עבודה מסכמת לתואר שני

זיהוי פינות בראיה ממוחשבת

מחבר: יבגני שמיס

ת"ז: 310256573

מנחה: ד"ר ג'ק וינשטין

האוניברסיטה הפתוחה

החטיבה למדעי המחשב

אוגוסט 2011

תוכן עניינים

1 מבוא 6

1.1 עיבוד תמונה וראיה ממוחשבת 6

1.2 ייצוג תמונה במחשב 7

1.3 חילוץ מאפיינים 8

2 מושגים בסיסיים 9

2.1 הגדרות 9

2.2 מדד ביצועי האלגוריתמים השונים 9

3 האלגוריתמים לזיהוי פינות 10

3.1 הגישה הכללית 10

3.2 מציאת מכסימום מקומי בשיטת Tomasi ו־Kanade 10

3.3 מציאת מכסימום מקומי בשיטה מקבילית 12

3.4 האלגוריתם של Moravec 12

3.5 אלגוריתם Harris & Stephens 19

3.6 אלגוריתם Shi & Tomasi 24

3.7 האופרטור של טרייקוביץ' 27

3.7.1 עקרונות האלגוריתם 27

3.7.2 חישוב מעשי 28

4 שימוש באלגוריתמים גנטיים 34

4.1 אלגוריתמים גנטיים 34

4.2 תכנות גנטי 36

4.2.1 ייצוג הפרט 36

4.2.2 יצירת אוכלוסיה התחלתים 37

4.2.3 פעולות אבולוציוניות 38

4.2.4 הערכת האוכלוסיה 38

4.3 אלגוריתמים גנטיים למציאת אופרטור זיהוי פינות אופטימלי 38

4.3.1 מרחב החיפוש 39
4.3.2 הערכת אלגוריתמים 39
4.3.3 התוצאות 41
5 סיכום 42
6 ספרות 43

רשימת איורים

1 דגימה מרחבית של תמונה 8
2 תרשים זרימה של מעבר בודד באלגוריתם NMS מקבילי 14
3 האופרטור של מורבץ 14
4 נקודה P שהיא פינה אמיתית 15
5 מפת פינתיות 16
6 תמונה לבדיקת אלגוריתמי זיהוי פינות 16
7 תוצאות הרצת אלגוריתם Moravec על תמונת בדיקה 18
8 תוצאות הרצת אלגוריתם Moravec על תמונות אמיתיות 19
9 M_c כתלות ב־$\lambda_{1,2}$ עבור ערכי k שונים 22
10 תוצאות ההרצה של אלגוריתם הריס־סטפנס 24
11 תוצאות ההרצה של אלגוריתם הריס־סטפנס על תמונות אמיתיות 24
12 תוצאות ההרצה של אלגוריתם שי־טומזי 26
13 תוצאות ההרצה של אלגוריתם שי־טומזי על תמונות אמיתיות 26
14 סימונים עבור האופרטור של טרייקוביץ' 27
15 מקרים שונים עבור אופרטור טרייקוביץ' 28
16 מיקום הנקודות 29
17 שיטת Interpixel Approximation 30
18 תוצאות הרצת אלגוריתם טרייקוביץ' 33
19 נקודה על שפה אלכסונית 33
20 נקודה שהיא פינה אמיתית 34
21 הפעלת אופרטור טרייקוביץ' על תמונות אמיתיות 34
22 הצגת פונקציה כעץ (מתוך [12]) 36
23 הרצת IPGP1 על תמונה אמיתית 42

רשימת אלגוריתמים

1 מציאת מכסימום מקומי של Tomasi ו־Kanade 11

2 אלגוריתם NMS מקבילי 13

3 אלגוריתם מורבץ 17

4 אלגוריתם Harris & Stephens 23

5 אלגוריתם Shi-Tomasi 25

6 אלגוריתם טרייקוביץ' 32

7 יצירת פרט בשיטת Grow 37

8 יצירת פרט בשיטת Full 38

9 אלגוריתם IPGP1 42

תקציר

העבודה מהווה סקירה של מספר אלגוריתמים של זיהוי פינות. תאור כל אחד מהאלגוריתמים יכלול רקע תאורטי המתאר את העקרונות המתמטיים בהם נעשה השימוש, מימוש האלגוריתם בשפת תכנות והדגמת פעולת האלגוריתם לקלטים (תמונות) שונים. האלגוריתמים המתוארים כוללים את האלגוריתם של מורבץ (Moravec), האופרטור של הריס וסטפנס (Harris&Stephens), האופרטור של שי וטומזי (Shi & Tomasi) והאופרטור של טרייקוביץ' (Trajkovic). בנוסף תתואר שיטה מתקדמת של תכנות גנטי ליצירת אלגוריתמי זיהוי חדשים.

חשיבות העבודה

תחום הראיה הממוחשבת, וזיהוי צורות ופינות בפרט, הינם חלק בלתי נפרד מתחום הרבה יותר רחב של *בינה מלאכותית*. החשיבות של הראיה הממוחשבת בתחום זה היא במתן אפשרות *לסוכנים חכמים* (אלגוריתמי בקרה המממשים סוגים שונים של *בינה מלאכותית*) לקלוט ולבצע ניתוח מידע בעזרת חיישני חוזי, המדמים ראיה של בני אדם ובעלי חיים. מידע זה נחוץ מכיוון שהוא כולל את התמונה המלאה ביותר על הסביבה של הסוכן הפועל בעולם הפיסי האמיתי. קיימים אינספור ישומים של ראיה ממוחשבת ועיבוד תמונה בתחומים שונים כגון: זיהוי ונעילה על מטרה במערכות נשק, זיהוי פנים בצילום ספרתי, בדיקת מעגלים אלקטרוניים מודפסים, גילוי תנועה במערכות אבטחה, חיפוש והתאמת תמונות במסדי נתונים וכו'. בכל אחד מהתחומים נעשה שימוש בניתוחים של תכונות שונות של תמונה ממוחשבת, בינהם גם בזיהוי הפינות, לו מוקדשת העבודה הזאת.

1 מבוא

1.1 עיבוד תמונה וראיה ממוחשבת

בין התחומים של מדעי המחשב המתפתחים בצעדי ענק בשנים אחרונות הינם התחום של *עיבוד התמונה* ותחום בעל קשר הדוק אליו של *ראיה ממוחשבת*. כאשר מדברים על עיבוד התמונה, בדרך כלל מתכוונים לפעולות חישוב אשר הקלט שלהן הוא תמונה ספרתית והפלט הוא תמונה ספרתית. פעולות החישוב יכולות להיות לצורך שיפור התמונה המקורית (תיקוני צבעים, ניגודיות ובהירות, הורדת רעשים, תיקונים גאומטריים), לצורך דחיסת התמונה על מנת להקטין את נפח האיחסון הדרוש לשמירתה או להתאימה לערוצי תקשורת איטיים או לצרכים אחרים. להבדיל מעיבוד התמונה, תחום הראיה הממוחשבת עוסק בעיבוד הקלט שהוא גם תמונה (או אוסף תמונות) על מנת לזהות תכונות ופרמטרים שונים לצורך הסקת מסקנות וקבלת החלטות על בסיסם. הראיה הממוחשבת, כפי ששמה מרמז, קשורה לתחום הראיה הביולוגית, ומטרתה לתת למערכות מלאכותיות יכולת "ראיה", כאשר תחת "ראיה" מבינים איסוף מידע חזותי ושימוש בו לביצוע משימה או תהליך מסוים. ישנם אינספור ישומים של ראיה ממוחשבת כמו בקרת תהליכים ורובוטיקה (רובוטים תעשייתיים, רכבים אוטונומיים, מערכות נשק וכו'), אבטחה (גילוי תנועה, זיהוי פנים), איסוף מידע (קיטלוג תמונות, זיהוי טקסטים) וכדומה.

1.2 ייצוג תמונה במחשב

לפני כל עיבוד ספרתי של תמונה במחשב יש צורך להמיר אותה לייצוג מתאים לעיבוד. במסגרת עבודה זאת נתמקד בתמונות ללא צבע, עם רמות אפור בלבד. תחת דרישה זו ניתן להציג כל תמונת רמות אפור בצורה של פונקציה $f(x,y)$, כאשר

$$\begin{aligned} f : (x,y) \in \Omega &\rightarrow [0,1] \\ \Omega &\subset \mathbb{R}^2 \end{aligned}$$

כאן Ω היא תחום התמך של התמונה (הרי שממדי התמונה מוגבלים לערכים סופיים). כלומר, f ממפה כל נקודה מתחום הגדרת התמונה לערך ממשי שבין 0 ל־1, המסמל את רמת האפור או את ה"בהירות" של הנקודה, כאשר הערך 0 הוא הרמה הנמוכה ביותר (הנקודה היא "שחורה") ו־1 הוא הרמה הגבוהה ביותר (הנקודה היא "לבנה").

הצגה כזאת עדיין לא מאפשרת עיבוד ספרתי של התמונה משתי סיבות:

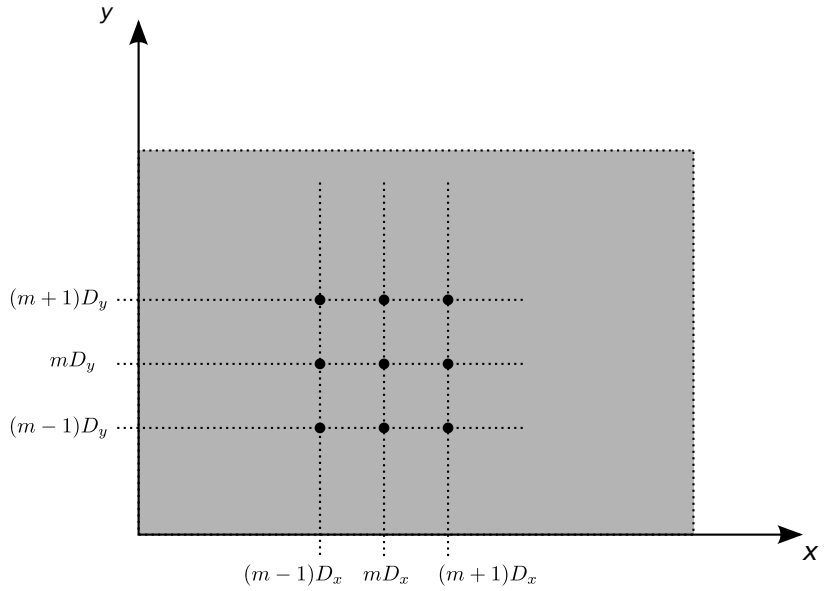
1. התחום של הפונקציה Ω הוא אוסף אינסופי של נקודות שאינו בר מנייה , ולכן לא ניתן לייצג את כל הנקודות באמצעים מקובלים.

2. הטווח של הפונקציה הוא כל ערך ממשי בקטע $[0,1]$, כלומר קיים מספר אינסופי של ערכים אפשריים, ולכן אין דרך לקודד ערכים אלה על ידי מספר סופי של סיביות.

את הבעיה הראשונה פותרים על ידי דגימה מרחבית של התמונה המקורית במרווחים של D_x ו־D_y בין הנקודות הנדגמות בצירים x ו־y בהתאמה, כך שיתקבל מספר סופי של הנקודות

$$f[m,n] = f(x,y)|_{x=mD_x,y=nD_y} = f(mD_x, nD_y)$$

כאן $f[m,n]$ היא הפונקציה הבדידה, כאשר m,n - מספרים שלמים (אשר ניתן להגדירם ללא הגבלת הכלליות בתחומים $[0,N-1]$ ו־$[0,M-1]$ בהתאמה).



איור 1: דגימה מרחבית של תמונה

בכך פתרנו את הבעיה הראשונה, אך הבעיה השנייה עדיין קיימת. הפתרון הוא פעולה בשם *קוונטיזציה* או *כימוי*. נניח כי ברצוננו להקצות לשמירת כל נקודה L סיביות. לכן כל נקודה תוכל לקבל 2^L ערכים שונים מהתחום $[0,1]$. הדרך המקובלת היא ליצג את הערך המקורי $f[m,n]$ על ידי המרה לערך $f^*[m,n]$ כך ש־

$$f^*[m,n] \;=\; round\left(\left(2^L-1\right)f[m,n]\right)$$

המרה כזאת מעבירה ערכים מהתחום הממשי $[0,1]$ לתחום השלם $\{0,1,\ldots,2^L-1\}$.

התמונה הספרתית $f^*[m,n]$ שקיבלנו היא אוסף של MN נקודות השמורה ב־MNL סיביות. על תמונה כזאת כבר ניתן להפעיל את המגוון הרחב של כלים חישוביים הקיימים בימינו.

