

פרויקט בעיבוד תמונה – לוח דמeka אינטראקטיבי

1. חברי הפרויקט:

אוריה פרג – 209339589
שלוי הנדל – 209494533
נדב לוי – 316409531
איתמר יהודאי – 205916703

2. כוורתה:

הכוורתה של הפרויקט היא "לוח דמeka אינטראקטיבי".

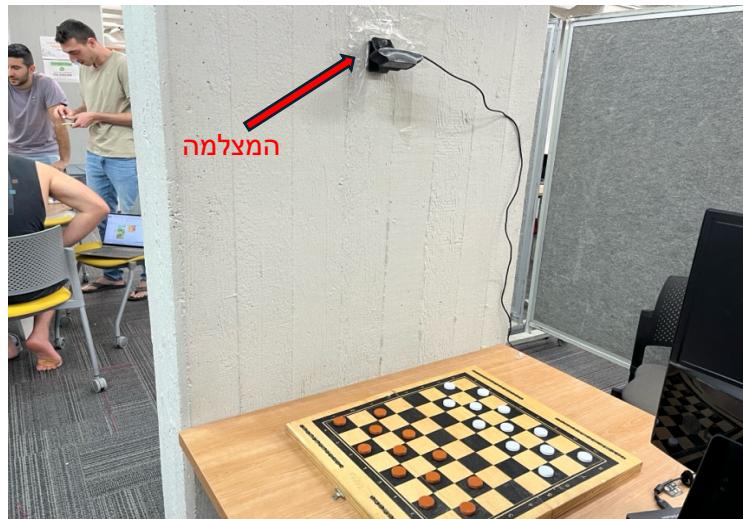
3. תקציר הפרויקט:

בפרויקט אנחנו מצלמים מלמעלה לוח דמeka ובאמצעות כלים בעיבוד תמונה מזוהים את הלוח וכלי המשחק ועוקבים אחר מהלכים של השחקנים.
 אנחנו מתעדים את מהלך המשחק ומציגים אותו על טלוויזיה בצורה אינטראקטיבית. לדוגמה,
 מודיעים על מהלכים לא חוקיים, מכריזים תור איזה שחקן בכל רגע נתון, מכריזים על ניצחון ועוד.

4. הקדמה:

המערכת שלנו מורכבת מצלמה אחת חיצונית, מחשב, מצלמה פנימית של המחשב ולוח דמקה עם כליו.

המצלמה נמצאת מעל הלוח ומצלמת את הלוח. ניתן לסובב את המצלמה ואף לעותות אותה מעט והתוכנה שלנו תדע לישר את התמונה (נראה זאת בהמשך).

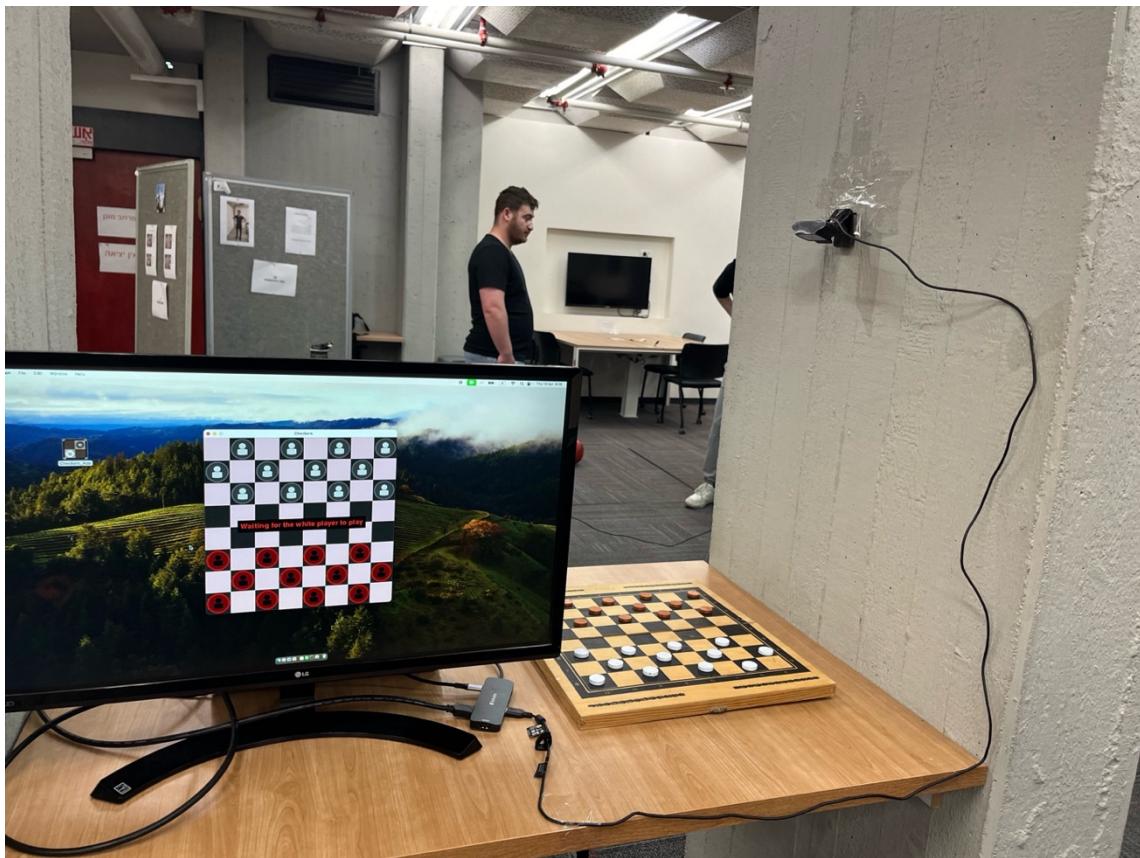


את המצלמה לחבר למחשב ודרכו ניתן להפעיל את האפליקציה Checkers App.



נצלם תמונות המשתתפים ונשים אותם על החיללים.





הסת אפ המלא

המטרה היא לזוהות את הלוח ואת מיקומי הכלים, לעקוב אחרי המהלךים של השחקנים, להציג על המסך, להתריע על מהלכים לא חוקיים ולהכריז על המנצח. נצפה שככל עוד אין הסטרות ללוח או הזזה של הלוח מחוץ לטווח אותו המכונה רואה, האפליקציה תצלילה לזהות את הלוח וכלייו וכן תוכל לבצע איזה מהליכים קרו. כמו כן, נרצה שאמם במהלך המשחק המכונה או הלוח ייזוזו, נדע להתמקד מחדש וזוהות את הלוח ולהמשיך את המשחק. כאשר תהיה הפרעה אשר מפריע ליזיהו, ניתן אינדיקציה על המסך על מנת שהשחקנים ידעו לטפל בה.

אתגרים שאנו צופים:

- התגברות על ידיהם של השחקנים שמסתיימות את לוח המשחק.
- התמודדות עם זוויות שונות של המכונה ויישור בהצלחה ובלי דילג גדול.
- בניית לוגיקה שתאפשר לעקוב ולהתריע על מALLECI המשחק.

