דו"ח מכין, חלק א' - מעבדה בענ"ת

עודד שלזינגר, 311356943

ניתאי ויגדרהאוס, 311154942

תרגיל הכנה – הצגת תמונה במטלב:

תמונה במטלב מיוצגת ע"י מטריצה אשר כל איבר בה מייצג פיקסל ומספק מידע לגביו. חלק מהתמונות מיוצגות ע"י מטריצה תלת-מימדית בה כל מימד מייצג אחד מצבעי RGB.

סוגי התמונות:

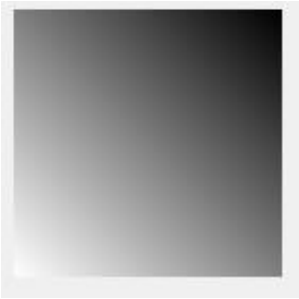
binary – כל פיקסל במערך הדו מימדי מיוצג ע"י 1 או 0 בתור לבן או שחור בהתאמה.

indexed - כל פיקסל במטריצה מיוצג ע"י אינדקס ממפת צבע.

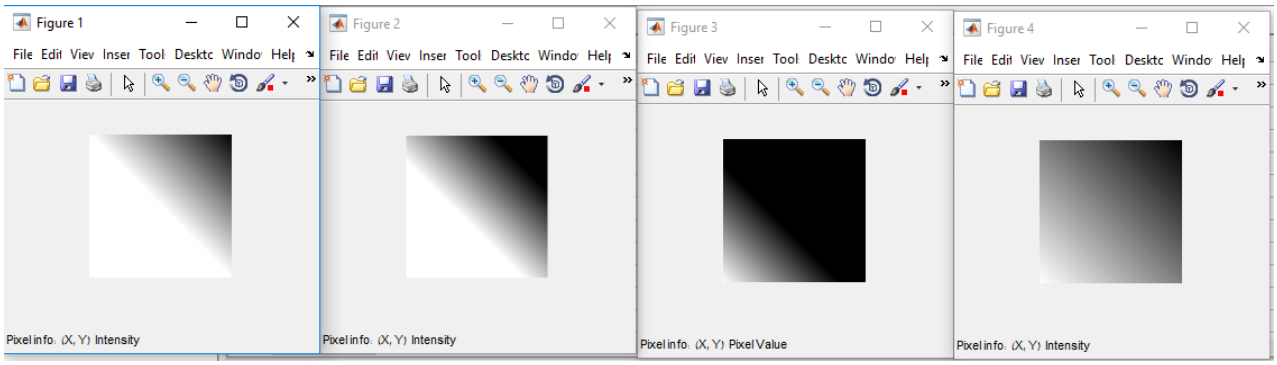
grayscale – כל פיקסל במטריצה מיוצג ע"י ערך העוצמה שלו.

colored – זוהי מטריצה תלת מימדית שבה כל פיקסל מיוצג ע"י איבר המייצג את ערכי העוצמה של כל אחד מצבעי אדום, ירוק וכחול.

עבור התמונה ראשונה נקבל את הערכים הבאים (גם באמצעות לולאה וגם באמצעות meshgrid):



וארבעת התמונות הבאות:



כעת התמונות שונות בערכי הפיקסלים ע״י כך שקבענו את התחום הדינאמי שבהן בפרמטר השני של הפונקציה imshow, כך שהערך הנמוך בתחום מיוצג ע״י הצבע השחור והערך הגבוה מיוצג ע״י הלבן. כאשר הסוגריים ריקות, התחום הדינאמי הנקבע הוא הטווח שבין הפיקסל המינימלי שבתמונה למקסימלי שלה. כך ניתן לראות שבתמונה הראשונה משמאל, שהתחום הדינאמי שלה הוא 0 עד 127, כל הערכים מעל 127 הם לבנים, ולכן רוב התמונה לבנה, לעומת התמונה השלישית, שבה התחום הוא בין 150 ל-255, כלומר כל ערך שקטן מ-150 מוגדר כשחור, ולכן רוב התמונה מוגדרת כשחורה. בתמונה 4 שבה התחום הדינאמי הוא בין הערך המינימאלי של הפיקסלים בתמונה למקסימאלי ניתן לראות בדיוק את השתנות עוצמת הגוון האפור לאורך התמונה.