כאן התייחסנו לתמונה בודדת. סרטים, או אוספים של תמונות, שיש בעיבודם עניין רב בראיה ממוחשבת, גם הם ניתנים לייצוג דומה, רק שהפעם לפונקציה f נוסף ממד נוסף של זמן, כך שלכל יחידת זמן מתאימה תמונה שונה מאחרות. כמובן שאיחסון ועיבוד כמויות כאלה של מידע דורש משאבים רבים הרבה יותר וכח חישוב הרבה יותר חזק מזה הדרוש לעיבוד תמונות בודדות.

1.3 חילוץ מאפיינים

חילוץ המאפיינים (feature extraction) הינו מושג המתייחס לשיטת *הורדת הממד* של המידע המועבד. כאשר הקלט כולל כמות רבה של נתונים יתירים (redundant), רצוי מאוד להמיר אותו לאוסף מצומצם

של תכונות או מאפיינים המספקים את כל המידע המעניין ללא יתירות. השיטה מאוד נפוצה בתחום של הראיה הממוחשבת אשר כאמור עובדת עם תמונות ספרתיות הכוללות הרבה מאוד נתונים, שרובם לא מוסיפים הרבה מידע. בהרבה מערכות המשתמשות בראיה ממוחשבת יש צורך בזיהוי עצמים וצורות בעלות מאפיינים מסוימים בתמונות ספרתיות למטרות שונות, כמו עקיבה, ספירה, השוואה או עיבוד אחר. קיימות הרבה שיטות לחילוץ המאפיינים השונים בתמונות ספרתיות. בעבודה זו נתמקד בשיטות שונות *לזיהוי פינות* בתמונה.

2 מושגים בסיסיים

2.1 הגדרות

הגדרה 2.1 פינה היא מפגש בין שתי שפות (edges) .

לחלופין, ניתן להגדיר פינה כנקודה ממנה יוצאות שתי שפות בשני כיוונים שונים בסביבתה של הנקודה.

הגדרה 2.2 נקודת עניין היא נקודה בתמונה, בעלת מיקום מוגדר היטב, אשר ניתנת לזיהוי באופן מדויק.

כלומר, נקודת עניין יכולה להיות גם פינה או נקודה בעלת מאפיינים מיוחדים אחרים, כגון נקודה מבודדת של מכסימום או מינימום מקומי, קצה של קו וכו'. רוב אלגוריתמי זיהוי הפינות מגלים למעשה נקודות עניין כלליות ולא פינות. לכן, אם נדרש למצוא דווקה פינות ורק פינות, נדרש עיבוד מקומי נוסף של כל אחת מהנקודות שנמצאו.

2.2 מדד ביצועי האלגוריתמים השונים

אלגוריתם טוב לזיהוי פינות צריך לעמוד בקריטריונים הבאים:

- יסמן את כל הפינות האמיתיות
- לא יסמן נקודות שאינן פינות אמיתיות
- המיקומים המסומנים יהיו קרובים ככל האפשר לפינות האמיתיות
- האלגוריתם יהיה חסין ביחס לרעשים בתמונה
- האלגוריתם יהיה יעיל

לא כל האלגוריתמים המוצגים בעבודה זו עומדים בכל הקריטריונים האלה, אך בהרבה מהמקרים ניתן להתפשר על קריטריון מסוים לטובת שיפור קריטריון אחר.

3 האלגוריתמים לזיהוי פינות

3.1 הגישה הכללית

הגישה הכללית בה משתמשים רוב האלגוריתמים לחילוץ מאפיינים ולזיהוי פינות בפרט היא בחלוקת העבודה לשלבים הבאים:

1. חישוב מפת העניין של התמונה, בה לכל נקודה משויך מספר המציין את מידת ההתאמה של הנקודה לנקודות העניין. במקרה של זיהוי פינות מדברים על *מפת הפינתיות (cornerness map)*, המציינת את מידות ההסתברות שהנקודות הן פינות.
2. מציאת נקודות בעלות ערכי הפינתיות הגבוהים ביותר בסביבתן, או במילים אחרות - מכסימום מקומי (local maxima).

האלגוריתמים השונים נבדלים בעיקר בשיטת ביצוע השלב הראשון (חישוב מפת הפינתיות). השלב השני מתבצע לרוב בעזרת שיטת *non-maxima suppression (NMS)*, בה באופן הדרגתי הנקודות בעלות ערך לא מכסימלי בסביבתן מוסרות מהפתרון. ניתן לפתור את הבעיה בשיטות הקלאסיות, על ידי גזירת התמונה בצירים שונים, אבל כאן נציג שיטה מעניינת יותר. האלגוריתם הוצע על ידי Tomasi ו־Kanade ב־[11].

3.2 מציאת מכסימום מקומי בשיטת Tomasi ו־Kanade

האלגוריתם מקבל כקלט את מערך הנקודות עם ערכי הפינתיות שלהן (מפת הפינתיות) ועובד באופן הבא:

1. ממיין את כל הנקודות בסדר היורד של ערכי הפינתיות שלהן.
2. החל מהנקודה הראשונה עובר על המערך ובודק את סביבת כל נקודה.
3. אם בסביבה נמצאה נקודה שכבר שויכה לפתרון (מכסימום מקומי או לא), אז הנקודה הנוכחית מסומנת כלא מכסימום מקומי (כי לפחות נקודה אחת שכבר סומנה נמצאת לפניה במערך).
4. אם אין בסביבה נקודה שכבר סומנה, מסמנים את הנקודה הנוכחית כמכסימום מקומי.

באופן כללי, זמן ריצת האלגוריתם הוא $O\left(N^2\right)$, כאשר N הוא מספר הנקודות הכולל. לכן עבור תמונות גדולות האלגוריתם עלול לרוץ זמן בלתי סביר. על מנת לשפר את זמן הריצה, בדרך כלל מצמצמים את מספר הנקודות במערך על ידי הורדת כל הנקודות הנמוכות מסף מסוים (*thresholding*). האלגוריתם 1 מממש את השיטה בצורה המותאמת ל־MATLAB, כלומר עושה שימוש בפעולות מובנות על מטריצות ומערכים.

```
2
% This function takes the input MxN matrix 'inp' and returns
% the local maxima 'label map' L performing thresholding with value
% 'threshold', while checking a window of size 'neighbourhood x 2 + 1'
function [L] = nms_tk(inp,threshold, neighbourhood)

[M,N] = size(inp);     % 'inp' has M rows and N columns

% Reshape the picture to 1D vector, scanning the 'inp' COLUMNWISE.
R = reshape(inp, M*N, 1);  % R is the 1D reshaped vector

% S - the sorted values vector,
% I - the corresponding indices
[S, I] = sort(R, 'descend');

% Now calculate the row and column numbers in the sorted order
P = mod(I-1, M) + 1;           % Sorted row numbers (the first index is 1)
Q = floor( (I-1) / M + 1 );  % Sorted column numbers (the first index is 1)

% The solution vector. Initially there is no points in solution
% e.g.set to '-1'
solution = zeros(length(P), 1) - 1;  % "-1"'s vector (zeros - 1)

% iterate through the points starting with the largest value
for cnt = 1: length(P)

    if (S(cnt) <= threshold)
        break; % exit the loop if reached points
               % below the threshold
    end

    % p, q are the coordinates of the current point
    p = P(cnt); q = Q(cnt);
    % delta_p/q are the vectors holding the horizontal and
    % vertical distances from the current point to the others,
    % located after the point in the vector
    delta_p = abs( P(:) - p ); % Subtract p from each point value in P
    delta_q = abs( Q(:) - q ); % Subtract q from each point value in Q
    % 'select' marks the points (with current index offset) of
    % located within the 'neighbourhood'
    select =( (delta_p <= neighbourhood) & ...
                    (delta_q <= neighbourhood) );

    % set current point within the solution if there is no
    % neighbour that is already set (that is, has greater value)
    if any(solution(select) ~= -1)
        solution(cnt) = 0;
    else
        solution(cnt) = 1;
    end

end
% Find only the coordinates of the solution
solution = (solution == 1);
P=P(solution);
Q=Q(solution);

% Build the label map
L = zeros(M,N);
for x=1:length(P), L(P(x),Q(x))=1; end;
```

3.3 מציאת מכסימום מקומי בשיטה מקבילית

השיטה המוצגת בסעיף זה הוצעה ב־[9] ועוסקת בהתאמת אלגוריתם ה־ NMS אשר הוצג בסעיף הקודם למעבדים גרפיים המותאמים במיוחד לחישובים מקביליים על מערכי נתונים גדולים. מעבדים אלה (הידועים כ־ *GPU - Graphics Processing Units*) מתפתחים במהרה בשנים האחרונות ומגיעים לביצועים מרשימים בעיבודי התמונה וגרפיקה ממוחשבת. האלגוריתם עובד במספר שלבים, כאשר קלט של כל אחד מהשלבים הוא הפלט של השלב הקודם. הפלט הסופי הוא *מפת התוויות (label map)* המסמנת עבור כל אחת מהנקודות, האם היא חלק מהפתרון או לא (inside או outside במונחים של [9]). כמו כן במהלך ריצת האלגוריתם נקודה יכולה להיות ללא תוית אם שיוכה לפתרון טרם נקבע (undefined). בכל אחד מהשלבים כל הנקודות מועבדות באופן מקבילי ומפת התוויות מתעדכנת בהתאם. שיטת החישובים המקביליים של GPU מבוססת על כך שכל אחד מהחישובים המתבצעים על נקודה מסוימת ממערך הקלט יכול לגשת לכל נקודה אחרת של הקלט, אך תוצאת החישוב יכולה להכתב רק בנקודה המתאימה בפלט. בכל אחד מהשלבים, החישוב מתבצע על בסיס מפת הפינתיות ומפת התויות מהשלב הקודם, כאשר במפת התויות ההתחלתית כל אחת מהנקודות מסומנות כ־undefined. החישוב מתבצע באופן הבא:

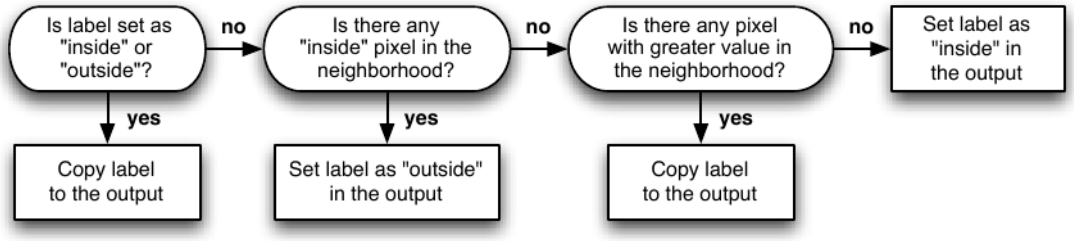
1. אם הנקודה כבר מסומנת בתור inside או outside, התוית מועתקת לפלט.
2. אחרת מתבצעת בדיקה של סביבת הנקודה, ואם יש בה נקודות שמסומנות בתור inside, מסמנים את הנקודה כ־outside.
3. אם אין בסביבתה נקודות שהן inside, בודקים האם יש בסביבה נקודות עם ערך פינתיות גבוה יותר.
4. אם אין, מסמנים את הנקודה בתור inside
5. אחרת משאירים את התוית כ־undefined
6. מסיימים כאשר לא נשארו נקודות לא מסומנות.

באיור 2 מתואר תרשים הזרימה שנלקח מ־[9]. כמו כן במאמר זה מוכחת נכונות האלגוריתם. אלגוריתם 2 מציג סימולציה של השיטה הזאת בעזרת MATLAB.

