2})$ שהוא לא פיקסל ב- f , אז איך ערך נבחר לשימם ב- g במקום ה- $(3, 3)$?

תשובה נבצע אינטראפולציה על הצבעים עם השכנים.

אפשר לעגל ולבחרו את השכן והקרוב ביותר, להשתמש באינטראפולציה בילינארית, bicubic, גאוסיאן וכו'.

אינטראפולציה בילינארית, לדוגמה, מאוד דומה ל-*splatting*.

אם נרצה לחשב את $f(x, y)$, כאשר משתמש בנוסחה הבאה:

$$\begin{aligned} i \leq x \leq i+1 \\ j \leq y \leq j+1 \end{aligned}$$

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

כלומר זהו מעין סכום משוקל של כל השכנים, כאשר המשקל הוא לפי הקרבה לכל שכן.

הערה נשים לב שכאן אין לנו בעיה של פיקסלים ריקים ("חורים") כמו ב-*Forward-Warping*, כי אנחנו שואלים כל פיקסל בתמונה של g מי הערך מ- f שמתחאים אליו? מי הערך שהוא אמור לקבל?

שאלה מה עדיף, *Backward-Warping* או *Forward-Warping*?

תשובה בדרך"כ נעדי, *Backward-Warping*, בדיק בಗל שאין לנו את הבעיה של החורים ולא נctrיך שום – *post-processing*. עקרונית ב-*Forward-Warping* אפשר לפתור את בעיית החורים, אולי באמצעות אינטראפולציה של הערכים הקיימים מסביב לחור בתמונה g , אבל זה יהיה רק אחרי שנסיים למלא את כל הערכים ב- g שהם לא חורים, ואז נctrיך – *post-processing*. בנוסף, אולי תהיה גם אפשרות שגיאת כי לא נדע מה הפרופורציה לפיה צריך למלא את הערכים, נctrיך לעשות טשטוש וכו'.

עם זאת, **בשאיין העתקה הפוכה נשתמש ב-*Forward-Warping*** (שכן אז בכלל אי אפשר להשתמש ב-*Warping*).

נקודות אפיון - Feature Points

ראינו שהינתן לנו זוגות של נקודות בתמונה f , ובתמונה g שהתקבלה מ- f ע"י הפעלה של טרנס' T , אנחנו יכולים לחשב מהי T (כמו הזוג שנדק לה היא בהתאם לדרגות החופש של T).

עם זאת, יש כמה נקודות לא ברורות כאן:

(i) מתי במציאות נגע לUMB שיש לנו את g , f אבל אין לנו את T ?

(ii) איך אנחנו מוצאים זוגות של נקודות מתאימות ב- g , f (אולי בגין זה יכול להיות קל, אבל איך מחשב יעשה את זה)?

(iii) למה אנחנו רוצים למצוא זוגות של נקודות כאלה? הרעיון כבר אלגוריתמים כמו *Lukas – Kanade*

ושםוצאים לנו מה ההזאה בין התמונות.

(i) נדמיין שאנו מצלמים תמונות פנורמה. פנורמה נעשית באמצעות צילום המון תמונות וחיבור חכם שלهن. אם צילמנו שתי תמונות בשביל הפנורמה, כמו באיר הבא, נרצה לדעת איך לחבר אותן, ולשם כך נדרש לדעת מה הייתה ההזאה בין התמונות. כמובן, אנחנו יודעים מהי f – זו התמונה הראשונה שצולמה – אנחנו יודעים מהי g – זו התמונה השנייה – אבל אנחנו לא יודעים מה הייתה T – ההזאה בין התמונות.

הערה נשים לב ש- g , f הם לא בדיקת ההזאה של זה, יש חלק די גדול מהתמונה משתמשת לשתי התמונות וזה ההזאה במקרה שלנו (והחלק שלא משותף לתמונות הוא מה שיוצר את האפקט של הפנורמה כש לחבר את התמונות).

הערה נוסף שבגלל שהיד שלנו לא יציבה, היא עלולה להזיז את המצלמה ככה שהטרנס' עלולה להיות יותר מורכבת מהזאה (אך היא עדין תהיה מטריצה 3×3 כפי שראינו).



איור 253: מימין יש תמונה אחת, ומשמאל תמונה נוספת נוספת שהתקבלה מהזאה אופקית של המצלמה

(ii) על שאלת זו נענה בהמשך בהרחבה.

(iii) הסיבה ש- LK, CC -*sh* לא בהכרח יהיו טובים עבורנו כאן היא שאם צילמנו תמונה אחת, ואז צילמנו תמונה נוספת בתאורה שונה תוך כדי שהיד שלנו זהה כמעט, LK, CC לא יצליחו לזהות מה הייתה ההזאה.

סיבה נוספת היא שבפנורמות למשל עלולה להיות ההזאה גדולה בין התמונות, ובגלל ש- LK, CC -*sh* משתמשים מאוד על זה שהתמונות מותלבשות טוב אחת על השניה. אנחנו נרצה לדעת מה הייתה בין שתי תמונות בפנורמה גם אם הן לא בדיק

מתלבשות זו על זו, כמו באирו לעיל: יש חצי שימושת לשתי התמונות, אך גם חלק שלא משותף (כלומר יש אישahi חפיפה לא שלמה) - LK, CC לא יזהו שהייתה פה הזאה של התמונה.

כדי למצוא זוגות מתאימים של נקודות נדרש כמה דברים:

(i) להזיהות נקודות מעניינות בשתי התמונות.

(ii) לתאר את הסביבה של כל נקודה מעניינת (בנייה *Descriptor*).

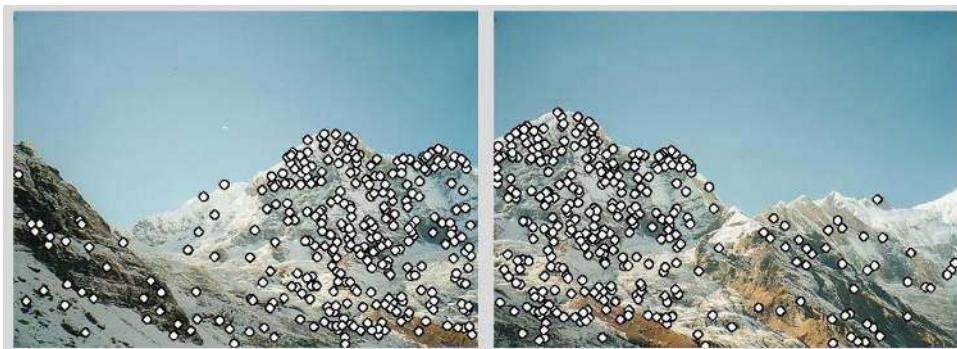
(iii) למצוא את הזוגות המתאימים של הנקודות המעניינות.

(iv) להשתמש בזוגות כדי ל指引 את שתי התמונות ולהתאים ביניהן.

lezahot Nekodot Meuniinot BeSheti HaTmonot - נרצה למצוא נקודות מעניינות (*feature points*) בתמונות, וביצעה התאמות בין שתי התמונות. נקודה שams נעשה את התהילה מספיק טוב לקבל שההתאמות שביצענו הן באמות נקודות מתאימות (זוגות מהצורה $T(x, y), (x, y)$). בנקודות אלה נשתמש כדי למצוא מהי T .

כאשנחנו אומרים נקודות מעניינות, אנו מתחווים שהן **צריכות להיות דיברויות ומשקל לזרותן**. לדוגמה, באירו לעיל של ההרים, לא נרצה לקחת פיקסל של שמיים, כי הוא כלל וכלל לא מיוחד ויהיה לנו קשה למצוא מי מתאים אליו.

הערה נזכיר שוב שבגלא שבחירה שלנו של זוגות הנקודות, אנחנו לא יכולים להיות בטוחים שבחרנו את הנקודות הנכונות, נבחר יותר נקודות ממה שצריך אידאלית. לדוגמה, אם אנחנו יודעים שהטרנס' הייתה טרנס' פרוייקטיבית, יש לנו 8 דרכות חופש ולכן צריך 4 זוגות, ולכן יותר בטעון ניקח לדוגמה 20 זוגות, ככה שהבחירות הלא-נכונות של הזוגות לא יגעו לנו בטרנס'.

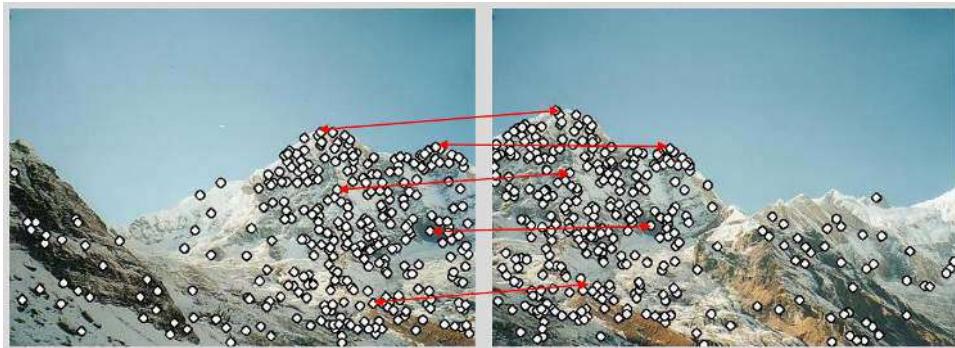


איור 254: מציאת נקודות מעניינות בשתי התמונות

لتאר את הסביבה של כל נקודה מעניינת - (בנייה *Descriptor*) כל נקודה מעניינת שבחרנו, לא ניתן לתאר סטם לפי המיקום שלו. אם נקודה (y, x) בתמונה f הוגדרה כמעניינת, זה לא מספיק עבורנו כדי להבין מי הזוג שלו ב- g . נרצה למצוא דרך לtarget את הסביבה של הנקודה, כדי שנוכל להבין מי הנקודה המתאימה בתמונה g ; לדוגמה, במקומות להגיד שהפיקסל ורוד, נרצה להגיד שהפיקסל ורוד ונמצא בפינה הימנית של המרשלמו העליון בצלחת על השולחן. במצב זה, נוכל רעיונית ללקת לתמונה g ולשים לב שגם בה יש פיקסל ורוד בפינה הימנית של המרשלמו העליון בצלחת על השולחן. אז נסיק שני הפיקסלים הללו מותאימים זה לזה והם מהווים זוג.

למצוא את הזוגות המתאימים של הנקודות המעניינות - כאמור, אחרי שנצליח למצוא נק' מעניינות ולבנות להן *descriptor*-ים,

נראה לאילו נקודות בשתי התמונות יש *descriptor*-ים מתאימים, ואלה יהיו זוג.



איור 255: מציאת זוגות מתאימים של נקודות לפי *descriptor*-ים שלם

להשתמש בזוגות כדי לישר את שתי התמונות ולהתאים ביניהן - אחרי שמנצנו את כל הזוגות, נוכל למצוא את העתקה T של ההזאה בין f ל- g כפי שכבר תיארנו, ונוכל לישר את התמונות באמצעות הפעלת העתקה הפוכה על התמונה השנייה. התוצאה לאחר כל השלבים האלה על האירור של ההרים תהיה כלהלן:



איור 256: חיבור שתי התמונות - סוף התהליך שתיארנו

כשאנחנו מוצאים נקודות - חשוב שהאלגוריתם **תמיד ימצא את אותן נקודות**. אם נרים את האלגוריתם על אותה תמונה כמו פעמים, וכל פעם הוא ימצא נקודות שונות (אם לדוגמה האלגוריתם בוחר ברנדומליות), אין סיכוי שנמצא התאמה ככה, כי אפילו אם התמונות זהות לחלוין לא נקבל את אותן הנקודות.

כשאנחנו מגדירים descriptor - חשוב שהוא יהיה **יחודי, טוב ואמין**. אנחנו צריכים לתאר מספיק טוב את הנקודה כדי שנצליח למצוא מי מתאים לה בתמונה השנייה.

תכונות נוספות שאנחנו רוצים בתהליך מציאת הזוגות:

- שהתהליך יהיה **קל לחישוב** ולא יקח יותר מדי זמן

- אינוריננטי לשינויים בתמונה (כמו רעש, scaling, viewing direction, סיבוב, הזזה, שינויים בתאורה וכו').

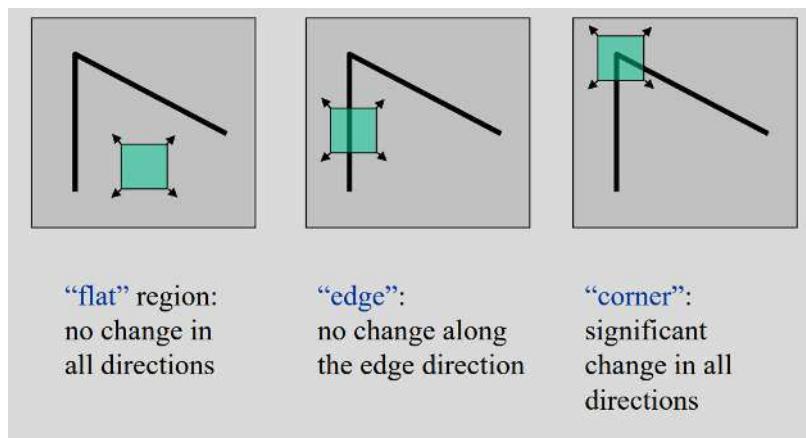
- קל להתקדמות בCOND מנגד מודר של נקודות בתמונה השנייה.

אלגוריתם Feature Points למציאת Harris corner detector

פינות הן נקודות מעניינות, כי הן ייחודיות מאוד. כאשר אנחנו אומרים ייחודי, אנחנו מותכוונים שם נסתכל על חלון קטן ונזיר את החלון הזה קצר, מה שנמצא בחלון ישנה מאוד. נקודה לא מעניינת היא נקודה בה החלון לא משתנה הרבה עם הזרזות (תחשבו על השמיים בתמונה של ההר).

שאלה למה שלא ניקח edges במקומות פינות?

תשובה נדמיין שיש לנו edge אובי, אם נזיר את החלון לעילו ולמטה, החלון עשוי להיות דבב. לעומת זאת, אם אנחנו נזיר איז החלון בטוח ישנה.



איור 257: איזור חלק הוא לא נקודה ייחודית כלל. edge הוא מעט יותר טוב, אך עדין לא ייחודי מספיק. corner הוא איזור ייחודי - לא משנה איך נזיר את החלון מקבל תוצאה שונה.

החלונות עליהם נסתכל יהיו חלונות קטנים בדר"כ (3×3 , 5×5).

שאלה מה קורה אם יש לנו בתמונה דפוסים (patterns)?

תשובה זו אכן בעיה, וקשה להימנע אותה. באյור הבא ניתן לראות כיצד צילו פנורמה של קיר שיש בו דפוס עץ יוצר פנורמה לא טובה, כי הדפוסים הקשו על האלגוריתם לחבר את התמונות כמו שצרכיך. גם אנחנו כבני אדם נתקשה לחבר המונGRAMS שמקורבבות מדים זרים/מאוד דומים.



איור 258: פנורמה שיצאה לא מוצלחת בגלל תבניות בתמונה

מתמטית, אנו מעוניינים למצוא את הנקודה , שאם נזיז את החלון סביבה בכל מיני הוצאות (u, v) , השינוי יהיה גדול לכל v .
את השינוי של מה שקרה לחלון בעקבות ההזזה, נמודד באמצעות הנוסחה

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

כאשר $I(x, y)$ הוא הערך המקורי בנקודה, $I(x + u, y + v)$ הוא פונק' חלון (בדרכ' או *rect* או גאוסיון).

כדי שלא נדרש לחשב המונן דברים, ולעבור על כל הפיקסלים, נעבוד בצורה חכמתה.
בין שאמ הוצאות (u, v) היא גדולה, מן הסתם ש- E -היא גדולה, שכן נרצה להסתכל רק הוצאות קטנות. בגלל שההוצאות קטנות,
nocל להשתמש בקירוב של טור טילור: כ- h קטן ניתן לומר כי $f(x + h) \approx f(x) + f'(x) \cdot h$. או במקרה הדו-מימדי,
בנוסף:

$$I(x + u, y + v) - I(x, y) \approx I(x, y) + I_x \cdot u + I_y \cdot v - I(x, y) = I_x \cdot u + I_y \cdot v$$

כלומר עכשו אנחנו רק מדברים רק על גודל התזזה כפול מה הוצאות. את הוצאות נוכל לחשב פעמי אחת בתחילת הריצה,
כך שלא נדרש לבצע המונן חישובים של הוצאות.
ניסי לב כי מכאן

$$[I(x + u, y + v) - I(x, y)]^2 \approx \left[\begin{pmatrix} u & v \end{pmatrix} \begin{pmatrix} I_x \\ I_y \end{pmatrix} \right]^2$$

כלומר

$$E(u, v) \approx \sum_{x,y} w(x, y) \cdot \left[\begin{pmatrix} u & v \end{pmatrix} \begin{pmatrix} I_x \\ I_y \end{pmatrix} \right]^2$$

ולכן אם נפתח את הריבוע נקבל (ונתעלם מפונק' החלון, נתיחס אליה $c\text{-rect}$, כדי שהיא לא תפריע בכתיבת):

$$E(u, v) \approx \sum_{(x,y) \in w} (u \ v) \begin{pmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = (u \ v) \left(\sum_{(x,y) \in w} \begin{pmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{pmatrix} \right) \begin{pmatrix} u \\ v \end{pmatrix}$$

נוכל לחשב מראש המטריצה בנוסחה ונסמן את המטריצה זו בתור

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

נשים לב ש- M היא מטריצה שמייצגת את ההתנהגות של הנזירות בחalon w .

כך נקבל את החישוב הפשטוט

$$E(u, v) \approx (u \ v) M \begin{pmatrix} u \\ v \end{pmatrix}$$

לכואורה, אנחנו עדין צריכים לעשות את החישוב של E לכל צמד (v, u) . עם זאת, לא כל התוצאות מעניינות אותנו: רק הקיצונית, התוצאה הקטנה ביותר והגדולה ביותר.

זאת כי אם התוצאה הגדולה ביותר והקטנה ביותר, שתיהן קטנות או כל התוצאות בין לבין מאוד קטנות והחלון לא השתנה הרבה.

אם שתיהן גדולות אנחנו יודעים שבטוח החלון השתנה הרבה בכל תוצאה.

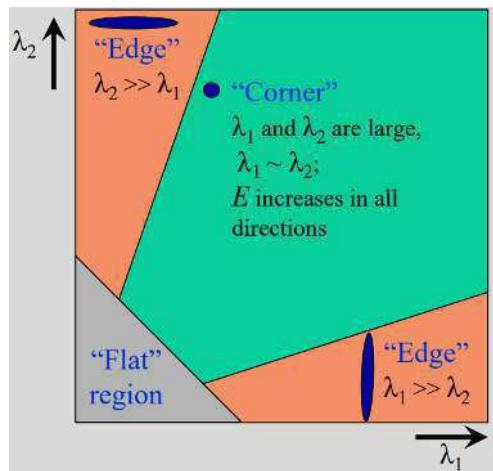
אם אחת גדולה ואחת קטנה אין שאר של איזה הרבה ויש אזורים שכן איזים הרבה.

כיצד נמצא את כיוון התוצאה המינימלית ואת כיוון התוצאה המקסימלית? וקטורים עצמיים! הווקטורים שנוטנים ערכיהם מינימום ומקסימום למשוואות מהצורה $(u \ v) M \begin{pmatrix} u \\ v \end{pmatrix}$ הם לבדוק הווקטורים העצמיים! (נובע מ-*Rayleigh Quotient* ועקרונו המקסימום לוקטורים עצמיים, ראו העשרה בנושא)

אם λ_1, λ_2 הם הע"ע של M , ו- $(u_1, v_1), (u_2, v_2)$ הם המטאים, התוצאה הגדולה ביותר תהיה בכיוון של (u_1, v_1) , והעומצתה תהיה פרופורציונלית λ_1 . התוצאה הקטנה ביותר תהיה בכיוון של (u_2, v_2) , והעומצתה תהיה פרופורציונלית λ_2 .

לא מעניין אותנו גודל התוצאה, אלא רק הפרופורציות ולכן נוכל להתעלם מהו"עים ולהסתכל רק על הע"עים - הם יספרו לנו כמה גדולה/קטנה ההזאה יכולה להיות.

נוכל לבצע את החלוקה הבאה בהתאם לגודל של הע"עים. אזור אפור הוא אזור "חלה" - החלון לא משתנה אף כיוון. אזור כתום הוא אזור פחות חלה - יש לנו ציר בו אין שינוי וציר בו יש שינוי (כגון edge). אזור ירוק הוא אזור בו השינוי גדול בכל כיוון (פינה).



איור 259: חלוקה לשוגים שונים של אזורים בתמונה, בהתאם לע"עים

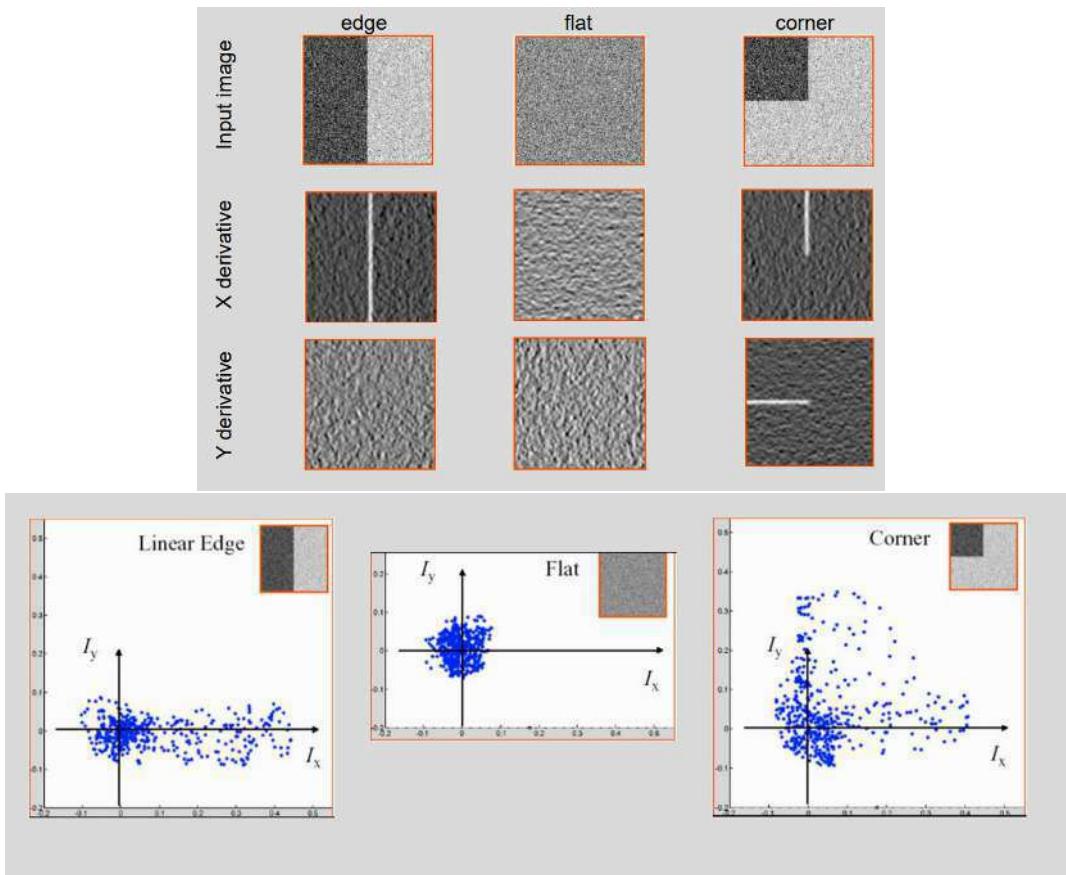
כדי למדוד את היחס בין λ_1 ל- λ_2 , הוצעו המדדים הבאים:

$$R_N = \frac{\det M}{\text{Tr}(M)} = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

$$R_H = \det M - k (\text{Tr}(M))^2$$

על הערכים האלה נוכל להפעיל סף, *threshold*, כדי להחליט האם אנחנו מתייחסים למקומות זהה כפינה. R_N, R_H יהיו גדולים מאוד כשווא פינה, גדולים פחות כשזה edge (אחד מהע"ע גדול והאחר קטן), וקטנים כשזה אזור חלק. אינטואיטיבית, M מתארת לנו כיצד מתנהגות הנגזרות בכיוון x וрок מהסתכלות על M הצלחנו להסיק כמה גדולים/קטנים השינויים בהזאות יהיה.

באזור שהוא edge, יש נגזרת חזקה בכיוון אחד אבל חלהה בכיוון השני. באזור שהוא שטוח, הנגזרת חלהה בשני הכיוונים. באזור שהוא פינה, הנגזרת חזקה בשני הכוונים. לכן זה הגיוני ש- M (שבוב, מרכיבת מהנגזרות) היא זו שמתארת את התנהגות השינויים.



איור 260: בתמונה מימין: התנהגות הנגזרות באזוריים שונים (שטוח, *edge*, פינה).
בתמונה משמאל: גרף של הנגזרות על איזוחו חלון באזוריים שונים (שטוח, *edge*, פינה).

נתאר עתה את האלגוריתם *Harris*

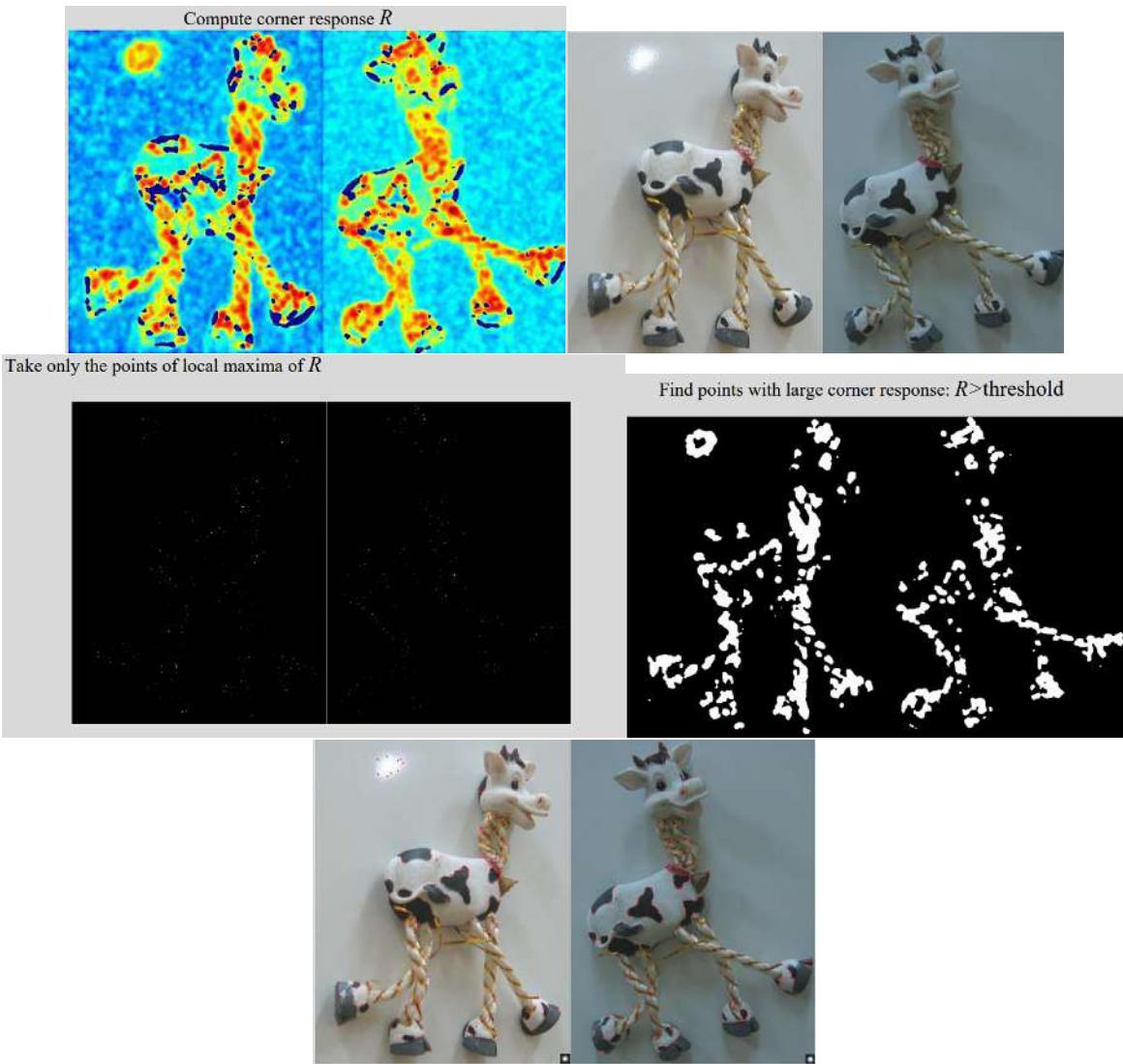
Algorithm 28 Harris Detector

- 1 : Compute corner response R
 - 2 : Find points with large corner response: $R > threshold$
 - 3 : Take only the points of local maxima of R
-

כאשר R יכול להיות R_N או R_H - סקלר שאומר (בעזרת הערכים העצמיים) כמה החלון משתנה בכל הכוונים. בשלב 3 אנו מבצעים NMS-Non-Maximum-Suppression (ליקחים רק את המקסימום בסביבה). זאת כי בפינה, גם פיקסל של פינה וגם פיקסל שלו יראו כמו פינה. לכן, סביב הנקודות המעניינות יהיו לי כמה חלונות שנראים מעניינים. בגלל שאחננו רוצחים נקודת אחת שתיהיה מאוד ייחודית, אנחנו נבחר את המקסימום באותה סביבה - נישאר עם חלון אחד שייצג את המקסימום באותה סביבה.

נדגים עתה על זוג קונקרטי של תמונות כיצד האלגוריתם עובד. שימו לב כיצד התמונה השנייה גם מסובבת וגם עם תאורה

שונה ביחס לשניה.



- איור 261: דוגמה קונקרטית לאלגוריתם *Harris* (תמונה המקור מימין למעלה).
(i) חשב את R - בתמונה השניה ניתן לראות **heatmaps** את הערכים השונים. (שמאל למעלה)
(ii) ניקח רק מקומות בהם $R > threshold$ - קיבל תמונה בולאנית. יש כאן מקומות בהם יש אזורים שלמים לבנים, ולא רק נקודותבודדות. (ימין באמצע)
(iii) על כן, ניקח מקסימום מקומי ואז בכל אזור תהייה רק נקודה לבנה אחת לכל היוטר. (שמאל באמצע)
(iv) נשארנו רק עם הנקודות בהם R הכי חזק. באותו נקודות נשתמש כדי לחפש נקודה מתאימה בתמונה השניה. אלה המועמדים הטובים ביותר. (תמונות תחתונות)

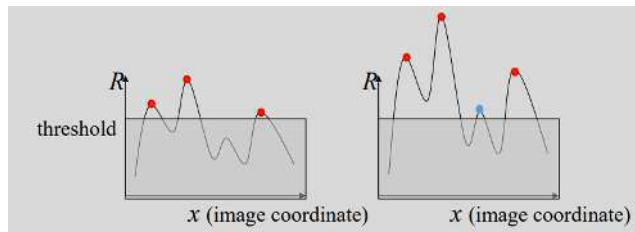
האריס - תכונות

- אינוריאנטי לסיבובים.** אם נסובב פינה, הכיוון של ה"ע"יים ישתנו, אבל הע"יים לא, ולכן R לא ישנה ולכן אם בחרנו נקודה בתמונה האחת, נבחר את אותה נקודה מתאימה בתמונה המסובבת.
- אינוריאנטי (באופן חלקי) לשינוי תאורה.** אנחנו משתמשים בגורמים, لكن זה לא משנה אם התמונה היא I או $I + b$.

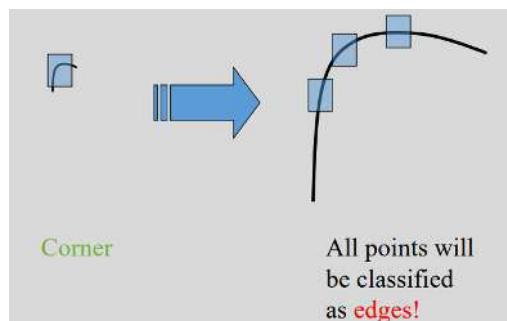
הנוצרת תהיה אותו דבר.
עם זאת, אם ניקח את I ונכפול אותו בקבוע: aI , הנוצרתلن תשתנה. ועדיין, הנקודות שעברו עם את הספ' ישארו עדיין מעל, רק יכול להיות שיתופסו לנו כמה נקודות שייעברו גם הן את הספ', אך הן בטוח לא יהיו מקרים באזור שכן הן יתוופסו כ"נקודה מעניינת" רק אם הן לא נמצאות ליד נקודות מעניינות שמצאו בתמונה המקורית I .
ראו המחשבה באירור.

3. **לא אינוריאנטי לסקלה גאומטרית.** אם נצלם פינה מרוחק, היא אכן תראה כמו פינה. אבל, אם נצלם אותה ממש מקרוב, היא תראה לנו כמו המונ *edges* כי החלון קטן מאוד ביחס לפינה. לכן אם נתקרב ונתפרק מהסצנה, אלגוריתם הא里斯 עלול להיות בעייתי!

מסיבה זו השתמש בפרמיידות יחד עם אלגוריתם האריס!



איור 262: אלגוריתם האריס אינוריאנטי באופן חלקי לכפל בסקלר. הנקודה הכחולה עוברת עכשו גם היא את הספ', לכן אולי נבחר גם אותה כנקודה מעניינת (אלא אם היא נמצאת ליד אחת הנקודות האדומות, אז היא טיפול ב-NMS).



איור 263: אלגוריתם האריס לא אינוריאנטי לסקלות. אם נתקרב לפינה אלגוריתם האריס יחשב שמדובר ב-edge.

Scale Invariant Detection

ניקח את התמונה, נעשה לה פרמידה גאוסיינית, ועל כל סקלה (על כל רמה בפרמידה), נפעיל את אלגוריתם האריס. כך, לכל נקודה נסתכל באיזו סקלה אנוTOPSIMS אותה הכי טוב.

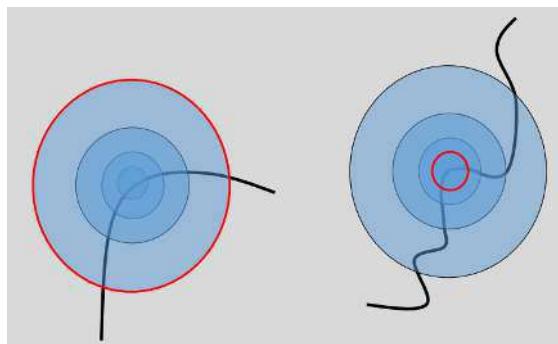
כשהיינו בתמונה בה הפינה קטנה, מצאנו את הפינה כבר ברמה הראשונה של הפרמידה.

בתמונה שngrait גודלה יותר, נצטרך לlect לرمות יותר גבוהות בפרמידה, מעין *zoom - out*, כדי לזהות שמדובר בפינה (שכל הפינה תיקנס בתוך החלון).

