INTRODUCTION TO DIGITAL IMAGE PROCESSING

361.1.4751

EXERCISE 4 - Edges and Hough Transform

Submission Date: 01.01.2023

Noam Atias – 311394357

Chanel Michaeli – 208491787

תוכן עניינים

1. Edge Detection

1.1 Reading the Image

נקרא את התמונה *'cameraman.fit'*, ננרמל בין [0,1] ונציג:

Cameramen image normalized

1.2 Prewitt Edge Detector

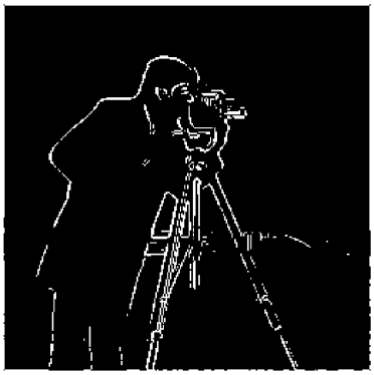
1.2.1 build the function *'dip_prewitt_edge(img,thresh)'*

בנינו פונקציה שנקראת *'dip_prewitt_edge(img,thresh)'* שמטרתה לממש זיהוי קצוות של התמונה על ידי הפונקציה *'prewitt edge detection()'* . הפונקציה מחזירה תמונת קצוות בגודל הזהה לזה של התמונה המקורית. זיהוי הקצוות נעשה על ידי הקרנלים הבאים :

$$G_{P_x} = \frac{1}{6}\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad , \quad G_{P_y} = \frac{1}{6}\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$$

הפונקציה מבצעת קונבולוציה דו ממדית של התמונה עם כל אחד מהפילטרים כדי לקבל את gradient magnitude ומשתמשת בערך סף (*thresh*) כדי לקבוע אילו פיקסלים בתמונה קיבלו ערך גבוה מערך הסף. פיקסלים אלו מקבלים את הערך 1 בתמונת הקצוות והשאר 0. נציג את תמונת הקצוות שנוצרת על ידי הפונקציה כאשר ערך הסף הוא 0.22 מול תמונת הקצוות שנוצרת על ידי הפונקציה המובנת של MATLAB:

Prewitt edge detection for cameramen image



Prewitt edge detection for cameramen image using matlab function

נשים לב כי עבור ערך הסף שבחרנו יש דמיון רב בין התמונות. כדי למדוד באופן כמותי את רמת הדמיון נשתמש בפונקציה *'ssim'* שבוחנת דמיון בין תמונות בהתחשב בפיקסלים השכנים. דמיון מקסימלי הוא כאשר הערך המתקבל הוא 1. הערך המתקבל אצלנו הוא 0.8636 , כלומר התמונות די דומות.

1.2.2 Display 2 edge images

נציג 2 תמונות קצוות שנוצרו על ידי הפונקציה שבנינו לזיהוי קצוות, כאשר נשתמש בערכי סף שונים בכל פעם:

Prewitt edge detection treshold = 0.05

Prewitt edge detection treshold = 0.3

נשים לב כי קיבלנו תמונות קצוות שונות לחלוטין. בתמונת הקצוות בה ערך הסף קטן יותר, קיבלנו המון קצוות של התמונה המקורית, מרבית מהקצוות מתארות את הרקע שמאחורי האיש עם המצלמה, ואילו בתמונה בעלת ערך הסף הגדול יותר קיבלנו קצוות מועטים מאוד, אין קצוות המתארים את הרקע אלא רק מעט קצוות המתארים את האיש והמצלמה.

לכן נוכל להסיק כי ככל שערך הסף גדול יותר, נקבל פחות פיקסלים עם ערך הגדול מערך הסף ולכן פחות פיקסלים יקבלו את הערך של הצבע הלבן בתמונת הקצוות.

בתמונה בה ערך הסף נמוך, נקבל קצוות שנראים כמו רעש, חלקם לא רלוונטיים לתיאור מתאר התמונה המקורית, במיוחד אלו שבתחתית התמונה. בנוסף קיבלנו שהקצוות המתארים את הפנים של האיש הם יותר מדי גסים. מצד שני, בתמונה בה ערך הסף גדול מדי, קיבלנו כי אין כלל תיאור של הרקע, וגם הקצוות המתארים את האיש והמצלמה הם מעטים מדי בשביל להבחין כי מדובר באיש עם מצלמה.

כלומר, קיבלנו כי יש לבחור ערך סף כך שתמונת הקצוות לא תכלול יותר מדי רעש מיותר אך עדיין תחזיק מידע מספיק טוב לגבי האובייקטים הרלוונטיים בתמונה.

1.3 Canny Edge Detector

1.3.1 MATLAB function *'edge(I,'Canny')'*

הפונקציה *'edge(I,'Canny')'* מקבלת תמונה ואת המתודה שלפיה היא מחשבת את הקצוות של התמונה הנתונה. אנחנו נשתמש במתודה *'Canny'*, מתודה זו מוצאת את קצוות התמונה על ידי חישוב של הגדיאנט של התמונה ומציאת המקסימום הלוקאלי שלה. חישוב הגרדיאנט בכל נקודה בתמונה מתבצע על

ידי הפעלת נגזרת של פילטר גאוסיאני. הפונקציה משתמשת בשני ערכי סף הקובעים אילו פיקסלים יהיו חלק מהקצוות של התמונה.

הפרמטרים האופציונליים שהפונקציה *'edge(I,'Canny')'* מקבלת הם:

Threshold – וקטור של שני ערכים, ערך גבוה וערך נמוך $[low, high]$ בין 0 ל – 1, הקובעים את ערכי הסף. הפונקציה מתעלמת מכל הפיקסלים שערכם קטן מערך הסף התחתון, ושומרת את כל הפיקסלים שערכם גדול מערך הסף העליון. פיקסלים בעלי ערך הנמצא בין ערכי הסף, המחוברים לקווי המתאר שנוצרו על ידי הפיקסלים שערכם גדול מערך הסף העליון, גם הם יהיו חלק מהקצוות של התמונה.