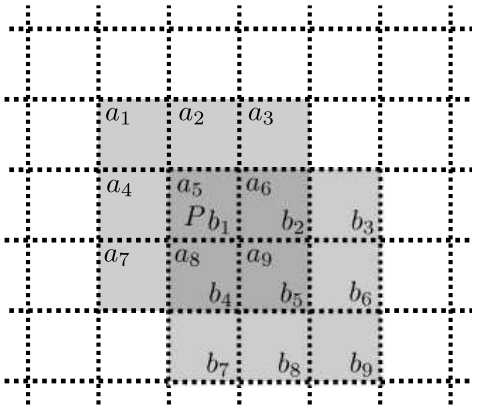
3.4 האלגוריתם של Moravec

אחד האלגוריתמים הראשונים לזיהוי פינות הוצג על ידי האנס מורבץ (Hans P. Moravec) מאוניברסיטת קרנגי־מלון בארה"ב ב־[1, 2]. מורבץ מגדיר את מושג "נקודות העניין" כאזורים בתמונה הניתנים להבחנה, ומסיק כי בעזרתן ניתן למצוא אזורים מתאימים ברצפים של תמונות (למשל לצורך עקיבה אחרי עצמים שונים).

```
2
function [L] = nms_tcg(inp,threshold, NH)

[M,N] = size(inp);
% L is the label map. '0' is for 'outside' point, '1' for inside,
% '-1' is for undefined. The initial values are all 'undefined'
% except to the "border" of width NH that always 'outside'
L = zeros(M,N);
L(1+NH:M-NH, 1+NH:N-NH) = -ones(M-NH*2,N-NH*2);      %

% Perform thresholding to reduce the run time
I =  ~(inp > threshold); %I is boolean matrix with ones for every point below
  the treshold
L(I)  = 0; % mark all of the points below the threshold as outside

cont = 1;  % flag indicating the algorithm should run another pass
while cont
     %perform single algorithm pass (see SinglePass subfunction)
    [L,cont] = SinglePass(inp, L, NH);
end

% Subfunction implementing single algorithm pass
% Returns updated label map and flag indicating that there is another pass needed
function [Lout, cont] = SinglePass(inp, Lin, NH)

% find indices of all undefined points
[P,Q] = find(Lin < 0);
cont = ~isempty(P); % cont gets 1 if there is undefined points
Lout = Lin;
for cnt = 1:length(P),
    p=P(cnt) ; q=Q(cnt);
    % Obtain a neighbourhood window of the current point's labels
    W_L = Lin(p-NH : p+NH , q-NH : q+NH);
    if any(any(W_L == 1))
     % some of the neighbour points is 'inside'
        Lout(p,q) = 0; % mark the point as 'outside'
    else
         % Obtain a neighbourhood window of the current point's values
        W_Inp =  inp(p-NH : p+NH , q-NH : q+NH);
        [I,J] = find(W_Inp > inp(p,q)); % Find neighbours with greater values
        if isempty(I)
            [I,J] = find(W_Inp == inp(p,q));              % Find neighbours with
              same value
            if (I(1) > NH && J(1) > NH)
            % the point has the maximal value within it's neighbourhood
            % and has a precendence before other equal values
                Lout(p,q) = 1;
            else
                % There is a point with equal value, but with higher precendence,
                % so copy it's label to output
                Lout(p,q) = W_L(I(1),J(1));
            end
        else
          % there is some of neighbours greater than the point
          % so copy their label to output
         Lout(p,q) = W_L(I(1),J(1));
        end
    end
end
```



איור 2: תרשים זרימה של מעבר בודד באלגוריתם NMS מקבילי



איור 3: האופרטור של מורבץ

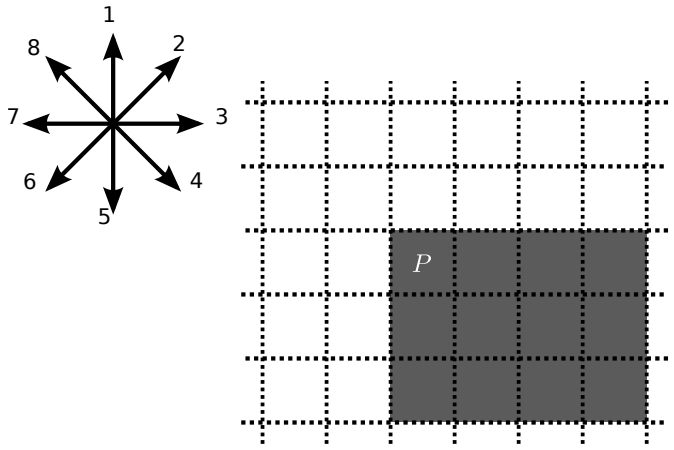
האלגוריתם נחשב לאלגוריתם המזהה פינות כי נקודות העניין אותן הוא מזהה הן הנקודות עם שונות גבוהה בבהירות בכיוונים שונים, שזה המצב בעיקר בנקודות שהן פינות.

מורבץ הציע למדוד באופן הבא את השונות (אותה נסמן ב־V) בנקודה P מסוימת של התמונה. מגדירים "חלון" ריבועי קטן (בדרך כלל בגודל 3×3 עד 7×7) סביב הנקודה P, ואז "מזיזים" את החלון בשמונה כיוונים שונים (אופקי, אנכי ושני האלכסונים) בצעד של נקודה אחת ואז מסכמים את ריבועי ההפרשים בין הנקודות המתאימות בחלון המקורי והמוזז. השונות של הנקודה P היא השונות *המינימלית* בין המחושבות עבור כל אחד מהכיוונים. נתבונן באיור 3. באיור מוצגת דוגמא להפעלת האלגוריתם על נקודה P עם חלון של 3×3. ב־a_i מסומנות רמות האפור של הנקודות בחלון שסביב P וב־b_i - רמות האפור המתאימות בחלון המוזז בכוון אלכסוני.

השונות, כפי שהוגדרה, עבור ההזזה הזאת היא

$$V = \sum_{i=1}^{9} (a_i - b_i)^2$$

על מנת להבין, למה הפעולה הנ"ל מזהה דווקה פינות, נתבונן בהתנהגות האלגוריתם במספר מקרים.



איור 4: נקודה P שהיא פינה אמיתית

לצורך פשטות נניח כי התמונה מורכבת מנקודות בעלות שתי רמות אפור בלבד - לבן ושחור, המסומנות ב־1 ו־0 בהתאמה. תחת הגדרה כזאת, ערך השונות בכל אחת מההזזות יהיה שווה למספר הנקודות בחלון ש"החליפו" צבע. במקרה של אזור תמונה אחיד הערכים של נקודות בחלון המקורי וכן בחלון המוזז יהיו זהים, ולכן עבור תמונה כזאת ערך השונות יהיה $V = 0$. במקרה בו הנקודה P נמצאת על שפת אזור אחיד, אך לא בפינה, השונות בכיוונים מקביליים לשפה תהיה אפס, והשונות בכוונים אנכיים תהיה חיובית. לכן השונות V המינימלית גם הפעם שווה ל־0.

במקרה שהנקודה P נמצאת אכן בפינה של אזור אחיד כמו שמתואר באיור 4, השונות תהיה $V = \min\{V_i\}_{i=1}^{8}$, כאשר V_i הן השונויות המתקבלות בהזזת חלון בכוון i כפי שמסומן באיור, והערכים שלהן

$$\begin{aligned} V_1 = V_3 = V_5 = V_7 &= 2 \\ V_2 = V_6 = V_8 &= 3 \\ V_4 &= 5 \end{aligned}$$

ולכן $V = 2$.

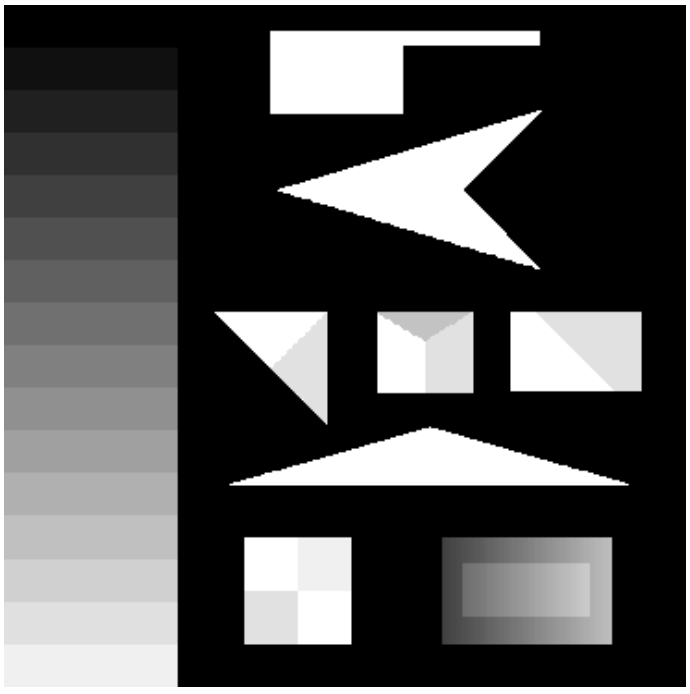
השונות הגבוהה ביותר תתקבל במקרה בו הנקודה P היא נקודה מבודדת (כלומר כל הסביבה שלה מורכבת מנקודות בצבע שונה). במקרה כזה בכל הזזה של חלון 3×3 שתי נקודות בחלון ישנו צבע, ולכן בכל כיוון השונות שתתקבל תהיה זהה ושווה ל- $V = 2$.

בצורה כזאת האופרטור של מורבץ יכול לתת את מידת *הפינתיות* ($cornerness$) לכל נקודה בתמונה נתונה. נתבונן ב*מפת הפינתיות* ($cornerness\ map$) של תמונה פשוטה באיור 5. כאן בתוך כל נקודה מסומן הערך המחושב עבורה על ידי האופרטור. ב־X מסומנות הנקודות הנמצאות על שפת התמונה, עבורן הזזת החלון תגרום לחריגה מחוץ לתמונה.

איור 5: מפת פינתיות

כפי שניתן לראות, במפת הפינתיות המתקבלת לאחר הפעלת אופרטור מורבץ, הפינות מסומנות בערך *מכסימום מקומי (local maximum)*. נקודות מבודדות גם הן מסומנות במכסימום מקומי, ולכן אם אנחנו לא מעוניינים בגילוי נקודות כאלה, נשתמש בחלונות גדולים יותר מ־3×3, כך שערך הפינתיות שלהן יהיה קטן יותר מזה של פינות אמיתיות, ונוכל לסנן אותן על ידי הצבת סף (threshold) על ערך הפינתיות. האלגוריתם 3 מממש את האופרטור של מורבץ ב־MATLAB.

על מנת לבדוק את ביצועי האלגוריתם, נשתמש בתמונה מלאכותית המוצגת באיור 6.



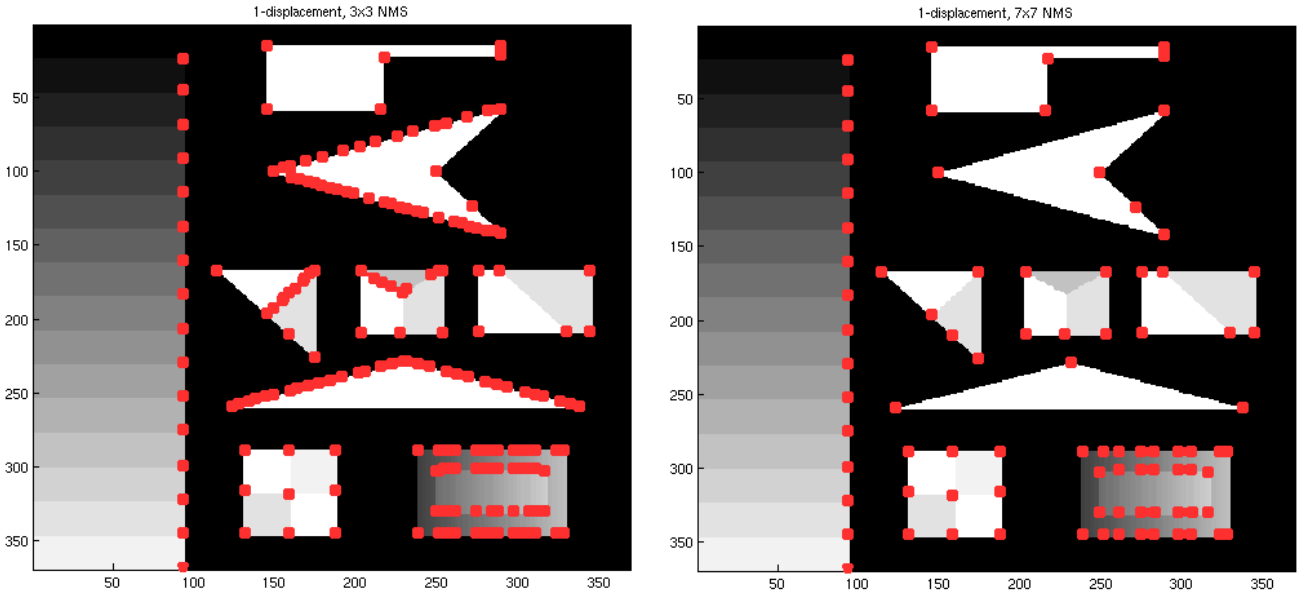
איור 6: תמונה לבדיקת אלגוריתמי זיהוי פינות

באיור 7 מתוארת תוצאת הרצת האלגוריתם על תמונת הבדיקה עם הזזת התמונה ב־1 ושימוש באלגוריתם