יכולות להיות פינות שנותנות בכמה סקלות, ואנחנו כמובן לא נרצה לשמור כמה עותקים של אותה פינה. על כן, בשלב של NMS נלק גם לרוחב וגם לעומק - נעבור גם על הסביבה של התמונה וגם על הרמות בפרמידה. כמובן, נשאיר את הפינה רק במקום בו היא נראית היטב (ערך R הכי גבוה) וגם ברזולוציה, *zoom*, הכי טוב. על כן, אנו שומרים לכל הנקודות בתמונה: האם מצאתי כאן פינה, ובאיזה סקללה? בתוננים אלה נוכל להשתמש לחישוב ה-*descriptor*.

כך יהיו פינות מכמה סקלות שונות, זה בסדר. **אנחנו בוחרים את הסקללה של הפינה הטובה ביותר.**

הבהרה בנגדו לשיטה הקודמת, בה כל נק' אפיון הייתה נק' (x, y), הפעם יש לנו גם אלמנט של עומק בפרמידה, ולכן כל נק' אפיון תהיה שלשה: ($x, y, level$) כאשר *level* הוא הרמה בפרמידה שבחרנו.



איור 264: בעט איתור פינות, יתכן שנבחר סקלות שונות שייצגו את הפינות השונות. בחלק הימני של האיור, הסטפנו בمعالג הפנימי ביותר, אך בחלק השמאלי הינו צריים לעשות *zoom out* לمعالג החיצוני כדי לԶוזות שזו פינה.

נסכם את מה שמצאנו עד כה:

1. מצאנו דרך לזוזות נקודות מיוחדות.
2. מצאנו דרך טרנספורמציה בהנתן נקודות מיוחדות המותאמות לנקודות בתמונה השנייה.
3. מצאנו דרך לזוזות טרנספורמציות بصورة גלובלית כאשר התוצאות נמוכות ויש חפיפה גדולה LK.
4. מצאנו דרך לחשב OpticalFlow באמצעות LK ופירמידות.

נותר לנו למצוא

1. דרך לתאר את הנקודות - *Descriptors* -
2. דרך להתאים בין *Descriptors*
3. לאחד בין התמונות בפנורמה לאחר שנדעஇeo טרנספורמציה כל אחת עברה.

36 תרגול 7 - תיאור הסביבה של נקודה - Descriptors

עד כה מצאנו נקודות מעניינות. נותר לנו למצוא התאמות בין נקודות. נלמד על אלגוריתם RANSAC ליזיהו התאמות לא נכונות. נזכיר כי התאמות לא נכונות יחרשו לנו את כל התמונה. נרשום את השלבים שביצענו ונבצע.

- Alignment - (מציאת הزاה) ביצענו.

- מציאת Feature-Points - ביצענו.

$$\frac{\det M}{\text{Tr}(M)} = M = \sum_{(x,y)} \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}$$

\Leftrightarrow פיניות, בכל כיוון יש שינוי. מצאנו את $\frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$.

\Leftrightarrow נעביר *.threshold*

\Leftrightarrow ניקח נקודות ממקסימום מקומי מסביבה של כל נקודה, כדי שלא נבחר כמה נקודות המיצגות אותה פינה.

\Leftrightarrow אם ביצענו *zoom-in-out* אנחנו עלולים Abed פיניות, נרצה להיות אינוריאנטיים בין שינויים מהתמונה הראשונה לשניה. לכן נחשב פירמידה גאוסיינית, ולכל רמה נריץ את האלגוריתם של האריס. יהיו נקודות שנמצא כבר בהתחלה, אבל פיניות גדולות שנפסיד ברמות גבוהות, יוכל לראות ברמות נמוכות יותר, בהן התרחקנו מספיק מהאובייקט.

\Leftrightarrow יחד עם זאת, אנו מקבל הופעה של נקודות בכמה רמות, וכך את *non-maxima-supression* ביצע גם לעומק.

- התאמה בין נקודות - לא ביצענו.

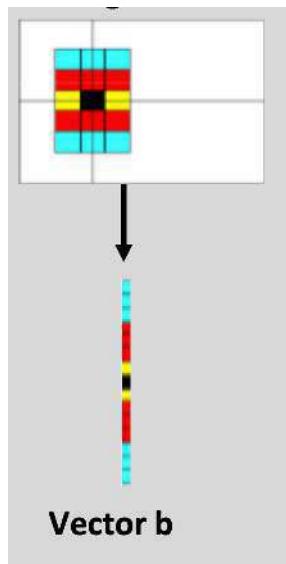
\Leftrightarrow נעיר כי כאשר יש רעש ותבניות שיטה זו לא מספיק טובה ואם ידוע שיש חפיפה נעדיף להשתמש ב-*LK*.

Feature-Descriptors 36.1

נרצה לאפיין סביבה של כל נקודה שתהייה אינוריאנטית לשינויים קלים כמו הزاה או סיבוב או תאורה. נרצה גם שהסביבה לא תהיה כללית מדי, כדי שתציג נכון את הנקודה. יש לנו כאן *Trade-off* בין כלליות לבין ייחודיות. כדי לתאר נקודה, ברור לנו שנטרך יותר מאשר צבע ומיקום הנקודה. נרצה לקחת אזור של הנקודה, ליציג אותה באמצעות וקטור ולהפעיל עליו כל מיני אופרציות שיאפשר להשוות אותו לסביבה אחרת.

פיתוח תיאור אינוריאנטי

ניקח את הסביבה ונציג אותה כווקטור:



איור 265: הפיכת הסביבה לוקטור

מה הבעיה בתיאור זה? הוא לא אינוריאנטי לתאורה, לשיבוב, והזזה. משחו חסר. מה יוכל לעשות כדי לתקן את זה? נחשב את ההסתוגמה של הצבעים, הנגזרות והградיאנטים. משתמש במדדי השוואה כמו EMD . הבעיה בתיאור היא שהוא עדין לא אינוריאנטי לתאורה, אף על פי שהוא כן עמיד יותר בפני שיבוב והזזה.

36.2 אלגוריתם Multi-Scale-Oriented-Patches - MOPS

באלגוריתם זה יש שני חלקים:

- מציאת הנקודות - עם *Multi – Scale – Harris*
- בניית Descriptor מנקודות אלה.

בבנייה זו, אנו נחתוך פיקסלים מהתמונה מטושטשת ולא מהתמונה המקורי, כדי להיפטר מהרעשים, וכן להימנע מהשפעה דרמטית של בחירה לא נכונה של פיקסלים. כאן אנו כן אינוריאנטים לשיבוב ו-*Scale*-*Orientation*. נותר רק להבין איך אנחנו חוטכים את החלון סביב הנקודה.

כשסיימנו את השלב של האריס קיבלנו שלשות של (s, y, x) כאשר s מייצג *scaling* בפירמידה. עכשו אנחנו רוצים למצוא את הזווית שלה - האוריינטציה כדי שנוכל ליחסו קו עם נקודות בתמונה השנייה.

שאלה איך מוצאים שינוי זווית?

תשובה נחשב את תМОונת הגרדיאנטים, נטושש את הגרדיאנטים, ניקח את הזווית של הגרדיאנט, כאשר אנו שוב מטושטשים כדי שנקבל ממוצע של האזור וככה יהיה קל יותר להתעלם מטיעויות של פיקסלקופה פיקסל שם (כמו למשל זווית 90° במקום 94°).

קיבלנו מכאן (θ, s, u, v). עתה, אנו יודעים לבדוק מה האוריינטציה של כל פינה. נחטו סביבות, למשל 40×40 , יחסית גדול, בזווית θ שמצאנו, ככה כשנשתכל על כל ה-*patches* מכל התמונות, נקבל מהם באותה זווית. במילים אחרות, אנו "מיישרים" את הסביבה.

הערה אם את החלון אנו חותכים לפי הזווית, יתכן שהזווית תהיה מספר לא יפה כמו 19° ואז אנחנו נקבל שאנחנו חותכים לכאהורה חזאי פיקסלים ולא רק פיקסלים שלמים. ניתן לטפל בדבר זה באמצעות דברים כמו אינטרפולציה, אך אלה כבר פרטי מימוש.

העיקר הוא שאם חתכנו את החלון לפי הזווית, אז עברו שתי תМОנות שסובבו באוויות שונות, החלון שיתקבל הוא אותו החלון בדוק בוגלן שאנחנו חותכים אותו לפי הזווית.

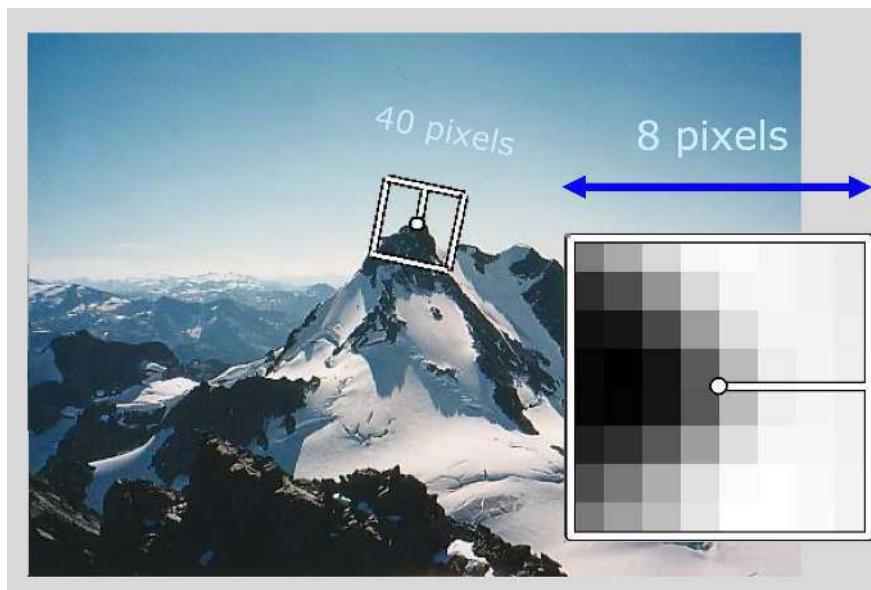
לסיכום פתרנו את:

(i) בעיית הסקלות - כי מראש לקחנו את החלון בסקללה הנכונה - בשתי התמונות בחרנו לחת את החלון הסקללה בו הפינה נכנסת יפה בחalon, ולכן זה מיישר את הדברים להיות באותו גודל.

(ii) בעיית הזווית - כי ללקחנו את החלון בזווית הנכונה.

עד כה:

- נחשב תמונה גרדיאנטים.
- נטשטש אותה.
- לכל נקודה מעניינת נחשב את הזווית לפי הגרדיאנט.



איור 266: זווית של נקודה והסביבה המסובבת

יחד עם זאת, מפתחי MOPS גלו שאפילו עם סביבה של 40×40 זה לא מספיק כלל. מסתבר, שמספיק לדגום את האזור בתדריות נמוכה יותר, ולקבל את כל המידע בסביבה בצורה מספקת. ניקח תדריות נמוכות, למשל נדגום כל פיקסל חמישי, ונקבל סביבה כללית יותר. במקרה זה, נקבל סביבה חדשה של 8×8 שמייצגת סביבה של 40×40 .

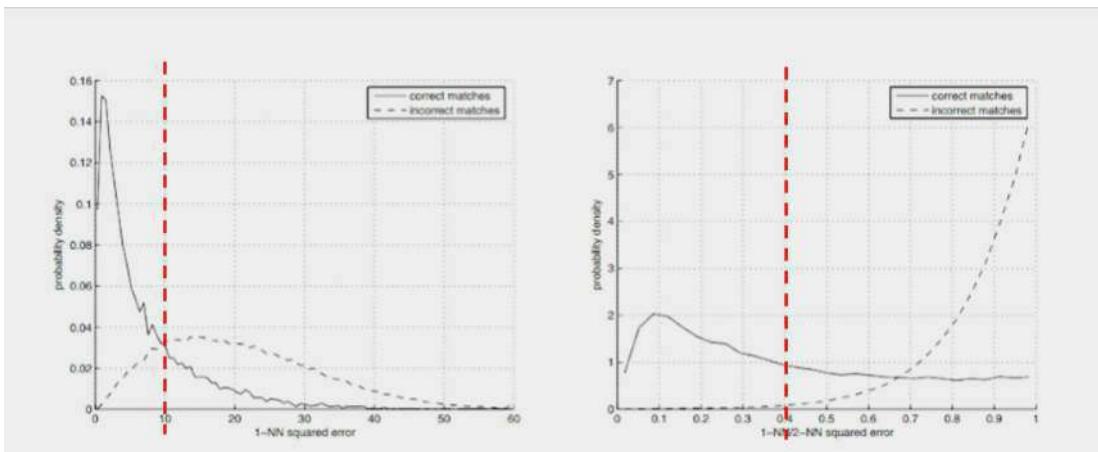
בעיה. דוגמה זו יוצרת *Aliasing*, לא טוב.

תמיד לפני שאחננו דוגמים, צריך לטשטש. לכן, כדי לטפל בבעיה זו, נדגום 8×8 מרמה נמוכה יותר בפירמידה, וככה נפטר מאפקט *aliasing*.
אבל, איך נטפל בשינויו תאורה? נרמל כל חלון לפי הנוסחה

$$I' = \frac{I - \mu}{\sigma}$$

כאשר μ הממוצע של החלון ו- σ השונות שלו. בצורה זו אנו מכניסים את הממוצע לאפס ואת השונות הופכים לקבועה, ככה של הסביבות בעלות אותה שוננות ואותו ממוצע. מכאן, גם אם שני *patches* עם תאורה שונה, אבל מאותו אזור, נקבל שהם מאוד דומים.

קיבלנו עד כה עירימות של *patches*, נותר לנו להתאים *patches* אחד לשני. אלגוריתם Mops משתמש ב-Wavelet Transform המשמש ב-*patches* אחד לשני. עליהם לא עמוקיק. התוצאה היא וקטור בגודל 64. ניקח את וקטור התוצאה, ונחשב את המרחק האוקלידי שלו מכל אחד מהווקטורים בתמונה השנייה. נחשב את היחס הווקטור הקרוב ביותר לווקטור שלו (NN - Neighbor Nearest) בין האחד הבא הכי קרוב. נעביר *Threshold* ההסתאמות:



אייר 267: בתמונה השמאלית, ניתן לראות את ההתפלגות של המרחקים עבור התאמות נכונות. רוב המרחקים נמוכים, הגוני, כי אם האזוריים מותאים אז המרחק קטן. בכו המוקו יש התאמות לא נכונות, ולכן גם נعتبر (הקו האדום) ניקח כמות גדולה של התאמות נכונות, אבל גם הרבה התאמות לא נכונות. עקב לכך, יהיה מוכנים יותר על כל התאמות נכונות כדי להימנע מבחירה של התאמות לא נכונות. בגרף הימני מוצג היחס בין ההתאמות לבין A לבין B ההתאמות השניות הכח הטובה. אם A הוא אכן השכן שלנו, אז B הוא לא השכן ולפניהם נצפה כי B גדול, יש כאן הבדל מהותי, ולכן $\frac{A}{B}$ קטן. אם A לא שכן, גם B לא שכן ולכן נצפה ש- A, B בערך מאותו פרופורציה ולפניהם נצפה ש- $\frac{A}{B}$ לא קרוב לאפס. על כן, נוכל להעביר (קו אדום) ולקחת את התאמות עם יחס קטן, וכך, אף על פי שאיבדנו התאמות טובות, לא ללחננו המון התאמות לא טובות ובכך מונעו הרבה ארטיפקטים. יתר על כן, נזכיר כי אנו צרכים מספר קבוע של התאמות לכל Patch, לא גדול מדי וכן גם אם איבדנו התאמות נכונות, עדין יש מספיק כדי למצוע את השגיאה. נבהיר כי התאמה טובה ביותר אינה בהכרח ההתאמה הנכונה, שכן ניתן דמיון בין אזוריים שונים בתמונה. לכן אנו משווים לתמונה הטובה ביותר הבאה, עושים פרספקטיבה.

36.3 אלגוריתם SIFT

רקע היסטורי אלגוריתם משנת 1999, מאוד שימושי בתחום ה-ComputerVision. מדובר בעבודת נמלים, ניתן למצוא שם המון מספרים "הזויים" שם תוצאה של ניסוי ותיהיה. בהתחלה לא קיבלו את המאמר, אך בסוף הוא הפך לסנסציה.

האלגוריתם מוצא את נקודות הקיצון, וגם מתאים את ה-*pathches* אחד לשני. את הנקודות מחשבים עם DOG-DifferenceOfGaussian אחד לשני. המוחזר שלשה s, y, x , גם כאן נמצא θ - את האוריינטציה ונקבל רביעה (x, y, s, θ) . כאן נבצע היסטוגרמה של הגרדיאנט וניצור סביבה עליו.

בדומה לאלגוריתם האריס, בורר לנו שלא מספיק להסתכל על סקלה אחת של התמונה. לכן, טבעי שנסתכל על פירמידה של התמונה. הפירמידה שנשתמש בה היא פירמידת DOG שכוללת טשטוש, דגימה חדש וחישור, לא בהכרח בין שתי רמות סמוכות. למעשה, פירמידת לפלאין היא מקרה פרטי שלה.

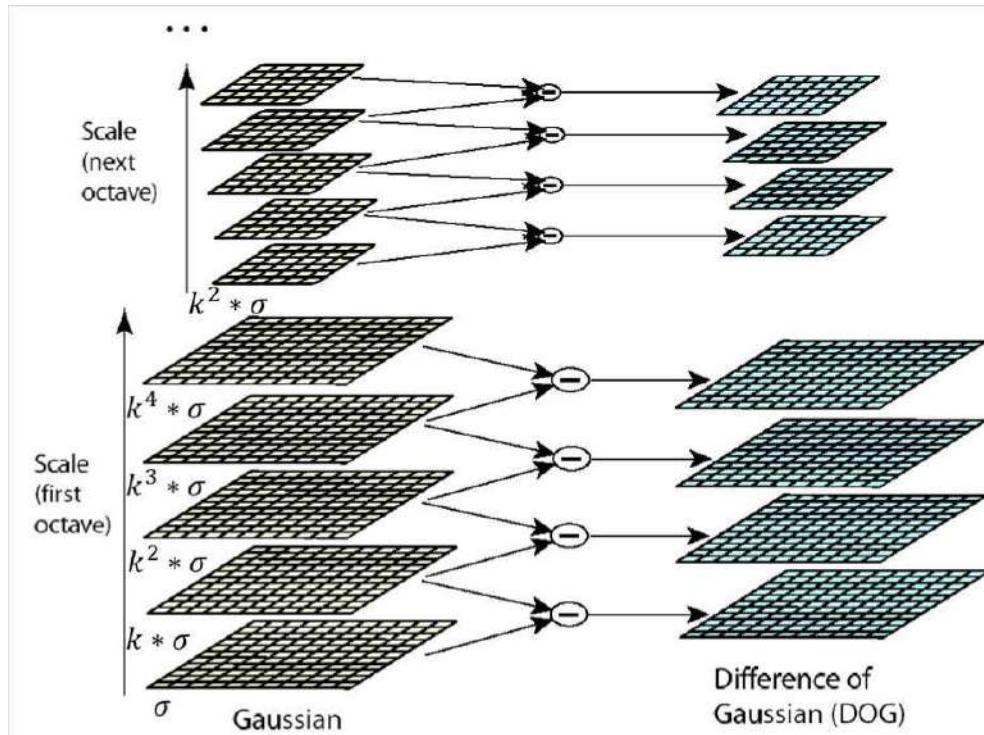
באלגוריתם עצמו אנו בונים פירמידה באופן הבא:

- נטשטש את התמונה ב-5 רמות, ללא הקטנה.

▷ למה 5 רמות? כי זה מה שנמצא הכי טוב.

- נחסר כל זוג ונקבל 4 תמונות של הפרשיים.

- לכל זה אנו קוראים אוקטבה.
- את 5 הרמות המטושטות לוקחים ומקטינים. מטושים שוב, ומתקבלים אוקטבה חדשה.
- סך הכל אנו יוצרים 3 אוקטבות.



איור 268: האינטואיציה מאוד דומה לפירמידת פלסייאנו. החיסור בין התמונות נותן כל מיני *Edges, corners* ושם נחפש ערכי קיצון. האינטואיציה היא שאמ לאחר הטשטוש משחו נושא, כנראה שיש שם משהו מעניין, הרי רקע אחד פול לאחר טשטוש והחסרה.

בכל החיסורים אנו מוחשיים נקודות מינימום ומקסימום, כדי לדעת האם היה שינוי קריטי בתמונה.

כיצד נמצא נקודות קיצון?

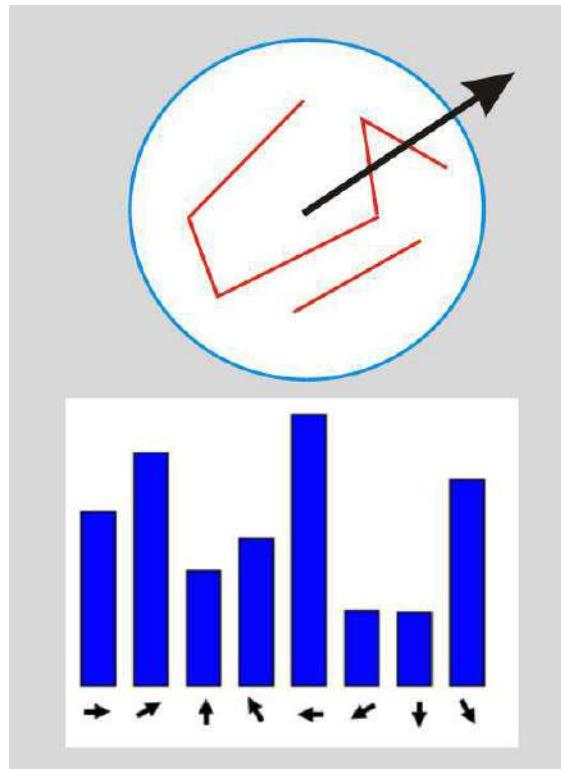
נסתכל על סביבה של הנקודה, ואם היא מקסימלית בסביבה נבחר אותה. אנו משתמשים על סביבה عمוקה - 8 פיקסלים מהרמה הנוכחית בפירמידה (כי זה לא כולל את הנקודה שלנו עצמה), 9 פיקסלים מהרמה הקודמת, ו- 9 פיקסלים מהרמה הבאה.

הערה. פרמטר הטשטוש של כל רמה קבוע מראש. לאחר שלב מציאת המינימום והמקסימום יש שלב זריקה של נקודות. לא נתעמק בהזאה.

עתה, עלינו לחשב את הزاויות כדי לקבל אוריינטציה, שכן קיבלנו עד כה רק (s, y, x) . כדי לעשות זאת, נחשב את הגרדיאנט של סביבה של כל נקודה. למשל סביבה של 5×5 וניתן את הכיוון הממוצע של הסביבה, כדי למצער שגיאות. בפועל, אנו בונים היסטוגרמא של 36 תאים, כלומר כפיצוות של 10 מעלות. לכל נקודה בסביבה נוסיף ערך לעמודה בהסתוגרמא המתאימה

לزاوية שלה. בנוסף, היחסה ממושקלת, אנחנו לא סתם מוסיפים 1 כל פעם: נקודות **קרובות** יותר למרכו החלון יקבלו משקל גדול יותר בעוד נקודות רחוקות יותר, יקבלו משקל קטן יותר - כך נמנע מרושים והשפעה של אלמנטים רחוקים. המשקל תלוי גם **במגנטיותה של הגרדיאנט** בנקודה - אם השינוי יחסית גדול, ניתן לזה משקל גבוה, אבל אם השינוי נמוך, לא נתיחס אליו יותר מדי, למשל עבור זווית 17 ומגנטיות נמוכה, ניתן לה חישיבות נמוכה.

לבסוף, ניקח את העמודה הגבוהה ביותר וזה יהיה הכיוון שלנו - θ .

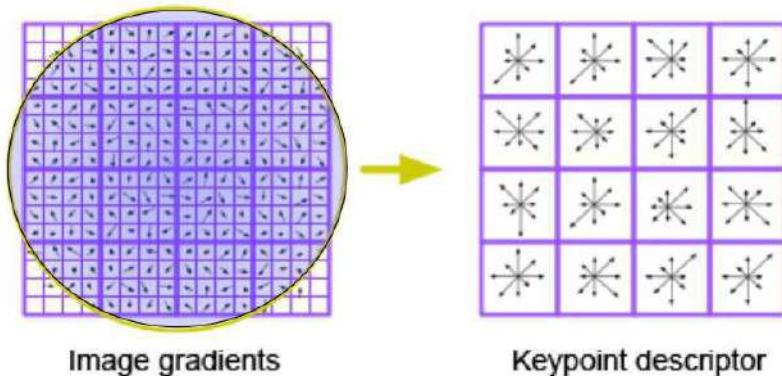


איור 269: המראה לעמודות ההיסטוגרמה של הזווית ולכיוון הממוצע בתמונה

במקרה של תיקו, אנו לוקחים את הנקודה **פערמים** - כל פעם עם זווית אחרת.

עכשו נוכל לבנות את ה-*descriptor*, כאשר יש לנו את כל המידע (x, y, s, θ)

נלק לנקודות המיעדות, אבל בתמונה הגרדיינט וניקח חלון של 16×16 . נחלק את החלון ל- 4×4 אזוריים בגודל 4×4 כל אחד. לכל אזור נציג גם היסטוגרמה, עוד יותר מצומצמת ממה שקיבלו עס רק 8 כיוונים הפעם, כדי שנדע את ההתנהגות כללית של האזור. לפני בניית ההיסטוגרמה אנו מטשטסים את האזור, כדי שנמציע את השגיאה. ההיסטוגרמה עצמה גם היא ממושקלת לפי קרבה ומגנטיותה. סך הכל קיבל *Descriptor* של 128 עמודות (4×4 תאים כפול 8 הכוונים בהיסטוגרמה).



איור 270: ניתן לראות את החלוקה ל- 4×4 אזורים ואת האזוטה של כל אזור

יחד עם זאת, כל האזוטים הנ"ל לא מיושרים! הם מסובבים באזוטה - לא התחשבנו באזוטה הראשית שמצאנו קודם. כדי לטפל בזה, נחסר מכל אחד מהם את האזוטה הראשית שמצאנו בהסתוגרמה הגדולה של ה-36. קיבלונו עד כה וקטור של 128 עמודות שהחסרנו מכל אחד את האזוטה הגדולה מההסתוגרמה הגדולה. האם סיימנו? לא, צריך לטפל בהבדלי תauraה. בשלב האחרון הוא נרמול תauraה, כדי להבטיח אינוריאנטיות טובה. כדי לעשות זאת, אנחנו מנורמלים את וקטור ה-128 עמודות לוקטור ייחידה. אנחנו מעבירים *Threshold* של 0.2 כדי לוסת את ההשפעה של האזוט על מדי הקרבה בהמשך. זהו, זה המתאר שלנו.

הערה. *SIFT* היה פטנט עד לפני כמה שנים והוא יקר לשימוש בו עד אז, והוא נחשב לפורץ דרך. עקב לכך התפתחו עוד אלגוריתמים.

36.4 אלגוריתם RANSAC

ממה שראינו עד כה קיבלנו זוגות של התאמות. אבל אנו רוצים להימנע מהתאמות לא נכונות בצורה מוחלטת, ולכן להוסיף שלב כדי לנחות נקודות כהן.

האלגוריתם שנלמד עכשו הוא אלגוריתם ממושפה של אלגוריתמים רובסטיים שבכלל נוגעים בבעיה כללית יותר - התאמת של מודל לממדיות.

הרעיון של שיטות רובסטיות הוא למצואו איזשהו פתרון שנוכל להתאים באמצעות שיטות מוהנוקדות עם התאמת לא טובה - נדע שהם קיימות, לא נדע איפה, אבל נוכל להתמודד עם זה.

נרצה לתת דוגמה לאלגוריתם שאינו רובסטי. *Least Square* למשל, מאפשר לנו למצוא התאמת של קו לנקודות, בצורה טובה. אבל, מה אם נוסיף שתי נקודות שלא על הקו? מזעור השגיאה תביא לכך שהקו יתעוזת לחולין. זו לא התנהגות רובסטית, אלא התנהגות הפוכה. *Least Square* למעשה אלגוריתם גרווע לעד זה.

אלגוריתם RANSAC (Random-Sample-Consensus) (1971) אمنם ישן אך עדין בשימוש. הרעיון המרכזי הוא בחירה של נקודות ראנדומליות אשר בוחן נשתמש כדי להתאים את המודל, נעשה זאת איטרטיבית ובתוך הכל נקבל בשלב מסוים אוסף ללא שגיאות

- *Outliers*.

לאלגוריתם שלושה צעדים עליהם נחזר:

- נדגם את המינימום נקודותו אותן צריך כדי להתאים למודל.

▷ 2 נקודות בשביל, *Rigid*, או נקודה אחת בשביל *Translation*.

- נחשב את המודל לפי נקודות אלה

▷ למשל - מה הטרנספורמציה המתאימה, מה קו המגמה המתאים.

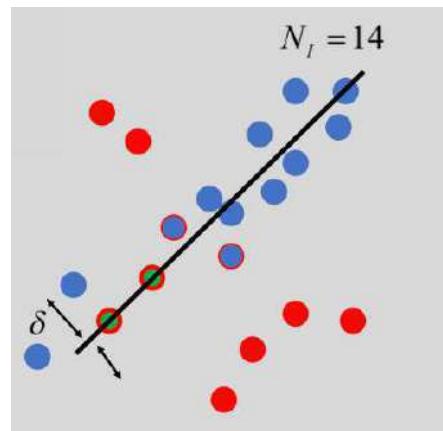
- ניתן ציון להטאמה זו לפי כמה הנקודות האחרות מסכימות אליה

▷ האם כל הנקודות האחרות נמצאות בקו זה?

▷ זהו למעשה ציון למידת ההסכמה של שאר הנקודות שלא בחרנו למודל.

בסוף נחזיר את המודל שהייתה בו הכפי הרבה הסכמה. האינטואיציה היא שאם קיבילנו הסכמה רחבה על אוסף מסוים, לא יתכן שהוא לא נכון.

הערה במקרים להחזיר את המודל שהייתה בו הכפי הרבה הסכמה, אנחנו יכולים להחזיר את כל הנקודות שהייתה ביןיהם הסכמה כי הם לבטח *inliers*, ואז להשתמש בהם בשיטת *least squares*.



איור 271: עבור שתי הנקודות הירוקות שבחרנו ראנדומלית, ההתאמנה טובה מאוד, בעוד עבור שתי הנקודות האדומות בחוץ ההתאמנה לא תהיה טובת.

נרצה להבטיח שהסתברות p מסוימת, למשל $p = 0.99$, נקבל התאמנה טובת לאחר מספר קלשהו של איטרציות. מסתבר שמספרם לרוב

$$N = \frac{\log(1-p)}{\log(1-\omega^s)}$$

כאשר

s - מספר הנקודות שצורך כדי לחשב את המודל. במקרה של קו מוגמת זה 2 במקרה של הומוגרפיה זה 4.

ω - ההסתברות שלנו לקבלת *inliers*. אנחנו לא יודעים אותה עדיין.

מכאן ω^s זו ההסתברות שבדגימת s נקודות כולן יהיו *inliers* (למענה זה לא מדויק להגיד כי אסור לבחור אותו פעמים אך זה טוב בקירוב).

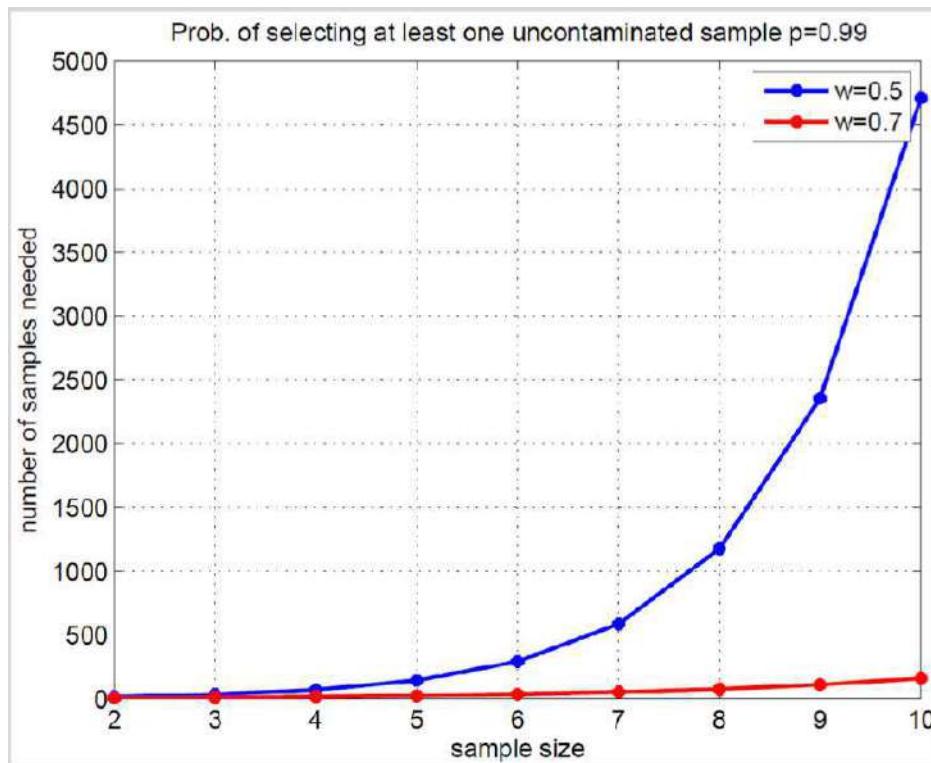
מכאן $\omega^s = 1 - \omega^{s-1}$ זו ההסתברות שהאיטרציה לא טובה - המודל לא יהיה מספיק טוב.

p - זו ההסתברות שנרצה להשיג.

כדי לחלק את הביטוי הזה נבהיר כי $p = 1 - \omega^{s-1}$ זו ההסתברות לקבלת איטרציה לא טובה. על כן, $1 - p = (1 - \omega^s)^N$ שכן החסתברת שאף פעם לא הייתה איטרציה טובה זו ההסתברות שמתוך N איטרציות לפחות אחת תהיה מושלמת. על כן

$$N \log(1 - \omega^s) = \log(1 - p)$$

ולכן $N = \frac{\log(1-p)}{\log(1-\omega^s)}$. כדי לקבל תחושת התמצאות נבייט בגרף הבא:



איור 272: עבור מספר נמוך של נקודות צריך מעט מאוד איטרציות כדי להגיע לוודאות אותה הן מוחפשים, אך ככל שהן עלות כך כמות האיטרציות עולה, כמוון שככל שכמות *inliers* גבוהה, כך צריך פחות איטרציות.