אילוצי הפרויקט:

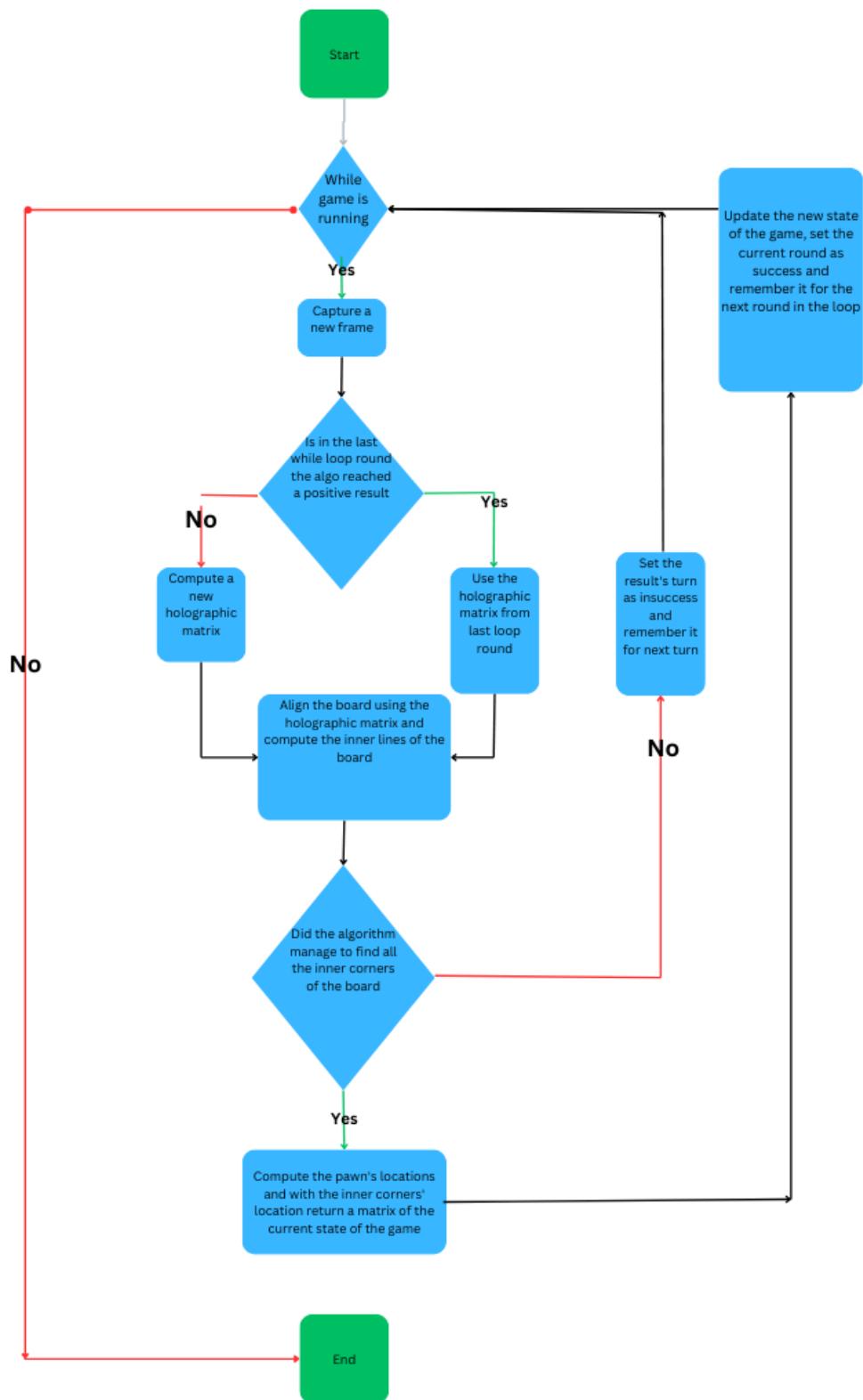
- המכונה צריך להיות מעל הלוח.
- על הלוח להיות לא מוסתר מהמכונה במהלך המשחק (מלבד הושטת יד לביצוע מהלך של שחון).
- לאחר ששחקן עושה מהלך, יש לחכות עד שהמהלך יゾהה לפני ביצוע המהלך הבא.
- על הכלים להיות בצבע לבן ואדום.
- הלוח צריך להיות באזור עם תאורה אחת.

הערה : על הכלים להיות בצבעים אלו מכיוון שהמערכת כוילה בצורה זו. ניתן להתאים אותה לצבעים אחרים.

הנחות הפרויקט:

- אנו מניחים שהמשחק מתקיים עם הלוח והחילילים שעלייהם פיתחנו את הפרויקט. במידה ומלחיפים לוח ו/או חילילים נדרש לכידל את המשחק בהתאם.
- אנו מניחים כי השחקנים מוכנים בין מהלך לעדכון של הלוח במאץ המחשב.
- אנו מניחים "שרופים" – כאשר חייל יכול לאכול חייל אחר, הוא חייב לעשות זאת.

5. תיאור אלגוריתמים בעיבוד תמונה:



תירושים : הלואה הריאשית שמרכזות את הליך עיבוד התמונה של הלוח ממעוף הציפור

הסביר מילולי על האלגוריתם לעיבוד התמונה:

אנחנו נתמקד בlolala הריאשית המתווארת בתרשימים מעלה ומأدגת תחתיה את כל הליק עיבוד התמונה. lolala דוגמת בכל סיבוב שלה את הפריים הנוכחי ומחזירה שני ערכים: סמן תוצאה שמעיד האם הצלחנו למצוא את חיליל המשחק והלוח, ולוח המשחק שבמקרה ששסמן התוצאה שווה ל-0 מחזיר את מיקומי משבצות המשחק והחילילים (מקוטלים לפי צבעים), ובמקרה ששסמן התוצאה הוא 1 מחזיר None. נשים לב לא להתבלבל – בסמן התוצאה דווקא הספרה "0" מעידה על תוצאה חיובית. הסיבה לכך היא שהתחלה היו לנו ספרות נוספות שהסמן החזיר אשר העידו על שגיאות שונות, אך לבסוף החלטנו לאחד את כל המקרים הללו.

כעת, נצלול אל המתרחש בכל סיבוב lolala, וכדי שההסבר יהיה רציף ככל שניתנו, ריכזתי את ההסבירים הקיצרים על האלגוריתמים המשומשים בסוף הפרק הנוכחי:

שלב 1 - יישור הלוח:

ה מערכת מחזיקה תמונה רפנס של הלוח ומנסה ליישר את לוח המשחק מתוך הפריים הנוכחי. פה, ישנו שני מצבים:

1. בסיבוב הקודם של lolala סמן התוצאה הראה תוצאה חיובית – במקרה זה נשמרת המטריצה ההומוגרפית שבה השתמשו בסיבוב הקודם ואנחנו משתמשים בה שוב בסיבוב הנ"ל כדי ליישר את הלוח.
 2. בסיבוב הקודם סמן התוצאה הראה תוצאה שלילית או שזהו הפריים הראשון שנקלט lolala – זה מקרה המופיע בו מתגלית האלגוריתמיקה. במקרה זה אנו מפעילים את אלגוריתם ה-ORB למציאת keypoints ו-descriptors. בין שתי התמונות, ממיינים את הדיקסריפטורים לפי מרחק האמיניג ביןיהם ושומרים אחו זמויים של ההתאמות הכי טובות מתוך סך ההתאמות. לאחר שקיבלו הצלבות Keypoints אנחנו מוחפשים מטריצה homography מתאימה בעזרת cv2.findHomography שמוצאת את המטריצה לפי LS מושלב RANSAC כמתואר בסוף הפרק. לצערנו, לאחר ניסיונות רבים בהם ניתחנו תמונות מזויות שונות ובתנאי תאורה שונים, לא הצליחנו להגיע לערך אופטימלי יחסית או למצוא חוקיות בקשר לכמה הצלבות keypoints למצוא ויאוז אחו מתוכן להשאיר. בנוסף, גם שינוי היפר פרמטרים כמו: שינוי הוויריאנס של ההגרלות בתהליך BRIEF (בחלק מ-ORB), שינוי ערך הסף ב-RANSAC או הגדלת מספר ההגרלות לא עזרו לנו למצוא קומבינציה שמכסה את רוב המקרים.
- על כן, אנחנו מרים lolala מקוננת שרצה על ערכים שונים של מס' הצלבות ואחו ההתאמות שנשאר עד שאנחנו מצליחים ליישר את הלוח כראוי (מנגנון הבקרה מוסבר בשלב 2). במידה שלא מצליחו יישור כנ"ל, אנחנו מסיימים את סיבוב lolala וסמן התוצאה לשיבוב זה הוא 1.