אם מכניסים לפונקציה ערך סף בודד, הפונקציה תגדיר ערך סף תחתון כ – default על ידי הכפלה של ערך הסף הקיים ב – 0.4.

אם לא הוגדרו בפונקציה ערכי סף כלל, הפונקציה תבחר כ – default ערכי סף כתלות בתמונה.

Sigma – סטיית התקן של הפילטר הגאוסיאני. אם לא מגדירים, הפונקציה מגדירה ערך דיפולטיבי השווה לשורש של 2. בהתאם לערך של sigma, הפונקציה בוחרת את גודל הפילטר.

1.3.2 Using the MATLAB functions *'edge(I,'Canny')'* with two sets of parameters

נשתמש בפונקציה *'edge(I,'Canny')'* עבור שני סטים של פרמטרים, סט של פרמטרים דיפולטיביים וסט של ערכים שקבענו מראש. נציג את התוצאות:

Canny edge detection using default parameters

Canny edge detection using parameters: [low,high]=[0.1,0.6], sigma=1.4142

עבור תמונת הקצוות שהתקבלה עבור הפרמטרים הדיפולטיביים, קיבלנו כי ערכי הסף הדיפולטיביים שהתקבלו הם $[low, high] = [0.03125\ , 0.078125]$ והערך של σ הוא הערך הדיפולטיבי $\sqrt{2}$.

בתמונה זו נשים לב כי הקצוות המתארים את האיש ברורים אך הרקע שמאחורי האיש מאוד בולט, יש הרבה קצוות בתחתית התמונה שלא ברור מה הם מתארים. בקו המתאר שמבדיל בין השמיים לארץ ניתן לראות בניין ועוד קצוות של מבנים מסוימים.

לעומת זאת, בתמונת הקצוות השנייה בחרנו ערכי סף $[low, high] = [0.1, 0.6]$ ואת הערך של σ השארנו להיות $\sqrt{2}$. בתמנה זו ניתן לראות כי הקצוות של האיש עם המצלמה נותרו בצורה טובה ואילו הקצוות של הרקע בחלק התחתון של התמונה נעלמו ונשאר רק קו המתאר שמבדיל בין השמיים לארץ. כלומר, קיבלנו תיאור של האובייקטים המרכזיים בתמונה שהם האיש והמצלמה בצורה טובה ללא קוויי המתאר של הרקע מלבד זה שמפריד בין השמיים לארץ שהוא החשוב ביותר מתוך כל הרקע, לעומת התמונה שנוצרה על ידי ערכים דיפולטיבים שבה מתוארים כל מיני פרטים הרקע שהם פחות משמעותיים בתמונה.

1.3.3 Change the parameters

לאחר שהבנו את ההשפעה של בחירת ערכי הסף על התמונה, נסיק כי אם נשנה את הערך של σ כך שהפילטר יבצע החלקה על התמונה בצורה טובה יותר ויפחית רעשים, ואז נשתמש בערכי הסף הדיפולטיביים של הפונקציה, נקבל תמונת קצוות ברורה יותר עם פחות רעשים. נבחר סיגמה גדולה יותר השווה ל-3 ונציג את התוצאות :

Canny edge detection using parameters: sigma=3, default thresholds

גם כאן קיבלנו כי ערכי הסף הדיפולטיביים הם [low, high]=[0.03125,0.078125]. נשים לב כי כעת התמונה המתקבלת היא הרבה פחות מורעשת, הרקע בחלק התחתון של התמונה חלק יותר ללא כל הפרטים הקטנים שהיו לפני כן, האיש עם המצלמה עדיין נותר ברור וכעת קו המתאר שמבדיל בין השמיים לארץ הוא הרבה יותר ברור ומכיל פרטים נוספים כמו בניין. כלומר, מקסמנו את איכות התמונה על ידי הבנת הפרמטרים ושינויים רלוונטי בלבד בערכים הביא לתוצאות טובות יותר. בתמונת הקצוות שהתקבלה על ידי הפרמטרים הדיפולטיביים קיבלנו רעש רב שנוצר משינויים קטנים ולכן כדי להוריד את רמת הרגישות של זיהוי הקצוות שינינו את הערך של σ בלבד וכך הצלחנו לנקות את הרעשים ולקבל תמונה טובה יותר.

2. Hough Transform

2.1 Hough line transform

(a) Read the floor.jpg image

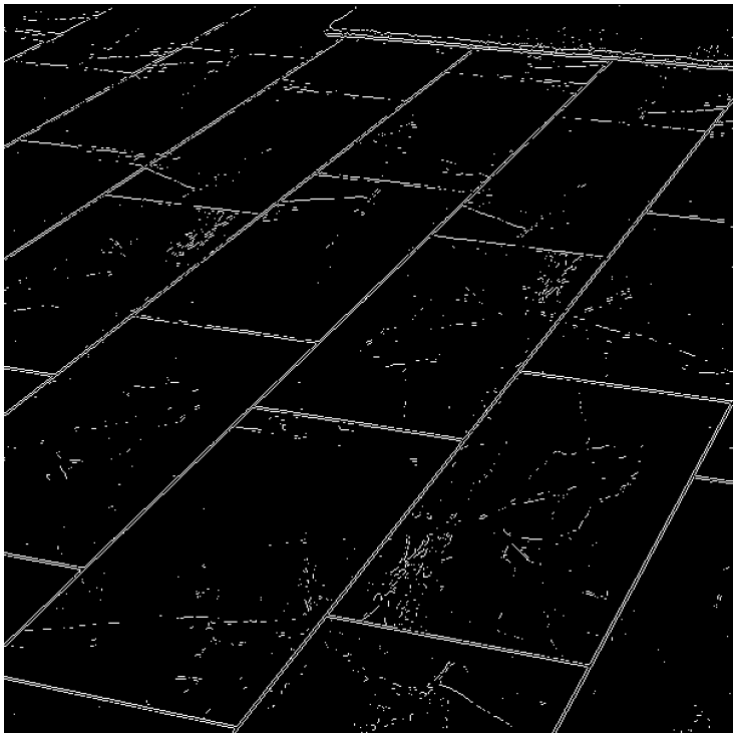
נקרא את התמונה *'floor.jpg'*, נמיר אותה ל – Grayscale וננרמל אותה בין 0 ל – 1:

Floor image grayscale normalized

(b) Extract the edges using MATLAB's BW = *edge(I)*

נשתמש בפונקציה של MATLAB *'BW=edge(I)* כדי לזהות את קצוות התמונה:

Default edge detection for floor image



קיבלנו כי הקצוות של התמונה הם הקווים המפרידים בין המרצפות ועוד רעשים שמתארים את קווי המתאר של טקסטורת המרצפות.

השימוש בפונקציה *'edge'* נעשה באופן דיפולטיבי. בשימוש כזה, המתודה המהווה ברירת מחדל של הפונקציה שבאמצעותה נעשה זיהוי הקצוות הוא *'Sobel edge detection'*. בשיטה זו זיהוי הקצוות נעשה על ידי מציאת נקודות המקסימום של הגרדיאנט.

הפרמטרים הדיפולטיביים של שיטה זו הם:

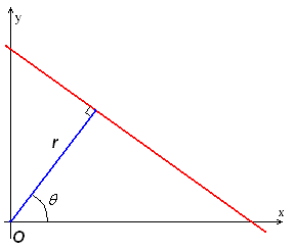
Threshold – ערך הסף נבחר כברירת מחדל. פיקסלים שערכם גבוה מערך הסף יקבלו את הערך 1 ויחשבו כקצה בתמונת הקצוות. פיקסלים שערכם קטן מערך הסף יקבלו את הערך 0 ולא יחשבו כקצה. כך מתקבלת מטריצה של אפסים ואחדות שמהווה את תמונת הקצוות. ערך הסף הדיפולטיבי נקבע על ידי פרמטרי התמונה.

Direction – פרמטר זה מאפשר לקבוע את כיוון הקצוות בתמונה שהפונקציה תזהה. אם לא קובעים מראש את הכיוון, הוא נקבע באופן דיפולטיבי וערכו יהיה כיוון אנכי וכיוון אופקי ('both').

(c) *'dip_hough_lines(BW,R_0,Θ_0)'* function

נבנה את הפונקציה $"dip\_hough\_lines(BW, R_0, \Theta_0)$ שמטרתה לחשב את מטריצת Hough למציאת קווים. הפונקציה מקבלת תמונת קצוות בגודל $N \times M$ ומייצרת מטריצה של אפסים בגודל $|R| \times |\Theta|$ כאשר $|R|$ הוא הגודל של הווקטור.

$R_0: \sqrt{M^2+N^2}: R = -\sqrt{M^2+N^2}$ המגדיר את אורך המשיק לקו, ו- $|\Theta|$ הוא הגודל של הווקטור $\Theta = -90: \Theta_0: 90$ שמהווה את הזווית בין המשיק לבין החלק החיובי של ציר x.



הפונקציה מוצאת את כל הפיקסלים $(x, y) \in BW$ בתמונת הקצוות שערכם צבע לבן, ולכל פיקסל כזה ולכל $\theta \in \Theta$ מחשבת את $r = x \cdot cos(\theta) + y \cdot \sin(\theta)$. הפונקציה מוסיפה 1 למטריצת Hough במקום ה-(r, θ) כאשר $r \epsilon R$ (כלומר נעגל את r שמתקבל).

כלומר קיבלנו המרה ממרחב (x, y) למרחב (r, θ) שמיוצג על ידי מטריצת Hough.

כל פונקציה $r(\theta)$ מייצגת סינוסאידה ונקודת החיתוך בין מספר סינוסאידות במרחב זה מייצגת את הקו שעובר דרך מספר נקודות במרחב התמונה. ככל שמספר הסינוסאידות העוברות דרך נקודת החיתוך הוא יותר גדול, כך הקו יותר דומיננטי במרחב התמונה.

נשתמש בפונקציה פעם אחת עם הערכים $(R_0, \Theta_0) = (1,1)$ ופעם נוספת עם הערכים $(R_0, \Theta_0) = (5,4)$.

(d) Display the Hough matrix

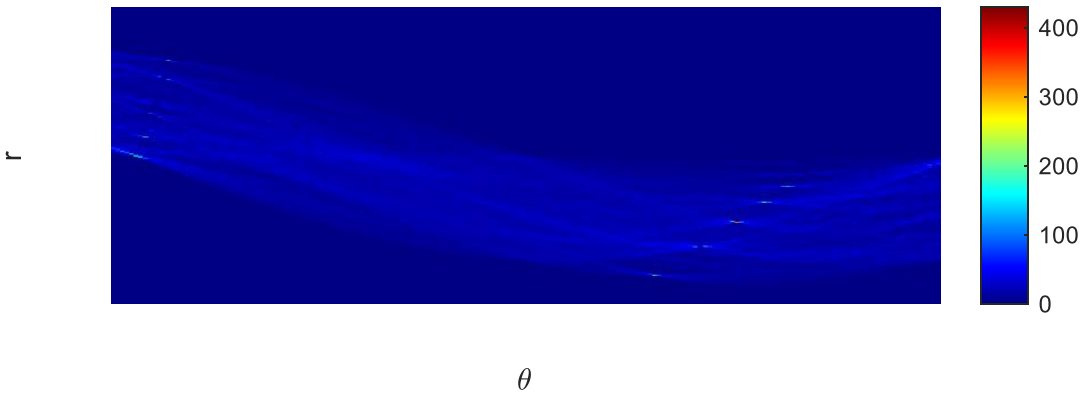
לכל אחד מהתוצאות של סעיף קודם, נציג את התמונה שהתקבלה:

Hough lines detection R_0=1, θ_0 =1

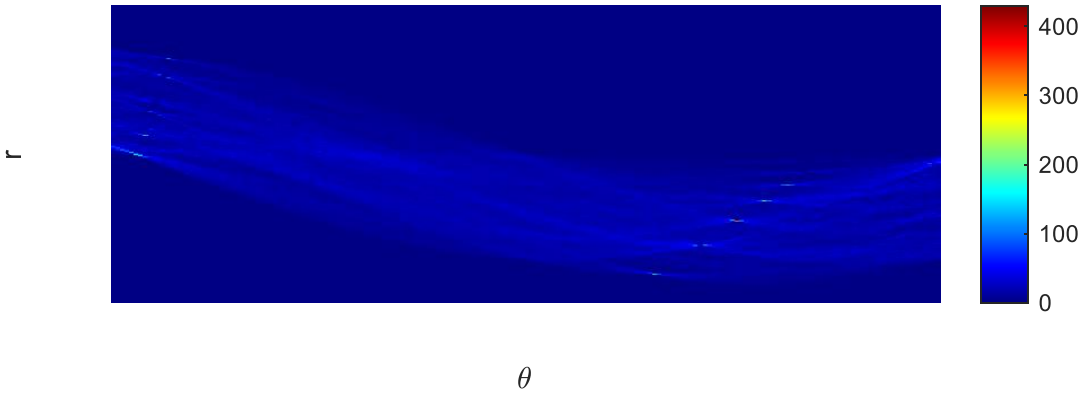
Hough lines detection R_0=5, θ_0 =4

נציג בצורה טובה יותר את התוצאות כך שיהיה ניתן להבחין בסינוסאידות:

Hough lines detection R_0=1, θ_0 =1



Hough lines detection R_0=5, θ_0 =4

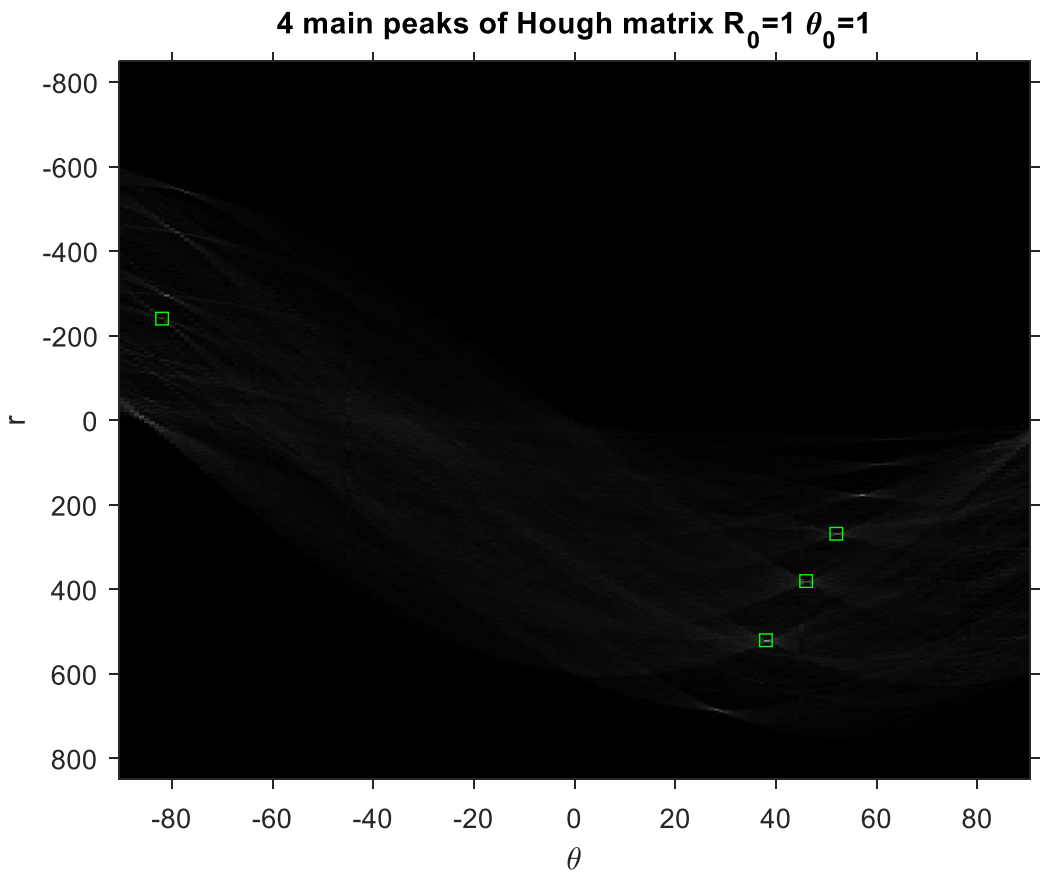


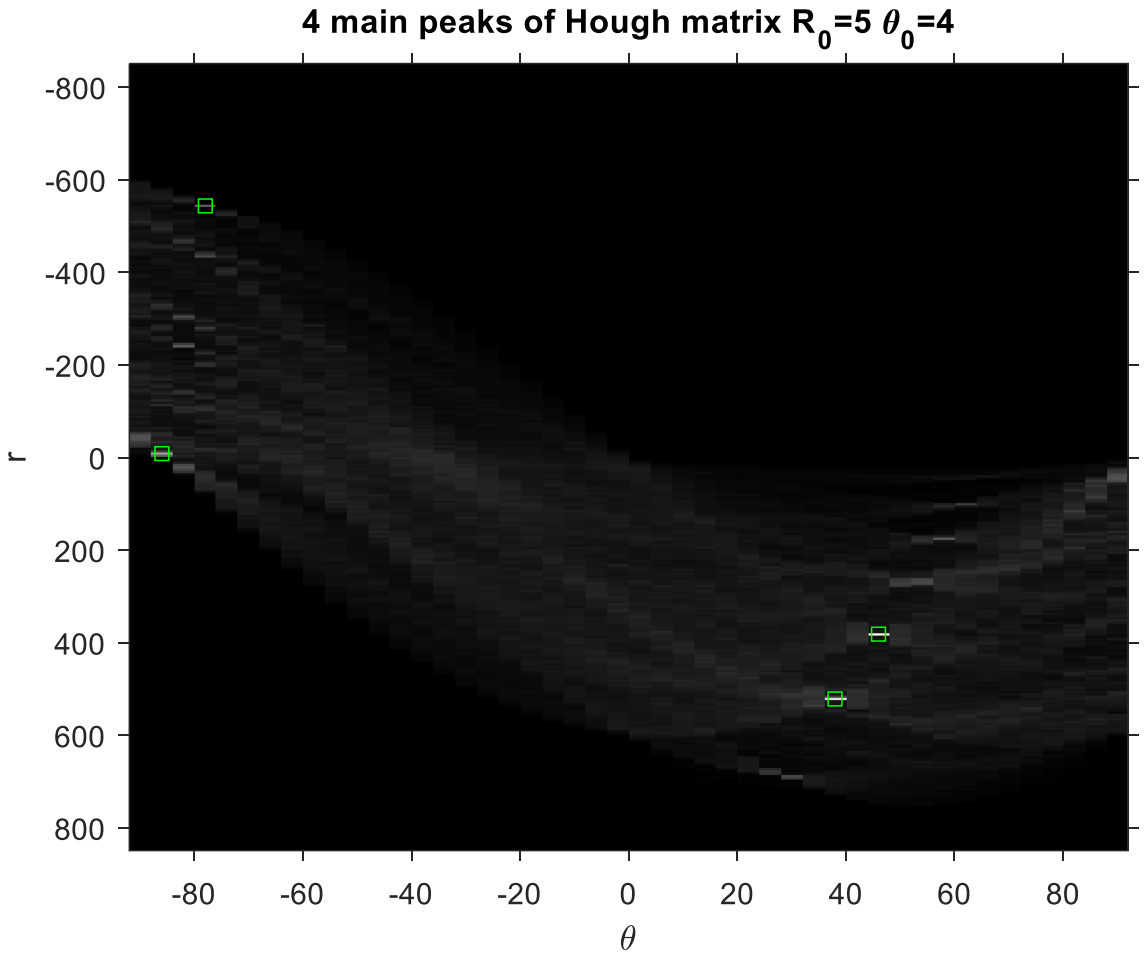
בהצגה זו נוכל לראות בבירור יותר את הסינוסואידות בשני הגרפים ואת נקודות החיתוך שלהם כאשר עוצמת נקודת החיתוך גדולה יותר ככל שיותר סינוסואידות עוברות בנקודת חיתוך זו, נשים לב כי נקודת החיתוך בעלת העוצמה הגדולה ביותר היא זאת שמסומנת בצבע אדום. כל סינוסואידה מבטאת פיקסל בצבע לבן בתמונת הקצוות.

התמונה שהתקבלה מערכים $(R_0, \Theta_0) = (5,4)$ מתקבלת ברזולוציה נמוכה יותר מאחר שהווקטורים המרכיבים את מרחב R, Θ זה הם בעלי אורך קצר יותר. כתוצאה מכך נקבל כי העוצמה של נקודות החיתוך גדולה יותר כי יותר סינוסואידות עברו דרכם למרות שבפועל הן לא באמת נחתכות.

(e) 4 most significant lines in the BW image

נרצה לחפש את ארבעת הקווים המשמעותיים ביותר בתמונת הקצוות. כדי לעשות זאת נשתמש במטריצת hough ונעזר בפונקציה של MATLAB שנקראת *'houghpeaks'* שמקבלת את מטריצת hough שמצאנו, מספר הקווים המשמעותיים שאנו רוצים למצוא וערך סף 'threshold' שמגדיר את הערך המינימלי שיש להתחשב בפיק. הפיקים במטריצת hough הם אלה שמחזיקים את המידע לגבי הקווים המשמעותיים ביותר של התמונה. נמצא את ארבעת הפיקים העוצמתיים ביותר במטריצה עבור כל אחד מהקונסטלציות ונציג אותם על גבי תמונת המטריצה:





באמצעות הפיקים שמצאנו, נוכל למצוא את הקווים המשמעותיים ביותר בתמונה על ידי הקשר בין התמונה למטריצת hough:

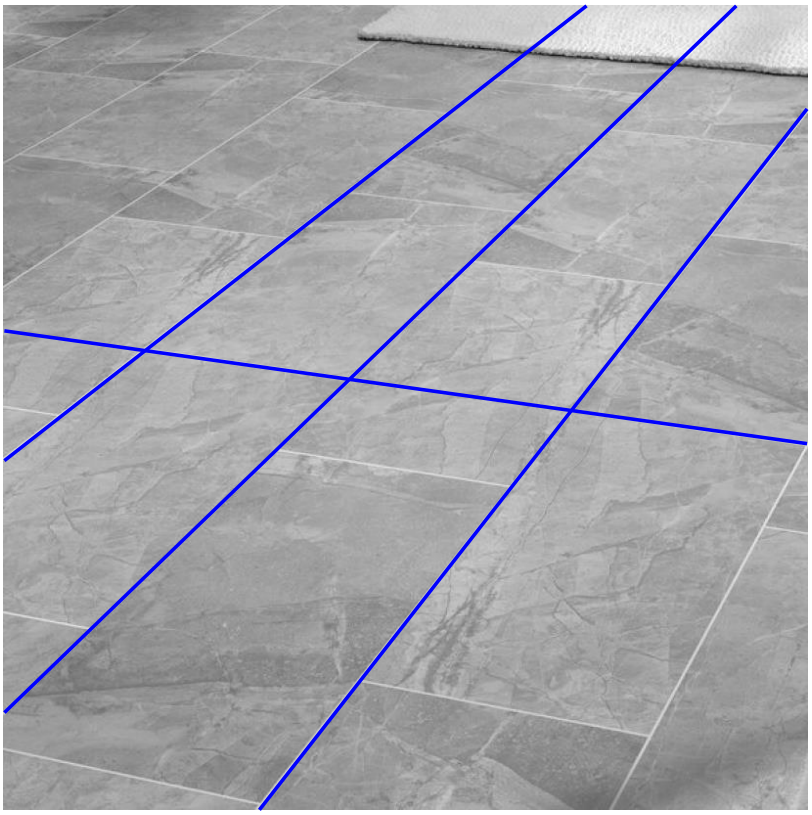
$$y = -x \cdot \cot(\theta) + \frac{r}{\sin(\theta)}$$

<u>כאשר:</u>

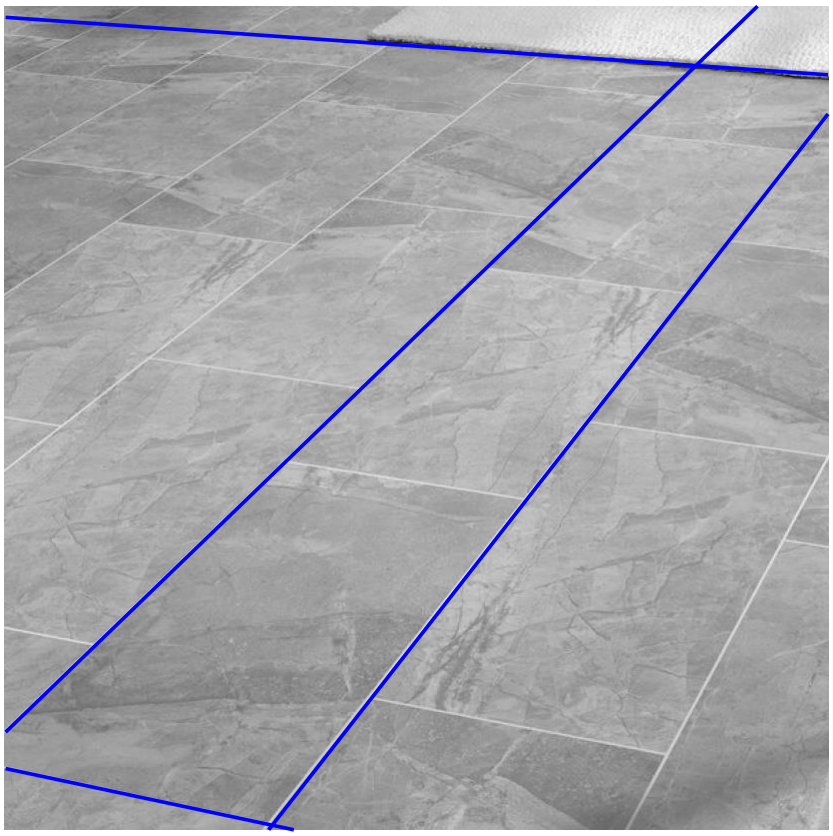
$$\theta \epsilon -90 : \Theta_0 : 90 \ \ , r \in [-\sqrt{M^2+N^2} : R_0 : \sqrt{M^2+N^2}]$$

נציג את ארבעת הקווים המשמעותיים ביותר בתמונה עבור שני הקונסטלציות:

4 most significant lines R_0=1 θ_0=1



4 most significant lines R_0=5 θ_0=4



(f) Explain the results

נסביר את התוצאות:

נראה כי עבור בחירה שונה של (R_0, Θ_0) קיבלנו קווים משמעותיים שונים, כלומר פיקים שונים במטריצת hough היו יותר דומיננטיים.

ברזולוציה גבוהה, כלומר כאשר בחרנו ערכים קטנים $(R_0, \Theta_0) = (1,1)$, קיבלנו 3 פיקים שנמצאים בחלק הימני של המטריצה בנקודות חיתוך תחתונות, פיקים אלו מתארים את שלושת הקווים האנכיים שנמצאו, ואילו פיק נוסף בחלק השמאלי של המטריצה בנקודת חיתוך עליונה שמתאימה לקו האופקי שנמצא.

ברזולוציה זו, קיבלנו דיוק גדול יותר בזיהוי הקווים המשמעותיים, אך עדיין הקו האופקי שזוהה הוא לא באמת קו ישר אחיד, מאחר שיש מרצפות שמפרידות בין הקווים. אך בכל זאת הקו הזה זוהה כקו ישר אחיד וזאת מאחר שברזולוציה גבוהה ישנה רגישות לרעש, כתוצאה מכך שהפיקים שזוהו במטריצה הן הרבה פחות בהירות, כלומר ההפרש בי הפיקים לרקע הוא הרבה יותר קטן ולכן נוצר זיהוי שקרי של פיקים. כמובן כי במקרה זה מטריצת *hough* היא הרבה יותר גדולה ורמת סיבוכיות החישוב היא גם כן גדולה יותר.