לשם המראה מספרית, עבור 4 דגימות למודל, קיבל שבסקרה הנורווגית ביוצר נציגך לרוץ 72 איטרציות, בהנחה שרוב הנקודות

הן לא :outliers

- $p = 0.99$
- w : inliers probability,
 $1-w$: outlier probability

Sample Size s	proportion of outliers ($1-w$)							
	5%	10%	20%	25%	30%	40%	50%	
2	2	3	5	6	7	11	17	
3	3	4	7	9	11	19	35	
4	3	5	9	13	17	34	72	
5	4	6	12	17	26	57	146	
6	4	7	16	24	37	97	293	
7	4	8	20	33	54	163	588	
8	5	9	26	44	78	272	1177	

איור 273: המקסימום הוא 72 איטרציות במקורה של הומוגרפיה, שזה בדיקת המקורה שלנו

בזאת הטבלה האנו נוכל להתאים בזמן הריצה את כמות האיטרציות שאנו רוצים לבצע:

נניח שאנו נמצאים במקורה הגרוע ביותר בו יש 50% outliers בלבד. נתחל משתנה $i = 0$ שיספור את כמות האיטרציות, ולפי הטבלה קבוע משתנה $i = 72$ (כי אנו רוצים $s = 4$).

נתחיל לזרז וכל איטרציה נעלמת $i = 1$. אם נמצא באיטרציה כלשהי שיש קונצנזוס של 80% מהנקודות, נבין שאלה בהכרח $i = 9$ ורץ $i = 9$ רק 20% outliers וכן לעדכן לפי הטבלה $i = 9$.

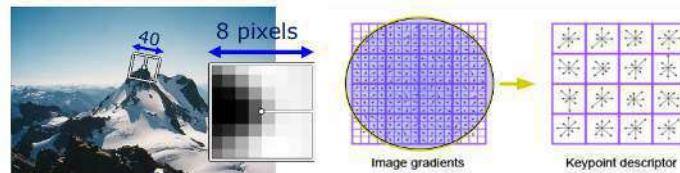
כך לא נצורך לזרז 72 איטרציות כדי שהחנו בהתחלת (ואיפלו אם בשלב זה $i > 9$) פשוט נעזר את הריצה כי נבין שסימנו, כי i גדול מהערך החדש של iterations, קרי $i > iterations$.

נבהיר כי אנו מניחים שאנו יודעים את אחוז-h-inliers המופיע ב- ω . כמובן שבפועל אנחנו לא יודעים אותו, אבל אם נניח שהוא לפחות 50%, נוכל להתחיל לזרז לפי המקורה הגרוע ביותר. במהלך הריצה נחשב שוב את מספר האיטרציות הדרוש לעדכן.

דמיון ושווי בין SIFT-L-Harris-MOPS

نبיט בטבלה הבאה המשווה בין MOPS-L-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)



אייר 274: השוואה בין *SIFT* ל-*MOPS*

הערה שימוש לב שב- $NN - 1$ לא אומר אחד פחות NN : הכוונה היא לשכן הקרוב ביותר. $NN - 2$ הוא השכן השני הקרוב ביותר.

גilio נקודות עניין

ב-*MOPS* אנו משתמשים באלגוריתם *Harris*.
ב-*SIFT* אנו מוצאים נק' קיצון בהפרש גאוסיאנים (*DOG*). בغالל שלפלסיאן זה נקודת פחותה הסביבה שלה, יתקבל הערך הגבוה ביותר כשהנקודה מאוד שונה מהסביבה שלה - כמו בפינוט.

מציאת כיוון (אוריגינטציה)

אם פעמי אחת צילמונו תמונה ישר ובפעם השנייה סובבנו את המצלמה, צריך להתחשב בכך. אם אנחנו מניחים שהצלמה ישירה אין צורך לעשות את זה, אך רוב הזמן אין זה המצב.
ב-*MOPS* אנו מוצאים את הכוון לפי הגרדיאנט.
ב-*SIFT* אנו עושים היסטוגרמאות של כיוונים ובורחים את הכוון הדומיננטי ביותר.

בנייה ה-*patch* ואינוריאנטיות ל-*scale*

ב-*MOPS* ה-*patch* שלנו הוא חלון בגודל 8×8 שנלקח מהתמונה ברמה שבוגה ב-2 רמות בפרמידה מהרמה שבה גילינו את הנקודה עם *Harris* (כלומר זה $40 \times 40 \simeq$ ברמה בה גילינו את הנקודה).
ב-*SIFT* אנו לא לוקחים חלקים מהתמונה, אלא בונים היסטוגרמות של כיוונים של גרדיאנטים: אנו לוקחים אזור בגודל 16×16 מהתמונה הגרדיאנט סביב הנקודה שלנו. אנו מחלקים את האזור ל-16 אזוריים בגודל 4×4 כל אחד. לכל פיקסל אנו מח�בים

את הגרדיאנט והכיוון שלו, ובונים היסטוגרמה של כיוונים לכל אחד מ-16 האזוריים. "ההצבעה" של כל פיקסל בהיסטוגרמה ממושקלת ע"י המרחק של הפיקסל מהפיקסל המרכזי (נקודות העניין) וע"י הגרדיאנט של אותו פיקסל. היסטוגרמות אלה, יהיו ה-*patch* שלנו, ככלומר טה"כ זה וקטור בגודל $128 \times 4 \times 8 = 4 \times 4 \times 8$.

אינוריאנטיות לשיבוב

ב-*MOPS* מראש ה-*patch* נלקח באותו כיוון של האזוטה העיקרית.
ב-*SIFT* אנו מחסרים את האזוטה העיקרית מכיווני הגרדיאנטים.

אינוריאנטיות לעוצמת אור

שני האלגוריתמים משתמשים במרקם פשוט
ב-*MOPS* לוקחים כל נקודה ב-*patch*, מחסרים ממנה את הממוצע של ה-*patch* ומהלקים בסטיית התקן.
ב-*SIFT* ה-*patch* הוא וקטור כיווניים. לכן אם נגיד נכפול כל ערך בתמונה המקורית ב-2 (הגברת של האור) אז הנגזרת גם היא גדלה פי 2. לכן, מנורמלים את וקטור ה-*patch* שלנו כדי שיחיה וקטור ייחידה. בנוסף, לוקחים סף ב-2.0.

השווואה בין שני patch-ים ומטריקת מרחק

ב-*MOPS* ה-*patch* הוא מטריצה 8×8 , לכן פשוט نتيיחס אליו כוקטור, ככלומר כדי להשוות בין שני *MOPS*-ים אנו מחסרים נקודתיות כל שני ערכיהם ומעלים בריבוע את התוצאה.
מטריקת המרחק, ככלומר הדרך להגיד כמה אני מתאים ל-*MOPS* שהכי דומה לי בתמונה השנייה, תהיה לחלק את המרחק שלו מהשכן הקרוב ביותר אליו בתמונה השנייה במרקח שלו מהשכן השני הקרוב ביותר אליו בתמונה השנייה.
כך גם ב-*SIFT*.

סיכום פעולות שני האלגוריתמים

נסכם את אופן הפעולה של *MOPS*:

- נמצא נקודות עניין באמצעות multi-scale-Harris-Corners (x, y, s) .
- Calculates the orientation of the feature point using the Harris-Corners algorithm.
- Computes the third-order derivative of the image at the feature point using a 3D Hermite filter.
- Computes the gradient of the image at the feature point using a 3D Sobel filter.
- Computes the Hessian matrix of the image at the feature point using a 3D Hessian filter.
- Computes the eigenvalues and eigenvectors of the Hessian matrix.
- Computes the orientation of the feature point using the eigenvectors.
- Computes the mean and covariance of the feature point's neighborhood.
- Computes the probability distribution of the feature point's neighborhood.
- Computes the feature vector using the mean and covariance of the feature point's neighborhood.

זה שקול ללקחת *patch* בגודל 40×40 מהתמונה המקורית רק בתדריות נמוכה יותר (טשטוש). הסיבה שהיינו צריכים את הטשטוש זהה היא עמידות לטבעיות, זהה של פיקסל הצידה וכו' (misregistration/misalignment).

- נormal את ה- $patch$: $I' = \frac{I - \mu}{\sigma}$
- (שלב שידלגו עליו) נפעיל *Haar wavelet transform* על ה-*patch* זה יהיה ה-*.descriptor* (.descriptor)
- סכם את אופן הפעולה של *SIFT*:
- מציאת נקודות עניין - נקודות קיצון בהפרשי גאוסיאניים (*DOG*): (x, y, s) .
- ▷ חישוב שלוש אוקטבות.
- חישוב אורינטציה באמצעות ליקית המקסימום מהיסטוגרמה ממושקלת של כיווני הגרדיאנטים בסביבה קטנה של הנקודה (θ).
- ▷ המשקל הוא ע"י מרחק מהנקודה ועוצמת הגרדיאנט.
- ▷ בהיסטוגרמה עושים קוונטייזציה ל-36 כיוונים.
- נחטוּ *patch* בגודל 16×16 (לחתוּ במקביל לצירים, לא לפִי θ כמו *MOPS*) סביב הנקודה שלנו מתומנת הגרדיאנטים בסקללה s שמצאנו.
- ▷ נחסר מכל הزواיות את θ ונחלק את ה-*patch* ל- $16 \times 4 \times 4$ קבוצות של 4×4 .
- ▷ לכל קבוצה נחשב היסטוגרמה שעשויה קוונטייזציה ל-8 כיוונים.
- ▷ את כל ה-*bins* מכל ההיסטוריה נקשר (על ידי החסירה של הزواית מההיסטוגרמה הגדולה) לוקטור אחד באורך $4 \times 8 \times 8 \times 4 = 128$. נormal אותו, ניקח *threshold* מעל 0.2 ונnormal מחדש.
- מטריקת המרחק שלנו תהיה לפי מרחק אוקלידי.

37 תרגול 8 - תפירה באמצעות Min-Cut, Blending, תכנון דינامي ו- Min-Cut

אחרי שמצאנו את הטרנספורמציות בין התמונות, נותר רק לחבר את התמונות.
תחילה, ניצור תמונה שחורה (מטריצת אפסים) שהיא תהיה הקנבס שלנו ועליה נדיביק את התמונות שלנו.

שאלה איך נמצא את גודל הקנבס?

תשובה כל שעליינו לעשות הוא להבין מהם קצוות התמונה (ימני עליון, שמאלית עליון וכו'). בעצם נפעיל על הפינות של כל התמונות את הטרנס' שלhn ונראה מהם מרכז ה- x , y הגדולים והקטנים ביותר - אלה יהיו הגבולות שלנו.

אחרי שהכנו קנבס, נעשה לתמונות *warping* לפי הטרנס' וכך נקשרו אותן ונקם אותן באותו מקום בו הן אמורות להיות.
הערה לכל התמונות אנו מבצעים *warping* ביחס לאיזה תמונה ייחוס שנבחר מראש (לDIG' האמצעית). וה-*warping* נעשה בזוגות כלומר כל פעם אנו מישרים תמונה כך שתתאים לתמונה הקודמת.



32

איור 275: נבצע לכל התמונות *warping* ונשים אותן על אותו קנבס

אחרי שעשינו *warping* והנחנו את כל התמונות על הקנבס שלנו, צצה בעיה חדשה. אף על פי שהכל יושר וモוכן לאיחוד, האיחוד עצמו בעיני - בחיבורים בין התמונות, במקומות החיבור בין כל שתי תמונות, יש פס לא יפה ולא טבעי, כמו שקרה כנסייתי לאחד שתי תמונות בטשטוש פירמידות. אנחנו רוצים שהתפרק יהיה בלתי-נראה!

שאלה למה כל כך קשה לתפור תמונות יחד?

תשובה יש לכך כמה סיבות:

- (i) יכול להיות שהוא שינוי חסיפה בין התמונות כך שהאות בהירה יותר מהמאוחרת.
- (ii) ישנו אפקט בשם Vignetting בו יותר אור מגע למרכז-*sensor* מאשר לקצוות ולכן המרכז בהיר יותר.
- (iii) יכול להיות שהוא טשטוש (עקב miss-registration)

(iv) אם אובייקט שזז מופיע בשתי תמונות, כשתפור אותו האובייקט עלול להראות מרוח בתמונה כמו רוח רפאים

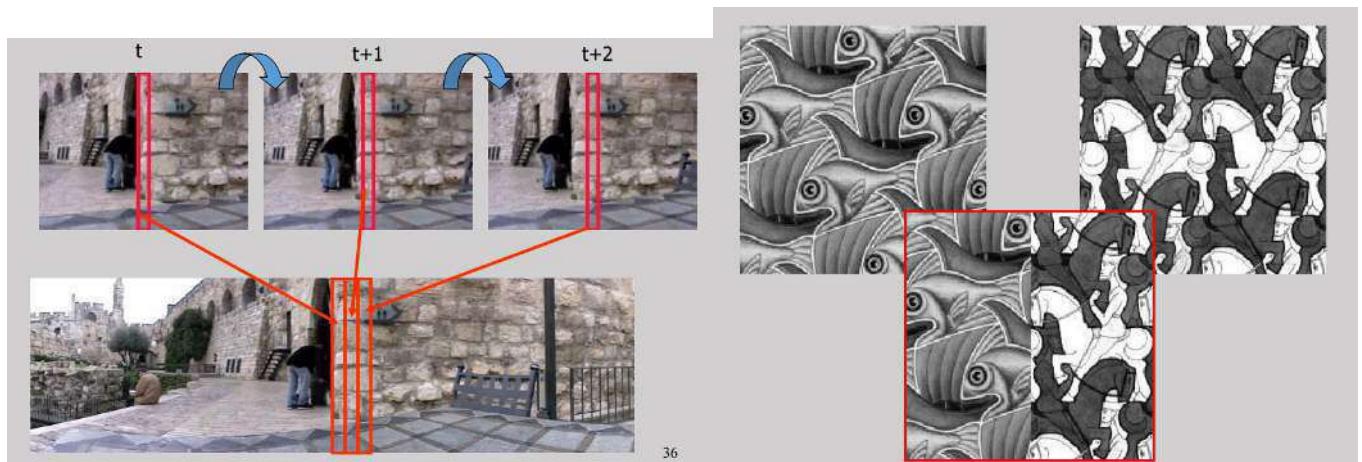
.(*Ghosting*)

על כן, מטרתנו עכשו תהיה לתרגם לתפר להיות כמה שפחות בולט: אנחנו רוצים שהתפירה שלנו לא תמציא *edges* כמו הפסים באירור.

הענ האנושית מאוד רגישה אפיו לשינויים הקלים ביותר ב-*edges*, ככה שהתפירה שלנו צריכה להיות ממש טובה. המחשה ידועה לרוגשות שלנו ל-*edges* היא האשלה האופטית הבאה. *A, B* הם באותו צבע על אף שאין לנו רואים זאת כך, וזאת בגלל הרוגשות שלנו לשפות ומה שקרה מסביב.

37.1 שיטה נאיבית

פשוט ניקח את הפס המרכזי מכל תמונה ונשים אותו אחת ליד השניה (כאשר עובי הפס תלוי בגודל ההזזה בין התמונות). כאשר אין שינוי תaura או אובייקטים שזזים התפירה נראה טוב. כשייש שינוי תaura - זה נראה מעוזע.



איור 276: משמאל: תמונה שנטפרה בשיטה הנאיבית ויצאה טוב כי אין שינוי תaura/אובייקטים שזזים. מימין: תמונה שנטפרה בשיטה הנאיבית ונראית גרווע כי שתי התמונות מאד שונות (תמונות כאלה שוניות נבחרו לצורך המחשב הרעינו).

37.2 תפירה עם Alpha/Pyramid-Blending

37.2.1 Alpha-Blending

הפעם לאחר שנשים את הפסים המרכזיים מכל תמונה, נבצע *blending* במעברים בין כל שני פסים.

ונכל לעשות זאת באמצעות טכניקה שנקראת *Feathering Alpha*.

ניקח מסיכה שאומරת לנו כמה לחתת מכל תמונה. המסיכה האחת תהיה פונק' α והאחרת תהיה $\alpha - 1$.

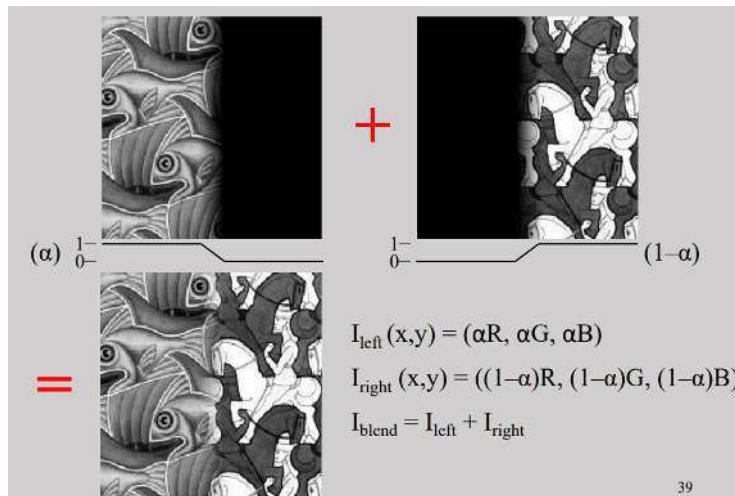
כך נוכל ליצור מתמונות חדשות:

$$I'_{left}(x, y) = \alpha(x, y) I_{left}(x, y)$$

$$I'_{right}(x, y) = (1 - \alpha(x, y)) I_{right}(x, y)$$

$$I_{blend} = I'_{right} + I'_{left}$$

והפעם המעבר בין התמונות יותר הדרמטי.



איור 277: חיבור תמונות באמצעות Blending Alpha

בעיה נורא קשה לנו להחליט מהי הפונק' α ? כלומר, קשה לנו להחליט כמה מהיר או איטי-Amor להיות המעבר מהתמונה אחת לשניה.

אם בחרנו α שמנשכת על כל התמונה, נקבל פשוט את התמונה האחת על השניה וזה יראה גרוע.

אם נבחר מעבר יותר איטי אך עדין גדול, עדין זה לא יראה טוב.

איך נבחר α שלא רחਬ מדי? אולי פשוט נבחר α קטנה מאוד כך שהמעבר יהיה חד? אבל אז נחזיר לבעה המקורית: אנו נראה פס של החיבור בין שתי התמונות.



איור 278: למעלה: בחירה גדולה מדי של α מובילה למיזוג של שתי התמונות. למטה: בחירה קטנה מדי של α מובילה לכך שאנו עדיין רואים את החיבור.

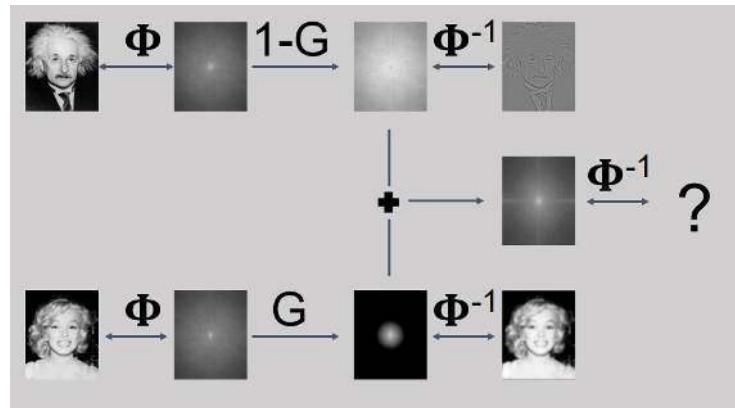
לסיום, אין דרך טובה לבחור α ולכן לא נשימוש בשיטה זו.

Pyramid-Blending 37.2.2

ב-*Pyramid-Blending* אנו בעצם עושים *Alpha-Blending* לכל רמה שונה של תדרים, ואז ה- α נהיית עם פרופורציה לתדרים בראש הפירמידה, שם התדרים נמוכים, וה- α גדולה: מטשטשים את שתי התמונות אחת לתוך השנייה. ככל שאחננו יותר דינמיות בפירמידה ה- α קטנה.

כך באמצעות *Pyramid-Blending* אנו מקבלים את ה-*Alpha-Blending* הכי טוב לרמות השונות של התדרים.

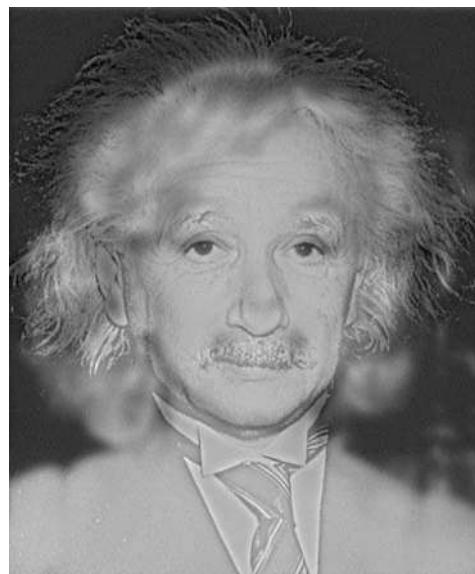
שאלה למה שלא נעשה *Fourier-Domain-Blending*? קלומר למה שלא ניקח תדרים גבוהים מהתמונה אחת ונמוכים מהאחרת, נסכם את תמונות התדרים ונעשה התמרת פורייה הפוכה?



איור 279: המחשב לחישוב Fourier-Domain-Blending

תשובה אנו נקבע אפקט די מגניב אבל שימושי לא שימושי לנו כאן. تستכלו על ת湊ת התוצאה מה-Blending-Domain למרلين מונרו ולאインשטיין.

סביר להניח שאתם רואים את איינשטיין ולא את מרליין מונרו. עכשו תצמצמו את העיניים שלכם - עכשו אתם אמורים לראות את מרליין מונרו! מגניב, לא?



איור 280: דוגמה לתוצאה של Blending Domain Fourier

הסיבה שזה קורה היא שכמשמעותם מקרוב רואים בעיקר את התדרים הגבוהים, ומרחוק את הנמוכים (להתרחק נוון אותו אפקט כמו לצמצם את העיניים). אבל זה קצת לא קשור לנושא שלנו :)

37.3 מיציאת תפיר אופטימלי

במוקם להשתמש ב-blending, נוכל לנסוטו למצוא את המקום שם נתפור בו, יהיו הći מעט ארטיפקטים. לא נבחר בהכרח לחתך פסים ממרכז התמונה. נציין שהחתך הזה לא חייב להיות קו ישר. כך אפילו לא נדרש לטשטש כלום!

הערה מיציאת תפיר אופטימלי תפטור לנו גם את הבעה של אובייקטים זזים, כי אם הינייעו עושים בין שתי התמונות הבאות, בהן האובייקטים זזים, נקבל ghosting.

תפיר אופטימלי יdag לא לתפור איפה שאובייקטים זזים ולא נקבל ghosting.



איור 281: ghosting בפנורמה נובעת מתנועה של אובייקטים בתמונות

במקרה זה, התפיר האופטימלי יהיה במרכז התמונה כי באמות שם אובייקטים לא זזים - איפה שיש הכי מעט שינוי בין התמונות, ככלומר איפה שההפרש בין שתי התמונות היכי קטן. במילים מדויקות אנחנו רוצים למזער את הגרדיאנטים בחיבור בין התמונות.



איור 282: משמאל: התמונה המקורית (יחד עם תמונה נוספת מופיעה באיר) מימין: ההפרש בין שתי התמונות. אזורים שחורים הם אזורים בהם לא היה שינוי גדול.

הערה אם יש שינויים תאוור, או תMOVנות ההפרשים לא בהכרח תלמד אותנו מהו החתך האופטימלי, ובמקרה כזה כדאי להשתמש Pyramid-Blending.

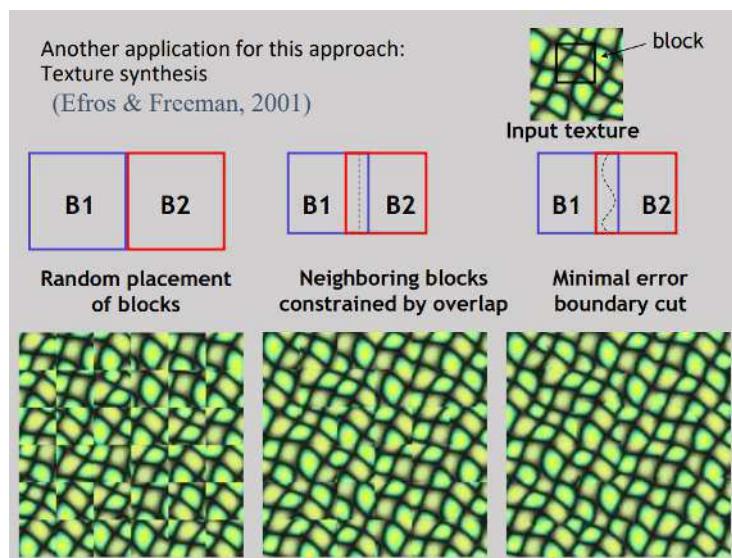
הרעין לדאג שהחיתוך שלנו יחלק כל אובייקט נוע לאזור נפרד (כך שכל אובייקט נוע יגיע למתחינה אחת בלבד) הגיע כבר ב-1998. את המעברים בין האזוריים אנו בוחרים להיות כאלה בהם אין הפרש בין התמונות כך שהמעבר יהיה ממש חלק.



איור 283: חלוקה של הפנורמה לאזורים בהם אין תזוזה כדי לאפשר תפירה טובה

משתמשים בדבר זהה גם לייצור של טקסטורות בגרפיקה ממוחשבת: לא לוקחים גרפיים שיצור טקסטורה ענקית. בדר'כ יוצרים טקסטורה מאד קטנה ומשכפלים אותה. חשוב לשכפל כך שכל המעברים בין השכפלים יהיו חלקים. אם סתם ניקח בלוקים ונשים אותם אחד ליד השני נקבל תוצאה גרוועה. (שמאל באירור) נוכל לנסות למצוא אזור *overlap* (או לא אותה תמונה זאת לא יפה כמו בפנורמה) כך שתיהיה אפשרי התאמה כדי שהתמונה יהיה יחסית דומות. אחר כך נוכל לשים אותם אחד ליד השני ונקבל תוצאה קצר יותר טובה אך עדין לא אופטימלית. (מרכז האירור)

לחלויפין, נוכל למצוא את החתכים שמאזערים את השגיאה באזור של *overlap* (בתמונות ההפרשים) ואז נקבל חתך אופטימלי שנראה טוב. (ימין האירור)



איור 284: ייצור טקסטורות באמצעות שכפול ותפירה בשיטות שונות

נדגים בפירוט את האופציה האחרונה: ניקח שני *patch*-ים שיש ביניהם *overlap* וונחר אוותם זה מזה ונעלם בריבוע (חשוב להעלות בריבוע כדי שכנזער לא ניקח שגיאה שלילית גדולה). נקבל מפת שגיאות - לבן זו שגיאה גבוהה (ערך גבוה) ושחור זו שגיאה נמוכה (ערך נמוך). נעבור על מפת השגיאות מלמעלה למיניה ונמצא איפה הכי כדאי להעביר את החתך כדי לקבל את התפר האידאלי.

משימה נותר להבין איך נמצא את החתך האופטימלי. נציג שתי שיטות: האחת עם תכנון דינامي והאחרת עם $Min - Cut$.

תכנון דינامي

נעבור שורה שורה בתמונה ונחשב סכום מצטבר של השגיאות עבור כל המסלולים:

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

נבחן כי בכל שלב אנחנו עושים אחד מהצעדים הבאים בלבד: יורדים למטה ופונים ימינה, יורדים למטה וזרים שמאליה

כאשר $e = (I_1 - I_2)^2$ (באזור של החפיפה בין התמונות).

シימו לב כי $E[i - 1, j - 1], E[i - 1, j], E[i - 1, j + 1]$ הם הערכאים המתאים לשולשות הפיקסלים שנמצאים מתחת לפיקסל שלנו (פיקסל אחד ימין למטה, פיקסל אחד בדיקת מתחת ופיקסל אחד משמאלי למטה).

כדי למצוא את החתך נעבור על השורה התחתונה, ונבחר את הפיקסל שהעלות שלו הכי נמוכה, ואז נבצע *backtrack* כדי למצוא את המסלול האופטימלי (שכאמור מסתiem בפיקסל שבחרנו).

האלגוריתם הזה יכול לזרוץ ב- $O(n^2)$ כאשר n הוא כמות הפיקסלים באזור החפיפה.

הערה אלגוריתם זה מוגבל במובן שאנו לא יכולים לעלות חזרה למעלה: אנחנו כל הזמן יורדים פיקסל אחד בכיוון האנכית. עקרונית הינו יכולים להרחיב אותו אך זה היה עולה לנו באופן משמעותי בזמן הריצה.



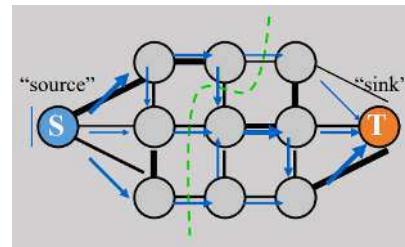
איור 285: שימוש בתכנון דינמי לשם מיציאת חתך אופטימלי

אלגוריתם Min-Cut

נזכר במשפט "ארימה מקסימלית - חתך מינימלי". יש לנו גראף $G = \langle V, E \rangle$ עם איזשהו מקור (*source*) ואייזחו שקע (*sink*).

כל צלע בגרף היא צינור עם משקל שמצוין את הקיבולת של הצינור (כמה מים ניתן להעביר בצינור).

בעיית Max-Flow היא "מה זרימות המים הגדולה ביותר שניתן להעביר מה-*source* ל-*sink*", כשהאנו מוגבלים ע"י הקיבולת של כל צלע.



איור 286: הממחשה לגרף שאנו רוצים לחשב לו Min-Cut. בירוק - החתך המינימלי, בכחול - הזרימה המקסימלית

בעה מקבילה ל-Max-Flow היא Min-Cut: מסתבר שהה-Min-Cut, ככלומר, החתך הכי קטן (לפי משקל הצלעות שהוא חותך) שאפשר להעביר שפעריד את ה-source וה-sink הוא גם זרימה המקסימלית. לבעה של מציאת Min-Cut יש פתרונות טובים בזמן פולינומיאלי (כמו Ford-Fulkerson).

אם נתאר את שתי התמונות שלנו כגרף כזה, נוכל למצוא את החתך המינימלי, Min-Cut, כדי לסייע את השגיאה ולמצוא תפר אופטימלי.

נمير את התמונות שלנו, A, B , לגרף כך:

(i) V - הקודקודים בגרף שלנו יהיו כל הפיקסלים באזור החיפוי בין שתי התמונות. באזור שלא חופף, כל מה שמנגע מהתמונה הראשונה יהיה ה-source ומה שמנגע מהתמונה השנייה יהיה ה-sink.

(ii) E - כדי לדאוג שהחתך לא יעבור באזור שלא חופף, ניתן משקל ∞ לכל צלע שיוצאת ישירות מה-source או מה-sink.

שני קודקודים p_1, p_2 מחוברים בצלע אם ו傒י $w(p_1, p_2) = (\star)$. המשקל ביניהם יהיה $\|A(p_1) - B(p_1)\| + \|A(p_2) - B(p_2)\|$.

(*) לפיקסל $p = (x, y)$ יש 4 שכנים אופקיים/אנכיים: $N4(p) = (x-1, y), (x+1, y), (x, y+1), (x, y-1)$. נאמר כי שני פיקסלים p, q מקיימים יחס $4-adjacency$ אם $q \in N4(p)$, כלומר אם p, q הם אחד מארבעת השכנים האופקיים/אנכיים של p .

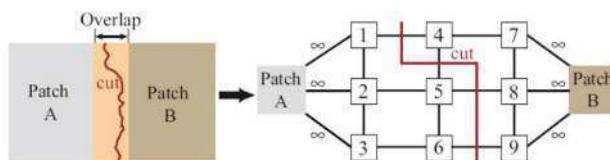


Figure 2: (Left) Schematic showing the overlapping region between two patches. (Right) Graph formulation of the seam finding problem, with the red line showing the minimum cost cut.

איור 287: הממחשה לחתך מינימלי בגרף הנוצר מהתמונות

יש לנו שני מקרים קיצוניים:

דומה ל- (p_1) (*i*) A יהיה מאות נמוך וכנראה נרצה לקחת אותו ולהעביר בו את החתך.

שונה מ- (p_1) (*ii*) A יהיה מאות גובה וכנראה לא נרצה לקחת אותו ולא נרצה להעביר בו את החתך.

סיכום

ראינו שיש לנו כמה שיטות לתפירת התמונה יחיד, ולכל אחת יתרונות וחסרונות.

- תפירה במקום קבוע (לדוגמה במרכז) ושימוש ב-*smoothing*

Feathering ↳

Pyramid Blending ↳

↳ פחות טוב כשי miss-alignment או אובייקטים נעים (yczor ghosting).

- חישוב חתך אופטימלי

↳ תכנון דינامي.

.Min-Cut ↳

↳ פחות טוב כשי שינוי תaura חזקים.

- אם יש שינוי תaura חזקים ואובייקטים נעים שתי השיטות שמצאננו יכשלו.

- שתי השיטות הנ"ל יוצאות מהשורה הראשונה בתמונה ומגיעות לשורה אחרת -

↳ ניתן לראות זאת בתכנון דינامي לפני הטליה.

↳ ב-*source-sink*, מחברים בין $minCut$ -ל-

38 תרגול 9 - Hough-Transform וצבע

Hough-Transform 38.1

מטרה נרצה להיות מסוגלים לאזהות בתמונה צורות כמו קוויים ישרים, עיגולים, מלבנים וכו'.

אנו יודעים בדיק איזו צורה אנו רוצים (לא מתחשים מהו באופן כללי), רק צריך לגלו איפה היא בתמונה (אם בכלל).

הטרנספורמציה שנראית לצורך השגת המטרה דומה לאלגוריתם Ransac מכיוון שגם היא **רובטית**.

יחד עם זאת, בניגוד>Ransac בו כל מודל קיבל הצבעה מהנקודות הנתונות, כאן כל נקודה תצביע לכל המודלים האפשריים שיכללו אותה, כמו למשל, כל הקווים היישרים שייעברו דרכה.

פתרון (נאיבי) נבנה קernel שייצג קו ישר ועל ידי קרוס קורלציה נמצא את הקו הישר המתאים לו.

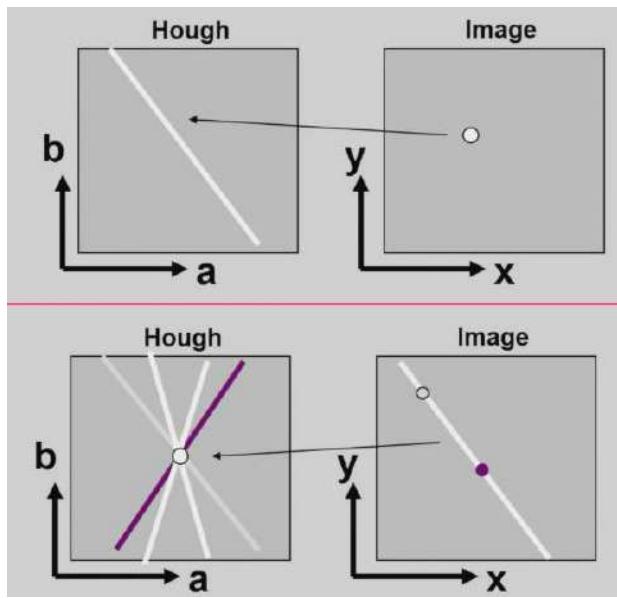
מדוע זה יכשל? ראשית, איןנו יודעים מה האוריינטציה של הקו הישר וכן נצטרך לבדוק יותר מקרים אחד, ולמעשה, علينا לבדוק את כל האפשרויות לקו ישר זה, וזה יקר חישובית.