2

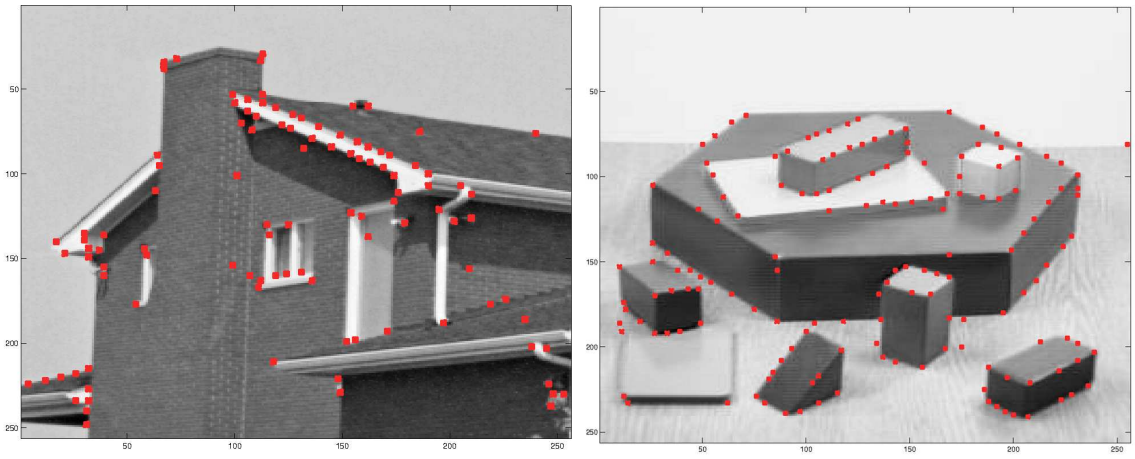
```
% This function implements the Moravec algorithm. It takes
% the image, and the displacement amount (how much points
% to shift the image) and returns the 'cornerness' map/
function [Map] = moravec(Image, disp)

% The next matrices are kernels for calculating differences
%  between two points with given image displacement
% 'displ' in each direction.
%  hor = [ 1 0 ... -1] of length displ+1
%  ver is transposed hor
% diag1 is (disp+1) by (disp+1):
%       [ 0 0 ...  1
%         0 0 ...  0
%           .......
%        -1 0 .....0]
%
% diag1 is (disp+1) by (disp+1):
%       [ 1 0 ...  0
%         0 0 ...  0
%           .......
%         0 0 ....-1]

hor = zeros(1, disp + 1);
hor(1) = 1;
hor(end) = -1;
ver = hor';
diag1 = zeros(disp+1, disp+1);
diag1(1,end) = 1;
diag1(end, 1) = -1;

diag2 = zeros(disp+1, disp+1);
diag2(1,1) = 1;
diag2(end, end) = -1;

Img = double(Image); % Convert to double data type for calculations

% Perform filter operation (correlation) with each one of the
% difference kernels and return matrices with same size as image
D1 = imfilter(Img, hor,'corr','same' ).^2; % Horizontal variances
D2 = imfilter(Img, ver,'corr','same' ).^2; % Vertical variances
D3 = imfilter(Img, diag1,'corr','same' ).^2; % Diagonal variances 1
D4 = imfilter(Img, diag2,'corr','same' ).^2; % Diagonal variances 2

[M,N] = size(D1);
V=zeros(1,4);        % Will store the variance in each direction
Map = zeros(M,N);    % will store the result map

% Iterate through the points and calculate the variances
for m = 1+disp: 1 : M-disp ,
    for n = 1+disp : 1 : N-disp ,
        V(1) = sum( sum( D1(m-1:m+1, n-1:n+1) ));
        V(2) = sum( sum( D2(m-1:m+1, n-1:n+1) ));
        V(3) = sum( sum( D3(m-1:m+1, n-1:n+1) ));
        V(4) = sum( sum( D4(m-1:m+1, n-1:n+1) ));
        Map(m,n) = min(V);
    end
end
```

non-maxima suppression עם גודל החלון של 3×3 ו־7×7.



איור 7: תוצאות הרצת אלגוריתם Moravec על תמונת בדיקה

מהאיור של ההרצה עם NMS מעל חלון של 3×3 ניתן לראות כי האלגוריתם מזהה נכון את כל הפינות בתמונה המלאכותית. אבל בנוסף לפינות האמיתיות האלגוריתם מסמן גם נקודות שאינן פינות. נקודות אלה הן בעיקר הנקודות הנמצאות על השפות שהן לא בכיוונים בהם מבוצעת הזזת התמונה באלגוריתם, ולכן האלגוריתם הוא *אניזוטרופי (anisotropic)*, כלומר תלוי כיוון. ניתן לשפר את התוצאות על ידי הגדלת החלון של אלגוריתם ה־NMS ל־7×7, אך משמעות הדבר תהיה שהאלגוריתם יחמיץ חלק מהפינות שיהיו קרובות לפינות אחרות. בנוסף הדבר מייקר את החישובים ולא מעלים את כל התוצאות הלא נכונות. תוצאות הרצה על תמונות בדיקה נוספות ניתן לראות באיור 8. ההרצות בוצעו עם הזזה של נקודה אחת, חלון NMS בגודל 7×7 וערכי הסף שהותאמו באופן אמפירי.



איור 8: תוצאות הרצת אלגוריתם Moravec על תמונות אמיתיות

3.5 אלגוריתם Harris & Stephens

האופרטור של מורבץ מחשב את השונויות של הנקודות במספר סופי וקטן של הזזות החלון (כלומר בשמונת הכיוונים העיקריים), ולכן תגובתו היא אניזוטרופית כפי שראינו. על מנת להתגבר על המגבלה הזאת, הריס וסטפנס ב־[3] הציעו נוסחה המאפשרת להתחשב בשונות הנקודה בכל הכיוונים האפשריים. אם נתבונן באופרטור של מורבץ, נראה כי ניתן לרשום את הערך המחושב על ידיו עבור הזזה בכל אחד מהכיוונים באופן הבא

$$E(u,v) = \sum_{\forall(x,y)} w(x,y) \left[I(x+u, y+v) - I(x,y) \right]^2 \qquad (1)$$

כאשר $I(x,y)$ - רמת האפור בנקודה (x,y) ו־(u,v) - ההזזה עבורה מחושב הערך. ההזזה בגודל של צעד אחד בכיוון אופקי היא $(\pm 1, 0)$, בכיוון אנכי - $(0, \pm 1)$, ובאלכסונים - $(\pm 1, \pm 1)$ ו־$(\pm 1, \mp 1)$.

$w(x,y)$ - פונקציית החלון הדו־ממדי אשר בדרך כלל המקבלת ערכים שונים מ־0 אם ורק אם $|u-x| \leq S$ וגם $|v-y| \leq S$, כאשר $(2S+1)$ הוא גודל החלון. על מנת לקבל תגובה *אניזוטרופית* הריס וסטפנס השתמשו בחלון גאוסי מעגלי הנתון על ידי

$$w(x,y) = \begin{cases} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) & , x^2+y^2 \leq S^2 \\ 0 & , otherwise \end{cases}$$

כאשר כאן s הוא רדיוס החלון המעגלי.

אם נבטל את ההגבלה על הערכים של u ו־v כערכים דיסקרטיים ונתייחס אליהם כערכים רציפים אזי הפונקציה הנ"ל תתן את ערכי השונות בכל כיוון אפשרי. על מנת לחשב את הפונקציה באופן מעשי, נקרב את הגורם $I(x+u, y+v)$ על ידי פירוקו לטור טיילור

$$I(x+u, y+v) \approx I(x,y) + I_x(x,y)\,u + I_y(x,y)\,v \tag{2}$$

כאשר I_x ו־I_y הן הנגזרות החלקיות של I. אם נציב את (2) ב־(1) נקבל

$$E(u,v) \approx \sum_{x,y} w(x,y)\left[I_x(x,y)\,u + I_y(x,y)\,v\right]^2$$

נשמיט את (x,y)לצורך פשטות הרישום ונקבל

$$\begin{aligned} E(u,v) &\approx \sum_{x,y} w\left(I_x u + I_y v\right)^2 = \\ &= \sum_{x,y} w\left(I_x^2 u^2 + 2I_x I_y uv + I_y^2 v^2\right) = \\ &= u^2 \sum_{x,y} w I_x^2 + 2uv \sum_{x,y} w I_x I_y + v^2 \sum_{x,y} w I_y^2 = \\ &= Pu^2 + 2Quv + Rv^2 \end{aligned} \tag{3}$$

כאשר

$$\begin{aligned} P &= I_x^2 \otimes w \\ Q &= (I_x I_y) \otimes w \\ R &= I_x^2 \otimes w \end{aligned}$$

כאן האופרטור $\otimes$ הוא אופרטור הקונבולוציה.

את 3 ניתן לרשום בצורה ביליניארית (bilinear form)

$$E(u,v) = \begin{pmatrix} u & v \end{pmatrix} A \begin{pmatrix} u \\ v \end{pmatrix}$$

כאשר A נתונה על ידי

$$A = \begin{bmatrix} P & Q \\ Q & R \end{bmatrix} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix}$$

כאשר הסוגריים הזויתיים מסמנים סכום משוקלל על x, y עם המשקלים הנתונים על ידי w (קונבולוציה עם החלון w). המטריצה הזאת ידועה גם בשם מטריצת הריס (Harris matrix). עם בתור החלון w ניקח חלון מעגלי כגון גאוסיאן, אז תגובת האופרטור תהיה *איזוטרופית*, כלומר בלתי תלויית כיוון.

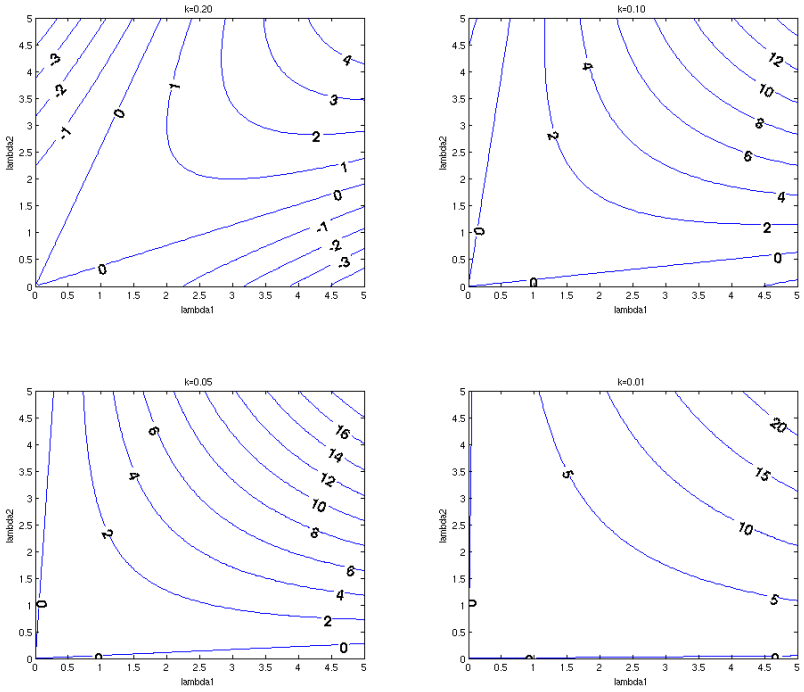
פינה (או נקודת עניין ככלל) מאופיינת על ידי שונות גדולה של E בכל הכיוונים של הוקטור (u, v). את הקריטריון הזה ניתן לאפיין במושגים של הערכים העצמיים של A. הריס וסטפנס מראים [3] כי הערכים העצמיים של A צריכים להיות גדולים עבור נקודות העניין ביחס לנקודות אחרות. לפי [3] אם λ_1, λ_2 הם הערכים העצמיים של A אזי מתקיים

- אם $\lambda_1 \approx 0$ ו־$\lambda_2 \approx 0$ אז הנקודה אינה נקודת עניין;
- אם $\lambda_1 \approx 0$ ו־$\lambda_2 \gg 0$ אז הנקודה נמצאת על שפה;
- אם $\lambda_1 \gg 0$ וגם $\lambda_2 \gg 0$ אז הנקודה היא פינה.

החישוב המדויק של הערכים העצמיים של A יכול להיות תהליך מסובך ויקר מבחינת זמן, ולכן הריס וסטפנס מציעים לחשב מדד M_c על ידי חישוב הדטרמיננטה והעקבה (trace) של A:

$$M_c = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 = \det(A) - k\,\mathrm{trace}(A)^2 \qquad (4)$$

כאשר k הוא פרמטר הקובע את "רגישות" האופרטור וערכו נקבע באופן אמפירי (הערכים הרגילים הם בין 0.04 ל־0.15). M_c מקבלת ערכים גבוהים עבור λ_1 ו־λ_2 גדולות וערכים נמוכים יחסית כאשר לפחות אחת מ־λ_1 או λ_2קטנה. באיור סססס אפשר לראות התנהגות M_c כפונקציה של λ_1 ו־λ_2 בצורה של קוי גובה של הפונקציה.