לעומת זאת ברזולוציה נמוכה, כאשר בחרנו ערכים גדולים $(R_0, \Theta_0) = (5,4)$, קיבלנו 2 פיקים בצד ימין בנקודות חיתוך תחתונות שמתאימות ל2 הקווים האנכיים ו-2 פיקים בצד שמאל בנקודות החיתוך העליונות שמתאימות לקווים האופקיים שנמצאו בתמונה. נשים לב כי הקו העליון האופקי שמחובר לשטיח הוא לא באמת קו ישר, אלא קו שמתעגל לאורך השטיח, אך ברזולוציה זו קיבלנו פרשנות לקו ישר. בעצם, גם פה נוצר זיהוי פיקים שקרי כתוצאה מכך שחלק מהקווים התחברו לקו אחד למרות שהם לא מחוברים או לא ישרים בדיוק. ברזולוציה זו מטריצת *hough* הרבה יותר קטנה ולכן העומס החישובי קטן בהרבה.

לסיכום, קיבלנו טרייד אוף בין הרזולוציות על ידי בחירת הערכים בקונסטלציה. ככל שנבחר ערכים גבוהים יותר נקבל רזולוציה נמוכה יותר, ואז הדיוק בסימון הקווים הדומיננטיים יפחת אך רמת הסיבוכיות ברזולוציה זו קטנה יותר. לעומת זאת, עבור ערכים נמוכים נקבל רזולוציה גבוה יותר, כלומר נקבל דיוק גדול יותר אך רמת הסיבוכיות גדולה יותר וגם תופס יותר מקום בזיכרון כתוצאה ממטריצת *hough* גדולה יותר, דבר שמגדיל את זמן הריצה.

2.2 Hough circle transform

(a) Read the coffee.png image

נקרא את התמונה *'coffee.png'* , נמיר אותה ל – Grayscale וננרמל בין 0 ל – 1:

Coffee image grayscale normalized

(b) Extract the edges using MATLAB's BW = *edge(I)*

נמצא את הקצוות של התמונה על ידי *'BW=edge(I)* ונציג :

Default edge detection for coffee image

(c) *'dip_hough_circles (BW,R_0,Θ_0)'* function

נבנה פונקציה שנקראת $'dip\_hough\_circles(BW, R_0, \Theta_0)$ שמטרתה לחשב את מטריצת *hough* כדי למצוא מעגלים בתמונה. הפונקציה מקבלת תמונת קצוות בגודל $N \times M$ ומייצרת מטריצה של אפסים בגודל $|N| \times |M| \times |R|$ כאשר $|R|$ הוא הגודל של הווקטור $R = R_{min}{:}R_0{:}R_{max}$. הגדרנו $R_{min} = 80$, $R_{max} = 100$.

הפונקציה מוצאת את כל הפיקסלים $(x, y) \in BW$ בתמונת הקצוות שערכם צבע לבן.

לכל פיקסל כזה, לכל $r \epsilon R$ ולכל $\theta \in \Theta$ כאשר $\Theta = 0{:}\Theta_0{:}360$, הפונקציה מחשבת:

$a = x - r \cdot \cos(\theta)$, $b = y - r \cdot \sin(\theta)$, זוהי פרמטריזציה של מעגל.

הפונקציה מוסיפה 1 למטריצת Hough במרכז המעגל, כלומר במקום ה-(a, b, r) לאחר שעיגלנו את הערכים.

כעת, נקודת חיתוך של הרבה מעגלים במחרב הפרמטרים מהווה מעגל דומיננטי במרחב התמונה.

נשתמש בפונקציה פעם אחת עם הערכים $(R_0, \Theta_0) = (1,1)$ ופעם נוספת עם הערכים $(R_0, \Theta_0) = (4,10)$.

(d) Measure the Run-Time of our function

נציג את זמן הריצה של הפונקציה שבנינו עבור הערכים $(R_0, \Theta_0) = (1,1)$:

```
The run time of calculating hough matrix to find circles is : 29.706
```

אנו יודעים כי ככל שהרזולוציה יותר גבוהה, כך זמן הריצה יותר ארוך. לכן כדי למצוא ערכים שמספקים רמת דיוק דומה אך זמן הריצה קצר יותר, נרצה ערכים גדולים יותר אך לא גדולים מדי כדי שרמת הדיוק תשמר. לכן נבחר $(R_0, \Theta_0) = (1,2)$, ערכים אלה קרובים מאוד לערכים המקוריים אך עדיין נקבל זמן ריצה קצר יותר:

זמן הריצה התקצר בערך פי 2 ונקבל:

```
The new run time of calculating hough matrix to find circles is : 15.974
```

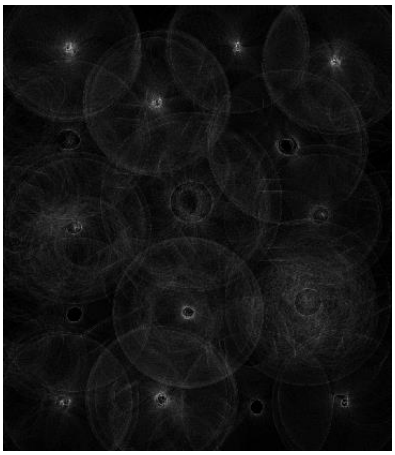
$$Speedup = \frac{T_{old}}{T_{new}} = 1.86$$

נשים לב כי גודל מטריצת ה- *hough* נשאר זהה בשני המקרים מאחר שהוא לא תלוי בגודל הווקטור של הזווית, ומבחינת רמת הדיוק, נחשב את ה-*MSE* ונקבל שהוא שווה ל-36.55, קיבלנו שגיאה לא גדולה מדי ולכן נוכל להגיד כי רמת הדיוק נשמרה ובהמשך נראה כי נקבל זיהוי זהה של המעגלים בשני המקרים.