שלב 2 – מציאת הפינות הפנימיות של הלוח:

לאחר שמצאנו מטריצה homography חדשה/nשארכנו עם המטריצה מהפריים הקודם אנחנו מנסים למצוא את הפינות הפנימיות של הלוח. אז אנחנו מישרים את הפריים בעזרת המטריצה ההומוגרפית ומכוון שאנחנו יודעים מהם מימדי תמונה הרפנס, והפריים החדש מיושר ביחס אליה, אנחנו חותכים את הפריים המיושר בגודל קבוע כך שאנחנו נשארים רק עם פנים הלוח לצורך מציאת הפינות בשלב זה.

הטריק ההנדסי הפשטוט הזה מאפשר לנו להתעלם מנקודות מסוימות שנוצרו כתוצאה הלוח או מהעיטורים שמצוירים סביבו.

לאחר החיתוך, אנחנו יוצרים תמונה שפות בעזרת Hough lines ועליה מרים theta. מכיוון שאנו חישבemos קווים ארכיטקטוניים ואופקיים בלבד אנחנו לוקחים רזולוציה גסה בציר theta (והיינו יכולים לחתות עוד יותר). זאת, על מנת שקוים שנוצרו מאותו קו בתמונה יתאחדו. אנחנו יכולים לעשות זאת כי כל היסרים שנוצרו הם או עם ערך theta שווה בערך -0 או שווה בערך $\pi/2$. לאחר מציאת מושאות הקווים אנחנו מוצאים את החיתוכים ביניהם שמתארים את הפינות הפנימיות של הלוח. בשלב זה מגיע הליק הבקרה – אם מספר החיתוכים שווה ל-49 צפוי והחיתוכים יוצרים בקרוב גריד מרוחק בדומה זו אמר שזכה לנו ליישר את הלוח כמו שזכה וסמן התוצאה של סיבוב זה יהיה חיובי. אחרת, נרים שוב את שלב זה בלוואה המקוננת של שלב 1 עם צמד חדש של כמות התאומות Keypoints ואחוז מתוכן שמסונן. במידה שלא הצליחנו למצוא שום צמד כנ"ל שעומד בתנאי הבקרה, סמן התוצאה של סיבוב זה יהיה שלילי, נסים את הסיבוב וניגש לדגום פריים חדש בלי לבצע את שלב 3.

בשלב זה, בזכות הטריק ההנדסי אותו תיארנו, וההבנה כי נוכל לחתות רזולוציה גסה יחסית בציר theta, בחירת רוב הפרמטרים הייתה קלה. הבחירה המשמעותית ביותר הייתה לגבי כמות החיתוכים המינימלית במישור ההאך על מנת לזהות תמונות אליהן נכנסה יד. נכנסו לעובי הקורה בעניין זה בחלק 7.

שלב 3 – מציאת כלי המשחק –

בשלב זה, אליו אנחנו מגיעים רק אם הצליחנו ליישר את הלוח כראוי (ראה בקרה שלב 2), אנחנו מוצאים את כלי המשחק. אנחנו עושים זאת באמצעות Hough circles על גבי התמונה המיושרת (ללא החיתוך של השלב הקודם) כאשר כלי המשחק הם ברדיוס אחד וידוע מראש. על-כן, אנחנו עוברים על טווח מצומצם יחסית של רדיוסים. הצליחנו למצוא שילוב טוב של רזולוציה של המרחב הפרמטרי, מספר חיתוכים מינימלי, ומרחיק מינימלי בין מעגלים למרחב המיקום שאפשר לנו לזהות את השחקנים כראוי.

מכאן, כדי לזהות האם זהו שחקן כתום או שחקן לבן, אנחנו ממיררים את התמונה ל-HSV ודוגמים את הצבע של כל כלי משחק במרקזו. השחקנים הכתומים נבחנים בערך מיוחד בציר ה-HUE שמייד על צבעם, בעוד שהשחקנים הלבנים נבחנים בערך Value מאד גבוהה. על-כן, ע"י הצבת ערכי סף בצירים המתאימים אנחנו מקטלגים כל כלי משחק לצבעו המתאים.

כאמור, בסוף שלב 3, שאלה נגיעה רק אם הצליחנו ליישר את הלוח, אנחנו מחזירים את מיקום הפינות הפנימיות של הלוח ואת מיקום וצבע כל כלי המשחק.

כדי לקבל פרספקטיבה על הפרויקט בכללתו ולא רק על הלוואה של עיבוד התמונה נשים לב שבמקרה שנכנסת יד לפריים שמיוזה שחקן, המשחק לא מתעדכן. רק לאחר שהיד זהה והמצטלה מזזה את הלוח שוב, היא מתחילה לחפש סיבובים מוצלחים רצופים, ואז ע"י majority vote לכל כניסה בין המטריצות שעובדו מ חמישת הסיבובים הנ"ל, נקבעת מטריצת מצב המשחק החדש. על גבי עיבוד התמונה הוספנו בדיקות לוגיות כגון: האם השחקן הנכנס שיחק? האם המהלך היה חוקי? האם שחקן שיחק אם שחקן שעוד שיש לו השמנות אכילה (שרופים)? (נסביר בהרחבה בהמשך)

כך, מסתכם כל הפרויקט למשחק דמeka אינטראקטיבי שmagbir עניין במשחק.

הסבירים קצרים על אלגוריתמי עיבוד התמונה בהם השתמשנו בפרויקט :

BRIEF – אלגוריתם למציאת keypoints descriptors מבוסס FAST למציאת keypoints descriptors. האלגוריתם מקבל שני תומנות אפור (במקרה שלנו הפריים החדש שנdagם למציאת keypoints descriptors). מהഴיר רשיימה של keypoints מותאים מהמצלה ותמונה הרפרנס שיבח ליראה לישר את הפריים) ובין התומנות שמשמעותם לפי המרחק בין הדיסקריפטורים (כמתואר מטה).

FAST הוא אלגוריתם שבו סביב כל פיקסל בוחנים 16 פיקסלים ברדיוס קטן סביב הפיקסל, אם מתוך ה-16 הללו יש רוב מוחלט של פיקסלים שייתר כהים או לחילופין יותר בהירים מהפיקסל הנבחן, הניל מסומן keypoint. על מנת להתמודד עם סוגיות הסקילינג האלגוריתם בונה תחיליה פירמידה ברזולוציה הקטנה בכל שלב פי 2 ומבצע את החיפוש על כל תומנות הפירמידה.

לאחר מכן, על מנת למצוא את האוריינטציה של הfatצ'ים סביב כל keypoint, האלגוריתם מחשב את "מרכז המשא" של עוצמת הפיקסלים (שהושבה לתמונה אפור מלכתחילה) סביב keypoint-המתאים. האוריינטציה של keypoint מחושבת כזווית הנוצרת בין הפיקסל המרכזי לבין "מרכז המשא". לאחר מכן, כל הfatצ'ים המתאים keypoints מסובבים לזוית אחידה.

"מרכז המשא" מחושב כדלקמן :

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y)$$

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right)$$

מכשנחנו את keypoints וסובבנו את הfatצ'ים סביבם לכיוון אחד, האלגוריתם משתמש ב-BRIEF כדי לתאר את הדיסקריפטורים. הרעיון הוא לתאר כל DISCRIPATOR כוקטור ביןאי אקראי באורך קבוע. האלגוריתם קודם כל מעביר את התמונה דרך פילטר גאוסיאני שמטרתו "להחליק" את התמונה ולהוריד את הרעש, ולאחר מכן מתחילה ביצירת הוקטור הבינהי : לכל כניסה בוקטור נבחרים שני פיקסלים מהfatץ' סביב הפיקסל המרכזי באופן אקראי. כבירות מחדל, שני הפיקסלים מוגרים מהתפלגות גaussית, כאשר הפיקסל השני מוגרל מתוך התפלגות גaussית עם וריאנס גדול פי שניים מהווריאנס של התפלגות שמננו הפיקסל הראשון והורל. אם הפיקסל הראשון בהיר מהשני, הכניסה הנבחנת מסומנת כ-1, ואחרת מסומנת כ-0.

כך, אנחנו מתארים כל keypoint עם descriptor אקראי ואז אנחנו מצלבים בין הדיסקריפטורים משתתי התמונות ומוגדים כמו הוקטוריים הבינהיים קרובים לפי מרחק Hamming.