ניסוי 1 – אינפורמציה על תמונה

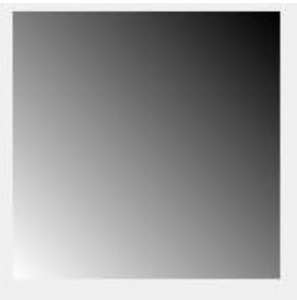
הפונקציה impixelinfo מאפשרת לשים את הסמן על התמונה, לקבל את הקואורדינטה הדו-מימדית שלה ואת ערך הפיקסל שלה.

הפונקציה histogram מספקת לנו את מספר הערכים של כל איבר במערך. כך, למשל, עבור תמונה, נוכל לקבל את היסטוגרמת הפיקסלים שלה, וכך ניתן לדעת את התפלגות הגוונים והצבעים השונים, לראות האם קיים פיקסל מסוים שבולט על פני אחרים וכד׳.

ניסוי 2 – מתיחת היסטוגרמה, שינוי בהירות וניגודיות

ניתן לבצע מתיחת היסטוגרמה על מנת להבין את התפלגות הגוונים בתמונה עבור תחום מסוים. למשל אם נקבע פיקסל מסוים וחלון מסוים שנמצא בסביבת הערך שלו, נוכל לראות כמה מהתמונה נמצא בתחום זה, כמה ממנו גדול מהתחום (אז ייצבע בלבן) וכמה קטן ממנו (אז ייצבע בשחור), וכך נוכל להבין איזה חלק בתמונה מכיל את התחום שאנו מתעניינים בו. נוכל לבודד את תחום זה ולבצע עליו שינויים בהתאם לרוחנו, ונוכל לראות בו את התפלגות הגוונים באיזור הספציפי (כיוון שאנו מבצעים מיפוי לינארי לערכי הבינארי עבור כל ערכי הגוונים).

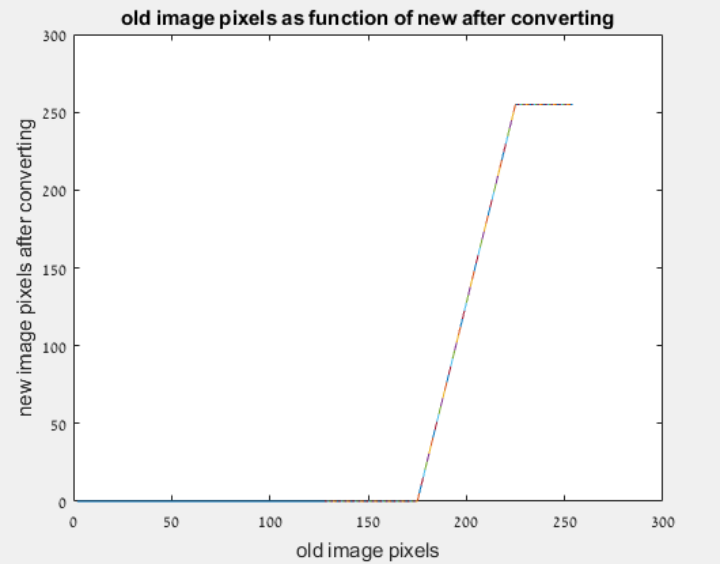
למשל, עבור פיקסל שערכו 200 וחלון בגודל 50 נוכל לראות איזה חלק בתמונה נמצא בתחום 175-225, מה שמעל ייצבע בלבן ומה שמתחת בשחור. כך ניתן לראות כיצד התמונה הבאה:



תהפוך לתמונה הזו:

ניתן לראות כי רק החלק שבתחום 175-225 נשמר וניתן לראות את התפלגות הגוונים בו, זה שמעליו הפך לבן ושמתחתיו שחור.

פונקציית ההמרה שקיבלנו:



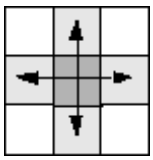
באופן כללי הפעולה אינה לינארית, כיוון שעבור התחום הרלוונטי (ערכי הביניים) נקבל את הפונקציה y=x, אך עבור x<level-windows/2 נקבל 0 ועבור x>level+windows/2 נקבל 255. באופן כללי ניתן לראות כי הפעולה אינה לינארית, כיוון שעבור ערכים מסוימים היא מקבעת אותם לערך קבוע מסוים ועבור ערכים אחרים (בחלון) היא לינארית.