במוקום זאת, נבצע תהליך אחר, למשל אם נרצה לאזהות קו ישר בתמונה. הנקודה הראשונה תצביע לכל הישרים שעוברם דרךה, הנקודה השנייה תצביע גם היא, אבל מכיוון שיש העובר דרך שתיהן, יש זה יקבל 2 נקודות בialogן לכל הישרים האחרים. נחזר על התהליך עם הנקודות האחרות ונבחר בסוף את הישר שקיבל הכי הרבה הצבעות. כל המידע יסוכם בטבלה.

מורחב הפרמטרים

משוואת ישר היא משווה מהצורה $b = ax + y$. בambilים אחרות, כל קו ישר מתאים לנקודה (a, b) במרחב פרמטרים כלשהו.

עליה השאלה, כיצד נזהה את הישרים העוברים דרך הנקודה? נציב (x, y) במשווה ונקבל את הקשר $ax - y = b$ כולם קיבלנו קשר ביןaries בין b ו- a . לכן נצטורך לבחור אך ורק ישרים המקיים קשר זה עבור b, a . על כן, לכל נקודה, העבור על כל ה- b, a שעונים על המשווה ונצביע עבורה במרחב הפרמטרים. במרחב הפרמטרים נקבל המונקוויים ישרים מהקשר שראינו, אשר תהיה להם נקודות חיתוך אחת עם cocci הרבה ישרים וו תכיל את ה- (a, b) שאנו חנו רוצים.



איור 288: המראהה למיפוי ישר לנקודה, ויצירת הקווים הישרים במרחב הפרמטרים

הערה. האלגוריתם שתיארנו הוא אלגוריתם כללי, אמנם המחשנו אותו באמצעות המקרה של קו ישר. כמו כן, אפשר לעשות דבר דומה עם שלושה פרמטרים - למשל עבר מעגל.

Algorithm 29 Hough-Transform

1 : Hough Transform

2 : Apply an **edge detector** on the image

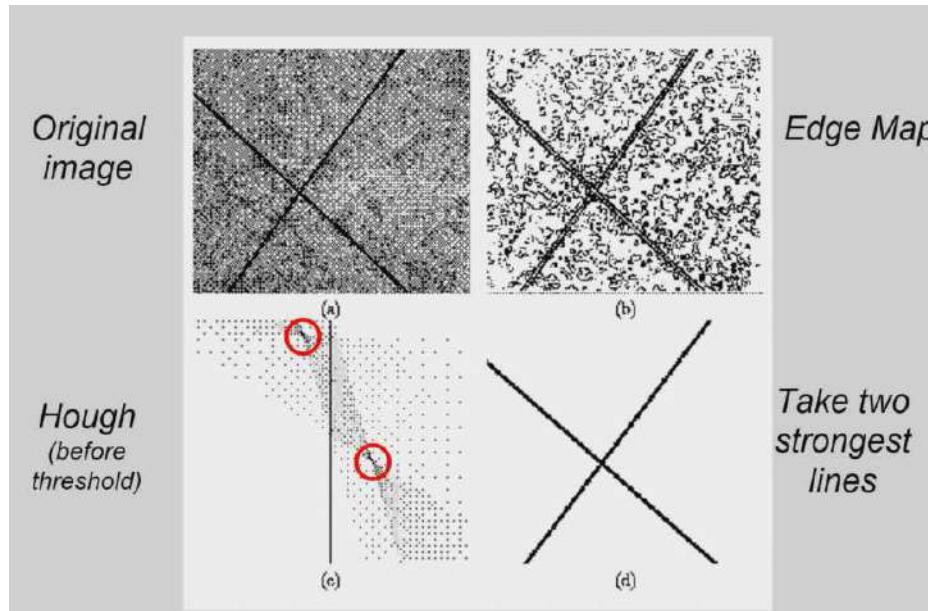
– gradient+threshold, or using Canny's method

3 : Prepare a **table $h(a, b)$** for a, b in a certain range

4 : Loop over the image: for each pixel (x, y) on an edge:

– **vote** for all the cells (a, b) which satisfy the equation $y = ax + b$

– meaning, increase the cell counter: **$h(a, b)++$**



איור 289: דוגמא לאופן פועלות האלגוריתם

בעיות

1. ייצוג: האלגוריתם לא יזהה קוים אנכויים, כי אין a, b , שמייצגים אותם.
2. טווח: צריך לבחור את טווח הטבלה.
3. קוונטייזציה: בחירת הגrid לטלבה.

הפתרון שנכיע הוא פולרייזציה.

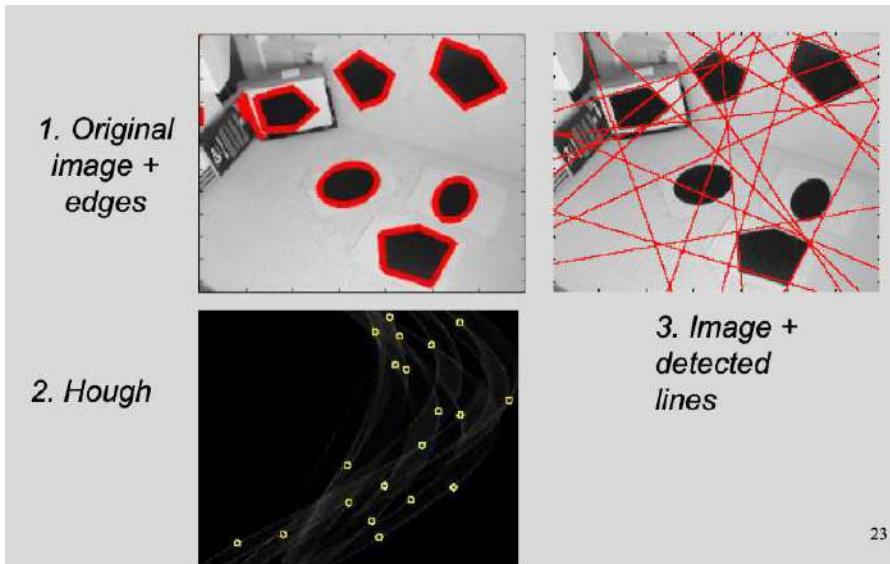
פולרייזציה

נשתמש בהצגה הפולרית

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

כאשר d הוא מרחק הישר מהראשית (מרחב ניצב), θ הזוויות בין הישר המחבר את ראשית הצירים עם הישר שלנו, לבין ציר x -ו- y (נקודה על הישר).

במצב זה נוכל ליציג כל ישר, רק שהפעם, בכל פעם שנקודה תצביע לזוג ערכים, היא תצביע לזוג (θ, d) , ואז במרחב הפרמטרים, במקומות הצגה לינארית, נקבל הצגות טריגונומטריות:



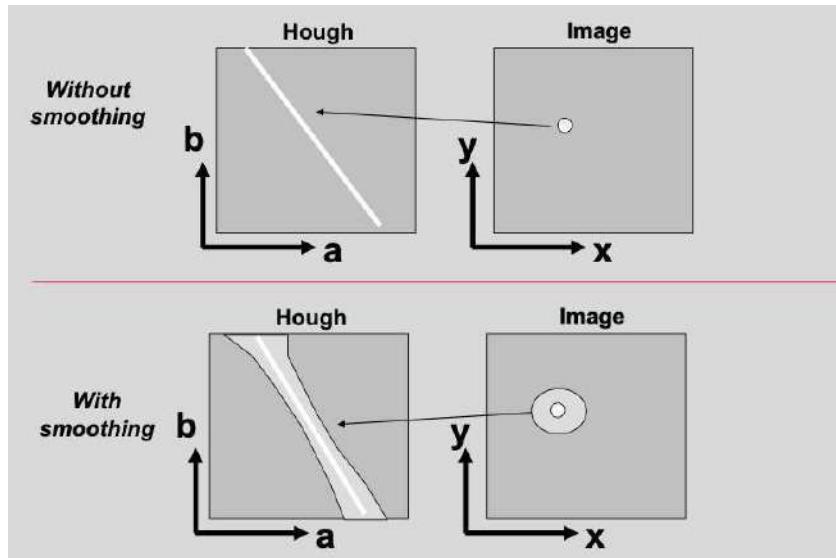
23

איור 290: האלגוריתם זיהה את כל הקווים הישרים, על ידי נקודות המקסימום בטבלה. ניתן לראות צורה של גלים, שמופיעים בכלל שנקודה מסוימת נמצאת על הרבה מאוד (θ, d) . חשוב לבדוק שהאלגוריתם לא מוצא את אורך הישרים, אלא רק את עצם קיומם.

טוחה וקונטיזציה

מבחינת טוחה שבחרנו נותנת טוחה קבוע $-360^\circ \leq \theta < 0^\circ$ ו- d חסום כי הוא המרחק של הראשית מהקו. אבל מבחינת קונטיזציה, יש עדין בעיה, למשל, קיבלנו את הזווית 3.4° ו- d שאינו מספרשלם, צריך להחליט מה עושים אם הזווית במספרים שלמים או אחרת? יחד עם זאת, האלגוריתם נותן לנו ניחוש ראשוני טוב. במילים אחרות, נותר לנו לקבוע מה הקפיצות של הנגיד.

בעיה. נבחן כי דרך נקודות קרובות עוברים ישרים דומים, וזה עשוי ליצור במרחב הפרמטרים נקודות חיתוך לא רצויות. כדי לטפל בזה, נחליק את התמונה ונקבל גם תמונה חלקה יותר במרחב הפרמטרים:



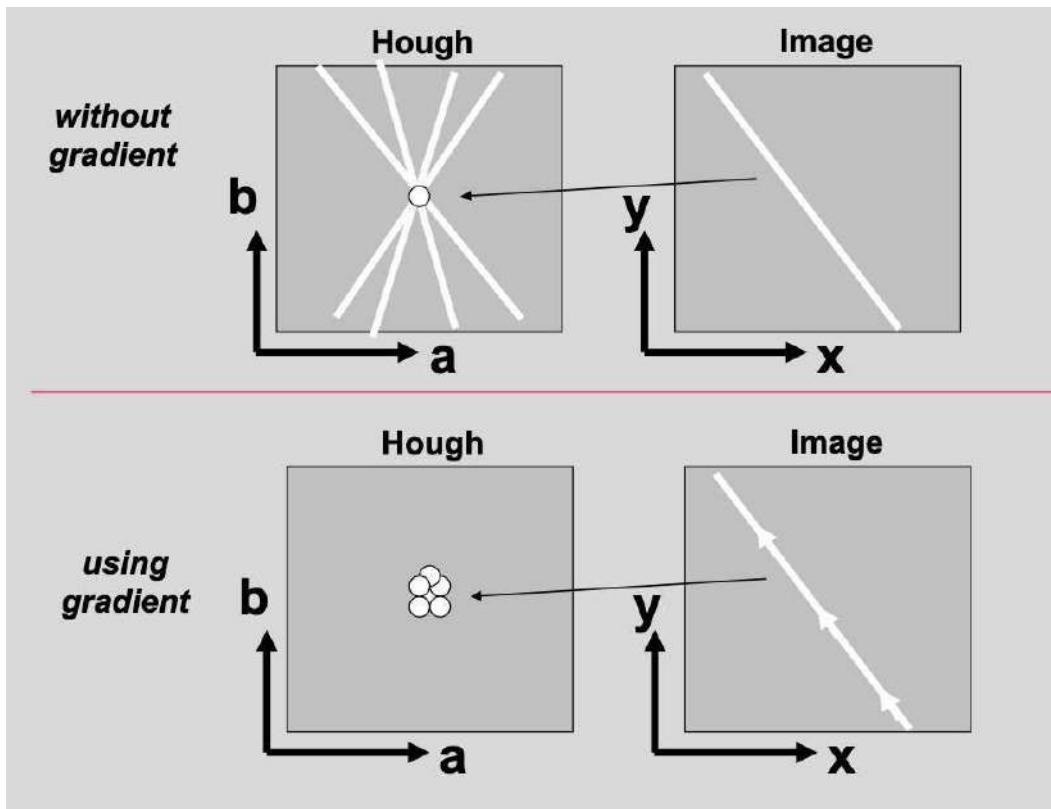
איור 291: ניתן לראות את ההחלה במרחב הפרמטרים

בעיה. ביעיה נוספת, היא מה לעשות כאשר כקו אחד חזק יותר מקו אחר. למשל, אם קיבלנו עבור קו מסוים שתי נקודות מקסימום - 100, 100 ועבור קו אחר נקודה אחת של 50, אם נבחר רק שתי נקודות, נבחר כמובן את ה-100, 100 שמתאימות לאותו הקו. זה קורה בדיקת מכך שנקודות קרובות בעלות קוים דומים - אם (θ, r) מקבל ניקוד גבוה גם $(\theta + \epsilon, r + \epsilon)$ מקבל ניקוד גבוה (מעין תכונת "רציפות").

כדי לטפל בזאת השתמש באלגוריתם Non Maximum Supression, ונודא שאנו בוחרים מקסימום לוקלי עבור כל קו.

בעיה. עילות - לכל נקודה יש המון הצבעות, זה יוצר המון מידע במרחב הפרמטרים ודורש עיבוד רב.

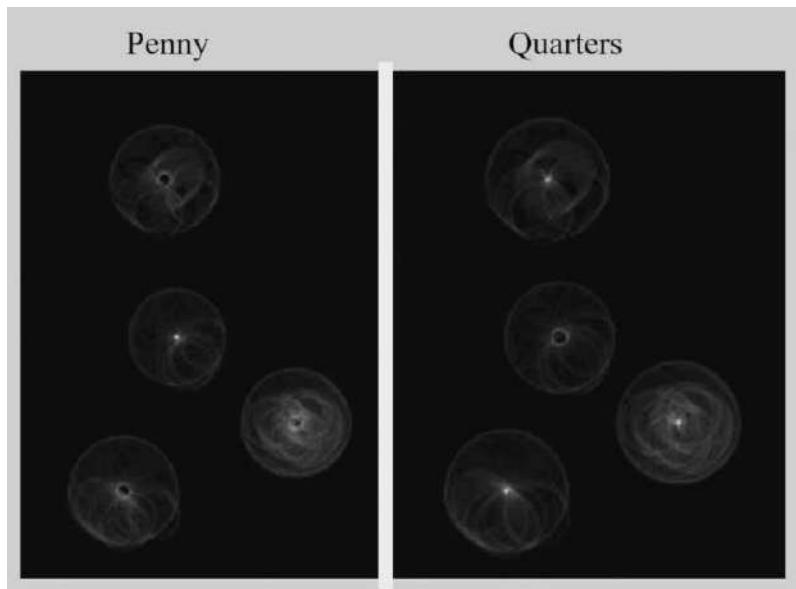
הפתרון הוא שימוש בגרדיינטס - הרי הגרדיינט יקבע את הקו דרכו עוברת הנקודה, ככלمر את הפרמטרים הרצויים במרחב התדר (כי הוא אנלוגי לשיפוע), וכך יוכל להציג רק עבור קו זה ובכך לחסוך המון הצבעות.



איור 292: השימוש בגרדיינט מפחית משמעותית את מספר הצבועות של כל נקודה.

זיהוי צורות מורכבות - מעגלים

נציע פתרון נאייבי. נציג מעגל C - $r^2 = (x - a)^2 + (y - b)^2$, לבני טבלה $h(a, b, r)$ ולכל פיקסל נבצע לכל הפיקסלים העוברים דרכו. הטבלה היא בגודל $O(n^3)$, ויש- $O(n^2)$ הצבועות. זה מאד יקר. כדי לטפל בזה, נחשב את הגרדיינט. הנקודה שלנו תהיה על edge שהוא שפת המעגל, שבקיים מדבר בכו ישר. לכן הגרדיינט יצביע לכיוון מרכז המעגל שהוא עליון, או לכיוון הפוך. בכל מקרה, נמיצמו את כמות הצבועות לשני כיוונים בלבד - עדין צריך לעבור על כל הרדיוסים.

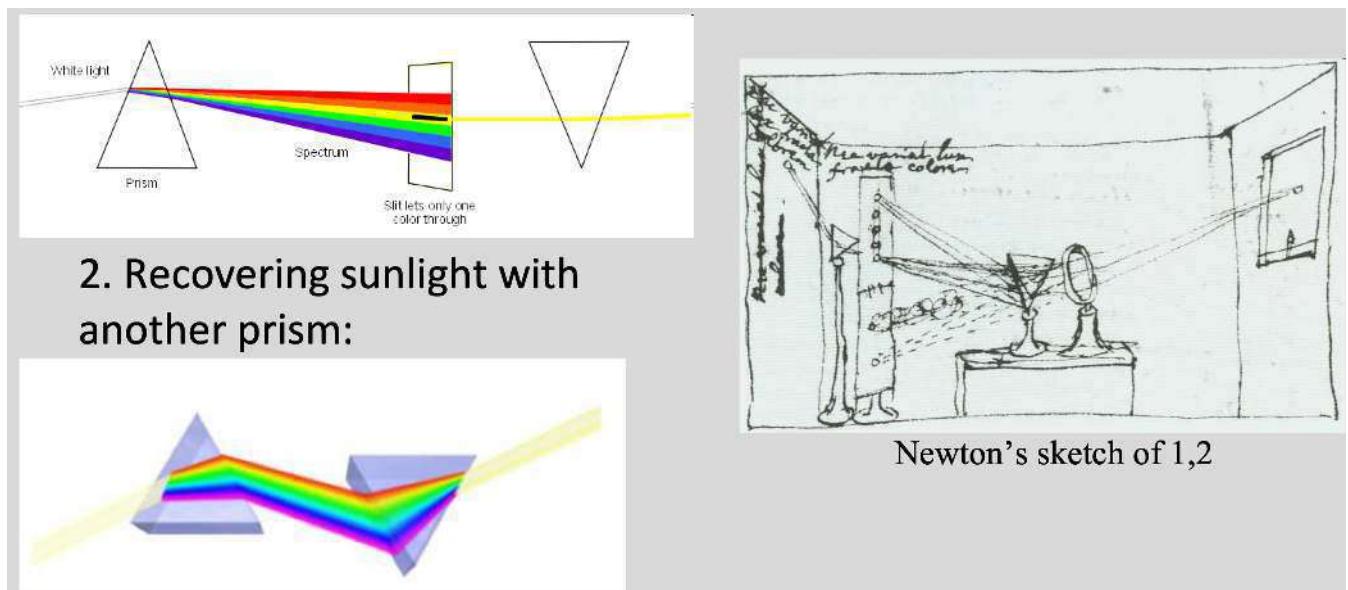


איור 293: לאחר מציאת *edges*, אנו מקבלים את החבוקות *the h*-*edges* בטללה. ניתן לראות שמרכזי המטבעות מקבלים חבוקות רבות וכי סבירם יש הרבה מעגלים נוספיםBeside them, there are many more circles. בדיק בגל שני הכוונים האפשריים שהוזכרנו קודם. נבחן כי כל נקודה בטבלה היא (a, b) (או (b, a)) ולכן היא תתן לנו גם את מיקומו של המעגל בתמונה וגם את הרדיוס שלו - בדיק את כל מה שאחננו כראיכים.

38.2 מרחבי צבע - Spaces Color

למד על מרחבי צבע נוספים, מלבד RGB , YIQ שכבר רأינו.

אור ניתן לפירוק לאורכי גל שונים



איור 294: נוכל לקחת קרן אור ולפרק אותה לכל אורכי הגל המרכיבים אותה, כמו שעשינו עם פורייה לתמונה. הניסוי בתמונה הוא ניסוי שערך ניוטון. ספקטרום האור הנראה נמצא בין (אורך גל באורך) $380 \sim 750 \text{ ננומטר}$ (אדום).

לעין האנושית, כמו שŁמְדָנָנוּ - יש שני קולטני אור - מדוכים וקנים.

- מדוכים - חישני בהירות, פעילים יותר בלילה. לא רגיסטים.

- קנים - חישני אור. יש מהם שלושה סוגים - כחול אדום וירוק. למעשה עבור שלשה זו (L, M, S) (גֶל אַרְוֹךְ, בִּינּוֹן, קָצֵר)

יתקיים הקשר

$$\begin{aligned} L &= \int L(\lambda) E(\lambda) \\ M &= \int M(\lambda) E(\lambda) \\ S &= \int S(\lambda) E(\lambda) \end{aligned}$$

המייצגת את טווח אורך הגל הנראים לכל אחד מסוגי הקנים.

עליה השאלה, מדוע דואען שלושה? ב-1802 הראו Young&Helmholtz שuboּר אדם ממוצע מספיקים שלושה צבעים כדי לייצג את כל הצבעים הנראים, ככלומר צבע שADB הוא צירוף לינארי של צבעי הבסיס $T = \sum_{i=1}^3 w_i P_i$. יתר על כן, מסתבר שהתאמה בין צבעים היא לינארית, ככלומר

$$\begin{aligned} C_1 + C_2 &= (R_1 + R_2) + (G_1 + G_2) + (B_1 + B_2) \\ \alpha C &= \alpha R + \alpha G + \alpha B \end{aligned}$$

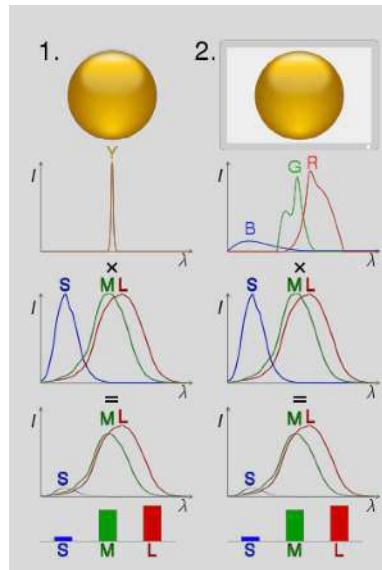
שלושה צבעים אלה, הם בעצם הבסיס למרחב הצבע RGB , שכן כל צבע במרחב זה הוא צירוף לינארי של RGB . יחד עם זאת, עלתה בעיה חדשה - יש צבעים שלא ניתן לייצגם על ידי סכום עם משקלות חיוביות על R, G, B , אלא רק אם אפשר למשקלות להיות שליליות. דבר זה הוא לא אינטואיטיבי, שכן החסרת צבע היא לא פעולה פיזיקלית מוגדרת. אבל זה המצב. על כן, ניתן להשתמש בפונקציות המופיעות צבע לעוצמות המתאים לו בשלשה הבסיסית כפונקציות של אורך הגל, ככלומר נמצא את הייצוג המתאים עבור כל אורך גל ($r(\lambda), g(\lambda), b(\lambda)$, או באופן כללי, קיבל את

$$C = \begin{pmatrix} c_1(\lambda_1) & \dots & c_1(\lambda_n) \\ c_2(\lambda_1) & \dots & c_2(\lambda_n) \\ c_3(\lambda_1) & \dots & c_3(\lambda_n) \end{pmatrix}$$

כאשר $\lambda_n, \dots, \lambda_1$ אורך הגל. עתה, נניח כי נתון לנו קלט $t = \begin{pmatrix} t(\lambda_1) \\ \vdots \\ t(\lambda_n) \end{pmatrix}$ הדרישה כדי לקבל את t היא בדיק

$$C \cdot t = \begin{pmatrix} c_1^T t \\ c_2^T t \\ c_3^T t \end{pmatrix}$$

שכן אנו מפרקים את התדר לאורך גל, ודורשים לבדוק את הכמות שמצאנו ש策יך עבור אורך גל זה לפי C . יחד עם זאת, נבחן כי לשני קלטים שונים אנו יכולים לקבל אותו קלט, כלומר $Ct = Cs$ כאשר $s \neq t$, על זוגות כאלה נאמר שהם מתאימים במרחב הצבע. זה נובע מזה שיש צבעים שאנו לא יכולים לייצג רק עם שלושה ערוצים.



איור 295: התמונות המכורפות נראהות זהות, אך בפועל הן לא. העין האנושית לא יכולה לקלוט את ההבדל, ולכן שימוש בשלושה צבעים הוא מספיק. כדי להיווכח שמדובר בתמונות שונות אפשר לבדוק השיפוק לתדרים שלhn שונה.

מרחב הצבע CIE-XYZ

מרחב זה מאפשר משקלות חיוביות בלבד, על ידי החלפה של RGB ב- XYZ כאשר XYZ הם שלשה בסיסית היפותטית שאיננה מייצגת צבעים אמיתיים בפועל. באמצעות מערכת זו אפשר לייצג את כל הצבעים הנראים לעין על ידי משקלות חיוביות.

ניתן לעבור בין הממערכות לפי הטרנספורמציה הבאה:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 3.240479 & -1.537150 & -0.498535 \\ -0.969256 & 1.875992 & 0.041556 \\ 0.055648 & -0.204043 & 1.057311 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

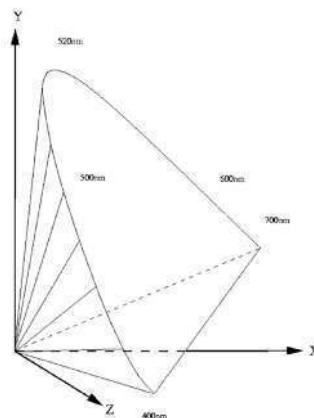
בדרך כלל יותר לעבור למרחב דו מימדי ולא בתלת מימדי, שכן ניתן להטיל את המרחב על המישור $X + Y + Z = 1$

ולקבל דיאגרמה כרומטית, לפי הקשרים הבאים של ההטלה

$$x = \frac{X}{X + Y + Z}$$

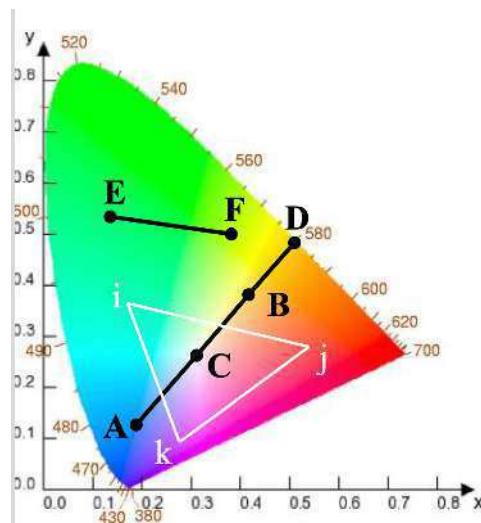
$$y = \frac{Y}{X + Y + Z}$$

$$z = \frac{Z}{X + Y + Z}$$



איור 296: ההטלה של המרחב $CIE - XYZ$ על המישור $X + Y + Z = 1$

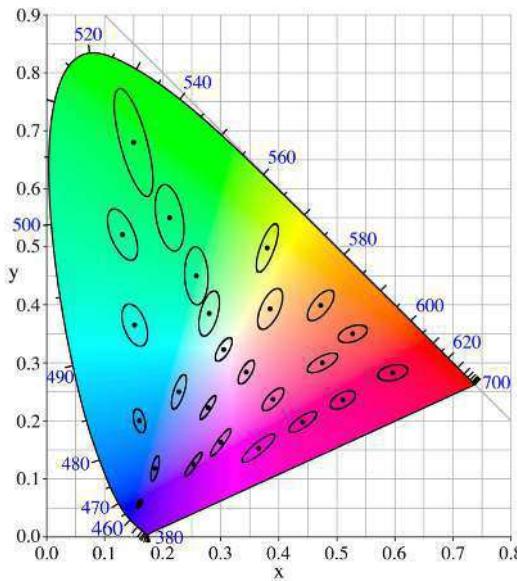
בצורה זו, לכל שלשה על המישור נדע לבדוק מה הצבעים שהיא יכולה לייצג לפי המשולש שהיא חוסמת:



איור 297: השלשה k, j, i יוצרת את כל הצבעים שבמשולש הלבן ולא מעבר לכך. מכאן ניתן לראותות שאין שלשה שיכולה ליצור את כל הצבעים במרחב זה. לצבעים אלה שהשלשה מייצגת אנחנו קוראים Gamuts.

מעבר לכך, מסתבר שהעין האנושית לא באמת מבדילה בין צבעים דומים במישור, לעומת זאת צבעים אלה לכדי צבע

אחד לפי תחומים אליפטיים:



איור 298: צבעים דומים מאוגדים לכדי אליפסה

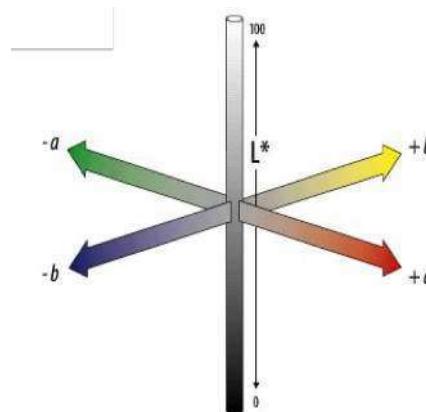
דבר זה בעיתוי, שכן במרחב צבע זה, אם הוספנו אחד לאחד הערכים, נרצה שהצבע ישתנה בהתאם, אך זה לא בהכרח המצב בכלל איגוד הצבעים.

CIE – Lab – ColorSpace

מרחב צבע זה בא לתת מענה לבועה של מרחב הצבע הקודם. כאן יש שלושה ערכאים *Lab* כאשר $L = Y^{\frac{1}{3}}$ עם $Y = 0.30R + 0.59G + 0.11B$ המיצג הארה. a, b מכילים את המידע על הצבע. ככלומר יש כאן הפרדה למידע על הצבע ורמת ההארה, בדומה למרחב הצבע *CIELAB*. בשונה ממנו, כאן המרחב לא לינארי. כדי לעבור מ-*Z*-*CIEXYZ* ל-*YIQ* משמשים בטרנספורמציה הבאה:

$$\begin{aligned} L^* &= 116f\left(\frac{Y}{Y_{white}}\right) - 16 \\ a^* &= 500 \left[f\left(\frac{X}{X_{white}}\right) - f\left(\frac{Y}{Y_{white}}\right) \right] \\ b^* &= 200 \left[f\left(\frac{Y}{Y_{white}}\right) - f\left(\frac{Z}{Z_{white}}\right) \right] \end{aligned}$$

$$f(t) = \begin{cases} t^{\frac{1}{3}} & t > \left(\frac{6}{29}\right)^3 \\ \frac{1}{3} \left(\frac{29}{6}\right)^2 t + \frac{4}{29} & \text{אחרת} \end{cases}$$

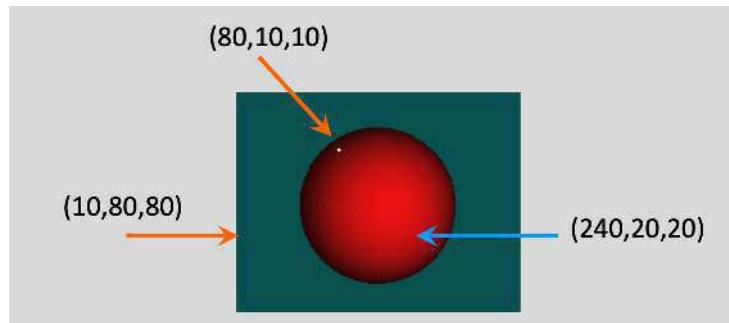


איור 299: המחשה למרחב הצבע Lab - כאן אם איזנו ב-1 גם הצבע a ב-1.

למרחב צבע זה יש ידיד והוא מרחב הצבע YIQ עליו למדנו וניתן להזכיר בו בתרגול 1.

חסרכנות מרחב הצבע RGB

ראשית החסרון העיקרי הוא חוסר היכולת לקבוע מוקטור ה- RGB האם מדובר בצבע דומה או לא, בניגוד למרחבי הצבע האחרים שמשמעותם בין צבע לתאורה. ניתן לקבל המחשה באյור המצורף.



איור 300: אף על פי שמדובר בשלשות שונות לחלוtin, הן מייצגות צבעים שונים

נוח להסתכל על מרחב ה- RGB כקוביה תלת ממדית בגודל $256 \times 256 \times 256$ שכל ציר בה מייצג אחד מהמימדים. כל נקודה פנימית בתחום הקובייה תחווה צירוי לנארוי כלשהו של השלשה, ותחווה צבע מסוים. אבל, במקרה להסתכל על המרחב כקוביה, אפשר לסובב אותו ולהסתכל עליו כגוף סיבוב.

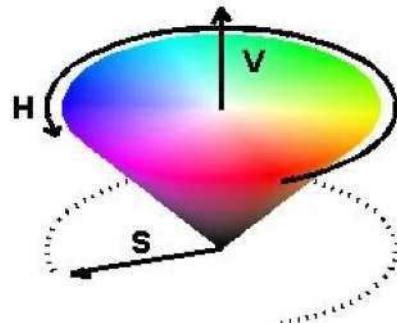
מרחב הצבע HSV

מרחב זה מבצע הפרדה של ראיינו עד כה:

• צבע הpigment הטהורה. Hue

• מידת כמיה צבע יש (למרחק מהצבע האפור). *Saturation*

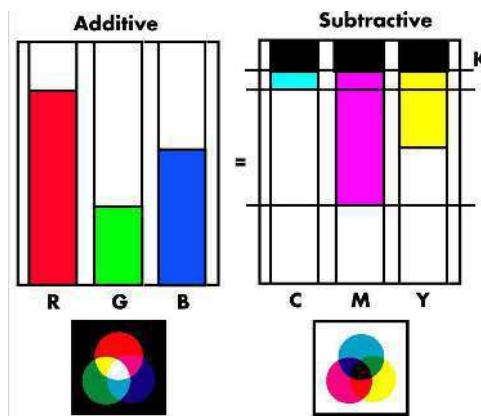
• מידת הבהירות. *Value*



איור 301: CAN המרחב הוא חרוט, בקוארדינטות ספריות (S, H, V)

מרחב הצבע CMYK

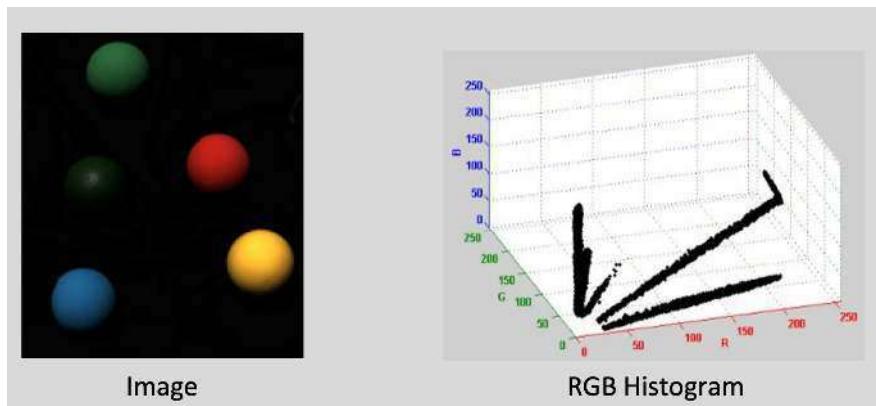
CAN בנגדם למרחבים הקודמים בהם סכמו צבעים בסיסיים וקיבלו צבע, CAN המרחב הוא *Subtractive*, כלומר אנו מקרים נורא על מנת להчисיר ממנו צבעים שאנו לא רוצים.