Cv2.findHomography – פונקציה שמקבלת keypoints תואמים בין שתי תמונות ומחשבת את המטריצה ההומוגרפית של ההעתקה ביניהן. הפונקציה הנויל בוחרת את המודל הטוב ביותר באמצעות LS RANSAC.

כלומר, אנחנו מגרילים מספר מסויים של הגרלות (ערך ברירת המחדל, שגם השתמשנו בו, הוא 2000) שבכל הגרלה מוגרים תשעה keypoints מתאימים ולפיהם מחושבים הפרמטרים שביאים למינימום את השגיאה הריבועית. בהינתן המודל שמתקיים, מחשבים את תוכאות ההעתקה לכל ה-keypoints-המקוריות. מתוך התוצאות הנויל מחשבים את השגיאה לכל הצלבה ובוחרים מתוך כל החרولات את המודל שבו המודל טעה במס' הגדול ביותר של הנקודות בכל היותר שגיאה מסוימת הנקבעת מראש.

Canny Edge Detector – אלגוריתם שמקבל תמונה אפור ומחזיר תמונה שפותה בינהarity. האלגוריתםראשית מעביר את התמונה דרך פילטר "נגזרת של גאוסיאן" שזה הרכבה של החלקת התמונה ולאחר מכן גוזרת ליזחיו שינויים. מכאן, מחושבים האוריינטציות והגודל של הגראדיאנטים בכל נקודה. הגודל משמש לשלב הבא והאוריאנטציה יכולה להיות שימושית בשילוב עםough Transform. למשל ככלי בקרה לכיוון השפט שימצאו בו. אם כן, בשלב לאחר מכן מוגדרים שני ערכי סף: גובה ונמוך. אנחנו משאים תחילת שפותה שعواמת הגראדיאנט שלחן עוברת רק את הערך הגבוה, ואז, שפות שעוברות את הערך הנמוך וגם "משלימות" את השפות שזוחו עם הערך הגבוה, נוספת גם הן לתמונה המוצאת.

Hough Lines – הקדשו זמן רב לטרנספורמציה הנויל בכיתה אז אנסה לתאר בקצרה יחסית. אלגוריתם שמקבל תמונה שפותה בינהarity ומוחזיר מספר משווהות קווים ישרים שמתוארים בתמונה לפי הפרמטרים השונים. הרעיון הוא להסתכל על מרחב פרמטרי שנקבע ע"י המשתנים rho ו-theta כך שכל נק' במרחב הפרמטרי מתארת קו במרחב המיקום ולהפוך. המשוואה המקשרת בין המרכיבים היא:

$$r = x \cos \theta + y \sin \theta$$

מכאן, נרצה להמיר כל נק' במרחב הפרמטרי החדש כאשר הקו הנויל מתאר את אוסף כל הקווים היישרים שעוברים בנק' הנויל במרחב המיקום. הנקודות בהן יתקבל במרחב הפרמטרי מספר החיתוכים הגדל ביותר מתארות את הקווים במרחב המיקום עליהם נמצא המספר הגדול ביותר של נקודות. פרמטר חשוב בהקשר זה שאיתנו שיחסנו במהלך בניית האלגוריתם שלנו הוא מס' החיתוכים המינימלי במרחב הפרמטרי על מנת שנק' במרחב הפרמטרי תוגדר כקו בתמונה המקורית.

שני פרמטרים חשובים מאוד הם הרזולוציות בכל ציר במרחב הפרמטרי – ככל שהרזולוציה נמוכה יותר, המודל יותר חסין לרעש כי קווים שנחתכים כמעט מוקוטלים לאוֹתנה נק' חיתוך. מצד שני, ככל שהרזולוציה גבוהה יותר, משווהות הקווים שמחושבות מדויקות יותר.

Hough Circles – אלגוריתם שמקבל תמונה שפותה בינהarity ומוחזיר מספר משווהות מעגלים שמתוארות בתמונה. הרעיון עם המרחב הפרמטרי הוא זהה גם פה, רק שכעת המרחב הפרמטרי הוא תלת ממדי כאשר המימד השלישי הוא רדיוס המעגלים. לכל רדיוס נתון שני פרמטרים הנוטרים הם a ו- b כשהכל צמוד כנoil ורדיוס נתון מתקיים מעגל במרחב המיקום, ולהפוך. פה המשוואה המקשרת היא:

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

פרמטרים נוספים שנווזרנו בהן בפונקציה :

הרזולוציות של המרחב הפרמטרי - מתקיים אותו הטריד אוף כמו במקרה של הקווים הישרים מינימום חיתוכים במרחב הפרמטרי – גם כאן זהה במקרה של הקווים הישרים.

מרחב מינימלי בין מעגלים במרחב המיקום – עוזר לאפליקציה שלנו ספציפית כי לא יכול להיות שיש חילוקים ש"עלולים אחד על השני".

רדיווס מינימלי ומקסימלי – קביעת הגבולות של ציר הרדיוס ערך סף גובה ל- Canny – בוגוד במקרה של ה-lines, בימוש של OpenCV הכניסו את ה-`houghCircles` לתוך הפונקציה של Hough Circles. על-כן, מדירים את הערך הגובה, והערך הנמוך נלקח כחצי ממנו.

מרחב צבע אלטרנטיבי ל-RGB שמורכב משלושה צירים :

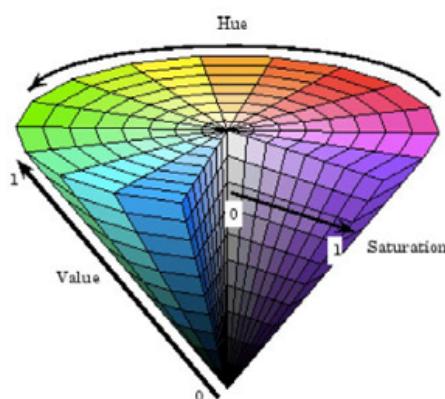
ציר הצבע Hue

גוון הצבע הנתון (כгалות ב-Value) Saturation

בהירות הצבע הנתון Value

זהו מרחב אלטרנטיבי שמתאים יותר לרוב לתפיסה שלפיה בני אדם חושבים על צבעים.

מרחב הצבע מתואר לרוב בצורה קונווס :



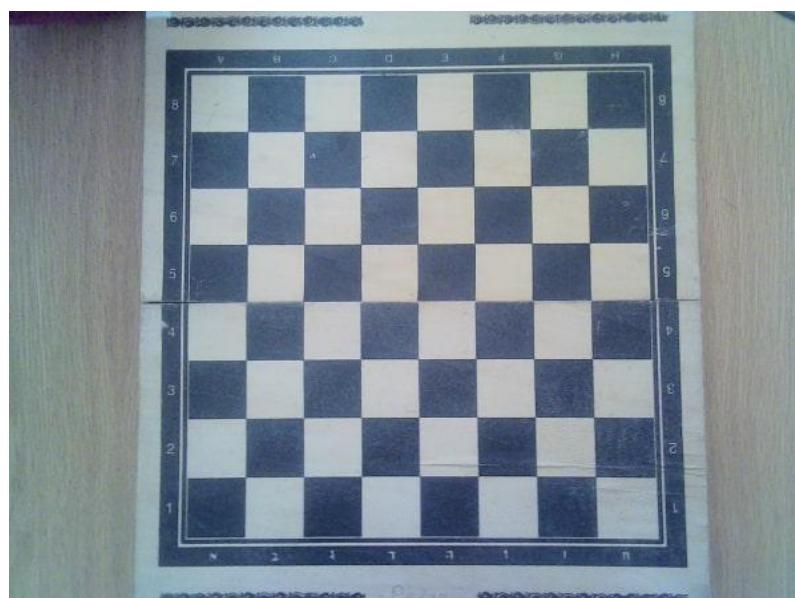
6. מקרי בוחן שמייצגים את חלקו האלגוריתם:

על מנת להציג את חלקו האלגוריתם שמרכיבים את מכלול עיבוד התמונה בפרויקט בחרנו לנתח שתי תמונות:

1. דוגמתא חיובית שבעזרתה נציג את התהליך ש庫ורה בסיבוב לולאה בודד כאשר נקלטת תמונה חדשה. לשם העניין נניח כי עליינו לחשב מטריצה הומוגרפית חדשה בשלב זה.
2. תמונה שלילית שבה יד נכנסת לפריים ואננו מזהים שימושו לא תיקון כמתואר במנגנון הבקרה בשלב 2 בחלק הקודם.