ניסוי 3 – סגמנטציה של תמונה

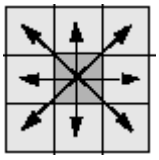
תפקיד הפונקציה bwconncomp הוא למצוא רכיבים מקושרים בתמונה בינארית. כפרמטרים מובאת התמונה הבינארית וסוג הקישוריות המבוקשת שתקבע האם רכיב הוא מקושר או לא. במוצא נקבל את השדות של המבנה: סוג הקישוריות שנקבעה, גודל התמונה, מספר הרכיבים המקושרים והאינדקסים של כל אחד מהם.

קישוריות 4 קובעת כי אם הפיקסלים מקושרים באחד מ-4 הקצוות שלהם (האנכיים והמישוריים), כלומר סמוכים אחד לשני ובעלי אותו ערך בפיקסל, אז הם חלק מאותו רכיב מקושר. לעומת זאת, עבור קישוריות 8, אם הפיקסלים סמוכים אחד לשני באנך, במישור ובאלכסון, אז הם חלק מאותו רכיב קישורי.

עבור קישוריות 4 אלו הכיוונים הרלוונטים כדי ששני פיקסלים סמוכים יהיו באותו רכיב מקושר:



ועבור קישוריות 8:



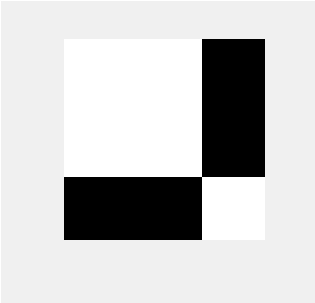
ניקח למשל את המטריצה הבאה:

1 1 0

1 1 0

0 1 1

שנראית כך:

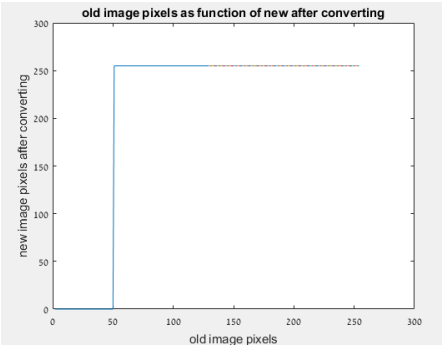


לאחר שנריץ עליה את הפונקציה בקישוריות 4, נקבל כי קיימים 2 רכיבים מקושרים ואילו עבור קישוריות 8 נקבל רכיב מקושר אחד, כיוון שהוא מחשיב שני פיקסלים באותו איבר גם אם הם באלכסון (בניגוד לקישוריות 4).

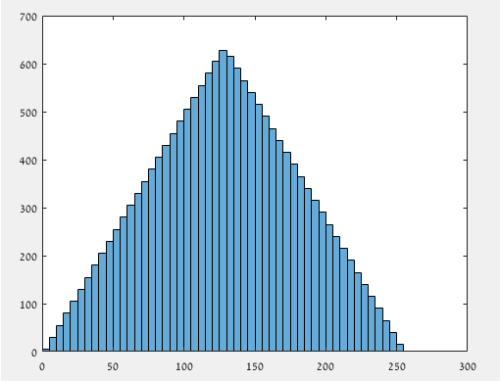
הפונקציה labelmatrix מאפשרת לנו לקבל מטריצה עבורה פיקסלים עם ערך 0 הם הרקע, ועבור כל ערך השונה מ-0, קבוצת הפיקסלים שערכה ערך מסוים היא חלק מאותו רכיב מקושר. דבר זה מאפשר לנו לעשות סגמנטציה כי הוא מבודד אובייקטים בתמונה ומאפשר לנו לבצע עליהם סגמנטציה.

עבור תמונה העוברת חיתוך סף נקבל את הגרפים הבאים :

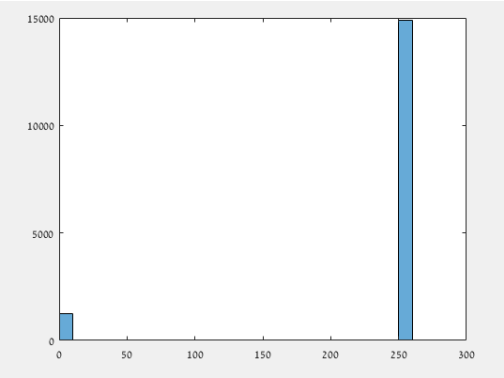
ההארה ההתחלתית (ציר x) לעומת הסופית (ציר y) :



ההיסטוגרמה של התמונה המקורית :



ההיסטוגרמה של התמונה הסופית :



לתמונות רמות אפור המכילות מספר אובייקטים המסתירים זה את זה באופן חלקי יש לבצע סגמנטציה באמצעות קביעת מספר ספים בתמונה על פי חלוקה בתחומים, כך שעבור אובייקטים המסתירים אובייקטיבים אחרים נבצע סגמנטציה על ידי קביעת ערך מסוים עבור תחום ערכי הפיקסלים בהם הם נמצאים וערך אחר עבור האובייקט המוסתר כך שהתחום שלו לא יחפוף עם התחום של האובייקט המסתיר.