איור 302: CAN השלשה היא *Cyan, Magenta, Yellow*, ומשתמשים בחישור במקום בהוספה, כלומר אנו מחסירים מהדך הלבן, צבעים שבולעים צבעים שאנו לא רוצים שיבלוו, עליינו לקבוע עבור צבע שאנו רוצים שיופיע, איזה צבע צריך לחריד כדי לקבל את הצבע. ככה עובדות מדפסות.

הסטוגרמת צבעים

נדיר פונקציית צפיפות $\mathbb{R}^3 \rightarrow \mathbb{R}$. לדוגמה, הערך של $B_{(ri,gi,bi)}$ בהסטוגרמת *RGB* יהיה כמות הפיסקלים בתמונה בעלי ערכי *RGB* השווים ל- (ri, gi, bi) . יחד עם זאת, כפי שהזכרנו בעבר, הסטוגרמה כזו תהיה בגודל $2^{24} = 256^3 = 16777216$ תאים, שהוא גדול מדי. אבל, מסתבר שרוב המקרים נקבל מעט מועט של המרחב ולכן זה יהיה בסדר.



איור 303: למעשה, מה שקרה בפועל זה שצבעים דומים (מה שקרה המון באובייקטים בעלי צבע אחד והשתקפות או משטנה) מופיעים בקבוקים ישרים בתוך מרחב הצבע (2004 Werman, M. & Omer I.).

ההסתטוגרמה של הצבעים תתן לנו וקטור (a, b, c) , שייצג את התנהלות הצבעים בתמונה. ככה נוכל להשוות בין תמונות. כמובן, זה לא יuid על כך שהתמונות זהות, אך זה כן יuid על דמיון בין הצבעים, או על דמיון בין הסצנות (גם אם התמונות שונות), נוכל לחשב זאת כך, לפי מרחק אוקלידי

$$d^2(h, g) = \sum_A \sum_B \sum_C (h(A, B, C) - g(A, B, C))^2$$

או את מידת החיתוך בין התמונות

$$s(h, g) = \sum_A \sum_B \sum_C \min(h(A, B, C), g(A, B, C))$$

ככה למעשה נהוג לבדוק האם מדובר בשתי סצנות דומות.

הערה מלבד שני אלה, דרך נוספת לחישוב מרחק בין שתי היסטוגמות נקראת EMD - Earth's Distance" Mover's "Earth - EMD

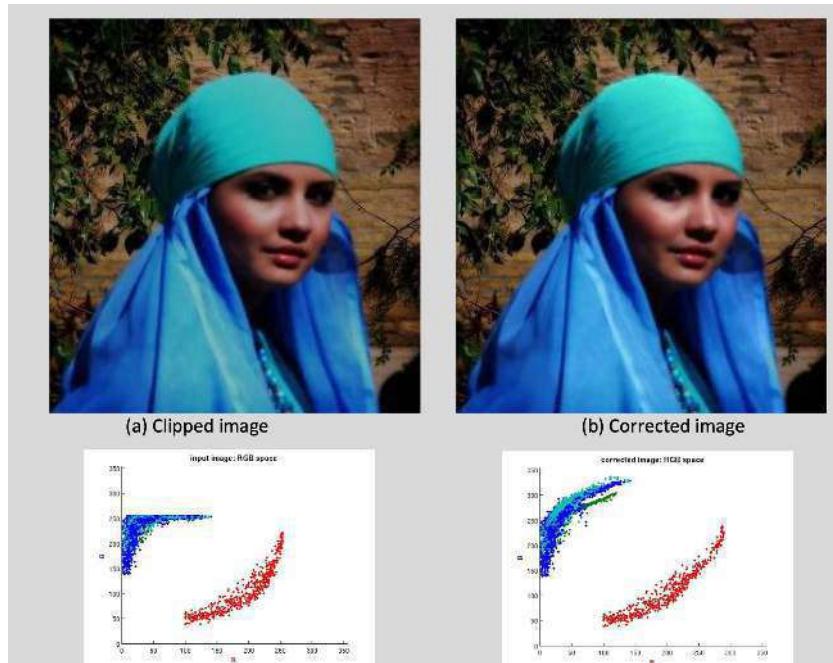
תיקון צבע רווי (Saturated) באמצעות הקווים בהיסטוגרמה

זכור שבהיסטוגרמה השינוי הדרמטי מצבע כהה לבהיר יוצר קו ישר בתוך ההיסטוגרמה התלת-מימדית שלנו. אחת התופעות שנוצרות מתקוונים הירשים⁸ היא שחלק מהעוזרים מגיעים למקסימום או למינימום והעוזרים האחרים ממשיכים להשתנות למשל $R = 255$ אבל G, B ממשיכים לגודל ועדין לא הגיעו ל-255.

⁸התופעה הוצאה לעומק במאמרו של פרופסור מארק ורמן מ-2011

דבר זה מוביל לאיבוד מידע על הצבעים בתמונה. אבל, אפשר לתקן בעיה זו על ידי ייצוג נרחב יותר של צבעים בגריד, למשל

להגדיל את הטווח ל-512 במקומות 256:



איור 304: ניתן לראות את התמונה ותיקונה. ההבדלים הם בעיקר הבדלי בהירות, הצבעים בהירים פחות בוהקים ואינם מסתירים מידע. התיקון נעשה על ידי השלמת הקו היישר שזיהינו ויישרו ("נתחה את הקו" נמשיך את הקו! כדי לייצג את התמונה נעשה scale-down להחל ונקבל את הצבעים כפי שהם היו אמורים להראות אם הסנסור לא היה מוגבל ל-256).³⁰

39 תרגול 10 - למידת מכונה ורשתות נוירוניות

למידת מכונה

המטרה שלנו היא לקבל תוצאות טובות מתכנית מחשב מבלי לתכנת זאת באופן מפורש. למשל, לתת קלט תמונה עם ריש ולקבל תמונה נקייה. למעשה, הבעיה היא כלהלן.

- נניח כי יש לנו חלוקה מתויגת לקלט ופלט רצוי .Set Training $\{ (x_i \in \mathcal{X}, y_i \in \mathcal{Y}) \}_{i=1}^N$, קבוצה זו היא ה- \mathcal{H} .

- מרחב הפונקציות האפשרות שיתאפשר בין x ל- y , שנסמןו $(Hypothesis Space) \mathcal{H} = \{f_\theta(x) \mid \theta \in \mathbb{R}^s\}$.

▷ למשל אם $f(x) = \sum_{i=1}^s x_i w_i$ לינארית עבור $w \in \mathbb{R}^s$.

▷ או טרנספורמציה לינארית $f(x) = Ax$ כאשר $A \in \mathbb{R}^{d \times s}$

▷ או פונקציות מורכבות יותר - רשתות נוירונים.

- המטרה היא למצוא θ כך ש- $f_\theta \in \mathcal{H}$ היא ההתאמה הטובה ביותר ביותר לקבוצת המדים.

- לא כל הפרמטרים נלמדים:

▷ הפרמטרים שאנו מאמנים אותם נלמדים.

▷ פרמטרים שלא משתנים - קיימים *Hyperparameters*, כמו למשל מבנה רשת הנוירונים שהוא קבוע ולא משתנה, סוג הפונקציה - לינארית, נשארת לינארית.

עליה השאלת - מהי ההתאמה הטובה ביותר? התשובה היא שזה משתנה. אבל בכלל - נגדיר פונקציית *Loss* שהנתן פלט התכנית, נגדיר את המרחק שלו מהתוצאה הרצוייה.

למשל, נוכל לבחור "පסד 1 – 0": ככלומר, $L_{0-1}(f, S) = \frac{\#\text{פעמים בהם } f(x_i) \neq y_i}{|S|}$ האוז ב- f טעה. זה מאד כללי ולא אינטואיטיבי, אבל מתאים במקרים מסוימים.

אפשרות אחרת היא Error Square Mean. $L_{MSE}(f, S) = \frac{1}{|S|} \sum_i \|f(x_i) - y_i\|^2$. ככלומר, את פונקציה זו נוכל לנזר כמו ב- LK ולמצוא לה מינימום ישרות, או לפעול בצורה איטרטיבית בדרך שמיד נראה. דבר זה פחות מתאים לביעות *Classification*, שכן אם קיבלנו כלב במקום חתול, השגיאה לא מאוד אינטואיטיבית.

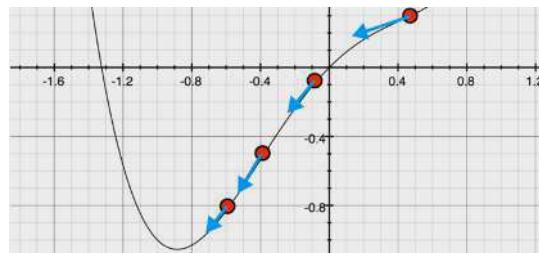
עוד פונקציית אפשרית היא *Cross – Entropy – Loss*: נניח כי $f(x)$ מוציאה וקטור הסתברות על ערכי x , כל קוארדינטה מייצגת הסתברות לשתיות של x למחלוקת מסוימת (למשל $1 = x$ בהסתברות 0.45 או $2 = x$ בהסתברות 0.55 או $3 = x$ בהסתברות 0). מודל זה יתאים בבעיות קלאסיפיקציה - כל קוארדינטה תייצג האם הגיעו לצורה מסוימת וקטור ההשווואה y יהיה 1 במקרה הנכון ו-0 בכל מקום אחר.

מazure פונקציית שגיאה

כדי למazure פונקציית שגיאה, נשימוש באלגוריתם Descent Gradient, אשר בכל איטרציה, נשימוש בעקרון המפתח הבא. באיטרציה ה- i נניח כי $L(f, S)$ היא פונקציה גזירה. נבצע איטרציות, כאשר בכל איטרציה, נשימוש בעקרון המפתח הבא. באיטרציה ה- i של האלגוריתם, ניקח צעד אחד בכיוון המינימום העמוק ביותר, דבר שאפשר להשיג על ידי הנוסחה

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

כאשר הצעד הוא בגודל η המסמלת Learning Rate והוא נקבעת לפי בדיקות שנעשה בעצמו (אם היא גדולה מדי נפספס את המינימום, אם היא קטנה מדי). פרמטר זה הוא גם Hyper-Parameter.



איור 305: אחרי כל צעד ב-Descent Gradient, נזכה להתקדם יותר ויותר למינימום

יחד עם זאת, עולה השאלה הבאה. אם S גדולה, איטרציות האלגוריתם יקרה מדי. למשל אם יש לנו 10^6 תמונות במדגם, לרוץ בכל איטרציה על מידע בגודל הנ"ל זה יקר מדי. על כן, אנו מקרבים את הקבוצה על ידי דוגמה ראנדומלית $S \subset B$. דבר זה מוביל לאלגוריתם הבא, המכונה Stochastic Gradient Descent:

Algorithm 30 Stochastic Gradient Descent

-
- 1 : **Sample** a random batch $B \subset S$
 - 2 : **Compute** the gradient of $L(f, B)$ w.r.t the parameters of f
 - 3 : **Update** parameters of f using the gradient
-

אנחנו מבצעים מספיק איטרציות עד שכל המידע ב- S נסרק, מעבר זה אנו קוראים Epoch - מעבר יחיד על המידע.

יחד עם זאת, SGD לא נוח מאוד לשימוש, ולכן יש וריאנט ושמו ADAM.

Overfitting

שאלה נניח שקיבלו מינימום Loss האם סיימנו?

תשובה לא.

מה שועלול לקרות הוא שהפונקציה שמצינו עובדת מעולה על המדגם, אבל במקרה של מידע חדש, היא קורסת. כדי לטפל בזה, נוסיף עוד מידע למדגם שלנו, עליו לא נאמן את הרשות, אלא בכל Epoch נבדוק שאנו חווים טובים גם על המידע הנוסף. מידע זה נקרא *Validation Set*. מכך ניתן לראות אם הרשות אכן נכונה.

אך על פי כן, גם לאחר שעשינו זאת וקיבלו משקלות, אנחנו נבדוק אותן על מידע בלתי תלוי לחלוטן שהרשות נכונה. בסיום התהילה נבדוק כמה היא טובה.

נסכם ונאמר כי:

- נרצה למצער את $(x) f_\theta$ על המדגם S .

- נמדד התאמת לפי פונקציית Loss שנסמך $L(f_\theta, S)$

▷ עבור קלסיפיקציה משתמש בוקטור הסתברותי - *Cross Entropy Loss*

▷ לרגרסיה - משתמש ב- *MSE*

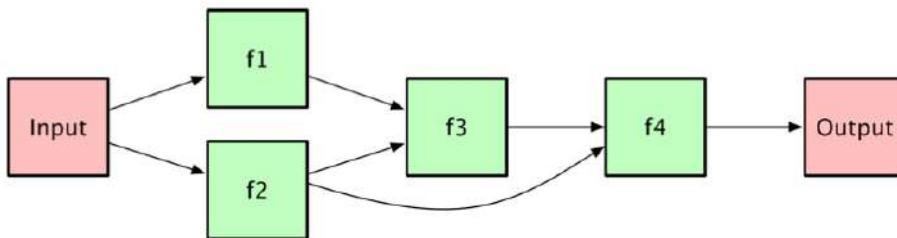
- נמצא את f_θ על ידי מצער ה-*Loss*

- במינימיזציה משתמש ב-*SGD* או וריאנטים עלייו

- נזהר מ-*overfit* ונבדוק תמיד את המודל על מידע שהוא לא ראה.

רשתות נוירונים - Neural Networks

רשת נוירונים היא גרפ' מכון חסר מעגלים של פעולות גזירות:



איור 306: הקטל עובר כמה פעולות (קודקודים) בגרף עד שהוא מגע לסוף ומוחזר לפט. העומק של הרשות מוגדר להיות המסלול הארוך ביותר מהקלט לפט, בדוגמה הנ"ל העומק הוא 4.

כל קודקוד בגרף נקרא שכבה והוא מוגדר לפי סוג ופרמטרים שלו. כל הפרמטריםividually נקראים המשקלות של הרשות. בנוסף, כאשר אנו ממלמים את הרשות אנו לא מושנים את מבנה הרשות, אלא רק את הפרמטרים, ולכן מבנה הרשות הוא קבוע. אנו כן נשנה אותו אם נראה צורך. *HyperParameter*

הערה. רשת נקראת *Feed – Forward* אם לכל קודקוד יש לכל היוטר קלט אחד ופלט אחד.

טנזורים

האובייקט/מנבה הנתונים בו אנו משתמשים ברשtotות נקרא טנזור. טנזור הוא פשוט מערך רב-מימדי כאשר \mathbb{N} $A \in \mathbb{R}^{M_1 \times \dots \times M_n}$, $M_1, \dots, M_n \in \mathbb{N}$. כלומר A הוא טנזור n -מימי עם המימדים (M_1, \dots, M_n) .

הקובננציה בספרית PyTorch היא לרשום בטנזור קודם את כמות העורצים ורק אז את מימדי האורך והרוחב, למשל בתמונה $(channels, height, width)$ נרשם RGB .

ביתר פירוט, נסמן:

N - מספר התמונות ב- $batch$ (i)

C - מספר העורצים בכל תמונה.

H - גובה כל תמונה.

W - רוחב כל תמונה.

ואז ב- $PyTorch$ הקובננציה תהיה טנזורי $NCHW$ (N בעוד שבספריות כמו Keras הקובננציה היא $NHWC$). כך הקלט והפלט של כל שכבה ברשtotת שלנו יהיה טנזור.

שכבות ברשת נוירוניות Dense

שכבה זו היא שכבה לינארית. כלומר, הפעולה שאנו מבצעים היא מהצורה $f(x) = Ax + b$

- קלט - וקטור $x \in \mathbb{R}^N$ (טנזוריים כלליים, כמו לדוגמה תמונה, משוטחים לטנזוריים חד-מימדיים).

1. פלט - הפלט $y \in \mathbb{R}^M$.

• מימד הפלט M : HyperParameters

- פרמטרים נלמדים: המשקولات $A \in \mathbb{R}^{N \times M}$ ו- $b \in \mathbb{R}^M$ שהוא ההיסט $(Bias)$.

שכבת קוןволוציה

הפעולה היא הפעלת קוןולוציה על הקלט עם כמות מסוימת של קרנלים.

- קלט - טנזור $(in_channels, height, width)$ 3D

- פלט - טנזור $(out_channels, height, width)$ 3D. כמות ה- $out_channels$ תלויות במספר הפילטרים בהם אנו

משתמשים⁹

:Hyperparameters •

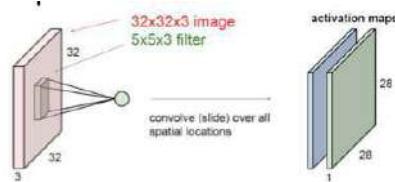
▷ הגודל של הקרナル: $K \times K$ ¹⁰

⁹כמו כן, יש אפשרות שפלט ה- $height, width$ יהיה שונים מהקלט אך לא נתיחס ליאת כרגע.

¹⁰בפועל הגודל של הקרナル יהיה $in_channels \times K \times K$, אך היפר-פרמטר היחיד שנשלט כאן הוא K .

↳ מספר הקרנליים M , כולל מספר ערוֹצִי הפלט

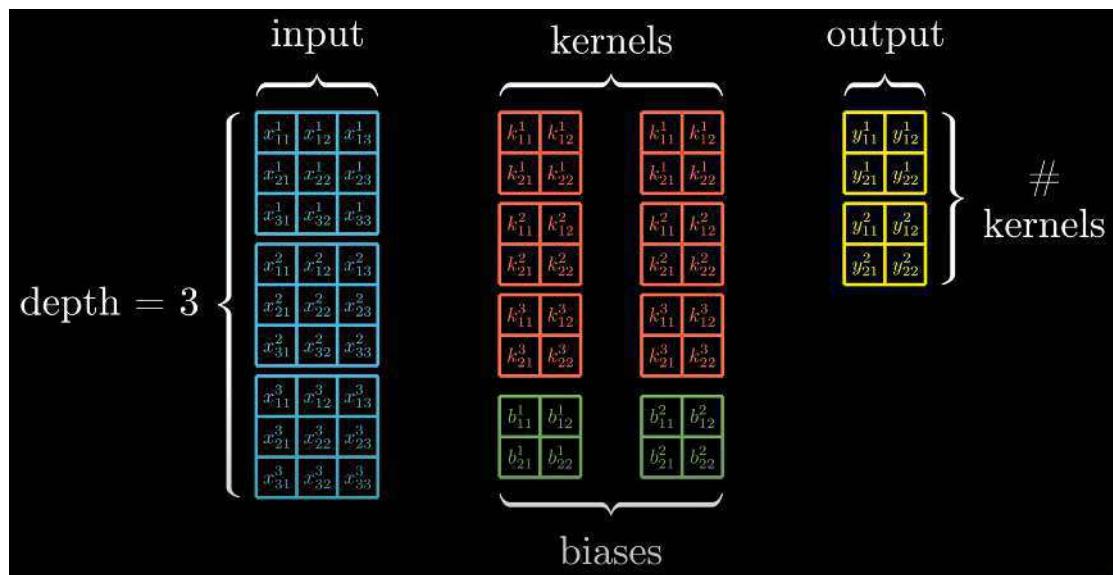
למוד את משקולות הkernel (ומלבד זאת יש גם *bias* שצריך ללמידה לכל kernel).



איור 307: המלצה לשימוש בكونבולציה על הקלט

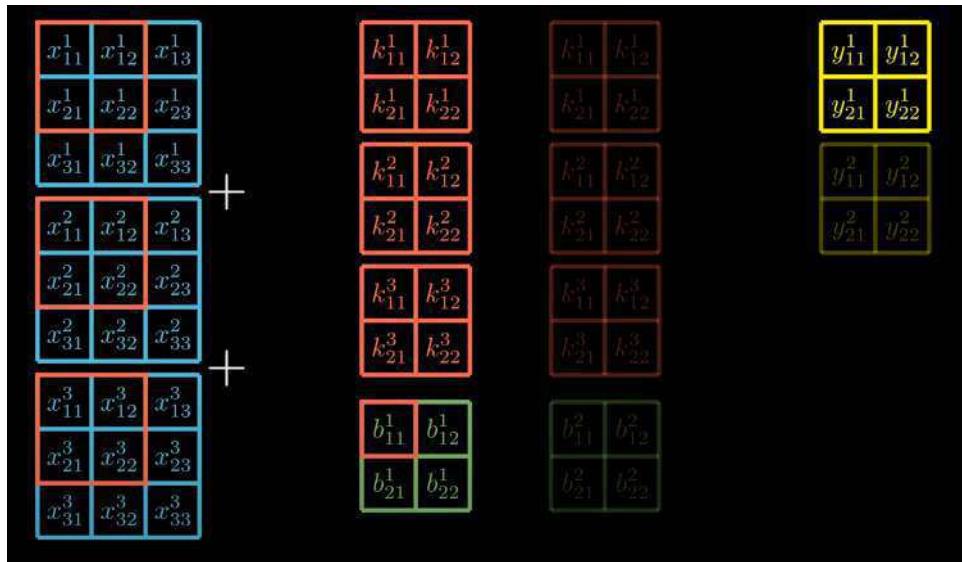
חשיבות לשים לב שכל קernel הוא עם אותו עומק כמו של התמונה, והעומק של הפלט הוא כמוות הקernelים שיש לנו בשכבה. חישוב הקונבולוציה נעשה ע"י סכימת הקונבולוציה של הקernel עם התמונה בכל רמה בעומק של התמונה, והוספת ה-*bias* המתאים.

ויזואלית, נראה זאת כך:



איור 808: דוגמה לפרמטרים המגדירים שכבה בראשת קונבולוציה

הчисוב נעשה כדלקמן:



איור 309: חישוב פלט בקונבולוציה עבור קרNEL אחד נעשה ע"י הפעלת הקרNEL על כל מקום בתמונה, סכימת התוצאות בכל העומקים והוספת ה-*bias* המתאים בכל נקודה.
באյור ניתן לראות זאת עבור פעולה הקונבולוציה הראשונית שמתאימה לפיקסל y_{11}^1 בפלט.
יש לחזור על הפעולה עבור כל קרNEL בשכבה כדי לקבל את הפלט המלא.

שכבה Pooling

נקטין את המידע על ידי צמצומו. אנו מחלקים את המידע לחלונות ובוחרים מכל חלון נציג.

• *AveragePooling* - מכל חלון נבחר את הממוצע שלו.

• *MaxPooling* - נחליף כל חלון במקסימום שלו.

▷ בדרך כלל בוחרים בדרך זו, בגלל האינטואיציה שבקרוטס קוורלציה רוצים גודל מקסימלי. זו רק השערה, אבל אכן נגלה שהה שזה עובד יותר טוב.

להלן התוכנות של שכבה זו:

• קלט: טנזר 3D מגודל ($in_channels, height, width$)

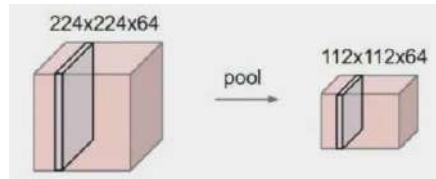
• פלט: טנזר 3D ($in_channels, \frac{height}{window-height}, \frac{width}{window-width}$)

• גודל החלון, בדרך כלל 2×2 - Hyperparameters

• אין - LeranedParameters.

במילים אחרות, אנו מתחמצחים את המידע, לטובת השכבות הבאות.

הערה שימושו לב שהשני הוי לא בכמויות העורכים, אלא רק ברוחב והגובה.



איור 3:10: הממחשה לתהליכי ה-*pooling*

שכבות האקטיבציה

נבחין כי עד כה הפעולות שהפעלו על המידע היו לינאריות, ולכן הכל מדובר בהרכבה של פעולות לינאריות שהיא פועלה לינארית. העולם שלנו הוא יותר מורכב מכפל במטריצה, ולכן נדרש מושהו נוספת. על כן אנו מפעילים פונקציית אקטיבציה שאינה לינארית.

אנו נפעיל פונקציית אקטיבציה $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ על כל איבר בטנזור הקלט, כאשר σ יכולה להיות:

$\sigma(z) = \max(z, 0)$ - *ReLU* •

$\sigma(z) = \tanh(z)$ - *Hyperbolic Tangent* •

$\sigma(z) = \frac{1}{1+e^{-z}}$ - *Sigmoid* •

אליה התכוונות של שכבה זו:

• קלט - כל טנזור n -ממדי.

• פלט - טנזור באותו גודל של הקלט.

- אין כמעט המקרה הבא: HyperParameters •

↳ במקרה של *LeakyReLU*, בו $\sigma(z) = \max(z, 0.01z)$, צריך לבחור כהווגן את הכופל של z .

- אין כמעט המקרה הבא: LearnedParameters •

↳ במקרה של *PReLU*, בו $\sigma(z) = \max(z, \alpha z)$, צריך ללמידה את α .

PyTorch

ספרייה פופולרית ונוחה מאוד לייצרת ואמון רשתות נוירוניות.

להלן נדריכים מפורטים:

<https://pytorch.org/tutorials/beginner/blitz/cifar10tutorial.html> •

<https://pytorch.org/tutorials/beginner/basics/intro.html> •

DataSets

קיימים כיוון *Dataset* גדולים לאימון רשותות - כמו למשל של תמונות, מספרים ועוד. בין מאגרים פופולריים כאלה נמנים *IMAGENET* ו- *MNIST, CIFAR10*

היום אנו מאמנים רשותות על מעבדים גרפים (*GPU*) ולא על ה-*CPU*. גוגל מאפשר שימוש של כ-12 ביממה על *s' GPU* שלו.

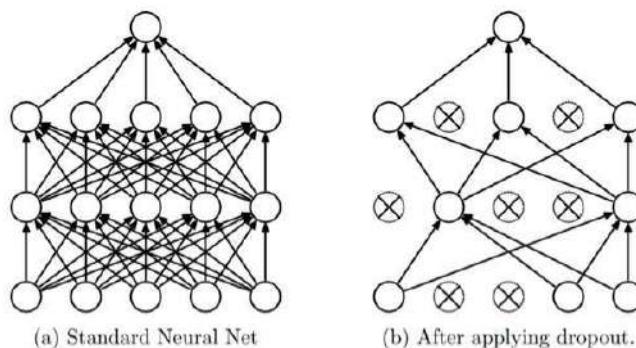
גוגל המציאו בנוסף צ'יפ בשם *TPU*, שהוא נוצר ספציפית לעבודה עם רשותות נוירוניות.

מניעת Overfitting

אחת השיטות היא *DropOut*: בזמן האימון נזרק באופן רנדומלי חלק מהנוירונים ונאמן את הרשות על מה שנשאר - ככה נאלץ אותה להתנהג בצורה כללית.

למעשה זה מונע *co-adaptations*: שמו לב שהרשות מתחילה ליצור תלויות בתוך עצמה - משקלות שמצוות על משקלות אחרות, משקלות שmagiba באופן מסוים אם משקלות אחרות magiba באופן מסוים וכו'. זה לא טוב לנו כי התלוויות הללו מורכבות מדי ולכן לא גורמות ל-*generalization* טוב.

כמו כן, ראו של השתמש בכמה רשותות ולהשתמש באגרגציה שלן מובילה לתוצאה יותר טובה. ככלומר, אם נאמן 7 רשותות (שנראתה בהמשך) ונעביר תמונה דרך ונעשה ממוצע *l-output* או משחזר דומה, התוצאה תשתperf. אנחנו כמובן לא נאמן רשותות המון פעמים, אז *dropout* נותן איזשהו קירוב לזה כי כל פעם הרשות קצרה משתנה.



איור 311: הממחה ל-*Dropout*

דרך נוספת היא *Data Augmentation*, אנו נהפוך את המידע ונazzi אותו במהלך האימון כדי למנוע *overfitting* על

, נוכל אפילו לשנות את בהירות ה-*RGB*. לעיתים מנורמלים כל נוירון:

$$b_{x,y}^i = \frac{a_{x,y}^i}{\left(k + \alpha \sum_{j=\max(0,i-\frac{n}{2})}^{\min(N-1,i+\frac{n}{2})} (a_{x,y}^j)^2 \right)^\beta}$$

אך היום משתמשים בזה פחות. למרות זאת,.cn משתמשים ברענון של נרמול אך באופן מעט שונה.

40 תרגול 11 - סוגים שונים של רשתות נוירוניים

בין השנים 2010–2017, התקיימה תחרות בשם ILSVRC (Large Scale Visual Recognition Challenge) בתחרות זו, מועמדים הגיעו לשנתו שמנסוחה לקטלוג תמונות בגודל 256×256 בין 1000 קטגוריות שונות (בעיתת קלסיפיקציה). לשם האימון עמדו לרשות המתמודדים התמונות ממאגר *IMAGENET* שכלה כ-1.2 מיליון תמונות! היה מאמץ רב לננות ולהשתפר בתחרות זו, וברוב השנים המנחים פיתחו רשתות שהיו פריצת דרך בתחום וחלקו נשארו איתנו עד היום. נציין שהמון פעמים מנצחיהם היו אנשים ידועים ומוסרים בתחום זה.



איור 312: רשימת המנצחים ב-ILSVRC לאורך השנים. בעמודה הימנית ניתן לראות את "Human" שזו התוצאה שהשיג נבדק אנושי כלשהו, כך ניתן להשוות בין האדם לרשתות.

AlexNet

- 8 שכבות - 5 קונבולוציה ו-3 שhn Fully-Connected

▷ באותו זמן נחשה לרשת מאוד عمוקה

- הרשת השתמשה ב-*ReLU*

▷ באותו זמן היו הפונק' הנפוצות, על אף שכבר היה שימוש כלשהו ב-*ReLU*. המון תולים את ההצלחה של AlexNet בשימוש ב-*ReLU* ומאז' כולם משתמשים ב-*ReLU* ובויריאנטים שלו.

- השתמשה ב-Batch Normalization (כבר לא בשימוש אבל משתמשים עדיין בנוורמליזציה: Local Response Normalization) והרמול הספציפי בו השתמשו היה באמצעות הנוסחה:

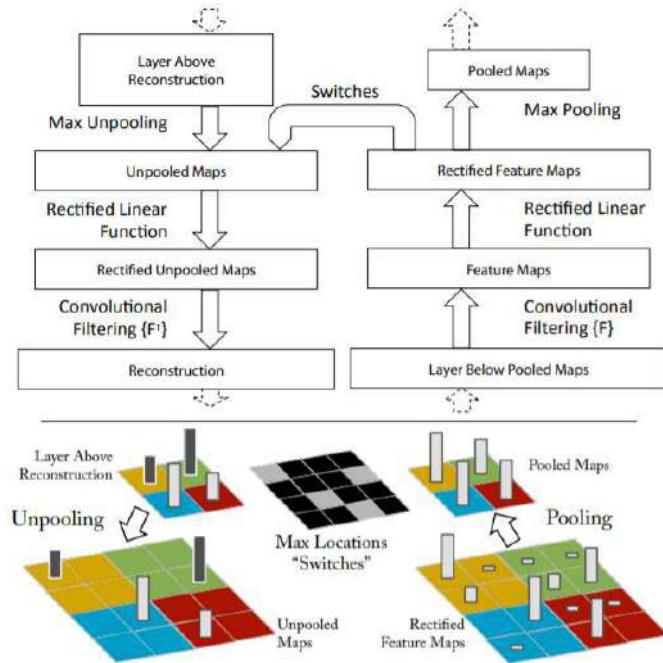
$$b_{x,y}^i = \frac{a_{x,y}^i}{\left(k + \alpha \sum_{j=\max(0, i-\frac{n}{2})}^{\min(N-1, i+\frac{n}{2})} (a_{x,y}^j)^2 \right)^\beta}$$

- השתמשה ב-Data Augmentation

- ▷ בתהליך האימון אנו לא רואים כל פעם את אותן תמונות. אנו מפעילים איזשיי טרנס' רנדומלית (כמו *Translation* או *Flipping* או משחק עם העוצמות של ערכי ה-*RGB*).
 ▷ כך אנו אינוריאנטיים להמוני דברים וגם האוגמנטציה מגדילה את ה-*DataSet*.
 • השתמשו ב-*DropOut* 0.5.
 • אמנו במשך שבוע על שני *GPU*-ים.

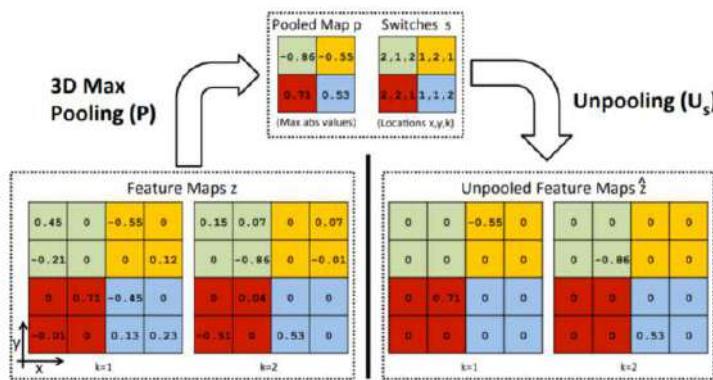
ZFNet

- חיפשו את ה-*AlexNet* Hyperparameters ביותר לארכיטקטורה של *AlexNet*
- ▷ שינו את שכבות הקונבולוציה בהן הernal היה 11×11 ל- 7×7
 ▷ הוסיפו עוד ערוצים בשכבות קונבולוציה
 מה שמעניין אותנו זה היזואליציה שם הציגו. הם השתמשו בكونספט בשם *DeconvNet* - לוקחים את הרשות ועשימים לה את התהליך הפוך: לקחת את האקטיבציה בכל מיני שכבות, ומעליהם אותה חוזרת למימד המקורי של התמונה.
 הרעיון כאן הוא לנסות להראות לנו מה קרה בכל שכבה כדי שנוכל להבין אינטואיטיבית מה קורה, כי עד כה רשותות נירוריות הן פשוט מעין *BlackBox*. להבין איך הרשות עובדת יכול להבהיר לנו אולי שינויים כאלה לבצע בראשת.
- כך נראה הארכיטקטורה של *DeconvNet*:



איור 313: מימין: אופן הפעולה של רשת ה-ZFNet.
משמאל: אופן הפעולה של ה-DeconvNet.
ניתן לראות שה-DeconvNet פשטוט "היפוך" של רשת המקורית.