איור 9: M_c כתלות ב־$\lambda_{1,2}$עבור ערכי k שונים

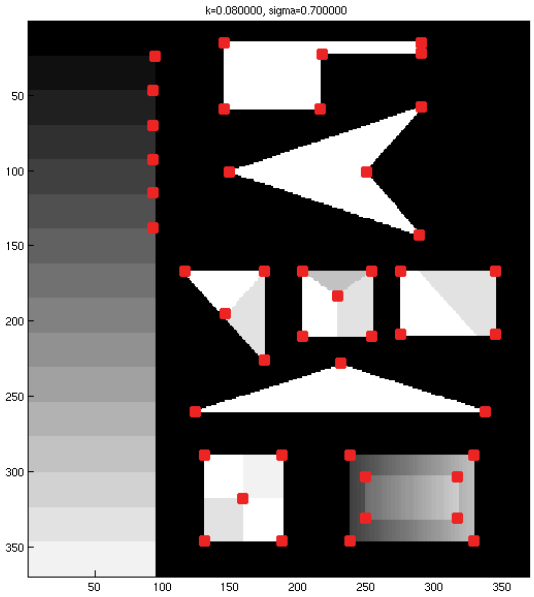
```
2
function Map = harris(Image, radius, sigma, k)

I = double(Image);
% The differentiation kernel
dx = [-1 0 1; -1 0 1; -1 0 1];
dy = dx';    % dy kernel is transposed dx kernel

% Gradients calculation
Ix = conv2(I , dx, 'same');
Iy = conv2(I , dy, 'same');

G = fspecial('gaussian',radius, sigma); % Gaussian Filter

% Elements of the structure tensor matrix A
Ix2 = conv2(Ix.^2, G, 'same');   % <Ix^2>
Iy2 = conv2(Iy.^2, G, 'same');   % <Iy^2>
Ixy = conv2(Ix.*Iy, G,'same');   % <Ix*Iy>

% M = det(A) - k*trace(A)^2
Map = (Ix2.*Iy2 - Ixy.^2) - k*(Ix2+Iy2).^2;
```

אלגוריתם (4) מממש את השיטה המתוארת עם חלון גאוסי בעל רדיוס ושונות נתונים.

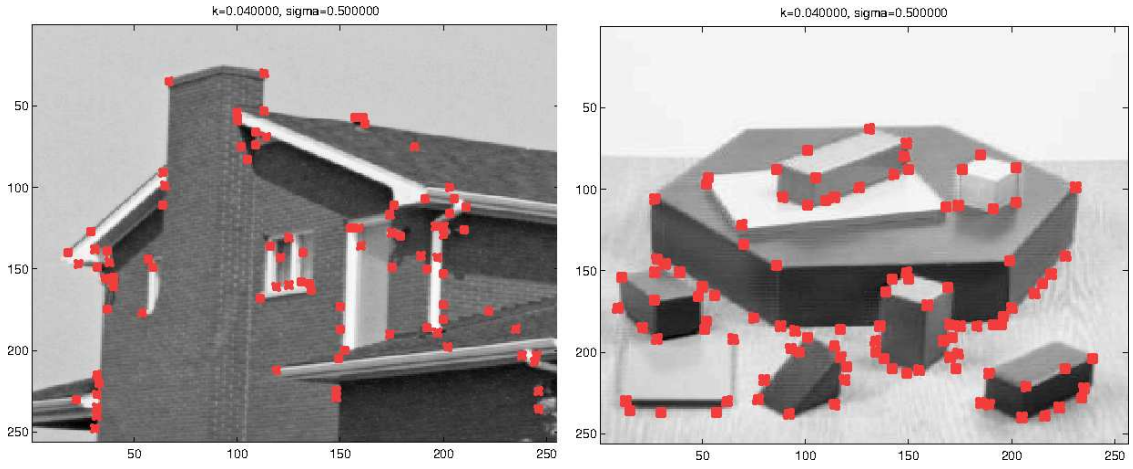
את תוצאות הרצת האלגוריתם (המועברת דרך אלגוריתם ה־NMS) ניתן לראות באיור 10. האלגוריתם הורץ עם הפרמטרים $k = 0.08$ ו־$\sigma = 0.7$ (השונות של הגאוסיאן) ורדיוס הגאוסיאן של 3. האלגוריתם מזהה את הנקודות שהן פינות אמיתיות, אך לא את כולן. ניתן לשנות את התנהגות האלגוריתם על ידי שינוי הפרמטרים השונים, אבל לכל תמונה הפרמטרים האופטימליים יהיו אחרים.

בין החסרונות של האלגוריתם:

- החישובים מסובכים ויקרים מבחינת זמן ומשאבים. פעולת הקונבולוציה עם חלון גאוסי היא פעולה כבדה.
- רגישות לרעשים בתמונה. האלגוריתם משתמש בגרדיאנטים של התמונה. אם למשל קיימת נקודה בתוך עצם שחור בתמונה, אשר כתוצאה מהרעש הפכה להיות בהירה יותר, הגרדיאנטים שלה לכל הכיוונים יהיו גדולים, והיא תחשב כנקודת עניין. ניתן להגביר את החסינות לרעש של האלגוריתם על ידי הגדלת החלון הגאוסי, כך שיותר נקודות ילקחו בחשבון והשפעת נקודה בודדת תקטן, אך הדבר יגדיל את סיבוכיות החישוב ואת זמן הריצה.
- מכיוון שהאיברים של מטריצת הקורלציה העצמית A מחושבים על בסיס גרדיאנטים של התמונה בשני כיוונים בלבד, תגובת האופרטור עדיין אניזוטרופית.



איור 10: תוצאות ההרצה של אלגוריתם הריס־סטפנס



איור 11: תוצאות ההרצה של אלגוריתם הריס־סטפנס על תמונות אמיתיות

3.6 אלגוריתם Shi & Tomasi

בשנת 1994 ג'יאנבו שי וקרלו טומזי מאוניברסיטת סטנפורד (Jianbo Shi & Carlo Tomasi) הציעו שינוי לאלגוריתם הריס והציגו שיפור בביצועים ב־[4]. שי וטומזי השתמשו במדד הפינתיות שונה מזה של הריס, אך מבוסס גם הוא על מטריצת האוטוקורלציה A של הריס. במקום המדד M_c המתואר ב־(4) הם

```
2
function Map = shi_tomasi(Image, radius, sigma)

I = double(Image);
% The differentiation kernel
dx = [-1 0 1; -1 0 1; -1 0 1];
dy = dx';    % dy kernel is transposed dx kernel

% Gradients calculation
Ix = conv2(I , dx, 'same');
Iy = conv2(I , dy, 'same');

G = fspecial('gaussian',radius, sigma); % Gaussian Filter

% Elements of the structure tensor matrix A
Ix2 = conv2(Ix.^2, G, 'same');   % <Ix^2>
Iy2 = conv2(Iy.^2, G, 'same');   % <Iy^2>
Ixy = conv2(Ix.*Iy, G,'same');   % <Ix*Iy>

% eigenvalues calculation
detA = (Ix2.*Iy2 - Ixy.^2);
trA = (Ix2+Iy2);
lambda1 = (trA+sqrt(trA.^2-4*detA)) / 2;
lambda2 = (trA-sqrt(trA.^2-4*detA)) / 2;

% The cornerness map
Map = min(lambda1,lambda2);
```

הגדירו מדד חדש

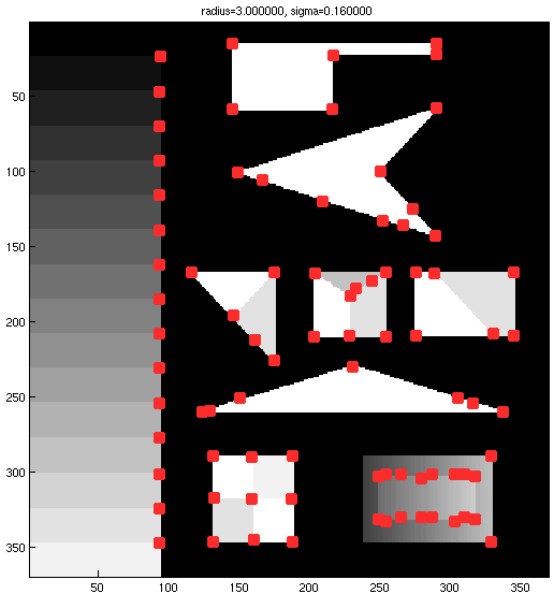
$$M_{st} = \min(\lambda_1, \lambda_2) \tag{5}$$

כאשר λ_1, λ_2 הם הערכים העצמיים של המטריצה A הניתנים לחישוב על ידי

$$\lambda_{1,2} = \frac{\text{trace}(A) \pm \sqrt{\text{trace}(A)^2 - 4\det(A)}}{2} \tag{6}$$

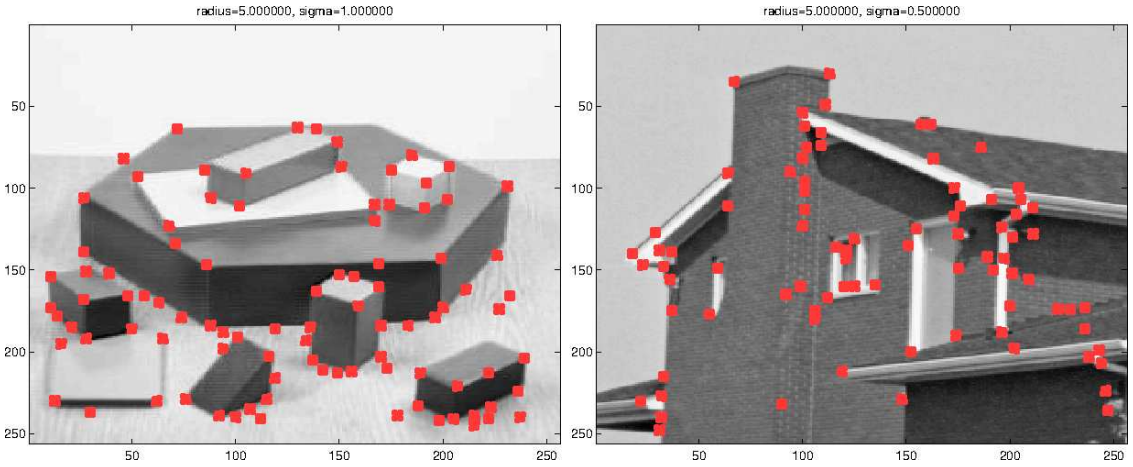
האלגוריתם (5) מתאר את מימוש השיטה ב-MATLAB.

באיור 12 ניתן לראות את תוצאות ההרצה עם פרמטרים $radius = 3, \sigma = 0.16$. האלגוריתם מזהה את כל הפינות בתמונה חוץ משתי הפינות של הצורה התחתונה מימין. ובנוסף לכך גם מספר נקודות הממוקמות על שפות אלכסוניות וכן על שפות בין איזורים עם רמות אפור המשתנות באופן הדרגתי.



איור 12: תוצאות ההרצה של אלגוריתם שי־טומזי

את הביצועים של האלגוריתם על תמונות אמיתיות ניתן לראות באיור 13.



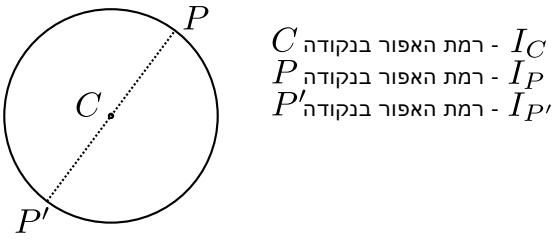
איור 13: תוצאות ההרצה של אלגוריתם שי־טומזי על תמונות אמיתיות

3.7 האופרטור של טרייקוביץ'

3.7.1 עקרונות האלגוריתם

האלגוריתם הבא פותח על ידי מירוסלב טרייקוביץ' ומרק הדלי (Miroslav Trajkovic & Mark Hadley) בשנת 1998 והוצג ב־[5]. מטרתם הייתה לקבל ביצועים קרובים לאלה של אלגוריתמי זיהוי פינות הנפוצים, תוך דרישת כח חישוב קטן יחסית. אופרטור טרייקוביץ' עושה שימוש באותה הגדרת פינה כמו מורבץ - פינה היא נקודה בה השינוי בבהירות הוא גדול בכל הכיוונים. כמו כן, גם בדומה למורבץ מידת הפינתיות נקבעת על ידי השינוי המינימלי מבין כל הכיוונים האפשריים. הביצועים של האלגוריתם טובים מאלה של מורבץ הודות לשיטת הקירוב של שינויי הבהירות בכל כיוון האפשרי לעומת מספר סופי של כיוונים של מורבץ.