נדרש לעשות סגמנטציה בתהליכים שברצוננו לבודד אובייקטים מסוימים מהרקע ולסווגם. למשל, אם נרצה לסווג סוג חיה על רקע נוף, נרצה לבצע סגמנטציה על החיה ולבודד אותה מהנוף, ורק אז לבודד אותה. כמו כן, סגמנטציה מאפשרת לנו באמצעות בידוד האובייקט בתמונה מסוימת לבצע רק עליו שינויים – למשל במקרה שיש לנו מספר חיות על רקע לבן, נוכל לבצע עליהן סגמנטציה ולהחליט שעבור החיה המסווגת כחתול נוכל "למחוק" אותה ע"י הפיכת כל הפיקסלים שלה ל-1.

ניסוי 4 – סינון רעש ושחזור תמונה

סוגי הרעש שניתן להוסיף לתמונה הם :

Gaussian – מוסיף רעש לבן לערכי הפיקסלים בתמונה עם שונות 0.01 ותוחלת אפס לתמונה. ניתן להוסיף פרמטר נוסף אשר קובע את הממוצע לא להיות האפס, וכן פרמטר נוסף הקובע את השונות.

Localvar – מוסיף רעש לבן עם שונות מוגדרת.

Poisson – מייצר רעש מהמידע באמצעות אלגוריתם פואסון במקום להוסיף רעש למידע ע"י אלגוריתמים מסוימים. אין פרמטר נוסף להוסיף לכך.

Salt & pepper – מוסיפה רעש "מלח ופלפל" עם צפיפות רעש דיפולטיבי של 0.05. הדבר משפיע על 5% מהפיקסלים בתמונה. ניתן להוסיף פרמטר הקובע את צפיפות הרעש, שמשפיע על d כפול מספר האלמנטים בתמונה.

Speckle – מוסיף רעש כפלי באמצעות המשוואה J=I+n*I, כאשר I מייצג את ערך הפיקסל ו-n הוא ערך שיכול להיקבע כפרמטר, ובאופן דיפולטיבי מוגדר עם רעש אקראי עם תוחלת 0 ושונות 0.04.

הפונקציה fspecial מייצרת מטריצה דו מימדית המשמשת כפילטר לתמונה. שולחים כפרמטר את סוג הפילטר, למשל פילטר של ממוצע, ואז ניתן באמצעות הפונקציה imfilter להשתמש בה עבור התמונה.

מסנן תנועה - מקרב את התנועה הלינארית של המצלמה, כאשר כפרמטרים נשלח אליה אורך התנועה וזווית התנועה. החיסרון בו הוא שקשה יותר לשחזר את התמונה המקורית בעקבות הסיבוכיות בהוספת הרעש בתמונה.

מסנן חציון - כל ערך פיקסל הופך לחציון של שכניו במטריצה 3x3. החיסרון בו הוא שיכול להיות שאם ערכי הפיקסלים קרובים מאוד בערכם אחד לשני אז השפעתו זניחה כדי שישפיע בצורה נראית לעין על התמונה.

הפונקציה imfilter מייצרת סינון N מימדי עבור תמונות N מימדיות באמצעות הפילטר המתאים והתמונה שאנו רוצים לסנן, אותם אנו שולחים כפרמטרים. הפלט שלו הוא מערך עם מידע מאותו סוג, למשל uint8.

הפונקציה deconvlucy משחזרת את התמונה המטושטשת בשיטת לוסי-ריצ'ארדסון. הפרמטרים ההכרחיים בה הם התמונה אותה יש לשחזר, אשר שונתה ע"י קונבולוציה עם פונקציית PSF, המתארת את התגובה של מערכת הדמיה למקור נקודתי, וייתכן גם ע"י הוספת רעש. פונקציית ה-PSF היא גם כן פרמטר הכרחי עבור הפונקציה, והאלגוריתם מבוסס על מקסום האפשרות שהפלט הוא דוגמא לתמונה המקורית תחת סטטיסטיקות פואסוניות.