נתחיל מקרה הבוחן העיקרי, התמונה הראשונה:

از יש לנו כאמור תמונה רפרנס שהוכנה מראש:



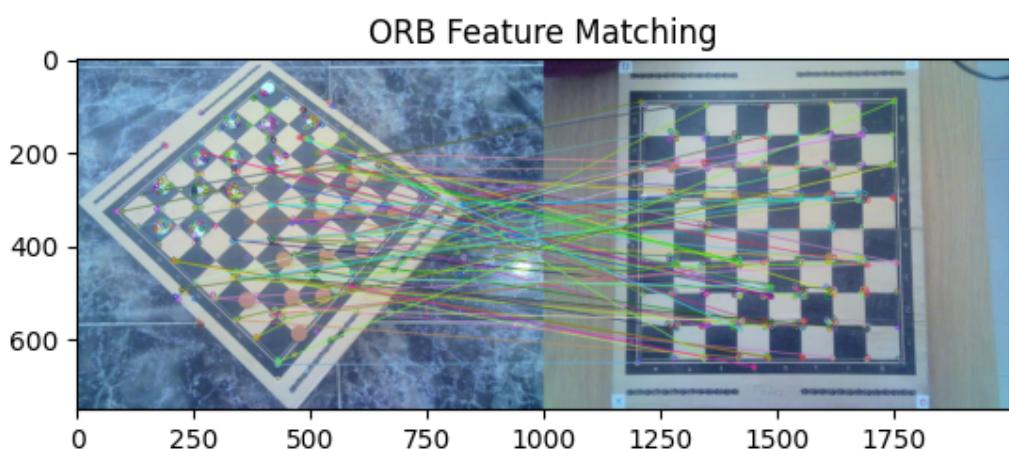
ועכשו המצלמה קראה פריים חדש :



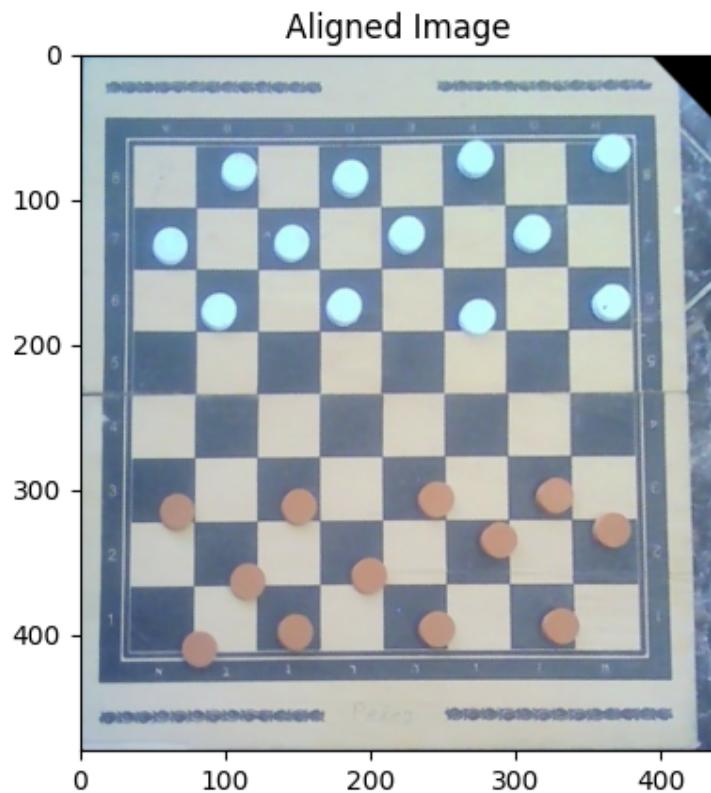
כפי שניתן לראות, על מנת ליישר את תמונה הרפרנס נדרשת העתקת 8 דרגות חופש.

בשלב זה הlolאה המקוונת של שלב 1 (ראה חלק 5) מנסה למצוא הצלבה של מספר זוגות keypoints לחפש ואיזה אחזו מתוכן להשאיר לאחר החיפוש. במקרה זה, אנחנו נצליח למצוא זוג כנ"ל שיעבור את הבדיקה של שלב 2 ולכן יישור התמונה יעבדו נכון אבל נפרק את התהליך שלב שלב.

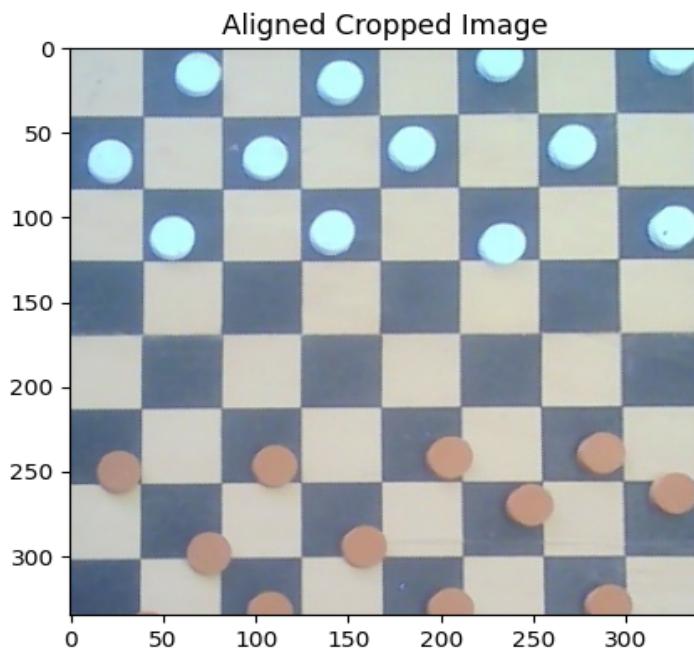
בහינתן שמצאנו הצלבה כנ"ל, נסתכל על התאמות keypoints בין שתי התמונות (למרות שהדבר אינו אינפורטטיבי במיוחד) :



כעת, ניישר את התמונה בעזרת ההעתקה ההומוגרפית שחישבנו בהינתן ההתאמות:

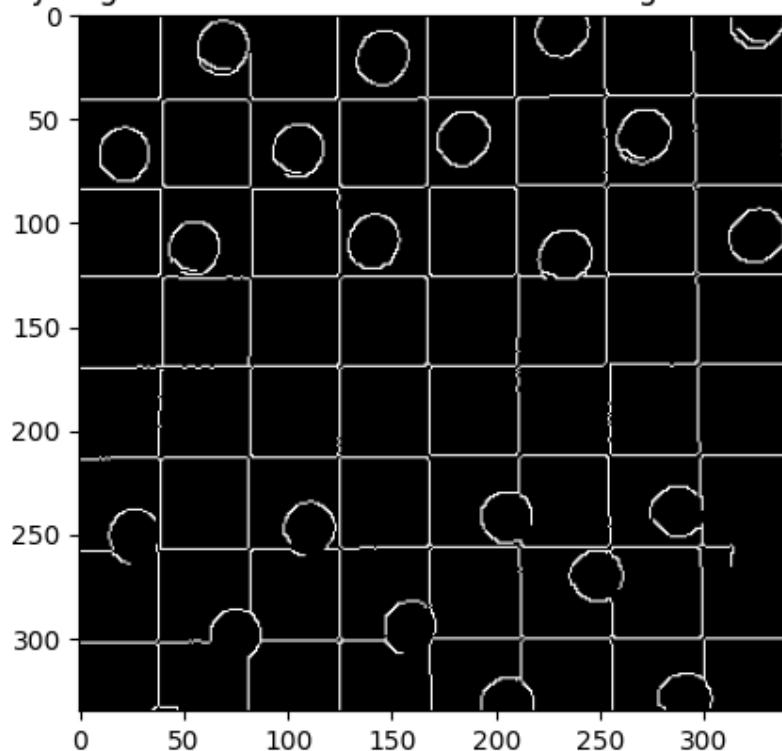


מכאן, כמוتواר גם כן בחלק 5, על מנת להקל על עבודה מציאת הקווים, ובגלל שימושם בתמונה המיוושרת
דועים בקרוב מראש נחתוך פנים אל הלוח בלבד:

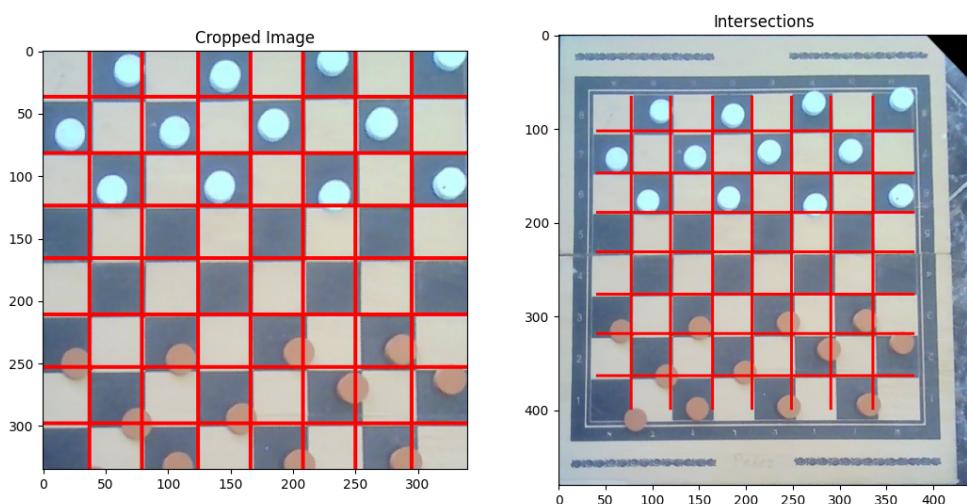


ניצור תמונה שפות בעזרת : Canny

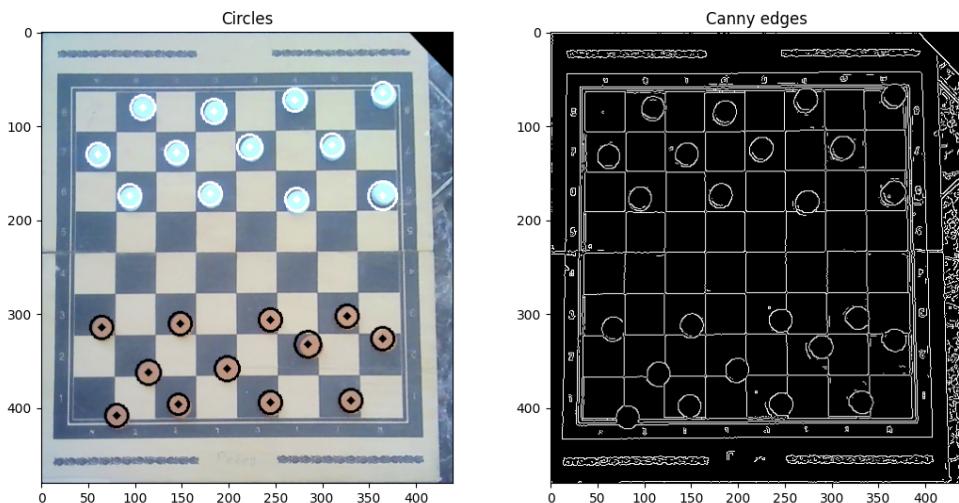
Canny Edges with low threshold: 150 and high threshold: 400



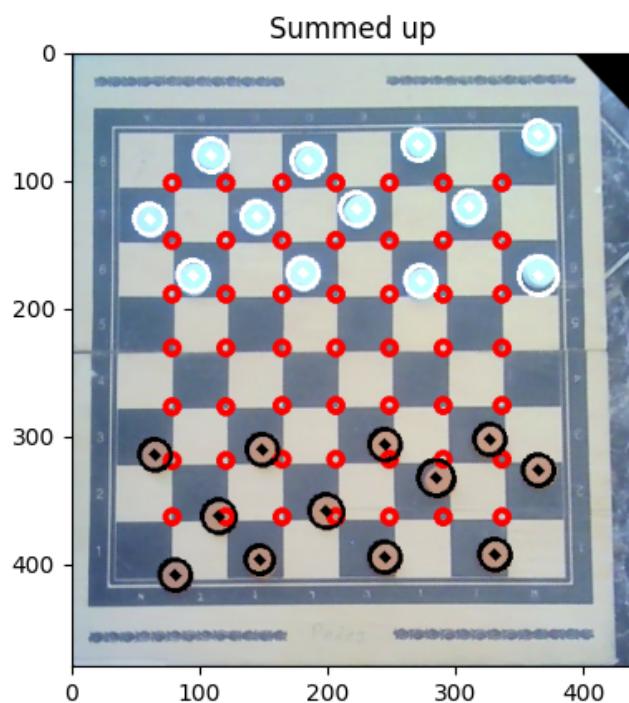
נפעיל את אלגוריתם ה-Hough Lines ונחזיר לתמונה הלא חתוכה :



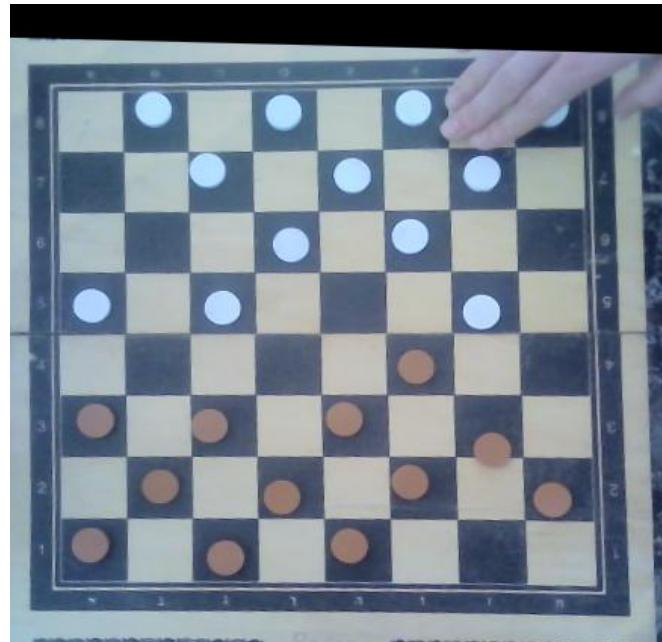
מחיתוכי משווהות הקווים אנחנו מוצאים את הפינות הפנימיות של לוח המשחק. בשלב 3, נמצא את כל המשחק באמצעות Hough Circles וננתח את צבעם כמתואר בחלק 5 :



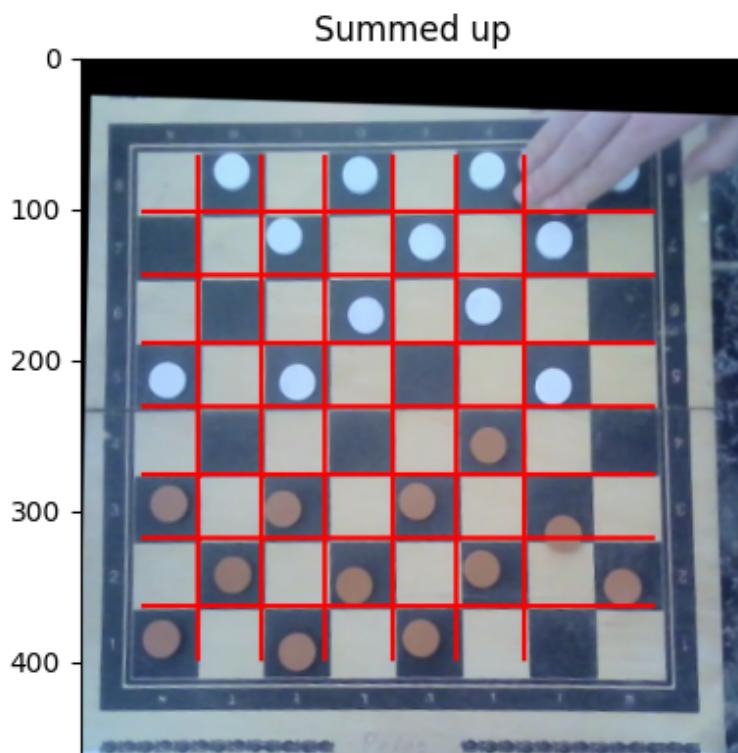
לבסוף, נסכם את כל התוצאות :