על מנת להעריך את מידת הפינתיות של נקודה, האלגוריתם משתמש בחלון מעגלי קטן ומחשב שינויי רמות האפור לאורך כל הקווים העוברים דרך נקודת המרכז. נקודת המרכז (שזו היא בעצם הנקודה, עבורה מחושב ערך הפינתיות) מסומנת באות C. נתבונן בקו העובר דרך C וחותך את המעגל בנקודות P ו־P'. נסמן את רמת האפור בנקודה X ב־I_x (איור 14).



איור 14: סימונים עבור האופרטור של טרייקוביץ'

במונחים אלה מידת הפינתיות של הנקודה C לפי טרייקוביץ' נתונה על ידי

$$M_{tr}(C) \quad = \quad \min\left\{(I_P - I_C)^2 + (I_{P'} - I_C)^2 \mid \forall P, P'\right\}$$

נבחן את התנהגות הפונקציה M_{tr} במקרים שונים:

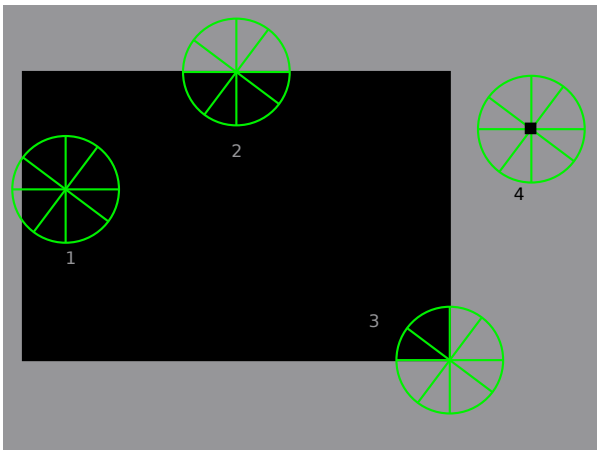
1. הנקודה C נמצאת בתוך אזור אחיד, כך שרוב שטח החלון נמצא גם הוא באזור האחיד. במקרה כזה קיים לפחות קטע PP' אחד, עבורו הנקודות P ו־P' יהיו עם רמות אפור קרובות לזו של הנקודה C (כלומר שינוי ברמת האפור לאורך הקטע יהיה קטן או אפסי). כלומר M_{tr} במקרה כזה תקבל ערך קרוב ל־0.

2. הנקודה C נמצאת על שפת אזור אחיד (נניח השפה היא קו ישר). במקרה כזה יהיה בדיוק קטע PP' אחד לאורכו רמת האפור לא משתנה. קטע זה עובר לאורך שפת האזור. כלומר גם במקרה כזה M_{tr} תהיה קרובה ל־0.

3. הנקודה C נמצאת על פינת האזור האחיד. במקרה כזה עבור כל קטע PP' לפחות אחד מ־I_P או $I_{P'}$ יהיה שונה מ־I_C ולכן ערך M_{tr} תקבל ערך גבוה.

4. הנקודה C היא נקודה מבודדת. במקרה זה עבור כל קטע PP' הערכים של I_P ו־$I_{P'}$ יהיו שונים מזה של I_C, ולכן M_{tr} תקבל ערך גבוה.

המקרים השונים מתוארים באיור 15.



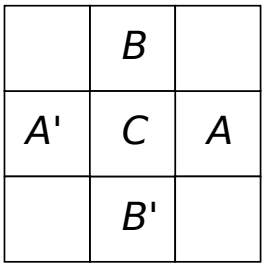
איור 15: מקרים שונים עבור אופרטור טרייקוביץ'

מהניתוח הנ"ל ניתן להסיק כי האופרטור יעבוד בצורה טובה רק על תמונות יחסית נקיות מרעשים, כי כל נקודה מבודדת הנובעת מרעשים כלשהם תפורש כנקודת עניין עם ערך פינתיות גבוה. באופן מעשי ניתן להקטין רעשים על ידי אופרטור טשטוש גאוסי.

3.7.2 חישוב מעשי

הפונקציה M_{tr} לא ניתנת לחישוב מדויק כי יש צורך לחשב ערכים עבור מספר אינסופי של נקודות P ו־P'. לכן משתמשים בשיטת קירוב *Interpixel Approximation* עם 4 שכנים או עם 8 שכנים. כאן נציג את שיטת הקירוב עם 4 שכנים.

נניח C היא הנקודה עבורה יש לחשב את ערך הפינתיות $M_{tr}(C)$. נניח גם כי הנקודות A, A', B, B' הן הנקודות השכנות ל־C כפי שמתואר באיור 16.



איור 16: מיקום הנקודות

ערכי השונות לאורך הקו האופקי (r_A) והאנכי (r_B) ניתנים לחישוב פשוט

$$\begin{aligned} r_A &= (I_A - I_C)^2 + (I_{A'} - I_C)^2 \\ r_B &= (I_B - I_C)^2 + (I_{B'} - I_C)^2 \end{aligned}$$

ומכאן ניתן לקבל את הקירוב הראשוני של M_{tr}

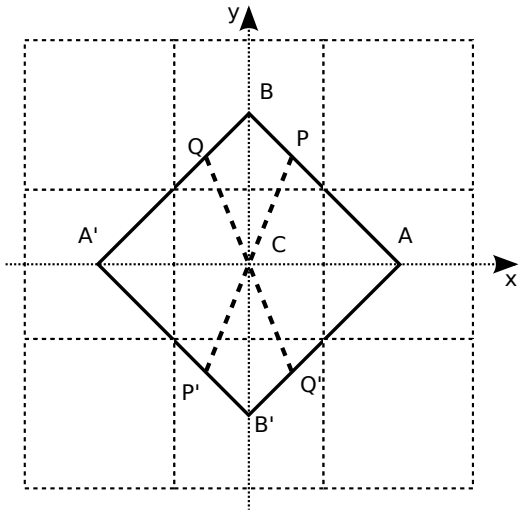
$$M_{simple} = min\,(r_A, r_B)$$

ערך זה מאפשר פסילה מהירה של נקודות בשלבים ראשונים של עבודת אלגוריתם זיהוי.

שיטת הקירוב *Interpixel Approximation* משתמשת בחישוב של נקודות לאורך היקף של ריבוע $ABA'B'$ כפי שמתואר באיור 17 ולא לאורך מעגל. נגדיר שתי נקודות P ו־Q לאורך הקטעים AB ו־$A'B$ בהתאמה, וכן שתי נקודות המתאימות להן P' ו־Q' על הקטעים $A'B'$ ו־AB' בהתאמה כפי שמתואר באיור 17. נגדיר גם פרמטר נוסף t אשר יקבע את מיקום הנקודות, כך שיתקיימו התנאים הבאים:

- **אם $t = 0$ אזי $P = Q' = A$ וגם $P' = Q = A'$**
- **אם $t = 1$ אזי $P = Q = B$ וגם $P' = Q' = B'$**
- **כאשר $t = 0.5$, הנקודה P נמצאת באמצע הקטע AB, הנקודה P' - באמצע $A'B'$, הנקודה Q- באמצע $A'B$ ו־Q' באמצע AB'.**

או באופן כללי, אם ניקח את אורך צלע הריבוע $ABA'B$ כיחידה, t יסמן את המרחק בין נקודות A ו־P ובין הנקודה A' ו־Q.



איור 17: שיטת Interpixel Approximation

כעת ניתן להגדיר את מדד הפינתיות המקורב

$$M_{interpixel}(C) = \min_{t\in[0,1]}\left(r_1(t), r_2(t)\right)$$

כאשר

$$\begin{aligned} r_1(t) &= (I_P - I_C)^2 + (I_{P'} - I_C)^2 \\ r_2(t) &= (I_Q - I_C)^2 + (I_{Q'} - I_C)^2 \end{aligned} \tag{7}$$

את רמות האפור בנקודות P, Q, P' ו־Q' ניתן לקרב באופן לינארי:

$$\begin{aligned} I_P &= (1-t) I_A + tI_B \\ I_{P'} &= (1-t) I_{A'} + tI_{B'} \\ I_Q &= (1-t) I_{A'} + tI_B \\ I_{Q'} &= (1-t) I_A + tI_{B'} \end{aligned} \tag{8}$$

על ידי הצבת (8) בתוך (7) נקבל

$$M_{interpixel}(x,y) = \min_{t\in[0,1]}\left(r_1(t), r_2(t)\right)$$

כאשר (x,y) הוא המיקום של הנקודה עבורה מחושב ערך הפינתיות וגם

$$\begin{aligned}
r_1(t) &= a_1t^2 + 2b_1t + c \\
r_2(t) &= a_2t^2 + 2b_2t + c \\
b_1 &= (I_B - I_A)(I_A - I_C) + (I_{B'} - I_{A'})(I_{A'} - I_C) \\
b_2 &= (I_B - I_{A'})(I_{A'} - I_C) + (I_{B'} - I_A)(I_A - I_C) \\
c &= r_A \\
a_1 &= r_B - r_A - 2b_1 \\
a_2 &= r_B - r_A - 2b_2
\end{aligned}$$

ניתן להוכיח ([5]) כי המינימום מתקבל כאשר

$$M_{interpixel}(x,y) = \begin{cases} c - \frac{b^2}{a} & ,(b<0) \wedge (a+b) > 0 \\ M_{simple}(x,y) & ,otherwise \end{cases}$$

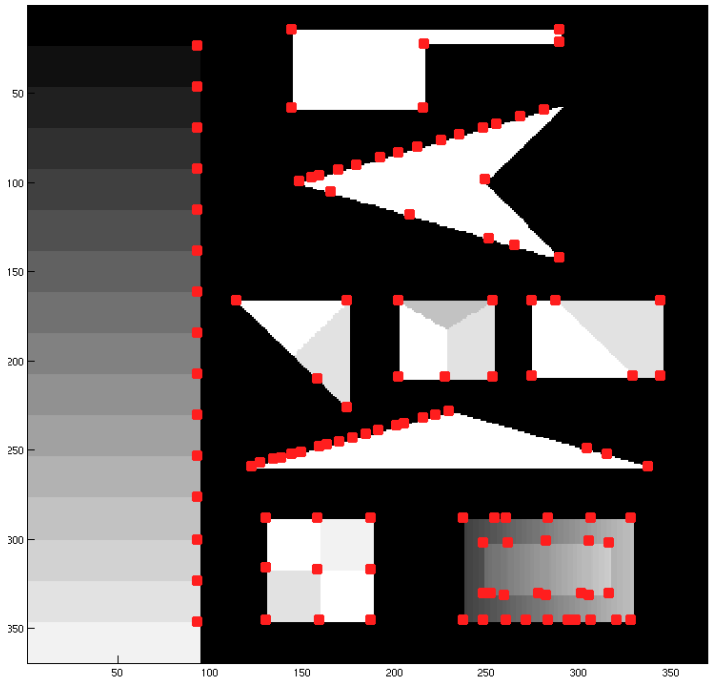
כאשר

$$\begin{aligned}
b &= \min(b_1, b_2) \\
a &= r_B - r_A - 2b
\end{aligned}$$

האלגוריתם 6 מציג תכנת MATLAB המממשת את החישוב הזה.