החלק המבלבל היחיד כאן, הוא כיצד נעשית ה-*Pooling*, *Unpooling*. בידוע, *Pooling* מקטין את כמות המידע שלנו, אז איך נשזרז אותו? אז לא ניתן לשזרז אותו, אבל לפחות נוכל לזכור מה המיקום של הפיקסל ממנו לקחנו את המקסימום ולהשתמש בו ב-*switches*. בכל מקום זהה אנו קוראים *switch* (כפי שניתן לראות באיור הקודם). להלן המחשה של ה-*Unpooling*:

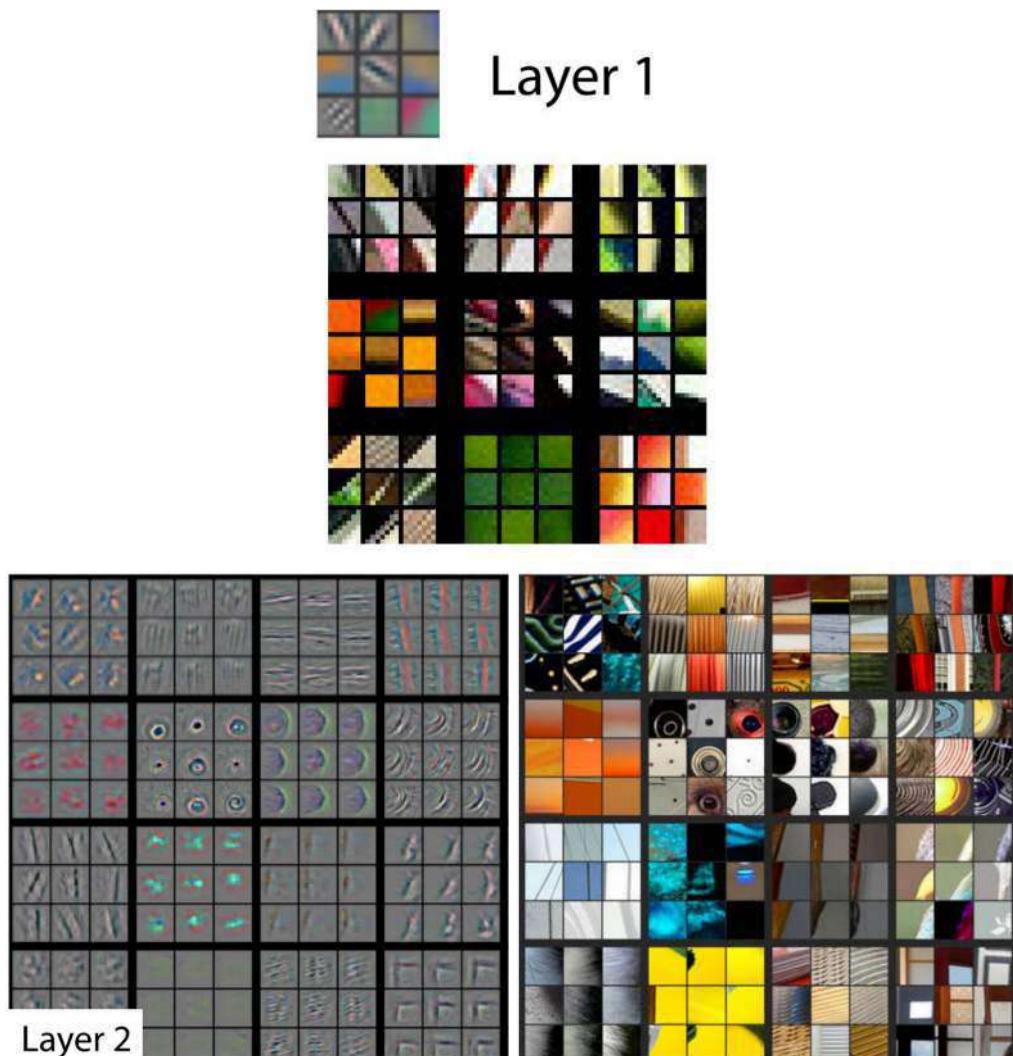


איור 314: אנו זכרירים בכל פעם מה המיקום של הפיקסל שבחרנו ב-*Max Pooling* כדי להשתמש בו ב-*Unpooling*.

הערה חשוב לשים לב שבניגוד לרשת המקורית, ב-*DeconvNet* אנו משתמשים בקונבולוציה רק **אחרי** הפונק' הלא-lienארית *(ReLU)*.

לאחר שימוש ב-*DeconvNet* רואו שככל שנכנסים יותר לעומק הרשת, הפילטרים מוחפשים צורות יותר ויתר מורכבות. לאחר

שהפעילו את התהילהז על רשותות אחרות, לא רק *ZFNet*, ראו שוכן מותנהגות ככה - בשכבות התחתיות מחפשים דברים פשוטים כמו קווים ולאט לאט עלולים ברמת המורכבות.



איור 315: בשכבה הראשונה הפילטרים מחפשים קווים ישרים, אזורים חלקיים וכו' (Features Low-Level). בשכבה השנייה הפילטרים מחפשים צורות יותר מורכבות (לדג' עיגולים, טקסטורות). ככל שנמשיך לעלות ברמות הפילטרים יחשפו דברים יותר ויותר מורכבים.

הדבר הזה פתח את הדלת ל- [Learning Transfer](#) - נניח שגוגל השתמשה בהמון משאבים כדי לאמן רשת מעולה על *ImageNet* ונניח שאינו חסר משאבים ורוצה לאמן רשת על *Data Set* אחר.

בגלל שהוא יודע שאם אמן רשת השכבות הראשונות שלי יהיה אותו דבר כמו השכבות הראשונות של גугл, אולי כדאי **לקחת מגוגל את השכבות הראשונות** ולאמן רק החל מהשכבות המאוחרות יותר. ככה נצטרך פחות כח עיבוד, פחות נתונים וכו'.

הערה אם ה- *Data Set* שלי מאד שונה מזו של גугл (לדג' הם אימנו את הרשת על תמונות של כלבים ואני רוצה לאמן רשת שמקבלת תמונות רפואיות של *MRI*, נראה ש- [Transfer Learning](#) לא יהיה רעיון טוב.

- רשות שנמצאת עד היום בשימוש (משמשת גם כבסיס טוב ל-Transfer Learning).

- הרשות הייתה עמוקה אף יותר מ-AlexNet שהכיל 8 שכבות: L-*VGG* הייתה גרסה עם 16 שכבות וגרסה עם 19.

- השתמשו בקרנלים פשוטים בגודל 3×3 במקומ 7×7 .

▫ 3 קרנלים של 3×3 הם בעלי אותו שדה קלט (receptive field) כמו קרナル אחד של 7×7 .

▫ ב-3 קרנלים של 3×3 יש $3 \cdot 3 \cdot 3 = 27C^2$ משקولات (C הוא מספר ערוצי הקלט והפלט, תחת ההנחה שבאמת כמוות ערוצי הקלט שווה לכמות ערוצי הפלט) כי יש לנו C קרנלים בגודל $3 \times 3 \times C$ בקרナル אחד של 7×7 יש $7^2C^2 = 49C^2$ פרמטרים, זהה 81% יותר פרמטרים!

▫ יש יותר Non-Linearities - במקום להפעיל פונק' לא-lieneariyat פעם אחת על פילטר 7×7 , אם ניקח 3 קרנלים של 3×3 אנו נפעיל פונק' לא-lieneariyat 3 פעמים. הכנסה של עוד אי-lienariyot הופכת את הרשות ליותר מורכבת, יותר אקספרסיבית שיכולה לבטא קשרים יותר מורכבים בתונונים שלנו.

▫ מאז אותה רשות תמיד משתמשים בקרנלים בגודל 3×3 (ולעתיתם רוחוקות גם ב- 5×5).

- הפסיקו להשתמש ב-Local Response Normalization (LRN) שהיה ב-AlexNet.

חשיבות גדולה מאוד של VGG היא ב-Perceptual Loss (המאמר המקורי של Li Fei – Fei, שאולי מוכרת לקוראים מהקורס CS231):

אנו מכירים כל מיני דרכים למדוד שגיאה בין שתי תמונות, כמו לדוגמה L^2 שעובדים באמצעות חישוב בין כל שני פיקסלים במקומות המתאימים. אולם, אם יש לנו שתי תמונות זהות, רק שאחת היא הזיה בפיקסל אחד של התמונה אחרת, השגיאה L^2 או L^1 תהיה עצומה(!) על אף ששתי התמונות הן ממשאותה תמונה.

יש צורך למדד שגיאה שהוא יותר קונספטואלי: שגיאה שמתאימה לאופן שבו אדם רואה תמונות. אם נסתכל על שתי תמונות בהזאה של פיקסל אחד, אנו נדע להגיד שהיא sama, אך L^2 לא מסוגלים לכך. לכן אנו צריכים שגיאה שהוא יותר High-Level: משווים יותר תוכן וסטיל.

שאלה כיצד נוכל למדוד Perceptual Loss?

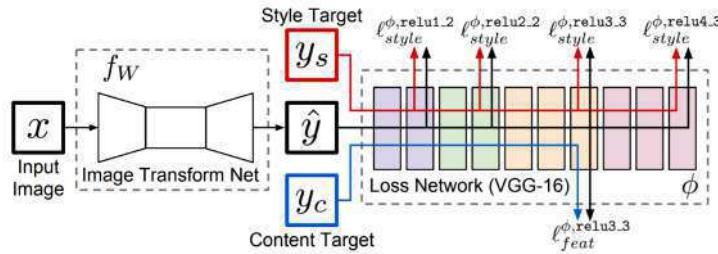
תשובה במקום השימוש ב- L_1 , ניקח רשות מיומנת (בדרכ' המשמשים ב-VGG16). שימוש לב, עכשו VGG16 היא לא רשות שאנו מאמנים, אנו רק משתמשים בה ככלי.

עתה, ניקח את התמונה הראשונה ונעביר אותה ב-VGG16, או בראשת כללית כלשהו. כל כמה שכבות ניקח את מפות האקטיבציה ונשמר אותן בצד. נעשה אותו דבר לתמונה השנייה ונשווה בין מפות האקטיבציה באמצעות ממד L_2 ,

לפי הנוסחה הבאה:

$$\ell^{\phi,j}(\hat{y}, y) = \frac{1}{C_j H_j W_j} \|\phi_j(\hat{y}) - \phi_j(y)\|_2$$

כאשר ϕ_j הוא וקטור ה- j -features מהשכבה ה- j של ϕ .



איור 316: המשכה ליצירת וקטור לצורך חישוב Loss Perceptual

שאלה למה זה עובד ולמה זה טוב?

תשובה הפלט של השכבות ב-VGG16 הוא תיאור קונספטואלי טוב של מה שמצאנו באותה תמונה, וזה תיאור יותר כללי של מה התוכן בתמונה שלא מתייחס למיקום ולערך של פיקסל ספציפי.
על כן, אם שתי התמונות דומות נצפה שഫוט האקטיבציה יהיו דומות, لكن שימוש ב- L_2 על מפות האקטיבציה הוא כליל יעיל.

העברת תמונה ב-VGG לצורך חישוב Perceptual Loss זה לא דבר מאד איטי (זה כמובן הרבה יותר איטי מאשר מסתם להימוש ב- L_2), لكن המון אנשים כן משתמשים בהזה.

GoogLeNet

- רשת עמוקה אף יותר ממוקדם: 22 שכבות.
- היה המון מאמץ ליצור רשת שהיא לא רק טובה, אלא גם עיליה. רשתות הן כבדות עם המון פרמטרים, אך חשוב ליעיל אותן.

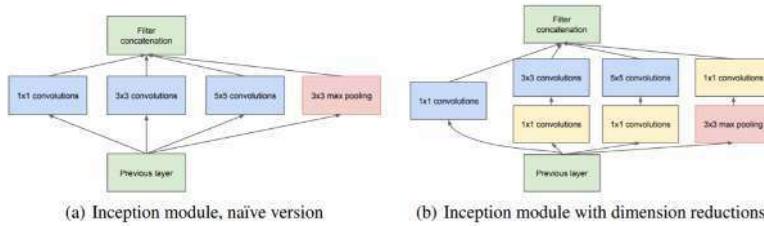
◀ הורידו את השכבות ה-Fully-Connected. אולי גם כיוון שאין להשתמש בהן. פשוט כי זה כל כך יקר חישובי ולא תורם מספיק.

◀ היו להם הרבה פחות פרמטרים: 5M לעומת 60M AlexNet עם 138M.

- השתמשו בבלוק בשם Inception Module

▷ במקומות לתוכנן את כל הארכיטקטורה של הרשת, הקדשו המון זמן בתכנון בлок של כמה שכבות שיהיה ממש טוב, וכל הרשת תהיה בנוייה ממחזירות של אותו בлок שוב ושוב. גישה של שימוש בבלוק חוזר תפסה ברשותות מודרניתות.

נסתכל עתה כיצד נראה בлок Inception Module אחד:



אייר 317: ארכיטקטורת ה-GoogLeNet Module Inception Module. הפלט של Module Inception הוא הקלט של ה-Module הבא.

בגרסתה הנאיבית (a) היו בעיות עם מימדים: הפלט של (a) הוא עצום בגלל שאחנו עושים *concatenation*, ולכן החישוב יהיה יקר.

על כן, בגרסתה המשופרת (b) הכניסו שכבת קונבולוציה 1×1 שהיא שימשה כ-*Dimensionality Reduction*. בפועל, קרנל 1×1 הוא צצ'ור $C \times 1 \times 1$ (כאשר C היא כמות העורצים שאחנו מקבלים כקלט). על כן, אם הקלט היה תמונה בעומק C , אז הפלט יהיה בעומק 1 - כלומר צמצמוו את כמות המימדים. אחנו לא נוגעים בגובה וברוחב, אלא רק בעומק. נשים לב שקרנל זה לא מסתכל על השכנים שלו כמו קרNELים גדולים יותר. אז קרナル 1×1 לא משמש למידה של משהו, אלא רק להורדת מים.

בנוסף לזה שהם משמשים להורדת מים, ניתן להתייחס לקרナル 1×1 כשכבה Fully-Connected (כה אמר Yann LeCun בפוסט הפיסבוק המפורסם שלו).

למה זה נכון? אם, כדוגמא פשוטה, הקלט הוא תמונה בגודל 1×1 עם 512 עורצים, ואחנו רוצים פלט עם 128 עורצים, נוכל להשתמש ב-128 קרナルים בגודל 1×1 (וכפי שאמרנו, זה בעצם קרナル $512 \times 1 \times 1$), שזו פשוט מטריצה בגודל 128×512 , וזה בדיקות דובר כמו להכניס את הקלט לשכבה Fully-Connected עם קלט בגודל 128 ופלט בגודל 128.¹¹

למה זה כל כך משמעותי? מחייבים אותנו לגודל מסוים של קלט - אם נכניס קלט בגודל אחר מהצפוי נקבל שגיאה בקוד. זאת בניגוד לكونבולוציה, שלא מצפה לגודל מסוים ולכן הקוד יכול לצלוח לרווח.

מסיבה זו, אוהבים המון פעמים לאמון רשתות קונבולוציה לא על תМОנות שלמות אלא על *Patch*-ים בתמונות (כי אין הרבה תМОנות וכו'). בזמן *test* כموון נתונים שלמות בגודל שאחנו מצפים לקבל.

¹¹ נציין שגם התמונה היא לא בגודל 1×1 , זה לא בדיקות דובר והמצב מסתובך מעת Yann LeCun השתמש במונח הלא-מובן (Full Connection Table). במקרה זה, כדי להפוך את הרשת ה-FCNN ל-Full Connection Table, יש לרשף קונבולוציה עם קרナルים 1×1 , נוצרך ממש להפוך את התמונה בגודל $C \times N \times M$ ל- $N \cdot M$ תMONות בגודל 1×1 עם C עורצים, ואז יהיו לנו $M \cdot N$ שכבות קונבולוציה עם פילטרים בגודל 1×1 . אך מתמטית, מעבר למקרה של תMONות בגודל 1×1 , שכבות FCNN הן לא אותו דבר כמו קונבולוציות 1×1 .

אך מי אנחנו שנענעים להתווכח עם Yann LeCun?

הערה בראשתות קונבולוציה אנחנו לא מצלמים לגובה ורוחב מסוימים, אך כן לעומק מסוימים.

ResNet

אחד המאמרים המפורטים והשימושיים ב-Deep Learning. Deep Learning ורשתות קבוצת חוקרים אימנה שתי רשתות: אחת עם 20 שכבות ואחת עם 56 שכבות. הם שמו לב שלא רק שה-test error של הרשת העמוקה יותר היה גבוה יותר, אלא גם ה-training error היה גבוה יותר!

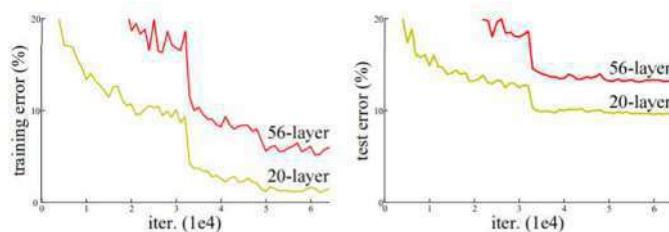


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

איור 318: הרשת העמוקה יותר הניבה תוכאות פחות טובות גם ב-testing וגם ב-training

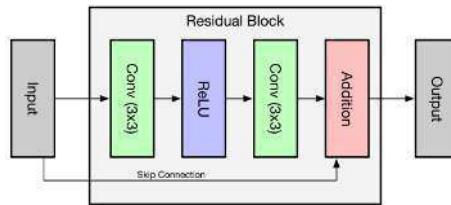
הדבר זה לא הגיוני: ככל שמעמיקים את הרשת אנו מוסיפים יותר non-linearities ומירות אמורה להיות יותר אקספרסיבית. וחוץ מזה, רשת בעומק 56 אמורה להיות טובה לפחות כמו רשת עם 20, כי רשת כזו יכולה לדמות במדויק את הרשת עם 20 השכבות באמצעות חיקוי הרשת ב-20 השכבות הראשונות, וב-36 השכבות הנותרות נבחר משקלות עם ערכים של 0 או 1 במקומות המתאימים כדי לא לשנות את התוצאה.

כלומר, ציפינו שרשת עם 56 שכבות תהיה טובה לפחות כמו רשת עם 20 שכבות, ולא גרווע יותר. בעקבות הבדיקה זו, הם ניסו להבין מה הבעיה באימון רשתות עמוקות.

הם הציעו להשתמש בבלוק שנקרא Residual Block. הרעיון מאחוריו הוא לחבר את ה-Input של הבלוק לא רק לשכבה שבאה אחריו, אלא “לידלג” על כמה שכבות ולהחבר אותן גם לשכבה יותר מתקדמת בראשת.

השכבה שהם חיבורו את ה-Input אליה היא שכבת Addition. כאמור, עבור קלט x וסדרת השכבות (F) הפלט של הבלוק

$$H(x) = F(x) + x$$



איור 319: מבנה ה-[Residual Block](#). ה-[Input](#) מחובר לשכבה כלשהי, אך מחובר גם לשכבה מאוחרת יותר. לחברו זהה אנו [Skip Connection](#).

שאלה למה זה טוב להשתמש בבלוק זה?

תשובה הבעיה הראשונה ברשת عمוקה היא גרדיאנט שהולך ונעלם ([Vanishing Gradient](#)). אנו מאמנים רשת באמצעות העברת התמונה ברשת מההתחלה לסוף, מחשבים Loss ומשנים גרדיאנט עבור כל משקלות באמצעות backpropagation (כל השרשרת). כך הגרדיאנט "מפעע" אחורה מהסוף להתחלה.

מה שקרה זה שככל שאחננו הולכים יותר אחורה עם הגרדיאנטים, הם הולכים וקטנים ויוצאים שהמשקלות בקושי משתנות. כך יוצא שהרשת عمוקה מדי והשכבות הראשונות בקושי מאמנות.¹²

ה-[Connection Skip](#) הוא מעין "בוסט אנרגיה" לגרדיאנט. במקום להעביר את הגרדיאנט מה-[input](#) ל-[output](#) דרך המון שכבות כך שהגרדיאנט יקטן ממש, אנחנו נסיף חיבור ישיר בין שכבת ה-[input](#) וה-[output](#) כך שהגרדיאנט לא ידען. כך אנו יכולים לאמן גם את השכבות המוקדמות ברשת. כמובן שגם השיטה הזאת מוגבלת, היא עבדה עם 152 שכבות אך כבר לא עבדה עם 1000 שכבות, אך בכל זאת מדובר בשיפור משמעותי.

נציג עתה את רשת ה-[ResNet](#):

- הורכבה משרשור של המון [Residual Blocks](#) והוא הכילה גם שכבות רגילים.

▷ מטפל ב-[Vanishing Gradient](#)

▷ אפשר [ארQUITקטורת Deep to Shallow](#) - אנו יכולים להתחיל במתן משקלות מאוד קטנים (כמעט אפס) לכל הבלוקים בלבד אחד. התוצאה האפקטיבית היא שהרשת מורכבת מבלוק אחד - ממש שטוחה.¹³ תוך כדי האימון המשקלות יתחלו גדול וזה-cailloו אנחנו מוסיפים שכבות תוך כדי האימון.

- הכילה 152 שכבות.

- השתמשה ב-[Batch Normalization](#) אחרי כל שכבת קונволוציה. (משתמשים בnormal הזה עד היום)

¹² קיימות בעיה נוספת, הפוכה, בשם gradient exploding עליה לא נדבר כאן. ברשת רגילה איפוס משקלות לא היה עובד, כי אז הפלט הסופי היה וקטורי האפס. אז לא בעיה: אם המשקלות בבלוק כלשהו הן אפס יש לנו את שיוויון ה-[Connection Skip](#) שיאפשר את הפלט של הבלוק הקודם ליקטור האפס של הבלוק הנוכחי.

Batch Normalization

יש לנו רצון לנרמל את הפלט של כל מיני שכבות בראשת כי שכבות שונות מוציאות דברים בטוחים שונים והרבה פעמים ההבדלים משמעותיים (אחת עם פלט ענק ואחת עם פלט נורא קטן). לרשותות נורא קשה להתכנס לתוצאה כיש "כאוס של ערכאים".

המטרה, על כן, הייתה לסדר את הפלטים שייהיו בטוחה יחסית קבוע ומוגבל, שאנו ידועים מראשות מתאמנות בו טוב. המטריה, על כן, הייתה לסדר את הפלטים שייהיו בטוחה יחסית קבוע ומוגבל, שאנו ידועים מראשות מתאמנות בו טוב. עבור כל מפת אקטיבציה שיצאה מהשכבה פועלם באופן Batch Normalization שמיים למשל אחרי שכבת קונבולוציה. עבור כל מפת אקטיבציה שיצאה מהשכבה פועלם באופן הבא:

1. נחשב ממוצע וסטיית תקן עבור כל הדוגמאות שיש ב-Batch.

2. נרמל את הממוצע ל-0 ואת סטיית התקן ל-1.

3. נבצע Scale ו-Shift באמצעות פרמטרים **למדים** γ ו- β . (לא ניכנס לה, זה פשוט נותן קצת יותר חופש)

יש אנשים שטוענים ש-BatchNorm הוא בעייתי כי הוא מתחנה שונה ב-train time וב-test time. זאת כי בזמן אימון משתמשים בממוצע וסטיית תקן של כל הדוגמאות ב-Batch, אבל בזמן test אין לנו Batch אלא תמונה אחת והממוצע והשונות של אותה תמונה קצת שונה.

באופן כללי BatchNorm משפר מאוד את תוצאות האימון. הוא עושה מעין "reset" של האקטיבציות לטוחה קבוע של ערכים בו אלגוריתם האופטימיזציה עובד היטב.

משימות נוספות ב-Computer Vision

מלבד קלסיפיקציה יש בעיות נוספות.

לדוגמה, *Detection*: לא להגיד סתם מה יש בתמונה אלא איפה. ממש להקיף במלבן את האובייקט.

דוגמה נוספת היא *Semantic Segmentation*: לסמן כל פיקסל בתמונה לאיזה אובייקט הוא שייך.



איור 320: מימין: *Semantic Segmentation*. משמאל: *Detection*.

מודלים גנרטיביים (Generative Models)

מודלים גנרטיביים הם מודלים של Unsupervised Learning, כלומר לא מקבלים תמונות עם ה-labelים שלהם. הפעם יתנו לנו המכון תמונות, לדוגמה של חתולים, ואני ננסה ללמוד מה התפלגות של תמונות של חתולים כדי שנוכל אחר

כך לדוגמה (לייצר) תמונות חדשות וריאליסטיות של חתולים.

נרצה שלמודל שלנו G יהיו שתי תכונות:

1. Sufficiency - לכל תמונה אמיתית x יהיה z עבורו $x = G(z)$. כלומר, המודל לא מייצר סתם תמונות של חתולים אלא יודע לייצר גם את התמונות שכבר ראיינו.

2. Compactness - לכל z , $G(z)$ צריך להיות תמונה תקינה ב- X (כלומר תמונה אמיתית ומציאותית).

יש לנו דרכיים את הבעיה שלנו (לא נדבר על כולן):

Auto – Regressive Pixel Generation •

VAE – Variational Autoencoders •

GAN – Generative Adversarial Networks •

Diffusion Models •

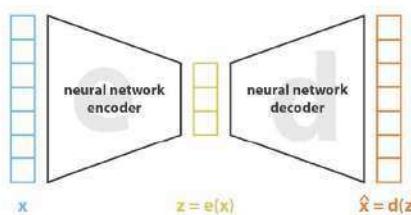
Auto – Regressive Pixel Generation

מייצרים פיקסל אחד כל פעם, החל מפהינה השמאלית העליונה. כל פעם מייצרים עוד פיקסל לפי הפיקסלים שייצרנו עד עכשו (מה הפיקסל הסביר הבא לייצר). התוצאות טובות אבל תהליכי היצור מאד איטי.

VAE – Variational Autoencoders

הרעין ב-*Autoencoders* הוא לייצר שתי רשתות: הרשת הראשונה היא *encoder* והשנייה היא *decoder*. כך אם נעביר תמונה ב-*encoder* נקבל וקטור (שמיידנו נזקק למינימום התמונה) שהוא יציג את התמונה. ווקטור זהה קוראים vector latent. וקטור זהה יהיה הקלט של ה-*decoder*, ונרצה שהפלט יהיה דומה ככל האפשר לתמונה המקורית.

פונק' ה-*Loss* שבדרכ' משמשים בה עברו *Autoencoders* היא *MSE*, אולם יש כיוון מאמריים שימושיים להשתמש דווקא *Perceptual Loss* ב-*Autoencoder*.



אייר 321: המחשה לבניה ה-*Autoencoder*

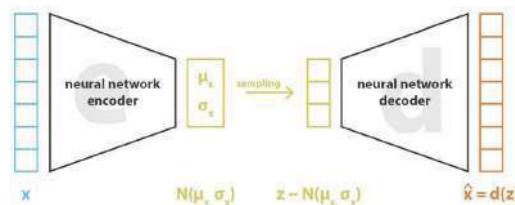
ה-vector latent הוא מימיד קטן יותר מהתמונה המקורי, שכן לא יכול להכיל את כל המידע שלה. על כן, נרצה שהוא יצலח להכיל רק מה שנחוץ כדי לתאר את התמונה "בהרבה פחות מילים" (לדוחס את המידע).

אנו מדברים כאן על מודלים גנרטיביים, שכן "נזרוק לפח" את ה-encoder ונרצה באיזה דרך לדגום ערכי z , להעבירם ב-decoder ולקבל תמונות חדשות.

עם זאת, אנו לא יודעים איך בנוי latent space וכאן אי אפשר סתם לבחור ממנו רנדומלית ערכי z ולצפות שנתקבל תמונות.

הרעיון ב-Variational Autoencoders הוא להכricht איזשהו "סדר" על latent space. כלומר, כאשרנו מאמנים את הרשותות נרצה שלא רק נכווץ את התמונה לאיזה latent vector, אלא גם שהוא יהיה אמיתי התפלגות שאנו מכירים, כך שנוכל לדגום ממנה.

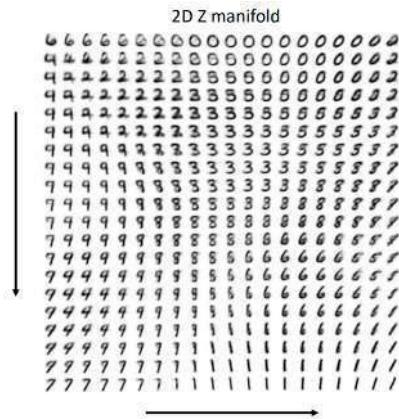
כדי להשיג זאת, נשנה מעט את המבנה של ה-encoder. במקום שה-encoder יוציא latent vector z , הוא יוציא ממוצע וסטיית. אז הקלט ל-decoder יהיה וקטור z שהתקבל מדגימה מתוך הגaussien שהמוצע והשונות שלו הוא המוצע והשונות שהוצאה ה-encoder.



איור 322: הפלט של ה-encoder יהיה ממוצע וסטיית תקין, והוקטור z , הקלט ל-decoder, יתקבל מדגימה לפי אותו ממוצע וסטיית תקין.

עכשו פונק' ה-Loss שלנו תהיה MSE , שזה loss של שחזור התמונה (reconstruction loss), וייה לנו גם regularization loss, שזה loss של השוני בין התפלגות הoriginale לבין התפלגות הנורמלית. קיימת מetric נוספת שמשתמשה בפער בין שתי התפלגות: Kullback-Leibler divergence (KL), אשר $KL[N(\mu_x, \sigma_x) || N(0, 1)] = \mu_x^2/\sigma_x^2 + \ln(\sigma_x/\sigma_0) - 1$. זה חשוב לנו כי אמרו היינו מתקבלים מה-encoder פלטים שהם גאוסיאניים כאשר כל גאוסיאן נמצא במקומות אחרים, ואז לא נוכל לדגום כמו שצריך, אבל כשאנו יודעים שהמרחב נראה בערך כמו גאוסיאן עם ממוצע 0 וסטיית תקין 1 (התפלגות הסטנדרטית), אנחנו יכולים לדגום מהגאוסיאן הזה ולקבל תוצאות טובות.

העובדת שהכרחנו את המרחב להתרחב בצורה הספציפית זו, הופכת אותו לסוג של "רציף". כל הנקודות הן משמעותיות ואנחנו יכולים לראות את המעבר הרציף בין הספרות השונות באյור הבא:



איור 323: ב-VAE יש מעבר רציף בין ערכי z שונים

GAN – Generative Adversarial Networks

הweeney-*GAN* הוא משחק בין שני שחקנים. יוצרים שתי רשותות והן יאמנו אחת את השניה, וכל רשות מנסה להטיעו את הרשות השנייה:

- רשות יוצרת - G (*Generator*), יוצרת תמונות שנראות אמיתיות מספיק כדי לגרום ל-D לחשב שהן אמיתיות. רשות זו היא *.unsupervised*
 - רשות בודקת - D (*Discriminator*), מנסה להבחין בין תמונות אמיתיות לתמונות שיוצרה G (מחזירה ערך בין 0 ל-1. שמבטא את מידת הבטיחו של האם התמונה אמיתית או לא). רשות זו היא *.supervised*

בהתחלת האימון שלה D קיבל תמונות של חתולים עם תווית שאומרת האם התמונה אמיתית או שהיא יוצרה ע"י G . תאמנו כדי לנסוט להבחין בהצלחה בין שני סוגים של תמונות.

G תנשה להשתפר ולהויא תמונה יותר ויוטר ריאלית כז- D -טעה.

כך באופן איטרטיבי נקבע את G , נוציא תכונות ונאמנו את D ונdag ש- D תשתפר.

נעוצר, נקבע את D , נגרום ל- G להוציא תמונות ונראה איך D מגיבה אליהם. אם D מזהה שהתמונה של G מזוייפת, אז מבינה שהיא צריכה להשתפר.

כך נמשיך בטהלה, ושתי הרשותות ישתפרו וישתףו עד שנהיה מסופקים.

בפועל יש לנו משחק minmax

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

באשר D מנסה לדאוג שיתקיים $1 \rightarrow D(x)$ ו- G מנסה לדאוג שיתקיים $0 \rightarrow D(G(x))$

כלומר פורמלית, במהלך האימון של D אנו מקבעים את G ומאמנים את D לפי הנוסחה:

$$\max_D \left[\mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log (1 - D(G(z)))] \right]$$

ובמהלך האימון של G אנו מקבעים את D ומאמנים את G לפי הנוסחה:

$$\min_G \left[\mathbb{E}_{z \sim p_z(z)} [\log (1 - D(G(z)))] \right]$$

הערה אימון של GAN -ים הוא מאד קשה ולא יצב! המונע פעמים אין התכונות, אבל הבעיה העיקרית נקראת Collapse Mode: במהלך האימון G מצליחה להזיאת תמונה ממש טובה ש- D לא מצליחה לקטול כראוי בשום שלב. על כן, אין ל- G אינטראס ליציר תמונות אחרות והיא מתחילה ליציר את אותה תמונה אחת ש- D נכשלה בה. קיימת בעיה נוספת, הפוכה, בשם Diminished Gradient באימון של G נעלמים לא מצליחה להשתפר.

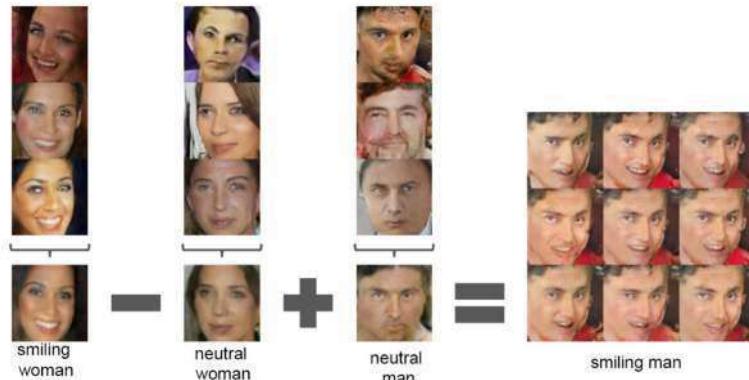
ה- GAN הראשון שעבד כמו שצרכיך הוא DCGAN – Deep Convolutional GANs. חוקרים ישבו וחקרו, עד שם מצאו כל מיני קווים מנהיים לנבי אייך צרייך לאמן ולבנות GAN :

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

איור 324: הקווים המנחים בנוגע לאימון GAN -ים אליהם הגיעו

דבר נוסף שעשו ב- $DCGAN$ הוא "latent arithmetic": לcko לדוגמה וקטוריים שונים תמונה של אישה מחייבת, חיסרו וקטור של אישה עם הבעת פנים ניטרלית, הוסיפו וקטור של גבר עם הבעת פנים ניטרלית וקיבלו וקטור של גבר מחייב:



איור 325: דוגמה ל-*latent arithmetic*. יש שמקפקים באמינות המחקר הזה.

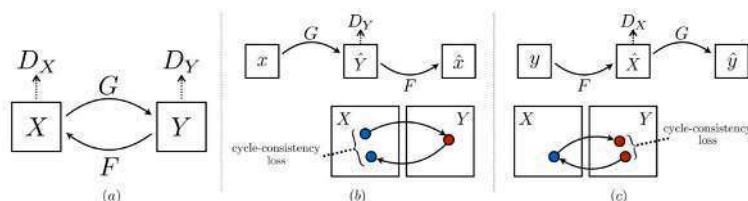
CycleGAN

המשימה ב-CycleGAN היא לחתת תמונה מדומין אחד X ולתרגם לדומין אחר Y . למשל, לחתת תמונה מדומין של סוסים ותרגם אותה לתמונה של זברה (לגרום לסוס להראות כמו זברה). כלומר G אמור להיות פונק' $Y \rightarrow X : G$ (שימו לב: הפעם לא מקבל latent vector אלא ממש תמונה של סוס).