הסיבות לטשטוש תמונה בזמן הצילום הן תזוזה של המצלמה או של האובייקט המצולם, מהירות צמצם מצלמה נמוכה מדי, עדשה מטושטשת ועוד.

ניקח מטריצת טשטוש ממוצע בגודל 5x5, בה כל איבר הוא 1/25. כך נראית התמונה מלפני (שמאל) ואחרי (מימין):

ניסוי 5 – גילוי שפות

המוצא של הפונקציה edge הוא תמונה (מטריצה) בינארית, אשר בה קיים ערכי 1 באיברים בהם הפונקציה מצאה גבולות (של אובייקטים) בתמונה.

סוגי האופרטורים שהיא מכירה הם:

- Sobel – מוצאת את הגבולות בנקודות בהן הגרדיאנט מקסימלי, באמצעות קירוב sobel.
- Prewitt - מוצאת את הגבולות בנקודות בהן הגרדיאנט מקסימלי, באמצעות קירוב prewitt.
- Roberts - מוצאת את הגבולות בנקודות בהן הגרדיאנט מקסימלי, באמצעות קירוב roberts.
- Log - מוצאת את הגבולות באמצעות חיפוש אחר שינוי סימן במטריצה ('חציות אפס'), לאחר שהתמונה מועברת בפילטר LoG – פילטר הלפלסיאן של הגאוסיאן.
- zerocross – מוצאת את הגבולות באמצעות חיפוש אחר שינוי סימן במטריצה ('חציות אפס'), לאחר שהתמונה מועברת בפילטר כלשהו שמגדירים עבור הפונקציה.
- Canny – מוצא את הגבולות באמצעות חיפוש עבור מקסימות מקומיות של הגרדיאנט של התמונה. הפונקציה מחשבת את הגרדיאנט באמצעות הנגזרת של פילטר הגאוסיאן.

- Approxcanny – משתמשת בגרסא מקורבת של אלגוריתם canny כך שמהירות הביצוע תהיה מהירה יותר על חשבון דיוק.

נתאר את אופן פעולת האופרטורים הבאים:

- Prewitt – האופרטור משתמש בשתי מטריצות גרעין בגודל 3x3 אשר מבצעות קונבולוציה עם התמונה המקורית, אחת עבור קירובי נגזרת בחלק האנכי ואחת בחלק המישורי. כדי לקבל את ערך הגרדיאנט נבצע משפט פיתגורס בין הערך שהתקבל על ידי קונבולוציה עם המטריצה בחלק האנכי לערך שהתקבל קונבולוציה עם המטריצה בחלק המישורי (את הכיוון ניתן לחשב עם הזווית).
- LoG – פונקציית הלפלסיאן של הגאוסיאן היא:

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

ועל פיה בונים את מטריצת הפילטר.
האופרטור מחשב את הנגזרת השנייה המרחבית של התמונה. באיזורים בהם העוצמה אחידה, כלומר האיזורים בהם הגרדיאנט הוא אפס, התגובה של ה-LoG תהיה אפס, ובאיזורים בהם יהיה שינוי, הפילטר יגיב בכך שייתן ערכים חיוביים בצד החשוך יותר, וערכים שליליים בצד המואר.

- Canny – ראשית התמונה עוברת קונבולוציה עם פילטר של פונקציית גאוסיאן, ואז נגזרת ראשונה שלו כדי לקבל את האיזורים הבולטים עם הנגזרת הגבוהה בתמונה. שנית, הפילטר מאפס את כל האיזורים בהם הנגזרת אינה גבוהה, ואז קובעים שתי נקודות סף, האחת גבוהה מהשניה, כאשר בהתחלה עוקבים אחרי האיזורים הגבוהים מנקודת הסף הגבוהה יותר, ואז ממשיכים לערכים נמוכים יותר עד שמגיעים לערך הסף הנמוך יותר.

ההבדל בין השיטות השונות הוא השוני בחישוב הנגזרת, כלומר בשינוי ערכי הפיקסלים באיזורים השונים בתמונה. בעוד ש-Prewitt נותן דגש על ערכי נגזרת ראשונה הניתנים על ידי הגרדיאנט, LoG נותן דגש על נגזרת שניה באמצעות פונקציית הגאוסיאן, ו-Canny נגזרת ראשונה באמצעות פונקציית הגאוסיאן. Canny, בניגוד לשתיים האחרות, מאפשר זיהוי גבולות גם עבור מקסימות מקומיות ולא גלובאליות, כך שהוא מאפשר זיהוי אובייקטים רב יותר, והוא האידיאלי יותר, אך הוא גם המסובך יותר משלושתם.