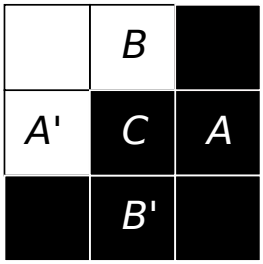
את תוצאת ריצת האלגוריתם על תמונת בדיקה עם הפעלת NMS על מפת הפינתיות ניתן לראות באיור 18.

```
2
function Map = trajkovic(Image)

Ic = double(Image);

% form the matrices Ia, Ib, Ia', Ib' as shifted Ic
Ia  = Ic(:, 3:end );
Ia_ = Ic(:, 1:end-2);

Ib  = Ic(3:end, :);
Ib_ = Ic(1:end-2, :);

% make the matrices same size
Ic = Ic(2:end-1, 2:end-1);
Ia  = Ia(2:end-1, :);
Ia_ = Ia_(2:end-1, :);
Ib  = Ib( :, 2:end-1);
Ib_  = Ib_( :, 2:end-1);

ra = (Ia - Ic).^2 + (Ia_ - Ic).^2;
rb = (Ib - Ic).^2 + (Ib_ - Ic).^2;

C_simple = min(ra, rb);

B1 = ( Ib - Ia ).*( Ia - Ic ) + ( Ib_ - Ia_).*( Ia_- Ic );
B2 = ( Ib - Ia_).*( Ia_- Ic ) + ( Ib_ - Ia_).*( Ia - Ic );
C = ra;
B = min( B1, B2 );
A = rb - ra - 2*B;

Ind = ( (B < 0) & ((A+B)>0) );

Map = C_simple;
Map(Ind) = C(Ind) - (B(Ind).^2) ./ A(Ind);
```

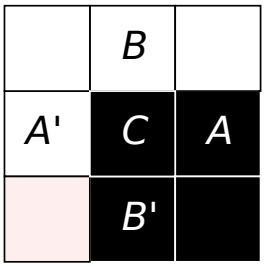


איור 18: תוצאות הרצת אלגוריתם טרייקוביץ'

האופרטור מזהה נכון את רוב הפינות בתמונה ובנוסף נקודות נוספות בעיקר על שפות אלכסוניות. נסביר את התופעה בעזרת האיורים 19 ו־20. באיור 19 הנקודה C נמצאת על שפה אלכסונית בזווית $45°$ ובאיור 20 הנקודה C היא פינה אמיתית. אבל, מכיוון שהאלגוריתם מתחשב ברמות האפור בנקודות A, A', B, B' ו־C בלבד, שהן זהות בשני המקרים, ערך הפינתיות שיתקבל עבור שתי התמונות יהיה זהה. אבל התמונה שבאיור 19 יכולה להתקבל על ידי סיבוב של תמונת שפה אופקית (או אנכית), אשר בה הנקודה C לא מסומנת בתור נקודת עניין ולכן ניתן להסיק כי גם האופרטור של טרייקוביץ' הוא אניזוטרופי בדומה לשאר האופרטורים שראינו.

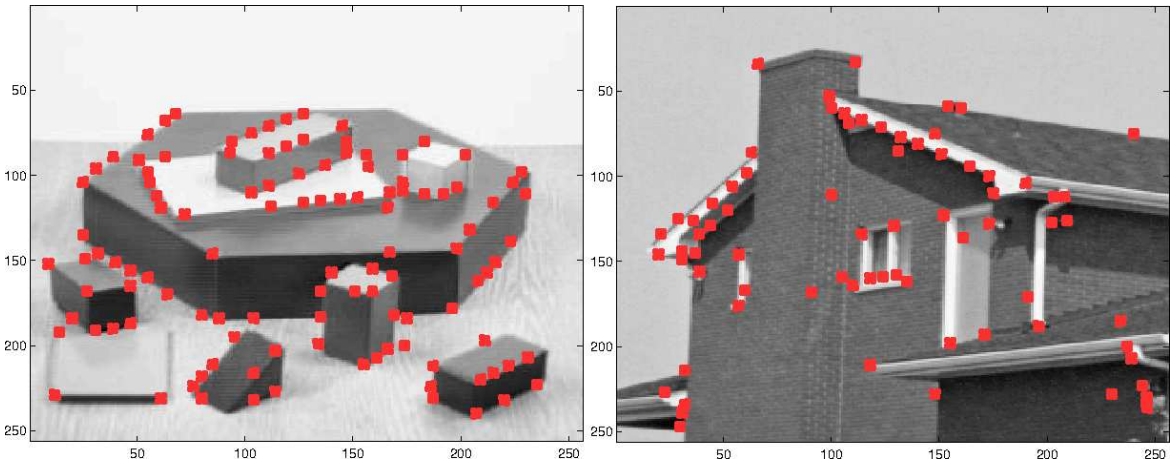


איור 19: נקודה על שפה אלכסונית



איור 20: נקודה שהיא פינה אמיתית

תוצאות ההפעלה של האופרטור על תמונות נוספות מוצגות באיור 21.



איור 21: הפעלת אופרטור טרייקוביץ' על תמונות אמיתיות

4 שימוש באלגוריתמים גנטיים

4.1 אלגוריתמים גנטיים

הגדרה 4.1 השם *אלגוריתמים גנטיים* מתאר משפחה של אלגוריתמים למיטוב (אופטימיזציה) שבהם משלבים זה בזה פתרונות אפשריים לבעיה, ומפעילים הליכים של ברירה טבעית כדי לבחור את המועמדים שיעברו לשלבים הבאים. רעיון תכנותי בסיסי זה מושפע מן היעילות של תורת האבולוציה, העושה שימוש ב־DNA של היצורים החיים, בפתרון בעיות אמיתיות [12].

תכנות גנטי הינו שם למודל תכנותי, בו אוסף (*אוכלוסיה*) של פרטים המיוצגים על ידי *כרומוזומים* עובר *תהליך אבולוציוני* במטרה לפתח פרטים חדשים, המתאימים ביותר לקריטריון מסוים. התהליך עובד בשלבים, כאשר בכל שלב מיוצרת אוכלוסיה חדשה (*דור*) על ידי ביצוע פעולות של *ברירה, זיווג ומוטציות* על הדור הקודם.

- *ברירה* הינו תהליך בו חלק מהפרטים מהדור הקודם נבחר כמתאים לזיווג ע"מ ליצור את הדור הבא. הברירה נעשית על סמך *ההערכה* של הפרט. בדרך כלל ההערכה מתבססת על פונקציה פשוטה יחסית, המקבלת כפרמטרים מספר מאפיינים של הפרט ומחזירה את הדירוג שלו המציין את מידת התאמתו לפתרון. בסוף תהליך הברירה נבחרים הפרטים שדירוגם הוא הגבוה באוכלוסיה.

- *זיווג* הוא תהליך יצירת הדור החדש של פרטים על ידי שילוב תכונותם של הפרטים מהדור הקודם ש"שרדו" את תהליך הברירה. במהלך הזיווג מתבצע מיזוג של מאפייני הפרטים ה"הורים" על מנת ליצור פרטים "צאצאים" המשלבים את התכונות החיוניות של כל אחד מה"הורים". בתור "הורים" לצאצא אחד יכול לבוא זוג פרטים, כפי שקורה בתהליכים ביולוגיים, וגם קבוצות גדולות יותר. האופן בו מתבצע המיזוג (כיצד בוחרים את התכונות אצל ההורים וכיצד משלבים בינהן) תלוי בשיטת ייצוג הפרטים, אופי הבעיה והפתרון הרצוי. בסוף תהליך הזיווג נוצרת אוכלוסיה חדשה אשר הדירוג הממוצע שלה גבוה יותר מהדור הקודם (במידה ותהליך הברירה והזיווג מתאימים לבעיה).

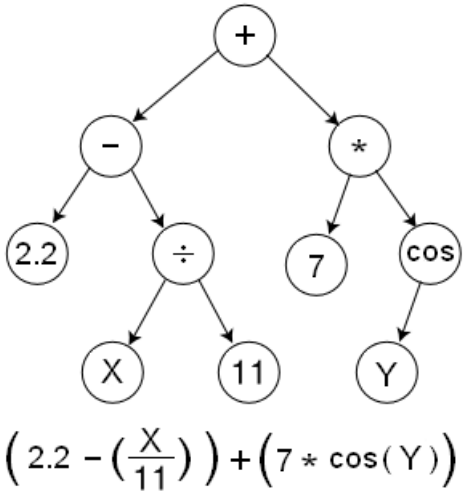
- *מוטציה* היא תהליך, בו חלק ממאפיני הפרט עובר שינוי אקראי. מוטציות באות להרחיב את מרחב חיפוש הפתרונות על ידי הכנסת תכונות חדשות של הפרט, אשר לא מורשות מההורים. המוטציות מונעות מאוכלוסיית הפרטים להגיע למצב, בו כל הפרטים דומים או זהים אחד לשני, ובכך עוצרים את תהליך ההתפתחות.

התהליך האבולוציוני מסתיים כאשר מתקיים אחד התנאים לסיומו (או שילוב של כמה) :

- נמצא פתרון אופטימלי.
- התבצע מספר מירבי של איטרציות.
- ניצול המשאבים (זמן, כסף, כח חישוב) הגיע למכסימום המוקצב.
- הדירוג של פתרונות הגיע למכסימום האפשרי.
- תהליך האבולוציה נעצר. כלומר אין הבדל משמעותי בדירוגים של הדורות האחרונים.
- עצירה ידנית.

באור מה שנאמר, באופן כללי האלגוריתם הגנטי נראה כך:

1. יצירת אוכלוסיה התחלתית
2. הערכת של כל פרט מהאוכלוסיה



איור 22: הצגת פונקציה כעץ (מתוך [12])

3. חזרה עד לקיום תנאי העצירה על השלבים הבאים:

(א) בחירת הפרטים המתאימים ביותר לזיווג על פי דירוגם

(ב) יצירת דור חדש על ידי זיווג ומוטציות

(ג) הערכה של פרטים מהדור החדש

4.2 תכנות גנטי

4.2.1 ייצוג הפרט

על מנת לייצג תכנה או אופרטור מסוים בצורה מתאימה לתכנות גנטי, משתמשים במבנה של עץ, כאשר כל צומת שלו מייצג פונקציה מסוימת. בצמתים של העץ נמצאות פונקציות מקבוצת הפונקציות F, ובעלים של העץ - הפונקציות מקבוצת הטרמינלים T, כאשר F היא קבוצת הפונקציות בעלות מספר פרמטרים *(arity)* גדול מאפס ו־T היא קבוצת פונקציות ללא פרמטרים (עם arity אפס) שהן בדרך כלל קבועים או הקלטים של הפונקצייה או אופרטור המיוצגים. כל צומת הוא פונקציה שהפרמטרים שלה הם הבנים של הצומת בעץ. לדוגמא הפונקציה $f(X,Y)=\left(2.2-\frac{X}{11}\right)+(7\times\cos(Y))$ ניתנת לייצוג באמצעות העץ באיור 22. כאן $F\subset\{+,-,\times,\div,\cos\}$ וכן $T\subset\{2.2,11,7,X,Y\}$.

4.2.2 יצירת אוכלוסיה התחלתים

יצירת האוכלוסיה ההתחלתית יכולה להתבצע בשלשה אופנים בסיסיים:

- שיטת ה־*Grow*
- שיטת ה־*Full*
- שיטת ה־*Ramped half-and-half*

בשיטות *Grow* ו־*Full* כל פרט נוצר על ידי הפעולה Initialize(root,1), כאשר הפסידוקוד של הפונקציה Initialize מוצגת באלגוריתם 7 עבור שיטת ה־*Grow* ובאלגוריתם 8 עבור שיטת ה־*Full*.

בשיטת ה־*Grow* העץ שנוצר יכול להיות בכל עומק עד לעומק מירבי $maxdepth$ ולא בהכרח מאוזן. בשיטת *Full* העצים הנוצרים תמיד מאוזנים ובעלי עומק מירבי.

אלגוריתם 7 יצירת פרט בשיטת Grow

```
2
InitializeGrow(node, depth)
    if depth = 1
        randomly select node from F
    else if depth = maxdepth
        randomly select node from T
    else
        randomly select node from F ∪ T
    end if
    if node ∈ F
        for n = 1 to arity of node
            InitializeGrow(n'th child, depth + 1)
        end for
    else
        return
    end if
end
```

אלגוריתם 8 יצירת פרט בשיטת Full

```
2
InitializeFull(node, depth)
    if depth < maxdepth
        randomly select node from F
        for n = 1 to arity of node
            InitializeFull(n'th child, depth + 1)
        end for
    else
        randomly select node from T
        return
    end if
end
```

שיטת *Ramped Half-and-Half* משלבת את השיטות הנ"ל. בשיטה הזאת נוצר מספר שווה של עצים הנוצרים על ידי *Full* ו־*Grow*. בצורה כזאת מובטח שבאוכלוסיה ההתחלתית יהיו עצים מגוון רחב של סוגים וגדלים.

4.2.3 פעולות אבולוציוניות

פעולת הזיווג בין העצים מתבצעת על ידי השחלוף של התת עצים של ההורים. באופן אקראי נבחרים צמתים בשני ההורים והתת עצים הנפרסים מהצמתים משוחלפים. בצורה כזאת מעל זוג של הדור הקודם נוצר זוג של הדור החדש. פעולת המוטציה מתבצעת על ידי החלפה של צומת אקראי בעץ בפונקציה אקראית מהקבוצה המתאימה T או F.

4.2.4 הערכת האוכלוסיה

כאמור פעולת ההערכה תלויה בבעיה עבורה נדרש למצוא פתרון. בבעיות של זיהוי נקודות עניין בתמונות ניתן להריץ את כל אחד מהאלגוריתמים של הדור על אוסף תמונות עם נקודות עניין ידועות מראש ולהעריך את ביצועי האלגוריתם מולן (כמות הנקודות שזוהו נכון, כמות הנקודות שלא זוהו נכון, מידת הדיוק בזיהוי מיקום הנקודות).