כעת, נראה מה קורה במערכת במקרה של דוגמה שלילית בה יש יד שנכנסה לפוריים. לאחר יישור הפורים מתקבלת התמונה הבאה :



לאחר הפעלת שלב 2 :



כפי שניתן לראות, היד שמשתירה חלק מהקו האנכי הימני ביוטר גורמת לאלגוריתם לא לזהות אותו, והדבר נופל בבדיקה הבדיקה לפי חיתוכי הקווים יוצרים גריד מרוחה באופן שווה של 49 נקודות. לעומת זאת, האלגוריתם זיהה את הפורים כפורים שלילי כי שאנחנו מצפים ממנו.

7. אנליזה של אחד מחלקי האלגוריתם

בחרנו לבצע אנליזה פרמטרית שלב 2 באלגוריתם עיבוד התמונה (ראה חלק 5). על כן, יצרנו מאגר תמונות של 40 תמונות של הלוח מזוויות שונות – 20 מתוךן הן הדוגמאות החיוביות שהבחן האלגוריתם אמור לזהות את גריד הלוח, וה-20 הנותרות הן הדוגמאות השליליות בהן יש יד שנכנסה לפירים, ועל כן האלגוריתם אמור להחזיר סמן תוצאה שלילי. נציין שעל מנת לבדוק את שלב 2 באופן בלתי תלוי משלב 1 הוא בלאי תלוי בכל מקרה מסוים (3), ווידאו שהאלגוריתם מצליח ליישר בשלב 1 את כל התמונות שבחרנו. לשם הדוגמה, התמונה הראשונה שניתחנו בחלק 6 היא דוגמה חיובית, והתמונה שבאה לאחר מכן היא דוגמה שלילית.

הפרמטר הראשון שבחרנו לבחון הוא כמות החיתוכים המינימלית במישור Hough כדי לקבוע שנק' מסויימת במישור הניל מייצגת קו בתמונה המקורית. ציפינו שהדבר העיקרי ישפיע על זיהוי התמונות השליליות מתוך ניסיון שהיה לנו בזמן בניית האלגוריתם. כאמור בפרק 5, בזכות הטريق של חיתוך אל פנים הלוח לא נדרש תחילה לעשות פיין טויניג להיפר פרמטר הניל במקרה של דוגמאות חיוביות. זאת, מכיוון שהקוואים היו ארוכים מספק ולא היו קווים מסתיקים אחרים בתמונה. לעומת זאת, במקרה של זיהוי תמונות שליליות, הפרמטר הניל היה פרמטר מאד משמעותי. בעזרת בחירה נבונה שלו יכולנו להדיח קווים שיש להם מספר קטן (אפילו במעט) של חיתוכים במישור ההאך כי היד מסתירה חלק מהקו.

בעקבות הניסיון הניל, יצאנו לבדוק את התופעה באופן מדויק יותר בעזרת מאגר התמונות שתיארנו מעלה.

יצרנו confusion matrices עבור שלושה ערכים שונים של הפרמטר הניל:

(1) מס' חיתוכים מינימי = 100 :

		Actual values	
		Positive	Negative
Predicted values	True	17	9
	False	3	11

(2) מס' חיתוכים מינימי = 160 (הערך שנבחר בהצגת הפרויקט) :

		Actual values	
		Positive	Negative
Predicted values	True	17	4
	False	3	16

(3) מס' חייטוכים מינימלי = 350 :

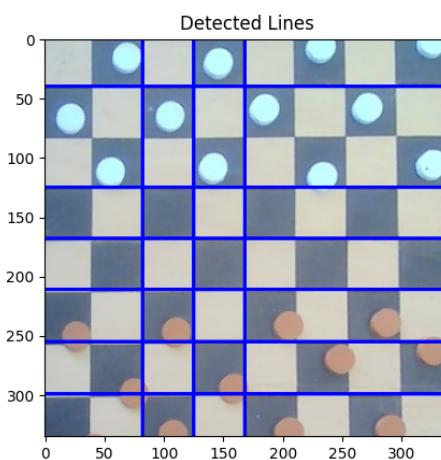
		Actual values	
		Positive	Negative
Predicted values	True	14	1
	False	6	19

כפי שניתן לראות, השערתינו הייתה נcona – ככל שהעלוינו את מספר החיטוכים המינימלי במשור האפ' מס' דוגמאות ה-FP פחתו, ומצד שני, כאשר העלוינו את המס' משמעותית התחלו לראות כי מס' הדוגמאות שם FN מתחילה אף הן לעלות כמתואר בתרiid אוף מעלה.

כעת, הfrmter השני שניגשנו לבחון הוא הרזולוציה של הfrmter θ במרחב האפ'. כידוע, ככל שהרזולוציה קטנה יותר אנחנו מקבלים קווים יותר מדויקים, אך מצד שני נק' במרחב יכולות לתאר אותו קו שמנצא במרחב המיקום בתמונה המקורית. לכן, קווים מסוימים יכולים להתפס או לחילופין האלגוריתם יזהה מס' קווים שונים שבפועל נוצרים מאותה השפה בתמונה המקורית. נציין שוב כי כמתואר בחלק 5, את הfrmter של ציר θ בחרנו בסות כי אנחנו מחפשים רק קווים אופקיים ומאונכים.

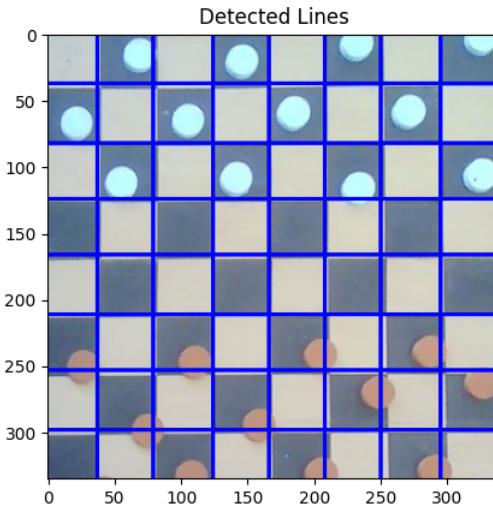
נכח שוב את התמונה החיובית שהוצאה מחלק 6 ונשים לב איך השינוי בfrmter משפיע עליה :

$$\text{Rho_resolution} = 1: \quad (1)$$



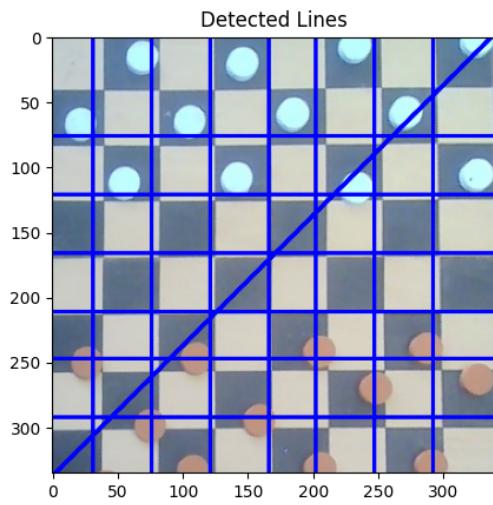
כפי שניתן לראות, הקווים מדויקים להפליא. מה שקצת קשה לראות בתמונה זה שמצוירים בה 14 הקווים המשמעותיים ביותר במרחב האפ', ואננס אנחנו רואים רק תשעה. הסיבה היא שבפועל אכן מצוירים 14 קווים, רק שחלקים מאד קרובים אחד לשני ומתחברים את אותה השפה בתמונה המקורית. כאמור, זה הוצאה אפשרית של רזולוציה גבוהה מדי.

$$\text{Rho_resolution} = 4: \quad (2)$$



זהו הפרמטר שנבחר גם בעת הצגת הפרויקט. כפי שניתן לראות אם נתבונן היטב, הקווים פחות מדויקים, אבל אכן האלגוריתם מזוהה את 14 הקווים שאנו חזו ממצפים ממנו.

$$\text{Rho_resolution} = 8: \quad (3)$$



כאן, הרזולוציה גסה מדי ואנחנו משלים פעמיים – גם הקווים לא מדויקים באופן גס, וגם מזוהה קו שלא היה בתמונה המקורית כי נראה שיווקר מדי נקודות במישור האט "שזילגו" לשכצת שלו.