בשיטת Laplacian of Gaussian אנו מקרבים את הנגזרות מסדר שני של פונקציית הגאוסיאן, והאופרטור מחשב את הנגזרת השנייה המרחבית של התמונה.

בעיות עיקריות בתהליכי גילוי שפות:

- אי הבחנה בין 2 שפות, דבר הנובע מכך שמספר מסוים של אותיות בשתי השפות דומה מאוד, והאותיות הניתנות בתמונה הן ברובן האותיות הדומות משתי השפות. דבר זה

עלול לגרום לזיהוי שגוי של שפה ע״י השפה השגויה האחרת שיש דימיון במספר מצומצם של אותיות ביניהן. כלומר, צריך כמה שיותר אותיות שמזהה הקצוות יעלה עליהן.

- ייתכן כי בשפות מסוימות האותיות מחוברות, או הכותב שלהן כתב את האותיות צמודות אחת לשניה, ויהיה קשה למזהה הקצוות להבדיל ביניהן והוא יבחין בהן כאות אחת. באופן דומה ייתכן כי אותיות יעלו אחת על השניה, או כל חפיפה שתימנע זיהוי אות (כי היא תכיל חלק של אות אחרת או שסופה יהיה מחובר להמשך אות אחרת).
- ייתכן שהפילטרים המזהים קצוות יזהו את האובייקט כבעל שטח רב יותר משטחו בפועל, והדבר יקשה על זיהוי האותיות (כלומר יכיל שטח לא רלוונטי בתמונה, בשונה מהבעיה הקודמת בה הייתה חפיפה בין אובייקטים שונים).
- ייתכן כי התמונה לא תכיל מספיק אובייקטים לזיהוי השפה.

ניסוי 6 – פעולות מורפולוגיות

הפונקציה strel מייצרת אלמנט בניה מורפולוגי, הפונקציה imerode מבצעת הצרת תמונה באמצעות אלמנט הבניה המורפולוגי, והפונקציה imdilate מבצעת הרחבת תמונה באמצעותו.

פעולות אלה יכולות לשמש אותנו למחוק עצמים לא רצויים בתמונה (במידה והם נבדלים מהשאר בצורתם הגיאומטרית ובגודלם), או לחילופין להשאיר רק עצמים מסוימים בתמונה.

הפעולות אינן הופכיות ובאמצעותן ניתן להעלים ולהחזיר צורות מהציור, אך לא בהכרח את כולן. למשל, ברגע שנעלים אוביקט מסוים שאינו בצורת אלמנט הבנייה, ייתכן שכדי להחזיר אותו נצטרך במדויק את אלמנט הבניה המקורי, וגם אז הדבר תלוי במה שנשאר בתמונה לאחר שהעלמנו אובייקטים מסוימים.

צורת האובייקט וגודלה יקבעו אילו אובייקטים יישארו או ייעלמו מהתמונה על פי צורתה וגודלה. ככל שצורת האובייקט תהיה מדויקת יותר וקטנה יותר (לפי גודל האובייקט שברצוננו להסיר או להשאיר), כך הפעולה תהיה מדויקת יותר.

ניתן לממש זאת עבור תמונה בינארית ע״י בניית מטריצה דו-מימדית בינארית המייצגת את אלמנט הבניה, וכאשר אנו עוברים על התמונה, יש לבדוק בהתאם לפעולת ההרחבה או ההצרה עבור כל פיקסל האם להשאיר אותו כשחור או כלבן (בהתאם לאלמנט הבניה – עבור הצרה יהיה זה הערך הקטן ביותר בסביבה, ועבור הרחבה יהיה זה הערך הגדול ביותר בסביבה). נוכל לעשות זאת אם עבור כל פיקסל בתמונה נמקם את אלמנט הבניה (שהוא המטריצה הדו-מימדית) כך שהפיקסל יהיה במרכז המטריצה, ועבור כל פעולה של הרחבה או הצרה נתייחס רק לערכים של האיברים הנמצאים במטריצה וערכם 1. עבור הרחבת התמונה, נתייחס לכל האיברים שהם אפס באלמנט הבנייה כבעלי הערך המינימאלי, ועבור הצרה נתייחס לכל האיברים שהם אפס כבעלי הערך המקסימאלי ע״י הפונקציות המתאימות, כך שרק ערך האיברים התואמים את צורת אלמנט הבניה ייכללו בחישוב עבור כל פיקסל.