הבעיה היא שגם אם הצלחנו לאמן GAN כזה בהצלחה, אין לנו דרך להבטיח ש- G יתן תמונה של זברה שמתאימה לאותו סוס. ומה שלא יזרוק את התמונה המקורית של הסוס לפח ויתן סתם תמונה רנדומלית של זברה? **אין קשר סמנטי בין הקלט לפלט!**

ב-CycleGAN הצעו פתרון באמצעות cycle consistency: החליטו לאמן שני F,G -ים, $X \rightarrow Y$ ו- $Y \rightarrow X$: F,G יהיו הפניות זה של זה. כלומר, שיתקיים $F(G(x)) \approx y$ ו- $G(F(y)) \approx x$. כדי לעודד זאת אנו מוסיפים cycle consistency loss.

עכשו ה- GAN חייב להשתמש בתמונה שהוא קיבל, והוא לא יוכל סתם לזרוק תמונה שלא קשורה לקלט שלו, בגלל שהוא צריך לשחרר את התמונה (כלומר שיתקיים $(G(F(Y))) \approx y$ ו- $F(G(x)) \approx x$).



איור 326: מכירח את הפלט של הרשת להיות קשור סמנטי לקלט, כי הוא דואג להעניש את הרשת אם $(G(F(x))) \approx y$ ו- $F(G(x)) \approx x$.

StyleGAN

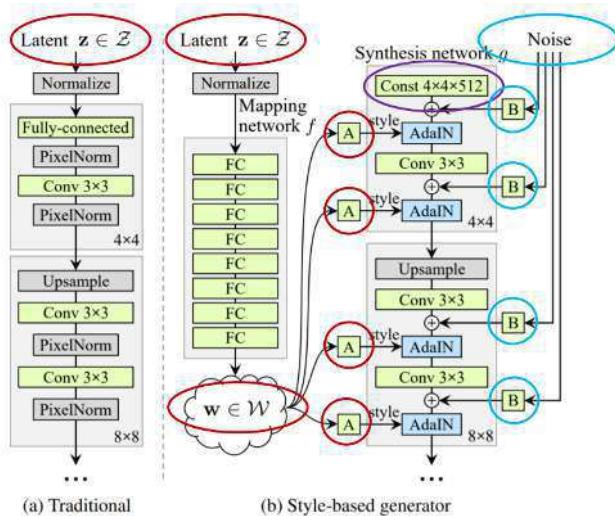
ב-StyleGAN הרעיון של החוקרים היה לנסות להבין יותר טוב latent space-GAN-ים ואת ה-latent space. הרעיון של החוקרים לקחחים מהעולם של ¹⁴style transfer רוצים שיהיה latent vector שஅחראי לסטגנון (style) של התמונה: האוזית של הפנים, מי הבן אדם, ועוד כל מיני תכונות כלליות. בנוסח, יהיה וקטור stochastic variation שהוא וקטור רעש שנotonin לרשת אפסיבי דברים שונים: לשחק עם השיער, לשנות קצת נמשים וכו'.

על כן, הפעם אנחנו לא נדגום רנדומליות וקטור z ונitin אוטו ל- G . במקומות זאת, הקלט ל- G יהיה קבוע במהלך תחיליך האימון (קידוד כללי של "מה זה פנים"). את הוקטור z נעביר בראשת אחרת שנאמן. הרשת האחרת, שנסמנה f , תהווה פונק' $f : Z \rightarrow W$, כאשר W הוא latent space אחר (המונ פעים גדול יותר). המטרה היא ש- f תתרגם את z לסטיל שאנו אנחנו רוצים.

כפי שניתנו לראות באיוור, הוקטור z מתרגם לוקטור w . הוקטור w לא נכנס קלט ל- G , אלא מוכנס אליו בכל מיני מקומות באמצע הרשת: זה כאילו אנחנו כל הזמן צועקים ל- G "תשמש בסטייל הזה!". מלבד זאת, אנו מזרים גם רעש רנדומלי שהוא stochastic variation עלייו דיברנו קודם. כך אם נקבע את w ונשנה את הרעש, נקבל את אותו אדם עם שינויים קלים (שיעור בתנוחה קצת שונה וכו').

הזרקה של w לתוך הרשת היא באמצעות שכבה בשם AdaIN (מזכיר קצת BatchNorm), עליה לא נפרט כאן.

שימוש לב שוב בראשת G השלב הראשון הוא להשתמש ב- $512 \times 4 \times 4$, שהוא הקלט הקבוע שאנו מאמנים שאומר ל- G "איך פנים נראה באופן כללי".



איור 327: מימין: ארכיטקטורת StyleGAN. משמאל: ה-GAN-ים "המסורתיים" שהשתמשו בהם קודם.

¹⁴לוקחים תבונה בסטייל מסוים, לדוגמה ציור של אן גון, ולהפוך אותה לסטיל אחר, לדוגמה ציור של מינה

41 תרגול 12 High Dynamic Range (HDR) - 12

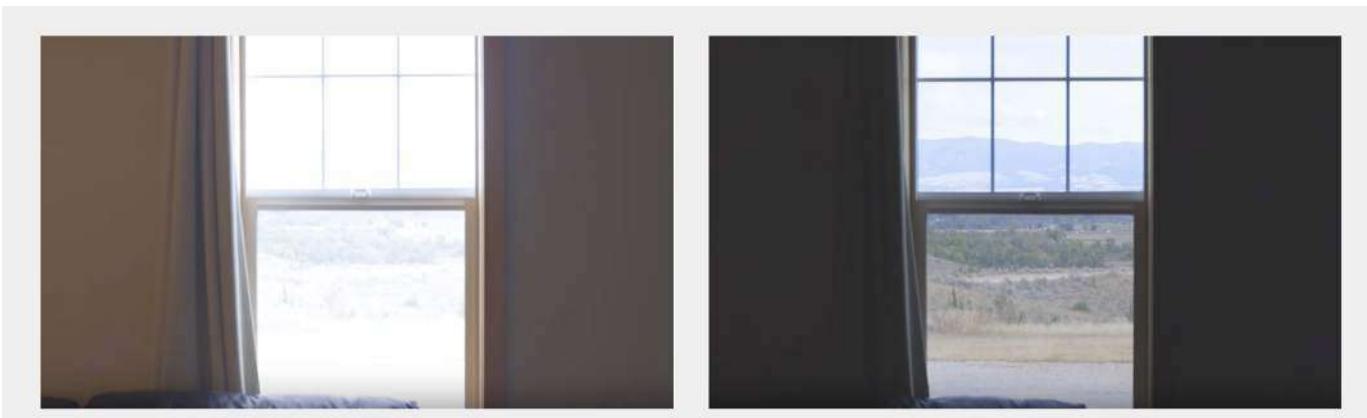
עתה עוסוק בתמונות *HDR*¹⁵.

תמונות HDR הן תמונות בעלות טווח צבעים גדול יותר מ-255 – 0, למשל 1024 – 0. טווחים אלה מאפשרים לשמר יותר מידע בתחום התמונה. ככלור כל פיקסל מיוצג על ידי יותר מ-8 ביטים. למעשה, אנחנו לא מסוגלים לראות תמונות כאלה. במצלמה, נניח כי כל סנסור מיוצג על ידי 8 ביט פר פיקסל.

רקע

בעולם שלנו יש הרבה תמורות שונות והעין האנושית שלנו מסוגלת להסתגל לתאורות אלה, למשל תחת תאורה נמוכה, נראה מעט מאוד בהתחלה, אך לאחר מכן יוכל לראות אובייקטים. מעבר לכך, העין האנושית מסוגלת לראות תמורות רבות שונות בו זמן קצר, ככלור לראות פרטים באזורי>bahirim>, וגם באזורי>cחיהם>.

המצלמה לעומת זאת, לא מסוגלת לעשות זאת עם טווח ערכים של 255 – 0, שכן בקונטיזיה שהמצלמה עשויה כדי להציג את העולם על המסך, ככל הנראה, תמורות שונות יפלו על אותו פיקסל.



איור 328: התמונה השמאלית צולמה בתאורה גבוהה, ושם רואים הרבה פרטים בווילון, ואילו מעט מאוד פרטים בחלון המואר. התמונה הימנית צולמה בתאורה נמוכה, ורואים פרטים רבים בחלון, ומעט מאוד פרטים בווילון.

HDR תמונות

הגדרה. ה-*Dynamic Range* של תמונה m מוגדרת להיות $\frac{\max m}{\min m}$.

הערה. רוב המקרים זה $\frac{255}{1} = 255$.

הגדרה. *Bit Depth* כמות הביטים שככל פיקסל מכיל.

הערה. רוב המקרים זה 8.

¹⁵ בשונה משאר התרגולים, תרגול זה הועבר על ידי דוקטור ענבר הוברמן

הערה. High Dynamic Range של תמונות מכיל ערכים בטווח גבוה יותר של בהירות ממלה שיש בתמונות רגילות (LDR). למעשה, כדי להציג את התמונה, נctrיך לדחוס את התמונה לטווח $0 - 255$, כמו "Imshow" נקבל דחיסה זו. על כן, נרצה לדחוס זאת לצורך שטיחור על המידע עבור העין האנושית.

איך יוצרים תמונות HDR

יש כמה דרכים.

1. כדי ליצור תמונה HDR, אנחנו מצלמים תמונה בכמה תאוות שונות, וכך כל תמונה מציגה אזור עניין אחר בסצנה. נוכל לאחד את התמונות באמצעות מסיכות ופירמידות. מטרתנו היא לאחד את התמונות כך שנשמר על ריאליות. תמונות שנוצרות מ-blending – α , והן מעתיקות פיקסלים מכל תמונה, ולכן נקבל תמונה לא מאוד ריאלית. יש אלגוריתם מתחכם יותר.
2. יש מצלמות עם חיישנים עם יותר מ-8 ביטים ואז ה-rawData שלחן מכיל את כל המידע. אך מידע זה הוא לא תמונה ה-Jpeg/Png שאנו מקבלים בסוף, אלא רק מידע בתוך המצלמה, ולכן יש עוד תהליך שחייב לעשות באמצעות בzew. בכלל, Jpeg/Png עישים דחיסה ל- $0 - 255$ עם שמונה ביטים לכל פיקסל.

שאלה אם נפעיל שיווי היסטוגרמה האם זה יספיק?

תשובה לא. שיווי היסטוגרמה מסתמך על האינפורמציה הקיימת בתמונה, ומוחת את כל הביטים, לכן לא נוכל לקבל חזרה את המידע שאבד במהלך הדחיסה.

Tone Mapping Operators (TMO)

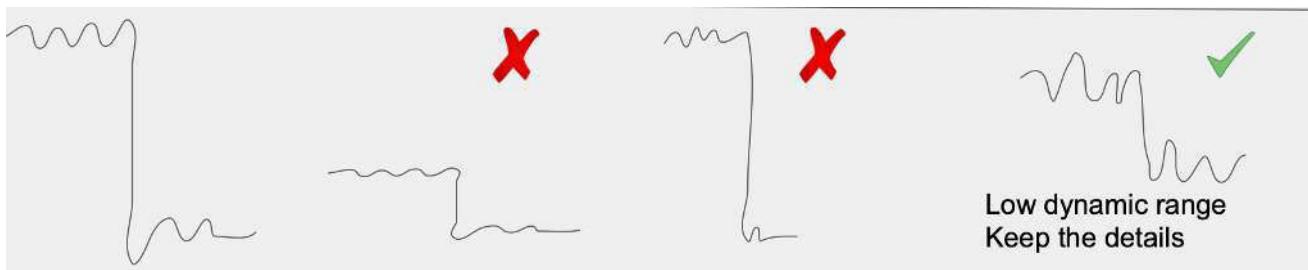
ה策עה. ניקח את תפיות ה-HDR m ונכצע $255 \cdot \frac{m}{\max m} + 0$ ויקבל ערכים בטווח $0 - 255$.

בעיה. התמונה שנתקבל לא מספיק יפה.

למעשה, שיטות שונות נבנות כדי להשיג מטרות שונות. אין אמת אבסולוטית בנוגע לטיב השיטה. יש מי שרצה תמונה יפה, יש מי שרצה לחוקת העין האנושית, ויש מי שרצה לשמור על התאורה המקורית.

מטרת אלגוריתם TMO כלל,

נניח כי יש לנו Edge מאוד חזק כמו בתמונה הבאה:



איור 329: הכי משמאל Edge חזק, ומימין צדדיו יש פרטם שנרצה לשמר עליהם, זה לא רוש. באמצעות דחיסה לא טובה של Edge ושל הפרטים, מכיוון שהיא מיותרת על הפרטים. מימין, דחיסה טובה של ה-Edge - טווח הערכים קטן, אך הפרטים נשמרים.

למעשה המטרה העיקרית של האלגוריתם היא לשמור פרטיהם בין Edges אבל כוונת דוחס את טווח הערכים.

אנו נלמד שלושה אלגוריתמים:



איור 330: משמאל אלגוריתם גלובלי, בהשוויה לאלגוריתמים האחרים הוא מעבד מידע באזורי הבהירותים ביחס לכחיהם. מימין אלגוריתם אחר שנותן תוצאות מעט שונות באזורי הבהירותים והכחים.

אלגוריתם OMO גלובלי

האלגוריתם משתמש ב-LUT כלומר $L_{out} = T(L_{in})$. הוא ממיר את התמונה לטווח ערכים 1 – 0 על ידי חלוקה בערך המקסימלי. לאחר מכן נוכל לעשות Correction – γ. אם התמונה בהירה נחזק אותה, אם היא כהה נבהיר אותה.

דבר זה בעייתי כי הוא נותן תוצאות מאוד גלובליות שלא מתחשבות בסביבה של כל פיקסל.

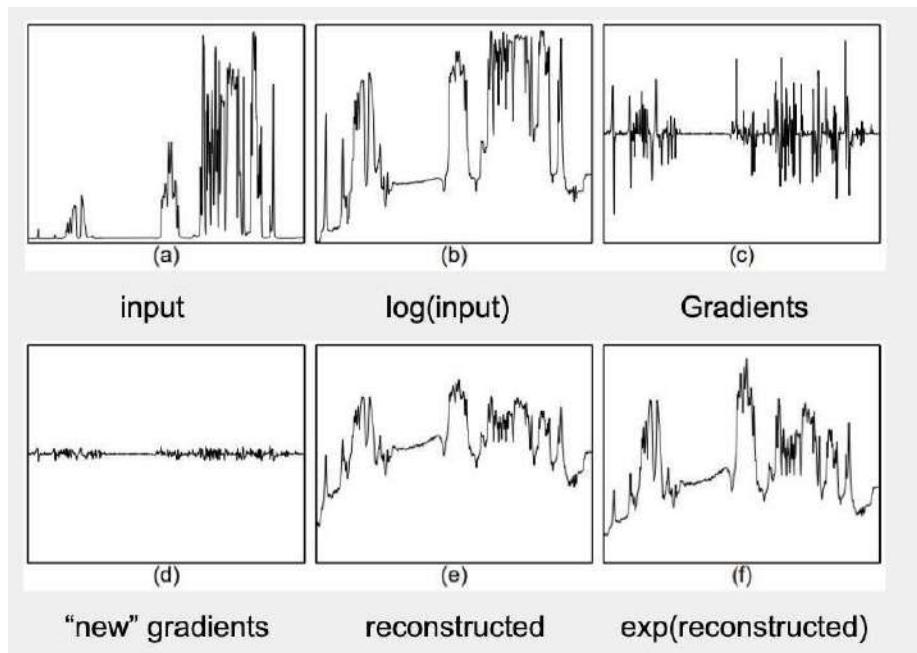
אלגוריתם TMO מקומי - 16 Gradient Domain HDR Compression

האלגוריתם משתמש על סיביות הפיקסלים באמצעות גרדיאנטים:

Algorithm 31 Gradient Domain HDR Compression

- 1 : **input:** Image m
 - 2 : **Compute** $H = \log(m)$
 - 3 : **Gradients** $H' = \nabla(\log m)$
 - 4 : **Attenuated Gradients** $G(x) = H' \cdot \Phi(x)$
 - 5 : **Reconstructed** $I(x) = C + \int_0^x G(t) dt$
 - 6 : **Fully Reconstructed** $\exp(I(x))$
-

לא נטען בביצוע אינטגרל לתמונה הגרדיאנטים. אך ניתן להבין מהי הפונקציה $\Phi(x)$ שעשתה פילטר על הגרדיאנטים.



אייר 331: המראה שלשלבי פעולה האלגוריתם

2002, Fattal et al.¹⁶

φ מוגדרת ע"י

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

כאשר β, α הם פרמטרים:

$$\alpha = 0.1 \cdot \text{mean}(\|\nabla H_k(x, y)\|)$$

$$\beta \in [0.8, 0.9]$$

בתווח הנ"ל. ו-מה זה אומר? נניח כי $\|\nabla H_k(x, y)\| = \alpha$ וכי בנקודת מסויימת הגרדיינט מאוד גבוה ביחס לממוצע. אנו כופלים את הגרדיינט בפקטור $\frac{\alpha^{1-\beta}}{\|\nabla H_k(x, y)\|^{1-\beta}}$ שלמעשה מקטין אותו. כמו בדוגמה שראינו קודם, לנקטו Edge והקטנו אותו.



איור 332: דוגמא למסיכה φ

אלגוריתם TMO מbas CNN

באלגוריתם זה נשימוש ברשומות כדי להשיג את מטרתנו. הבעה היא שאין לנו אמת אבסולוטית - No Ground Truth, בונגע להאם תוצאה טובה או לא. לכן התמונות יתחלקו לשתי קבוצות

1. תמונה ללא זוג שמייצג את התוצאה שאנו רוצים להשיג.

2. ותמונה שיש לה זוג.

למעשה יש לנו שלוש מטרות

1. הורדת טווח הערכים ל-0 – 255.

2. סידור הצבעים ככה שנקלם תמונה שנראית טוב בעיני.

.HDR Tone Mapping .3

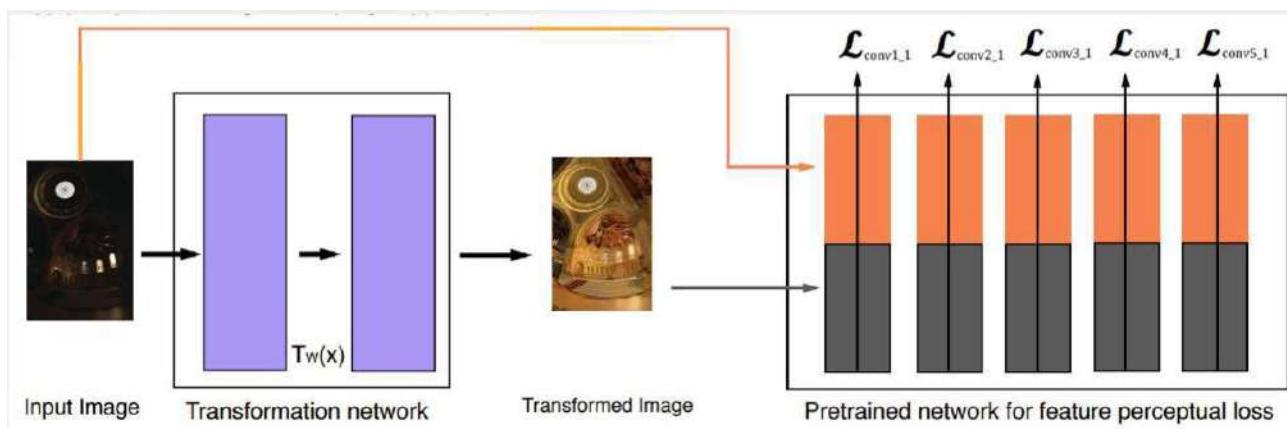
האלגוריתם של Hou, 2017 למקה בו אין זוגות

בהתחלת, הרשת לא מאמנת, אבל כן מוציאה תמונות שניות לחיצג. לכן משתמש ב-*Perceptual Loss*. כמובן שברשת נירונים המאמנת כבר על תמונות ומיועדת לקלאסיפיקציה, האם אתם מכירים רשות כזו? כן, VGG. לכן אנחנו משתמשים ברשת נירונים המאמנת כבר על תמונות ומיועדת לקלאסיפיקציה, האם אתם מכירים רשות כזו? כן, VGG. על כן, מתקובל האלגוריתם הבא:

Algorithm 32 CNN Training Using Perceptual Loss

- 1 : **input:** HDR Image m_{in}
 - 2 : Pass m_{in} through the transformation network and output output image m_{out}
 - 3 : Pass m_{out} and original HDR image m_{in} through VGG
 - 4 : **Compute Loss function** by simply computing the **perceptual Loss**
 - 5 : **Update** the transformation network's weights
-

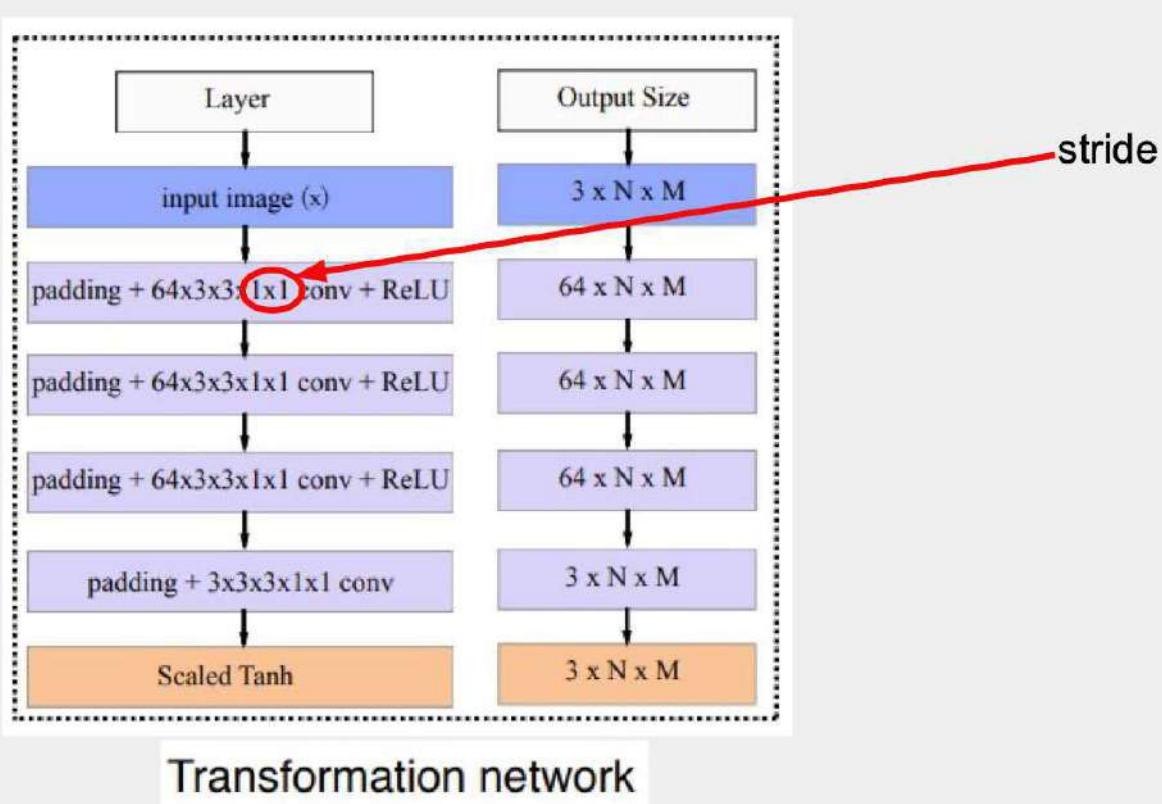
באיור הבא ניתן לקבל המראה לפועלות האלגוריתם.



איור 333: המראה לפועלות האלגוריתם

מסתבר שההשוואה לפלטיהם של רשת ה-VGG מספיק טובה כדי שהרשת שלנו תתפתח כמו שאנחנו רוצים.

באיור הבא נוכל לראות את מבנה הרשת שלנו:



איור 334: מבנה הרשת - שלוש שכבות של קונבולוציה + ReLU ולאחר מכן שכבה של Pooling.

מציאות Ground Truth

יש שלוש שיטות עיקריות ליצירת Ground Truth אך אלו נציג רק שתיים.

1. השתמש באלגוריתמי TMO ונבחר את התוצאה הטובה ביותר ביחס לשני מדדים - טבעיות ופירוט המידע.

2. ניתן לצלמים מצלזונים את תמונות HDR ובעזרת ערכיה גרפית ניצור תמונות רגילים כדי יפות שניתן.



איור 335: המלצה לשיטה הראשונה

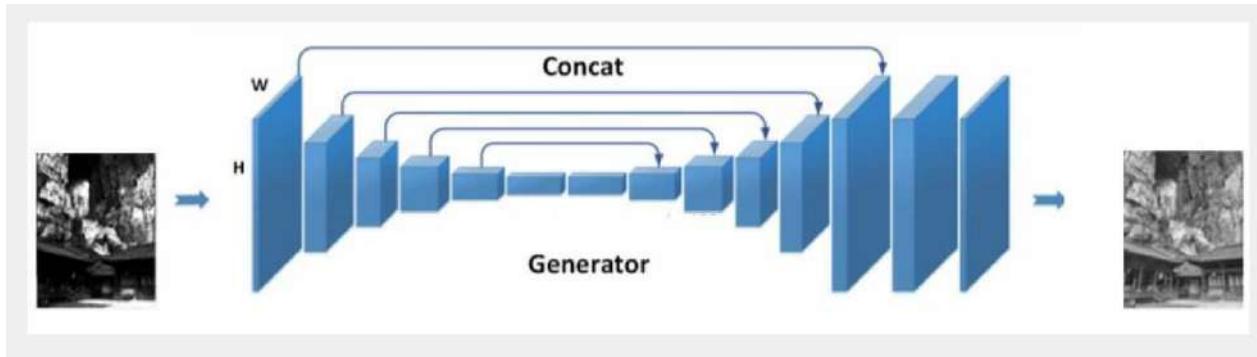
לאחר שמצאנו את הזוגות, עלינו לאמן רשת. כיצד נעשה זאת?

למעשה יש גנרי למטרות אלה, המתאים כמעט לכל שימוש Comuter Vision Tasks

Algorithm 33 Generic Pipeline For Computer Vision Tasks

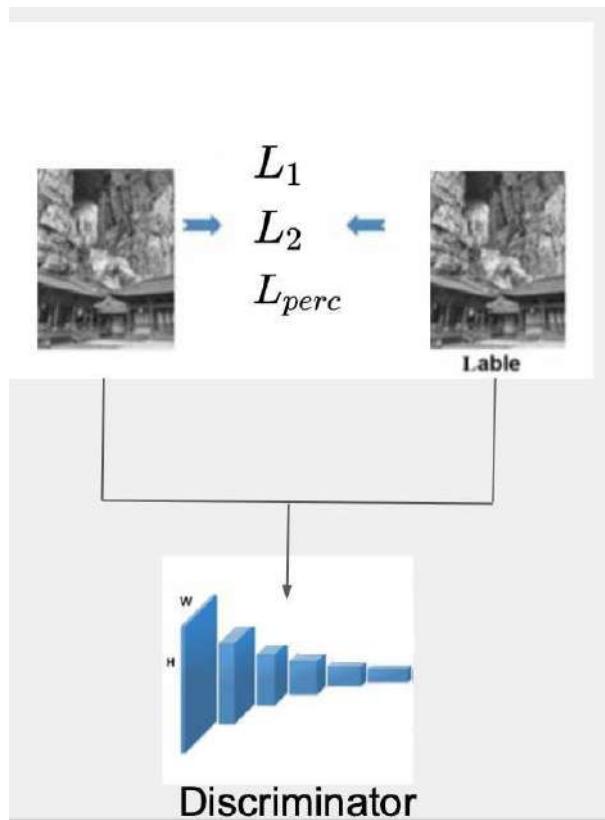
- 1 : **input:** set of HDR Images and their pairs. Denote each image by m_{in}
 - 2 : **Pass** m_{in} through the transformation network and output output image m_{out}
 - 3 : **Compute Loss function** by using L_1/L_2 or perceptual Loss with the paired image
 - 5 : **Update** the transformation network's weights
-

ובאיור הבא ניתן לקבל המ חה לאלגוריתם זה: |



איור 336: הרשות היא רשות הקטנה וגדלה חזקה. המפרט הטכני של הרשות פחות מעוניין אותנו כרגע, אם נתיחס אליה כקובסא שחורה, נקבל גנריות. לאחר שקיבלו את הפלט נחשב את השגיאה ביחס לוג.

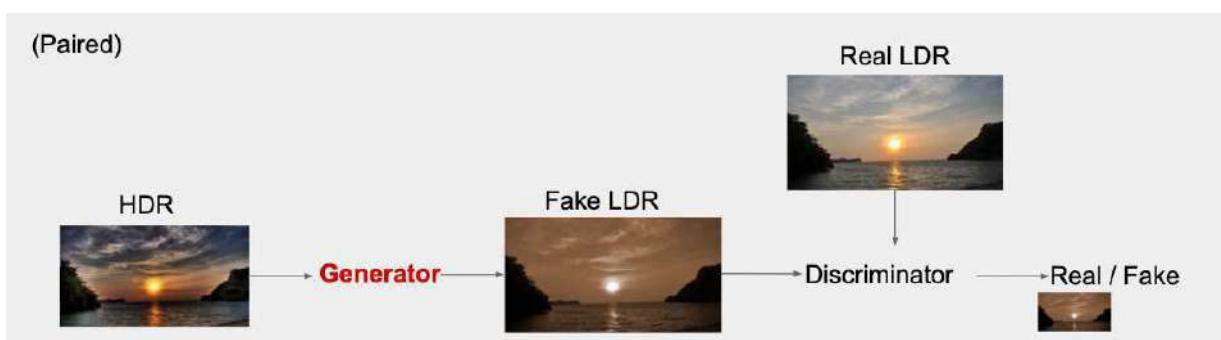
:Discriminator, כיצד נוכל לשפר את זה? במקומם להשתמש בפונקציית Loss פשוטה השתמש ב-Adversarial Loss וב-



איור 337: נשתמש ברשת המסוגלת לקבוע כמה קרובה התוצאה לתמונה ה-LDR.

למעשה מה שיקר זה שהרשת שלנו תנסה את המשקלות שלה לפי התוצאה של ה-Discriminator. בהתחלה היא לא תצלית, אך לאט היא תלמד איך להוציא תכונות שיבלבו את ה-Discriminator.

זה נראה כך:



איור 338: הרשות שלנו היא רשות ה-Discriminator אותה אנו מאמנים באמצעות רשות ה-Generator שמוסיפה True/False ליפוי ההתאמה של תמונה הפלט לתמונה ה-LDR.

ככה נודע שתמונה הפלט לא תהיה מוטושטשת.

להלן קישור למאמר של מעבירות תרגול זה - דוקטור ענבר הוברמן.

42 תרגול 13 - סיכום ויזאו (Video Summarization)

ההערכה כיום היא ש-85% מהתוכן הדיגיטלי מורכב מסרטוני ויזאו. כאשר מעריצים כי כ-70% בערך צולמו על ידי מצלמות אבטחה. הבעיה ביזאו היא האורך שלו: אנו נרצה להיות מסוגלים לסכם אותו (Video Summarization) בצורה כזו שנקבל את המידע הרלוונטי.

למשל, עבור צילום אבטחה של הבית לאורך כל היום, לא נרצה לראות את כל הסרטון כדי להזות מי הבן שנכנס אל הבית כי זה כמו לחפש מחת בערימות שחת. אנו נציג שני מאמרים המציגים פתרון לעביה:

- **המאמר הראשון:** הפיכת היזאו לתמונה סטטית - נפהוך את היזאו לתמונה סטטית המורכבת מפרייםים שונים ביזאו
 - מעין תמונה פנורמה שתכילה את כל האינפורמציה.

- **המאמר השני:** נסכם את היזאו ליזאו דחוס יותר -¹⁷ Video Synopsis

42.1 סיכום היזאו לתמונה אחת - Video Indexing Based On Mosaic Representation

נעסק במאמר על ¹⁸ Video Indexing Based On Mosaic Representation ויזאו הוא למעשה רצף של המון פרייםים של אותה סצנה. לכן יש חזרתיות, למשל, עד שאובייקט נעלם לגמרי מהפריים עוברים הרבה פרייםים.

הרעין בשיטה זו, הוא לוקח ויזואו המוצג בצורה פרייםים וליצג אותו בצורה סצנות, אנו מחלקים את הרכיבים העיקריים בסצנות לפני הקטגוריות הבאות:

- **מידע מרחבי סטטי** - למשל, כסאות בכיתה,لوح, קירות וכו'. רקע סטטי.
- **תנועות** - אובייקטים זזים.
- **מידע גאומטרי** - מבנה הסצנה התלת מימדית, הטרנספורמציה הגאומטרית שנוצרת עקב תזוזת המצלמה. המידע החדש שנחשפנו אליו עקב הזזת המצלמה.

מידע מרחבי סטטי (Extended Spatial Information)

נרצה להציג את כל המידע הסטטי ביזאו בתמונה אחת. כיצד? ניצור תמונה פנורמה המורכבת מפרייםים שונים ביזאו שתכלול את כל המידע הסטטי בחדר.

¹⁷ הטכנולוגיה פותחה באוניברסיטה העברית על ידי שמואל פלאג, מרצה הקורס.

Michal Irani and P. Anandan (IEEE 1998)¹⁸

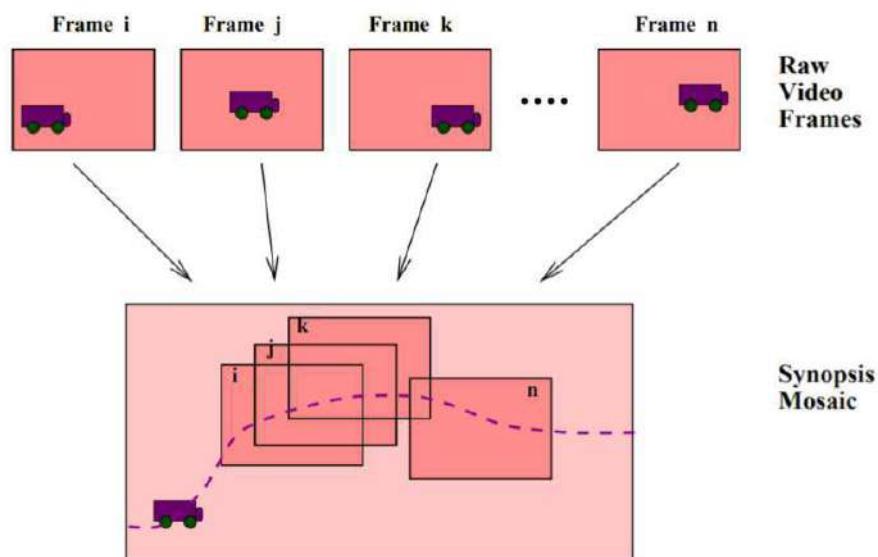
מידע על תנועות (Extended Temporal Information)

התפתחות האירועים לפי הזמן מפוזרת על כל הפריים, ולכן נרצה לסכם אותה. למשל, אם אדם קופץ כשהוא מגיע לסוף החדר נרצה לדלג על החלק שבו הוא הולך עד הקפיצה. נרצה לראות את כל התנועה בפעם אחת. לכן, אנו נזהה את מסלול התנועה של האובייקט ונציג אותו כאובייקט אחד. (Trajectory)

מידע גאומטרי (Geometric Information)

נרצה למפות אירועים מהוידאו אל תוך הפנורמה בצורה כזו שנוכל לחזור חזרה לוידאו ומשם לפנורמה. זהו למעשה מעבר בין מערכת הקוארדינטות של הוידאו למערכת הקוארדינטות של הפנורמה.