4.3 אלגוריתמים גנטיים למציאת אופרטור זיהוי פינות אופטימלי

את עקרונות התכנות הגנטי למציאת אלגוריתם זיהוי פינות אופטימלי ישמו לאונרדו טרוחיליו וגוסטבו אולג (Leonardo Trujillo & Gustavo Olague) והציגו בעבודה [10]. על ידי ניתוח של מספר האלגוריתמים הקיימים (שאת חלקם ראינו) הם הציגו אוסף של פונקציות F וטרמינלים T אשר השימוש בהם חיוני לזיהוי נקודות העניין, את אופן הערכת האלגוריתמים וביצוע פעולות אבולוציוניות.

4.3.1 מרחב החיפוש

לפי טרוחיליו ואולג F ניתנת על ידי $F = F_{1ary} \cup F_{2ary}$, כאשר F_{1ary} ו־F_{2ary} הן קבוצות הפונקציות האונריות והבינריות בהתאמה.

$$\begin{aligned} F_{2ary} &= \{+, -, |-|, \div, \times\} \\ F_{1ary} &= \left\{A, A^2, \log_2(A), EQ(A), G(\sigma = 1), G(\sigma = 2)\right\} \end{aligned}$$

כאן A הוא פרמטר שיכול לקבל ערכים של כל אחד מטרמינלים ב־T או את תוצאות הפונקציות מ־F. EQ היא אופרטור איזון ההיסטוגרמה של תמונה ו־G הוא אופרטור טשטוש גאוסי עם פרמטר σ. הטרמינלים שהוגדרו הם

$$T = \{I, I_{\sigma=1}, L_x, L_{xx}, L_{xy}, L_{yy}, L_y\}$$

כאשר I היא תמונת הקלט, $I_{\sigma=1}$ היא תמונת הקלט המטושטשת על ידי אופרטור גאוסי ו־L_w הן הנגזרות הגאוסיות בכוון w המחושבות על ידי קונבולוציה עם נגזרת הגאוסית בכוון w:

$$L_w = I * G_w(\sigma = 1)$$

הטרמינלים נבחרו בצורה כזאת מתוך הנחה כי לצורך זיהוי נקודות העניין בתמונה חיוני להתחשב לא רק בתמונה עצמה אלא גם במידע על קצבי השינוי של רמות האפור בכל נקודה, ומכאן באות הנגזרות.

4.3.2 הערכת אלגוריתמים

להערכת אלגוריתמי זיהוי נקודות עניין טרוחיליו ואולג משתמשים במידת *ההדירות* (repeatability) שהיא המדד הנפוץ ביותר בישומים אלה. ההדירות מציינת באיזו מידה התוצאות של ריצת האלגוריתם יציבות ביחס לשינויים שונים של דגימת תמונת הקלט, כגון סיבובים או שינוי בתאורה, כלומר האם זיהוי נקודה מסוימת חוזר על עצמו במספר תמונות שונות. זיהוי הנקודה x_1 בתמונת קלט I_1 חוזר על עצמו בתמונה I_i אם נמצאה בה נקודה מתאימה x_i. לצורך הפשטות מניחים כי התמונות בגודל זהה. היחס בין הנקודות x_1 ו־x_i ניתן להצגה על ידי הטרנספורמציה ההומוגרפית $H_{1,i}$, כאשר

$$x_i = H_{1,i}x_1$$

הטרנספורמציה ההומוגרפית היא היחס בין שתי הנקודות המתאימות על שתי תמונות שונות, אשר צולמו בזוויות או הזזות שונות של מצלמה.

מכיוון שקיימים איזורים שהם לא משותפים לכל התמונות, על מנת להעריך את הדירות הזיהוי יש לספור רק את הנקודות הנמצאות באיזורים המשותפים. יתר על כך, יש לקחת בחשבון גם שגיאת זיהוי מסוימת. זיהוי הנקודה x_i נחשב אם זוהתה נקודה בסביבה בגודל ϵ של x_i. אוסף זוגות הנקודות (x_1^c, x_i^c) הנמצאות באזור המשותף לשתי התמונות, ומתאימות עם שגיאה ϵ נתון על ידי

$$R_{I_i}(\epsilon) = \{(x_1^c, x_i^c) \,|dist\,(H_{1,i}x_1^c, x_i^c) < \epsilon\}$$

טרוחיליו ואולג מגדירים את מידת ההדירות $r_i(\epsilon)$ של הנקודות המזוהות בתמונה I_1 על ידי הנוסחה

$$r_i(\epsilon) = \frac{|R_{I_i}(\epsilon)|}{\min(\gamma_1, \gamma_i)}$$

כאשר $\gamma_1 = |\{x_1^c\}|$ ו־$\gamma_i = |\{x_i^c\}|$ הם מספרי כלל הנקודות שנמצאו בתמונות I_1 ו־I_i בהתאמה. נגדיר את $r_{K,J}(\epsilon)$ כהדירות הזיהוי על ידי האופרטור K על כל אוסף התמונות J כך ש־

$$r_{K,J}(\epsilon) = \frac{1}{N-1}\sum_{i=2}^{N} r_{K,I_i}(\epsilon)$$

$r_{K,J}(\epsilon)$ הינו מדד הביצועים של האופרטור K המופעל על רצף התמונות J המורכב מתמונות I_i.

המדד הוא מדד טוב לביצועי האופרטור, אבל אם משתמשים בו בצורה שהוצגה, אלגוריתם החיפוש עלול להעצר במכסימום מקומי לא רצוי. למשל החיפוש יכול להביא למציאת האופרטורים המזהים נקודות לא מעניינות המקובצות זו ליד זו באיזורים חלקים של התמונה. במקרה כזה אם המרחק בין הנקודות השכנות יהיה קטן מ־ϵ נקבל ערך ההדירות גבוה יחסית. יתר על כך, מכיוון שבתהליך החיפוש משתמשים בתמונות עם הרבה נקודות עניין המפוזרות באופן אחיד (זוהי דרישה לתמונות הקלט), אופרטור זיהוי טוב צריך לזהות נקודות מפוזרות באופן אחיד. כלומר, מצד אחד יש לזהות נקודות עם הדירות גבוהה ומצד שני מפוזרות באופן מירבי. לכן לפונקציית ההערכה הסופית f נוספים גורמים כפליים

$$f(K) = r_{K,J}(\epsilon) \cdot \phi_x^{\alpha} \cdot \phi_y^{\beta} \cdot N_{\%}^{\gamma}$$

כאשר הגורמים ϕ_u מקדמים את תנאי הפיזור של הנקודות ו־$N_{\%}$ הוא הגורם המקטין את הציון אם האופרטור החזיר פחות נקודות ממה שנדרש:

$$N_{\%}^{\gamma} = \left(\frac{extracted\,points}{requested\,points}\right)^{\gamma}$$

ϕ_u היא פונקציית סיגמואיד (sigmoid)

$$\phi_u = \begin{cases} \frac{1}{1+e^{-a(H_u-c)}} & H_u < H_u^{max} \\ 0 & otherwise \end{cases}$$

כאשר H_u היא האנטרופיה של ההתפלגות המרחבית של נקודות העניין שזוהתו על תמונת הייחוס I_1 מתוך האוסף J לאורך הציר u:

$$H_u = -\sum_j P_j(u) \log_2 [P_j(u)]$$

כאשר $P_j(u)$ היא פונקציית ההתפלגות של הנקודות לאורך ציר u, המקורבת על ידי ההיסטוגרמה של מיקום נקודות העניין לאורך הציר. הערכים של H_u^{max} נקבעים בצורה אמפירית מתוך תמונת הייחוס.

4.3.3 התוצאות

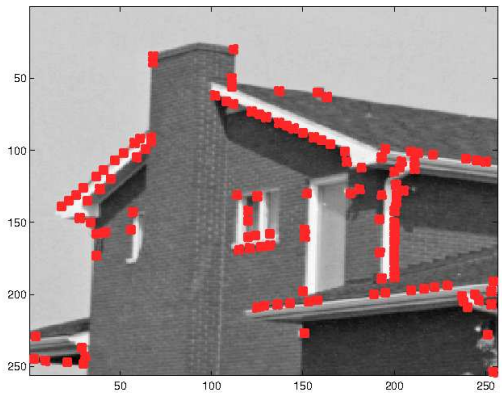
טרוחיליו ואולג יישמו את אלגוריתם החיפוש באמצעות הכלי GPLAB של Matlab. בתור תמונות ייחוס הם השתמשו בתמונה של וינסנט ואן גוך "הזורע" עם סיבובים שונים. בעבודתם [10] הם מתמקדים בשני אופרטורים שונים מתוך תוצאות החיפוש של האלגוריתם - IPGP1 ו-IPGP2 אשר הביטויים המתמטיים שלהם

$$IPGP1: \quad G(\sigma=2) * [G(\sigma=1) * I - I]$$
$$IPGP2: \quad G(\sigma=1) * [L_{xx} \cdot L_{yy}] - G(\sigma=1) * [L_{xy} \cdot L_{yx}]$$

כאשר כל אחד מהאופרטורים המרכיבים מוגדר בסעיף 4.3.1.

אלגוריתם 9 מציג את מימוש האופרטור $IPGP1$ ב-MATLAB ובאיור 23 ניתן לראות את תוצאת הרצתו על תמונה אמיתית.

```
2
% Function returning the "cornerness" map
% of the Image using the IPGP1 operator with
% radius of gaussian filter of 'radius'

function Map = ipgp1(Image, radius)

I = double(Image);

% gaussian filter with sigma=1
G1 = fspecial('gaussian', radius, 1);
% gaussian filter with sigma=2
G2 = fspecial('gaussian', radius, 2);

tmp = conv2(I, G1, 'same') - I;      % G(1)*I-I
Map = conv2(tmp, G2, 'same');        % G(2)*[G(1)*I-I]
```



איור 23: הרצת IPGP1 על תמונה אמיתית

5 סיכום

בעבודה זו הוצגו והודגמו מספר אלגוריתמים קלאסיים לפתרון בעיית זיהוי הפינות שהיא חלק מתחום רחב יותר של ראייה ממוחשבת. בנוסף תוארו האלגוריתמים הממנצלים את כח החישוב המקבילי הממומש בחלק מהמעבדים המודרניים. כמו כן הוצגה שיטת פיתוח אלגוריתמים חדשים באמצעות תכנות גנטי. קיים מספר רב של עבודות בנושא אלגוריתמי זיהוי הפינות מעבר לאלה שהוצגו בעבודה זו, אבל את המושג הבסיסי של העקרונות והגישות העקריות לבעיה העבודה אכן ממחישה. תחום הראייה הממוחשבת הינו תחום חדש יחסית ומתפתח במהרה ונושא זיהוי הפינות הינו חלק קטן אך חשוב שלו.

6 ספרות

[1] Hans P. Moravec, Towards Automatic Visual Obstacle Avoidance, Proc. 5th International Joint Conference on Artificial Intelligence, 1977, pp 584.

[2] Hans P. Moravec, Visual Mapping by a Robot Rover, Proc. 6th International Joint Conference on Artificial Intelligence, Tokyo, Japan, August 1979, pp. 599-601.

[3] C. Harris and M. Stephens, A Combined Corner and Edge Detector, Proc. 4th Alvey Vision Conf.,1988, pp. 147-151.

[4] Jianbo Shi and Carlo Tomasi, Good features to track, In IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94), Seattle, 1994, pp. 593-600.

[5] M. Trajkovic and M. Hedley, Fast Corner Detection, Image and Vision Computing, Vol. 16(2), 1998, pp. 75-87.

[6] Edward Rosten and Tom Drummond, Fusing points and lines for high performance tracking, Proc. 10th IEEE International Conference on Computer Vision, Vol. 2, 2005, pp 1508-1515.

[7] Edward Rosten and Tom Drummond, Machine learning for high-speed corner detection, European Conference on Computer Vision, 2006, pp. 430-443.

[8] Sudipta N Sinha, Jan-Michael Frahm, Marc Pollefeys and Yakup Genc, GPU-Based Video Feature Tracking and Matching, Technical Report 06-012, Department of Computer Science, UNC Chapel Hill, May 2006.

[9] Lucas Teixeira and Waldemar Celes and Marcelo Gattass, Accelerated Corner Detector Algorithms, Proc. British Machine Vision Conference (BMVC), Leeds, UK, 2008.

[10] L. Trujillo and G. Olague, Automated Design of Image Operators that Detect Interest Points, Evolutionary Computation. MIT Press, Vol. 16, No. 4, 2008, pp. 483-507.

[11] C. Tomasi and T. Kanade, Detection and tracking of point features, Technical Report CMU- CS-91-132, Carnegie Mellon University, 1991.

[12] Wikipedia: The free encyclopedia. (2004, July 22). FL: Wikimedia Foundation, Inc. Retrieved from http://www.wikipedia.org