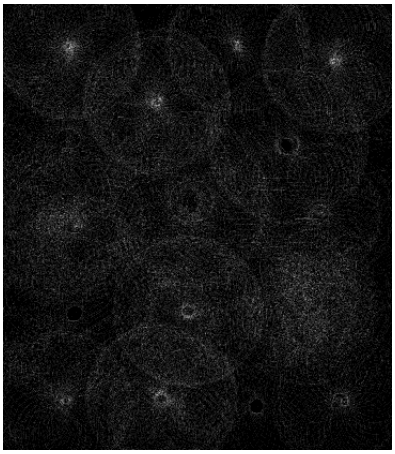
(e) Our Hough matrix is 3D, display one slice

נציג ערוץ בודד מתוך מטריצת *hough* עבור כל אחד מהקונסטלציות של הערכים:

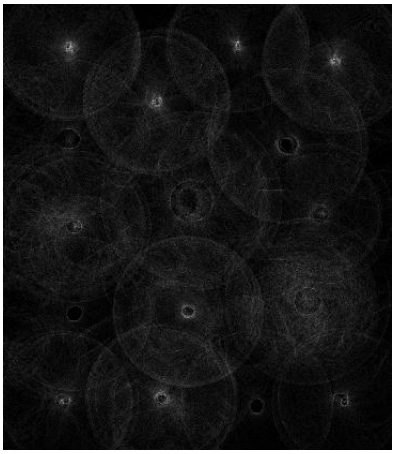
One Silce of Hough 3D with R_0=1,θ_0=1



One Silce of Hough 3D with R_0=4,θ_0=10



One Silce of Hough 3D with R_0=1,θ_0=2



נשים לב כי קיבלנו עבור הערכים $(R_0, \Theta_0) = (1,2), (1,1)$ דמיון גדול, בשני מקרים אלה, רמת הרזולוציה היא עדיין גבוהה ולכן קיבלנו מעגלים חלקים ורציפים, ניתן לראות את המעגלים נחתכים במרכז הפיקים. לעומת זאת, בתמונה האמצעית שבה הערכים גדולים יותר $(R_0, \Theta_0) = (4,10)$, קיבלנו מעגלים מחוספסים יותר, שנראים יותר מורעשים וזאת מאחר שהרזולוציה נמוכה יותר והרבה פחות פיקים מרוכזים בנקודה אחת.

(f) *'dip_houghpeaks3d(HoughMatrix)'* function

נכתוב את הפונקציה *'dip_houghpeaks3d(HoughMatrix)'* שמטרתה למצוא את חמשת המעגלים המשמעותיים ביותר בתמונה. הפונקציה מקבלת את מטריצת *hough* התלת-ממדית ומחזירה את האינדקסים של חמשת הפיקים הדומיננטיים ביותר שכל פיק מתאים למעגל שונה בתמונה. כדי למצוא חמישה מעגלים שונים ולמנוע מהפונקציה להשתמש בפיקים שכנים שמייצגים את אותו מעגל, נאפס במטריצה את הפיק שנמצא יחד עם שישה הפיקסלים השכנים שלו מכל כיוון, כך בכל איטרציה נמצא מעגל השייך לספל שונה.

לאחר שמצאנו את חמשת הפיקים הדומיננטיים ביותר, נוכל לבנות את המעגלים המשמעותיים ביותר על גבי התמונה של הספלים, נציג את התוצאות עבור כל אחד מהקונסטלציות:

5 most significant circles R_0=1 & $theta_0$=1

5 most significant circles R_0=1 & $theta_0$=2

5 most significant circles R_0=4 & $theta_0$=10

(g) Explain the results

נסביר את התוצאות:

קיבלנו עבור כל אחד מהערכים שהפיקים שנבחרו אכן מתארים מעגלים בתמונה של הספלים. נשים לב כי עבור הערכים הנמוכים $(R_0, \Theta_0) = (1,2), (1,1)$ שעבורם הרזולוציה גבוהה יותר, קיבלנו את אותו סט של מעגלים, דבר התואם את הטענה כי רמת הדיוק של ערכים אלו מספיק טובה כך שזיהוי המעגלים יהיה זהה.

לעומת זאת כאשר הערכים גבוהים יותר $(R_0, \Theta_0) = (4,10)$ קיבלנו זיהוי של מעגלים שונים, כלומר רמת הדיוק נפגע אך מצד שני זמן הריצה היה קצר משמעותית. במקרה זה, הפיקים במטריצת *hough* לא נחתכו באותם המקומות ולכן קיבלנו צורה מפוזרת יותר של פיקים שאיתה התמודדנו בפונקציה *'dip_houghpeaks3d(HoughMatrix)'*. בנוסף קיבלנו כי ערכי הפיקים במטריצת *hough* היו נמוכים יותר דבר שהקשה על הזיהוי של המעגלים.

מצד שני, ברזולוציה גבוה אנו מקבלים זמן ריצה ארוך וסיבוכיות חישובית גבוה לעומת רזולוציה נמוכה. כלומר, כפי שציינו, ישנו טרייד אוף בבחירת רמת הרזולוציה בין רמת סיבוכיות לבין רמת דיוק.

בהתבסס על מידע מקדים וניתוח נוכל לזהות את הרזולוציה המתאימה שתביא לביצועים טובים ומדויקים בזמן ריצה הקצר ביותר האפשרי.

אם נרצה למצוא מעגלים בתמונה שהם לאו דווקא הדומיננטיים ביותר, נוכל להשתמש ברזולוציה נמוכה יותר שתביא לביצועים מספיק טובים בזיהוי המעגלים.