בקיצור, התהיליך יראה כך:



איור 339: פריימים שונים ממופים לפנורמה ויוצרים יחד את מסלול המשאית כאובייקט יחיד.

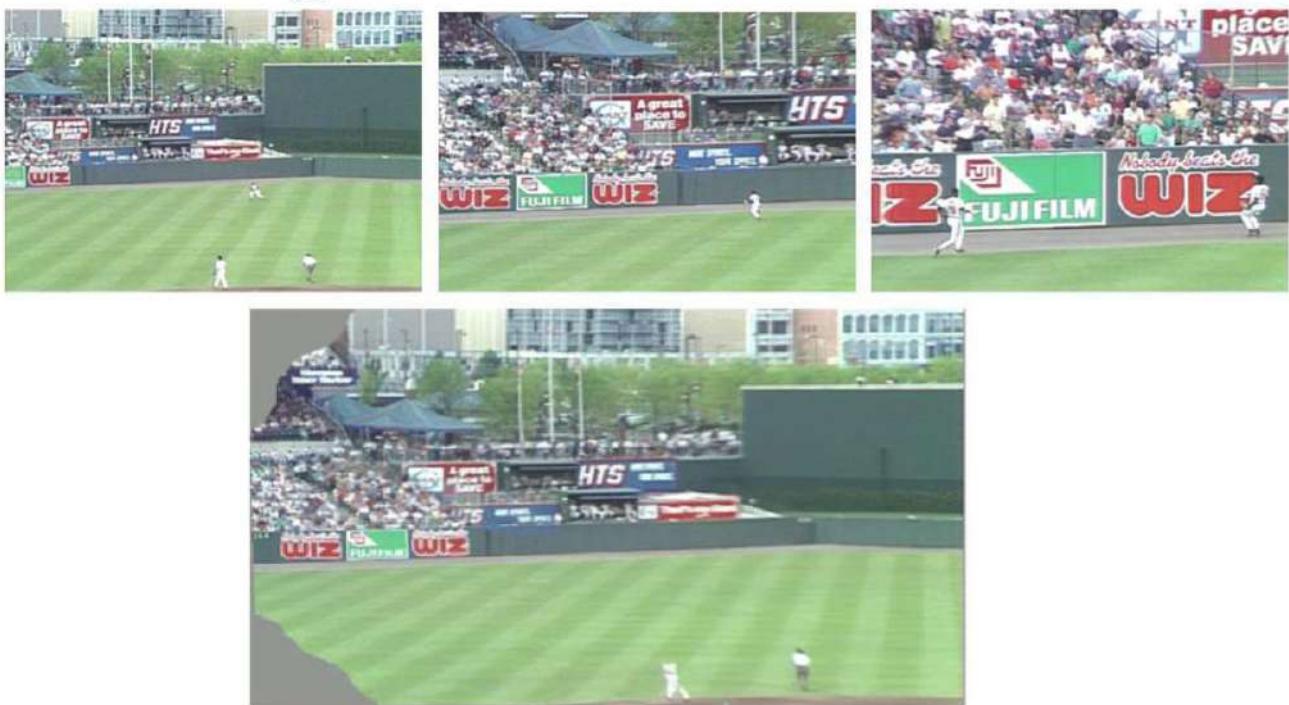
יצירת התמונה הסטטistica Static Background Mosaic

את התמונה הסטטistica, כפי שהוזכר לעיל, נוצר על ידי יצירה פנורמה של פריימים. אך נרצה לכלול אך ורק אובייקטים סטטיים, ללא אובייקטים זזים. כשিיצרנו פנורמות, רצינו לכלול אובייקטים זזים פעם אחת בתמונה (לכן השתמשנו למשל ב-*MinCut*), ואילו כאן רוצים לכלול את כל ה움ות שלהם, אך נרצה להפריד את שתי המשימות - תמונה סטטית ולאחר מכן נוסיף אובייקטים דינמיים.

עליה השאלה, כיצד נמחק את האובייקטים הנעים? נסתכל על פיקסל מסוים בכל פריימים, נצפה שברוב המוחלט של פריימים, יהיה אובייקט סטטי ואילו בכמה רצפים קצריים יופיעו אובייקטים נעים, لكن נמצע את ערך הפיקסל לפי ערכיו לכל

אורך הפריים וכן נקלט שאובייקטים זרים יפלו במשמעותם. דרך נוספת היא לkishת החזין, או יצירת הסטוגרמה של כל ערכי הפיקסלים ולkishת הערך הנפוץ ביותר (כך נפתר מרעשים).

זה נראה כך:



איור 340: ניתן לראות שכל הרקע הסטטי נשאר, וכך על פי שהשחקנים בתמונה השמאלית נעים, מכיוון שהם נעים במקום, אנחנו מחשבים אותם כאובייקט סטטי. לעומת זאת השחקנים בתמונה הימנית נעלים, כפי שרצינו. ניתן לראות שבפנורמה הסופית השחקנים שנשארו מעט מושלמים, זה בסדר, מכיוון שאנחנו רוצחים לראות התנהלות כללית בתמונה, שתכלול כמה שימוש מידע, איננו מעוניינים בפנורמה הכייפה.

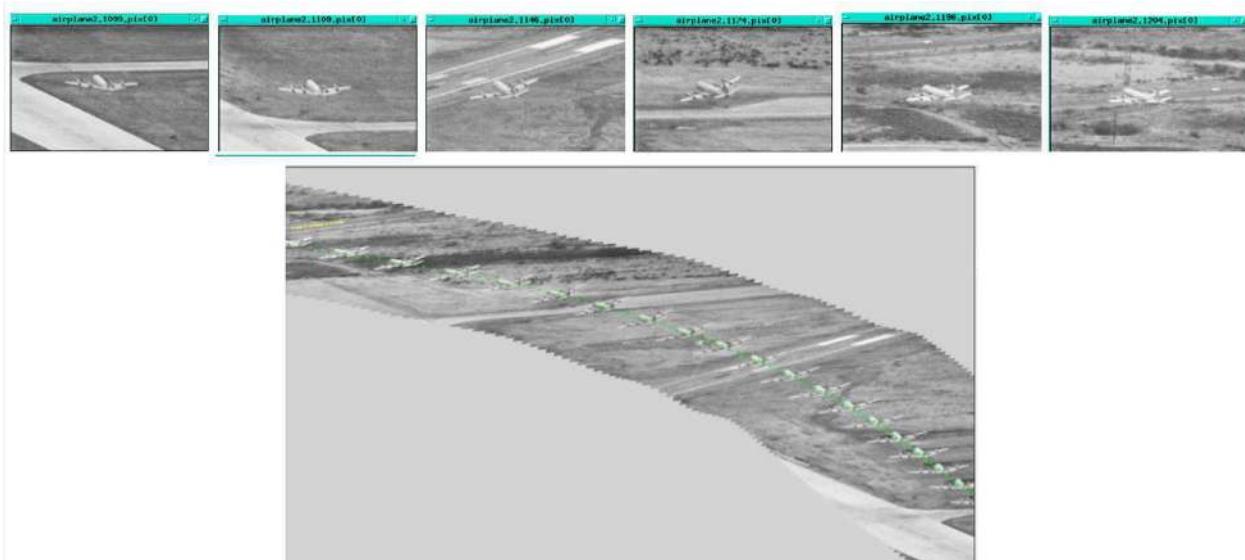
יצירת התנועה Synopsis Mosaic

עכשו אנו רוצים לחת את כל האובייקטים שנפלו בשלב הקודם, הם יחשבו חלק הדינמי בתמונה, ונחשב את ה-trajectory של כל אובייקט - התנועה של האובייקט בין הפריים.

כדי להזות את התנועה של אובייקטים בין הפריים נוכל לחשב את ההומוגרפיה בין שני הפריים באמצעות נקודות תואמות, או פשוט להשתמש בשיטות יישירות כמו LK.

את ההומוגרפיה נשמר לשלב הגאומטריה בסוף כדי שנוכל לעבור בין מערכות הקוארדינטות.

למעשה, אנו לוקחים קבוצת פריים עוקבים והופכים אותם לתמונה אחת שמייצגת את תנועת האובייקט ומציגים את כל הפעולות שלו. ככה למעשה, יחד עם התמונה הסטטית אנו קולטים את כל האינפורמציה בתמונה:



איור 341: אנו מאחדים את כל התמונות לכדי תמונה אחת בה המטוס מופיע בכל המופעים שלו ונוצר לנו המסלול (Trajectory)

למעשה, כל תנועה של אובייקט נציג על ידי Tube - צינור שדרכו האובייקט נע.

אנחנו לא מתייחסים כרגע להתנסויות ולפרטים הקטנים, את זה נראה במאמר הבא.

42.2 סיכום היזdeo ליזdeo קצר - Nonchronological Video Synopsis and Indexing

אנו נהפוך את היזdeo ליזdeo קצר יותר. יש שתי שיטות מוכרות לביצוע משימה זו:

- **סיכום מבוסס פרימיים** - נזהה פרימיום חשובים וחלקים קרים מהיזdeo, ואך נבצע FastForward.

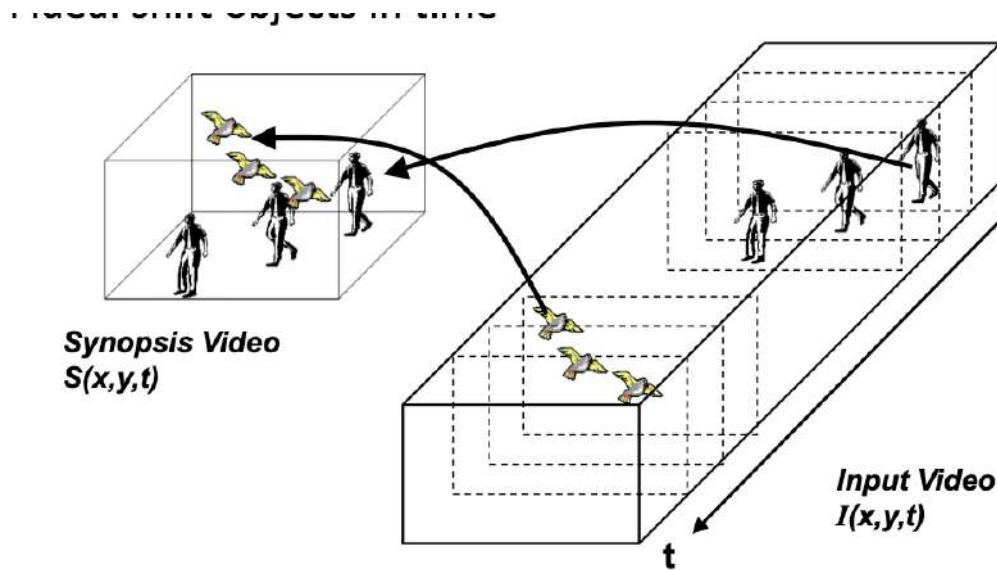
↳ סיכומים כאלה מאוד מוגבלים כי הם כוללים את עצם בתוך הפרימיום: מה שהיה בפרימיום זה מה שנכון ולא נשנה את אותו (אולי) נשנה את הסדר של הפרימאים או הקצב, אבל לא את תוכן הפרימאים.

• **סיכום מבוסס אובייקטים** - כאן אנו מותרים על הסדר הכרונולוגי של הופעת האובייקטים ומאפשרים להם להופיע במקביל, ללא FastForward, ובכך נשמר את הדינמייה. כמובן, אם אובייקט רץ בסרטון הוא יירוץ גם בסרטון המקורי, ההבדל הוא שנציג את כולם ביחד.

↳ המשך די טבעי של השיטה הקודמת של סיכום ויזdeo לתמונה אחת.

אנו נבצע סיכום מבוסס אובייקטים¹⁹, שהוא פתרון חדשני:

Yael Pritch, Alex Rav-Acha and Shmuel Peleg (PAMI 2008)¹⁹

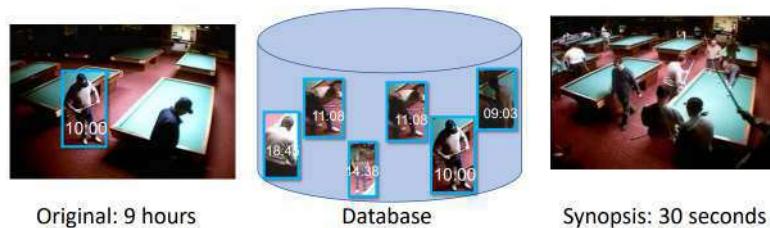


איור 342: ניתן לראות כאן המחשה לסיקום מבוסס אובייקטים, אנו לוקחים התנהגות של אובייקטים מסוימים שומנים ומציגים אותן באותו זמן בסרטון החדש.

הערה כפי שנראה, אנו נרצה לשמר רפנס זמן ממנו לקוח כל אובייקט, בדומה ל-Geometric Information. כך אם נסתכל על סרטון כלשהו ונראה אדם חדש, נוכל לחזור לסרטון המקורי ולהתמקד בו. כמו כן, השיטה שאנו נציג עתה עובדת טוב הסרטונים ארוכים (כמו אלה שיש במצולמות אבטחה). על כן, אנו נוסיף בשיטה זו שינויים שלא התייחסו אליהם בשיטות קודמות כי שם עבדו על סרטונים קצרים.

אנו נבדוק בשני שלבים:

1. **online phase** - נרצה לאזות את האובייקטים שקיימים הסרטון ולעקוב אחריהם (ציורות) ולשמור אותם ב-database.
 2. **query phase** - לפי בקשה, query, של משתמש (לדוגמה, "תן לי 30 שניות של היממה الأخيرة), נבחר את האובייקטים הרלוונטיים ונפתחו אותם מתוך הסרטון המקורי.



איור 343: השלבים ב-Video Synopsis

תכונות

נניח שיש לנו N פריטים מודיאו הקלט. נוכל לצייר אותו בצורה תלת מימדית (x, y, t) אם נוסיף מימד זמן שיציג את מספר הפריטים, כלומר $N \geq 1 \leq t \leq N$ המיקום הקרטזי.

נרצה שודאו הפלט $S(x, y, t)$ יקיים את התכונות הבאות:

1. S קצר יותר משמעותית מ- I .

2. נרצה הופעה מקסימלית של מידע מהוידאו המקורי I (מקסימום פעילות תנועה ייכנס ל- S).

3. נרצה לשמר דינמיות, כלומר ללא שימוש ב-FastForward - מי שהלך הלא ומירץ רצ.

4. נמנע מתפרים גלוים ואובייקטים מפוצלים.

זיהוי תמונה וסגנונציה

עלינו להזוהה תזואה מעניינת של אובייקטים. ההנחה הכללית היא "מעניין \approx זז". הנחה זו כמובן אינה לגמרי נוכנה: למשל, תנועה של עננים ועלים היא לא מעניינת ואילו אובייקטים מעניינים כמו אנשים עלולים להיות סטטיים. כדי להתגבר על הבעיה, את זיהוי האובייקטים נוכל לבצע באמצעות שיטות זיהוי קיימות. כמובן, את התנועה ניצג **כגינור** במרחב התלת מימדי (x, y, t) - מי היה איפה ובאיזה זמן.

יחד עם זאת, עלינו להתחילה בזיהוי רקע ובנитו $B(x, y, t)$. בזיהוי הרקע ובניטו בעיה שנוצרה להתחמיך איתה היא שההתנועה עלולה להשנות לאורך זמן, למשל, אם נצלם פארק לאורך כל היום תהיה תאורה שונה בין הבוקר לצהרים הערב.

עשויות ממווצע בין כל הערכאים יהיה מגוחך. לכן, כדי לפטור את הבעיה ניקח סביבה של הזמן (למשל 10 דקות) וניקח ממש את החציוון, או כמו שעושים בפועל: נעשה היסטוגרמה לאורך מרחב הזמן per-pixel עברו אותה סביבת זמן. כמובן, לכל פריטים ניקח את הפריטים שנמצאים 10 דקות לפני ו-10 דקות אחרי, ונשתמש בהם כדי לחשב את הממווצע. כן, בסרטון התוצאה לאורך הזמן רקע ישתנה.

ככה אנו נקבל תמונות רקעים מייצגות לטוחי זמן מן מסויימים במקומות תמונות רקע לכל נקודת זמן. על תמונות רקע אלה, נוסיף את האובייקטים הנעים באותו זמן - להחליפם רקעים מזומנים שונים זה דבר ענייתי שכן אנו עלולים לראות פחות טוב את האובייקטים.

מיציאת צינור התנועה - Activity Tube

לכל תמונה I_t ניצור פונקציית Label שנסמנה $f_t \in I_t$ כך שלכל r יתקיים

$$f_t(r) = \begin{cases} 1 & \text{אובייקט על הרקע (גע)} \\ 0 & \text{רקע} \end{cases}$$

על כן, נרצה למשער את ההפרש בין תמונה הרקע לבין הרקע שנסיך באמצעות f_t . כלומר, נמצא את f_t באמצעות הגדרתنا להיות הפונק' שמשמערת איזהי שגיאה.

ונכל לעדן את המזעור לפי הפרמטרים הבאים, מבלי להסתמך על תמונה רקע, שאין לנו:

(data term) E_1 • פונקציית שגיאה שתווידא שפיקסלים שונים מהרקע יהיו מותגים כאובייקטים על הרקע.

(smoothness term) E_2 • פונקציית שגיאה שתווידא חלקות - פיקסלים הקרובים האחד לשני ודומים, יקבלו אותו תיוג.

על כן, אנו מקבלים את פונקציית השגיאה הבאה ($-f_t$ צריכה למשער>):

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

כאשר $N(r)$ היא סבירה קטנה של הפיקסל r .

הערה אמם רשותנו ש- E_1 מקבלת את (r, f_t) , אך היא כמובן צריכה לדעת מהו r .

הערה ה-data term הוא הניחוש הראשוני הטוב שלנו. ה-smoothness term מחלק לנו את זה - גם אם הפיקסל ליד קצת שונה וחישוב הרקע לא יהיה מספיק טוב אני צריך להתנהג כמוותו.

לאחר שמצאנו פונקציה f_t לכל פריים I_t , נוכל לקבל את כל האובייקטים בתמונה ואת הרקע. עתה, לפי החפיפה בין אובייקטים בפריים עוקבים נוכל להסיק את צינור התנועה. אנו מקבלים מעין רכיב קשריות - כל פריים מחובר לפריים הקודם לפי החפיפה של האובייקטים והמסלול הקשר אוותם הוא הצינור (קיים הблокים השונים לפי התנועה של האובייקט ביניהם). למעשה, כל צינור b מוגדר בזמן סופי בוידאו המקורי $t_b = [t_b^{start}, t_b^{end}]$ ונוכל להציג את הצינור לפי הפונקציה האופיינית

שלו

$$\chi_b(x, y, t) = \begin{cases} \|I(x, y, z) - B(x, y, t)\| & t \in t_b \\ 0 & \text{אחרת} \end{cases}$$

ואז אנו מקבלים בדיקת הצינור, ללא הרקע.

התנשויות

נchap פונק' מיפוי M שתקבל את הציגור b ותיתן לנו צינור חדש: לכל צינור b קיבלנו מרוחז זמן בוידאו המקורי $t_b = [t_b^s, t_b^e]$ נרצה למת לו טווח זמן בוידאו החדש $\hat{t}_b = [\hat{t}_b^s, \hat{t}_b^e]$. אם b לא כולל סרטון נרצה כי $M(b) = \emptyset$. נרצה להשיג סרטון אופטימלי - הקצר ביותר מכל הסרטונים שמרתא את כל האובייקטים, עם התנשויות מינימלית.

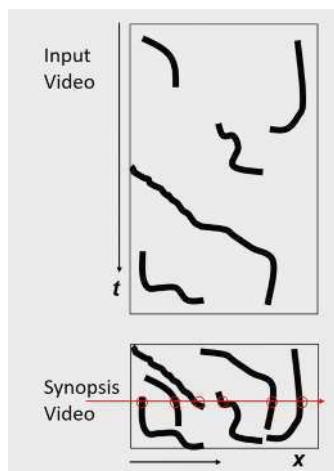
את מחיר ההתנשויות נגדיר באופן הבא:

$$E_c(\hat{b}, \hat{b}') = \sum_{\substack{x,y \\ t \in \hat{t}_b \cap \hat{t}_{b'}}} \chi_{\hat{b}}(x, y, t) \chi_{\hat{b}'}(x, y, t)$$

כאשר $\hat{t}_b \cap \hat{t}_{b'}$ הוא זמן ההתנשויות הסרטון החדש בין b ל- b' , כלומר זמן ההתנשויות בין \hat{b} ל- \hat{b}' .

על כן, אנו מקבלים מעין טריד אוף - בין אורך הסרטון לכמות ההתנשויות.

כל שהסרטון יותר קצר, כך מקבל יותר התנשויות, אך ככל שהסרטון ארוך יותר, מקבל פחות התנשויות. בנוסח, אם ניתן חשיבות רבה יותר למשקל של E_c מקבל פחות התנשויות ולבן וידאו יותר ארוך, אך אם ניתן לה שטיבות פחות מכובעת מקבל הרבה יותר התנשויות.



איור 344: אנו מנסים "לאرؤ" את הצינורות באמצעות מציאות M אופימלי. אנו דוחשים את כל האובייקטים הסרטון הקלט הסרטון קצר יותר. שימוש לב שבקו האדום יש כמה אובייקטים שמופיעים באותו זמן, על אף שבמציאות הם לא קרו באותו זמן.

Stroboscopic Panoramic Effect

שאלה נניח כי צינור הוא באורך שעה שלמה, כיצד נוכל לסכם את הסרטון המלא הסרטון קצר משעה? הרי הנחנו שאנו לא חותכים תנוצה.

אנו נחתוך את הצינור לחלקים ונציג אותם במקביל, למשל, נראה אצן רץ בתחילת המסלול, באמצע ובסוף במקביל. לפקט זה אנו קוראים Stroboscopic Panoramic Effect.

Stitching Synopsis

לאחר שמצאנו את כל האובייקטים, את הצינורות ואת אופן הציגה הנוכחי, علينا לאחד את הכל לכדי סרטון אחד. אממה, יתכן כי מושינוי הזמן, התאורה של האובייקטים אינה אחידה. כיצד נפתרו זאת? נתפרק את אותם על רקע.

אייר הבא בשביל ה-Poisson editing, אך שיטות כמו Pyramid blending无缝拼接, בהחלט יהיו טובות.



אייר 345: ניתן לראות משמאלי את הפרשי התאורה בעוד בצד ימין את התפירה החלקה. התפירה כMOVIN לא מושלמת, אך מספיק טוביה למטרתנו - קל לצפייה.

Clustered Synopsis Of Surveillance Video 42.3

החומר הבא שנציג לוקח את מה שראינו עד כה צעד אחד קדימה.²⁰

הרעין הוא לחלק את הוידאו ל-clusters (מקבצים) ולהציג באותו זמן תנועות דומות בתחום המקבצים. כך נוח יותר לצפות בסיכוןים ותנועות לא רגילות קלות לאיזהו.

כדי לאחד תנועות לכדי מקבץ נרצה להגדיר אישיה פונקציית מרחק מתרנעה i לתנועה j נגדיר

$$D_{ij} = \alpha S_{d_{ij}} + (1 - \alpha) M_{d_{ij}}$$

כאשר $S_{d_{ij}}$ הוא המרחק הנראה ו- $M_{d_{ij}}$ הוא למרחק התנועה - זה בעצם משקל של שני הפרמטרים האלה אותם נציג מיד בהרחבנה.

המרחק הנראה

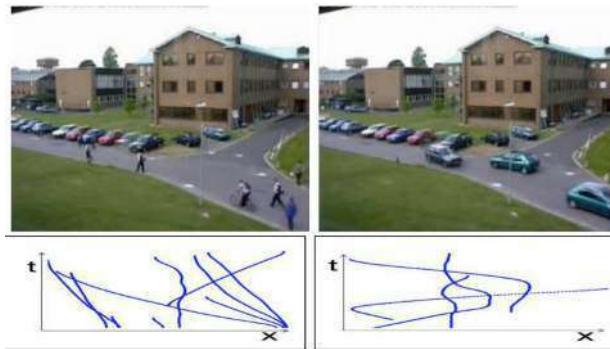
כדי להגדיר את $S_{d_{ij}}$ נפעל באופן הבא: לכל אובייקט N נגדיר דסקריפטורים באמצעות אלגוריתם SIFT, מפרייםים שונים.

²⁰ Yael Pritch, Sarit Ratovich, Avishai Hendel and Shmuel Peleg (AVSS 2009)

את המרחק הנראה בין שני אובייקטים נגידיר להיות המרחק מהשכן הקרוב ביותר, לכל אורך הצינוור. כלומר,

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

כאשר S_k^i הוא הדסקריפטור ה- k בציינור ה- i - j והוא הדסקריפטור הקרוב ביותר ל- S_k^i בציינור ה- j . ככה נזהה האם אובייקטים דומים או לא.



איור 346: באמצעות "המרחק הנראה" נוכל לקבל synopsis שיש בהן רק אנשים או רק מכוניות. ניתן לראות בתחרתית שהתנועות (ה-trajectories) מאוד שונות, כי לא הגבלנו את התנועה.

מרחק התנועה

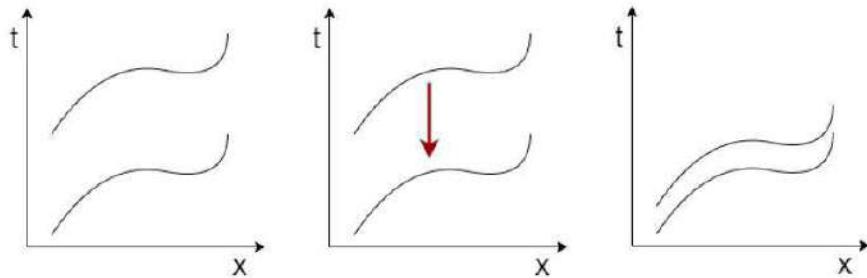
מלבד אובייקטים דומים, נרצה להתאים בין תנועות האובייקטים לפי ה-MotionDistance. אם התזוזה של אובייקט אחד דומה לתזוזה של אובייקט אחר.

ניצג את התנועה של אובייקט i לפי ה-trajectory t שלו:

$$\{(x_t^i, y_t^i, r_t^i)\}_{t \in t_i}$$

כאשר r_t^i הוא רדיוס האובייקט בפריים t ו- (x_t^i, y_t^i) הם קוואדרינטות ה- (x, y) של מרכז האובייקט בפריים t .

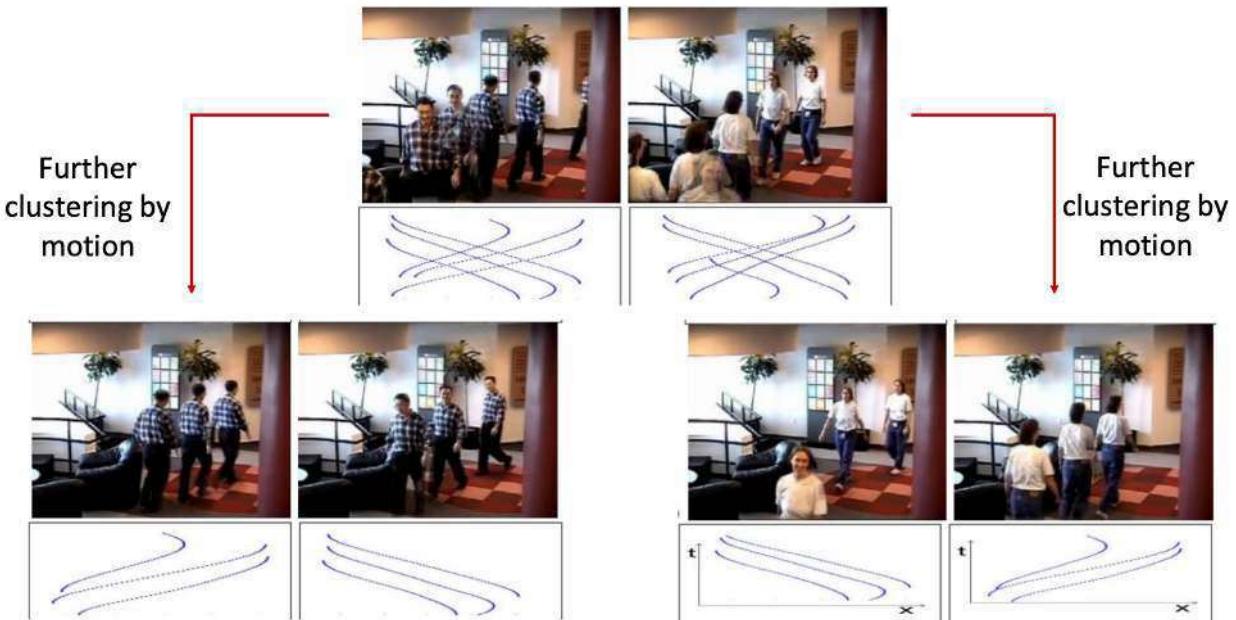
בהתאם תנועות j, i נגידיר את המרחק התנועתי שלהם להיות קטן, אם קיימת זהה k (הזה בזמן!) כך שם נזיז את j ב- k פריים נקבל חפיפה גדולה, ובו זמינות הפרדה מוחכית נמוכה. דהיינו התנועה של האובייקטים דומה בפריים שונים:



איור 347: אובייקטים שונים בעלי תנועה דומה מאוד בפרייםים שונים. בתמונה הימנית ביותר ניתן לראות את תנועת האובייקטים לאחר ההזאה, כאילו שהם נעים במקביל באותו פריים. דבר זה נכון לעין האנושית.

ככה אנו מקבלים שתי דרכים להציג תנועה - לפי המרחק הנראה ולפי המרחק התנועתי:

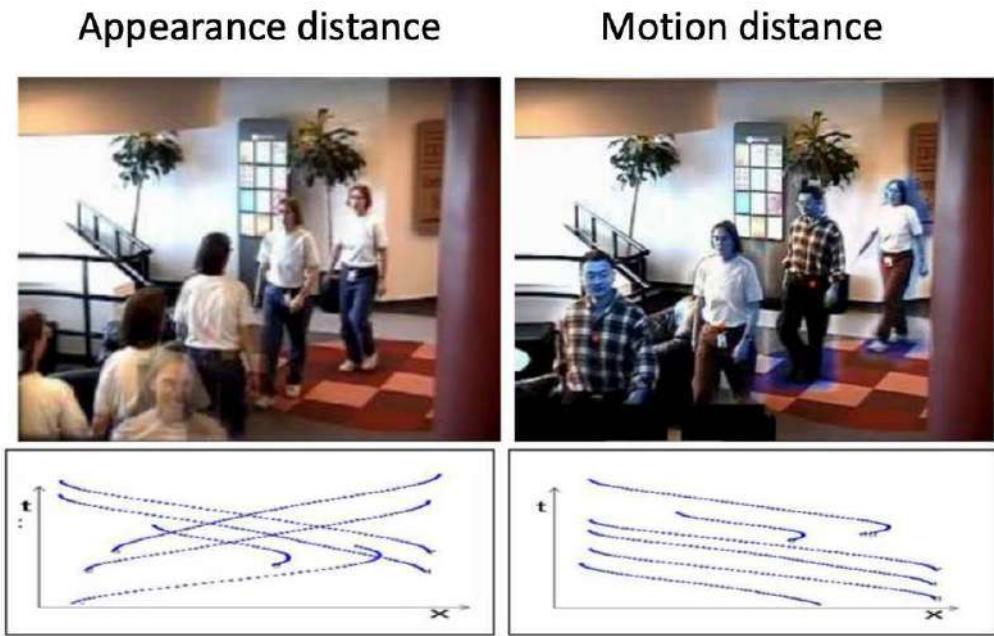
Clustering by appearance



איור 348: בשתי התמונות אנו נתונים משקל לדמיון התנועתי ומקבלים תנועה מקבילה של האובייקטים בעוד תנועה המקורית התנועה שונה לחלוין (בכוונים שונים).

הערה יכול להיות שבין שתי תנועות לא תהיה התאמה, אין מיפוי שייתן לנו את ה-*overlap* שאנו מחפשים. במצב זה, נקבל גובה ולא נקבע את שני האובייקטים הללו יחד. MotionDistance

באיור הבא ניתן לראות השוואה בין שתי השיטות:



איור 349: בתמונה הימנית ניתן לראות את התנועות בצורה כזו שמתאפשרת תנועה כמעט זהה של כל האובייקטים. בתמונה השמאלית לעומת זאת מתאפשרת תנועה בעלת תנועות שונות, כאשר אובייקטים דומים לפי מרחק נראה ככלمر לפ_{ij} הם באותו הקלאסטר.

לבסוף, לאחר שמצאנו גם קלאסטרינג וגם דרך לטפל בהתנגשויות, נרצה לאפשר הכל בפונקציה מחיר אחת. עלינו להגדיר פונקציית מחיר שתארוז את כל מה שמצאנו (Packing Cost) ותגדר במדויק כיצד להציג אובייקטים בתחום מקבץ וכיידן להציג את המקבץ עצמו - עם התיקחות להתנגשויות:

$$Pk_{ij}(k) = \beta \cdot M \cdot d_{ij}(k) + (1 - \beta) Colisions_{ij}(k)$$

142

$$Pk_{ij} = \min_k Pk_{ij}(k)$$

כך גם הקלאסטרים וגם האובייקטים בתוכם נכנסים לידיו לפי הפונקציה. למעשה כאן מסתמכים על המרחק התנועתי זהה וגם מה שיטור נפוץ (פחות מעוניין אותנו "מרחב הראייה"). גם כאן, אנו ממשקלים לפי שני פרמטרים - התנגשויות ומרחבי תנועה ולפי הפרמטר β נוכל לשלוט בחשיבות של כל אחד. למשל β גדול יכול להוביל לסדרות נקודות קצרים יותר עם יותר התנגשויות. "תמצאו לי דרך שהיו התנגשויות מינימליות, דמיון מקסימלי בתנועה ו-activity מקסימלי".



אייר 350: באמצעות השיטה שהצגנו אפשר לראות באותו אזור זמן רק מכוניות שימושיות ימינה, רק מכוניות שעולות בהר, רק מכוניות שייצאות, רק מכוניות שנכנסות וכו'. זאת כי לכל נסעה בכל כיוון יש trajectory שונה, והקיבוץ לפי התנועה ייב את התוצאה האו.