זו הייתה רק דוגמה ויזואלית כմובן, אך היא הייתה חשובה כדי להראות את רמת הדיוק בקווים שהזוהו משתנה שकצת יותר קשה למדוד. על-מנת לתת לדבר תמקך טיפה יותר מספרי, ללחנו רק את הדוגמאות החיוביות ובדקנו בכמה מתוכן הצלחנו למצוא גרייד. את הדוגמאות השיליות הזוחנו למרות שהפרמטר משפיע גם עליהם. זאת כמובן, מכיווןuai היוצרים הגרייד במקרים אלו עלולה להיות גם מהיד המסתירה ולא רק מערכ הפרמטר.

ב-8 תמונות בלבד הצלחנו לזהות גריד. = 1 : Rho_resolution (1)

ב-17 תמונות הצלחנו לזהות גריד. = 4 : Rho_resolution (2)

ב-12 תמונות הצלחנו לזהות גריד. = 8 : Rho_resolution (3)

התוצאות תואמות לתיאוריה ולדוגמא הוויזואליות שעליהן פירטנו מעלה.

8. תיאור הלוגיקה המגשרת בין הנ吐נים מעיבוד התמונה למשחק:

* לא נכתב בהוראות לדוח שצריך לפרט חלק זה, אך בפרויקט שלנו לוגיקה זו היא חלק ממשמעותי מהפרויקט ואנגריו ולכון הוסיףנו חלק זה.
מהחלק של עיבוד התמונה אנחנו מקבלים מיקומים של מרכזים העיגולים עם צבעיהם ומיקומים של 49
הצמתים הפנימיים של הלוח.
מנ吐נים אלו אנו נרצה להפיק את מיקומי הכלים על הלוח :

שלב 1 – המرة של הנ吐נים למטריצה 8x8:

באמצעות מיקומי הצמתים נוכל לחשב איזה כלי נמצא באיזה משבצת על ידי חילוק הלוח שלנו ל- 64
תחומים ונמפה כל מרכז מעגל לפי התוחום בו הוא נופל.
ואז אם בנקודה j , יש חייל אדום אז נקבל :

```
board[i, j, 0] = 1
```

```
board[i, j, 1] = "red"
```

שלב 2 – בדיקה האם היו שינויים בלוח ומה הם היו :

על מנת להבין אם היו שינויים בלוח, בכל איטרציה נשתמש במטריצת `old_board` (אותה אנחנו נשמר ונדכן בכל איטרציה).
נדיר את מטריצת השינויים :

```
Changes_matrix = board[:, :, 0] - old_board[:, :, 0]
```

כעת נוכל לבדוק באיזה אינדקסים `Changes_matrix` שווה ל- 1 (כלומר באיזה משבצות לא היה חייל ועכשיו יש) ובאיזה אינדקסים `Changes_matrix` שווה ל- -1 (כלומר באיזה משבצות היה חייל ועכשיו אין).

לפי נתוניים אלו נוכל לדעת איזה מהלך נעשה :

- חיל התקדם - החיל עבר מהמשבת עם הערך 1 - למשבצת עם הערך 1 .
זהה מהלך זה כי יהיה רק משבצת אחת עם ערך 1 ו ורק משבצת אחת עם ערך 1 - והMOVE ביןיהם יהיה 1 בציר ה- x ו - 1 בציר ה- y .

לדוגמה :

```
[[0, 0, 0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 1, 0, 0, 0, 0],  
 [0, 0, -1, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0]]
```

- חיל אכל חיל אחר - חיל עבר מהמשבת עם הערך 1 - המתאימה לצבע החיל ב- old_board ו עבר למשבצת עם הערך 1 .
זהה מהלך זה כי יהיה רק משבצת אחת עם ערך 1 ושתי משבצאות עם ערך 1 . המOVE בין המשבצת ממנה עבר למשבצת אליה עבר יהיה 2 בציר ה- x ו 2 בציר ה- y וביניהם יהיה משבצת עם הערך 1 .

לדוגמה :

```
[[0, 0, 0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 1, 0, 0, 0],  
 [0, 0, 0, -1, 0, 0, 0, 0],  
 [0, 0, -1, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0]]
```

- שחקן עשה מהלך לא חוקי - קיימות דוגמאות רבות לכך :
דוגמה – חיל התקדם צעד לא חוקי :
זהה מהלך זה כי יהיה רק משבצת אחת עם ערך 1 ו רק משבצת אחת עם ערך 1 - והMOVE ביןיהם יהיה שונה מ- 1 בציר ה- x ו/או שונה מ- 1 בציר ה- y .

```

[[0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 1, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, -1, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0]]

```

בדיל מקרה זה מהמקרה בו חיל_1 אכל חיל_2 אך השחקן לא הרים את הכליל של חיל_2 שהוא אכל.
במקרה זה `Changes_matrix` יהיה זהה ולכון נבדק ב- `board` האם יש חיל_2 בין המשבצת ממנו עבר חיל_1 למשבצת אליה עבר.

כאמור ישנו הרבה מקרים כאלו וניסינו לזהות את colum ובהתאם להתרעה על לוח המשחק במחשב.
כאשר זיהינו מהלך לא חוקי המשחק לא ימשיך עד שהלוח יחזיר למצבו הקודם (לפני המהלך הלא חוקי).
דוגמאות נוספות למקרים בהם אנחנו מזהים מהלך לא חוקי:

- 2 שחקנים עשו מהלך במקביל.
- השחקן הלא נכון ביצע מהלך.
- חיל זו לכיוון הפוך.
- חיל נעם מהשחקן.
- חיל נוסף למשחק.
- ועוד..

כך נעקוב אחר כל המהלים, נציג על הלוח את המהלים שנעשה, נטריע תורו איזה שחקן לשחק בכל רגע נתון, נטריע על מהלים לא חוקיים ועל ניצחונו של אחד השחקנים.

9. סיכום :

בפרויקט שלנו עשינו משחק דמקה אינטראקטיבי בו צילמנו משחק דמקה והקנו אותו על מסך המחשב עם התמונות של השחקנים בתור כלិ המשחק. במסך המחשב אנו מראים תור איזה שחקן עכשו לשחק, מה המהלך שנעשה, האם נעשה מהלך לא חוקי ומני ניצח.

עשינו זאת באמצעות אלגוריתמים של עיבוד תמונה שלמדנו בכיתה :
השתמשנו ב-algorithms Canny Edge Detector ו-Hough Lines על מנת למצוא את הגריד של המשחק, Hough Circles על מנת למצוא את השחקנים.
נעזרנו גם באלגוריתמים שלא נלמדו בכיתה : descriptors keypoints ו-ORB
כמו כן יישמנו לוגיקה לזיהוי המהלים והתמודדות עם מהלים לא חוקיים והתראה עליהם.
את הפרויקט העלו ל- git hub וניתן לראות אותו בקישור :

https://github.com/oritalp/DIP_Final_project

ביבוגרפיה : .10

OpenCV - <https://docs.opencv.org/4.x/index.html>