Scanner assist

מגישים:

דן אביגדול

איתן נצר

מנחה:

אמיר גבע

תוכן עניינים

מבוא

מסמך זה מתאר ומסכם ביצוע של פרויקט בתכנות אפליקציה עזר בסריקת מסמכים בעזרת המצלמה שבטלפון החכם. הפרויקט נעשה במסגרת המעבדה למערכות נבונות בפקולטה למדעי המחשב בטכניון.

הפרויקט משלב אתגרי תכנון ומימוש תוכנה במספר שפות תכנות וכן אתגרי אלגורית מיקה בעיקרם בתחום הראיה הממוחשבת ועיבוד תמונה.

נתאר במסמך זה את כל תכולת הפרויקט החל מהאלגוריתמים בהם השתמשנו עבור באתגרים איתם התמודדנו וכלה במימוש של האפליקציה עצמו.

סקירה של האלגוריתם

פרק זה יעסוק בסקירה כללית של האלגוריתם. הפרקים הבאים יפרטו וירחיבו את השלבים השונים והשימוש האלגוריתמי בכל שלב.

מטרת האלגורתמים היא לזהות בתמונה את מסגרת הדף ולחשב כיצד להציב את המצלמה כדי לצלם את הדף באיכות אופטימאלית.

שלבי האלגורתמים

1. רכישת התמונה ממצב הpreview.
2. עיבוד מקדים לתמונה.
3. ניקוי רעשים.
4. מציאת השפות בתמונה.
5. זיהוי מסגרת הדף (מציאת השפה הסגורה בעלת השטח הגדול ביותר).
6. מציאת ארבע נקודות המייצגות את פינות הדף.
7. חישוב מיקום המצלמה ביחס לדף.
8. מתן הוראות למשתמש כיצד להניע את המצלמה להשגת איכות צילום גבוהה.

עיבוד מקדים

פרק זה יתאר את השיטות השונות לביצוע עיבוד מקדים לתמונה. השיטות השונות משמשות עבור סיטואציות שונות עבורן המשתמש מבקש להשיג סריקה איכותית.

ראשית מתבצעת דגימה מחדש לתמונה, כך שהתמונה תוקטן פי 4 (פי 2 בכל ציר). הקטנת התמונה מבוצעת בשביל להגיע לשיפור בביצועים וזמן ריצה מהיר יותר.

בכול פריים נבחר בין אחת שיטות העיבוד המקדים, אם השיטה הייתה לא מועילה במשך כמה פריימים, עוברים לשיטת העיבוד הבאה. המעבר בין שיטות העיבוד המקדים הוא בשיטה ציקלית: שיטה רגילה ← שוויון היסטוגרמות ← מסכה פנימית ← מסכה חיצונית ← שיטה רגילה...

שיטה רגילה

עבור המקרה הכללי, נמיר את התמונה ממרחב RGB למרחב רמות אפור.

שיטת שיוויון היסטוגרמות

עבור מקרים בהם הדף מונח על גבי משטח בעל גוון צבע דומה לצבע הדף, לדוגמא, מסמך לבן על גבי שולחן בצבע אפור בהיר. במצב כזה קשה ליצור הפרדה בין הדף לרקע. לכן, נרצה לגרום להפרדה ברורה יותר בין רמות האפור של הרקע לרמות האפור של הדף.

נמיר את התמונה ממרחב צבע RGB למרחב רמות אפור. כדי להקל על ההפרדה נשתמש בשוויון היסטוגרמות (ראה נספח), בעזרת שיטה זו ניצור הבדל גדול יותר בין הרקע לדף ע"י ניצול של מירב התחום הדינמי של מרחב רמות האפור. החסרון של שיטה זו, היא בכך רעש קטן מוגבר כתוצאה משיוון ההיסטוגרמות.

שיטת מסכה פנימית

עבור מקרים בהם הרקע מכיל טקסטורה עשירה, דוגמת שולחן עץ. הטקסטורה עלולה להיחשב כשפה השייכת לדף ולשבש את מציאת המסגרת. במצבים כאלה נרצה לנצל את גוון המסמך בשביל ליצור הפרדה מהרקע.

נמיר את התמונה ממרחב צבע RGB למרחב צבע HSV (ראה נספח). נחשב את ההיסטוגרמה של ערוץ הרוויה עבור טלאי במכרז התמונה, מתוך הנחה כי שם ימצא הדף אותו אנו מבקשים לסרוק. מכיוון שאנו מחשבים היסטוגרמה המבוססת על הדף אותו נרצה לסרוק, צפוי כי עיקר הערכים יתארו את אופי הדף הנסרק ולכן הערך עבורו מתקבל מקסימום של ההיסטוגרמה, יתאר את האופי של הדף. בעזרת ערך זה נבצע תהליך של יצירת מסכה לתמונה, תהליך זה יגרור כי כל פיקסל בעל ערך רוויה סביב ערך שחישבנו קודם יקבל ערך 255 וכל פיקסל אחר יקבל ערך 0. בצורה זו קיבלנו תמונת רמות אפור בה בכל מקום הצפוי להכיל את הדף הפקסלים דולקים ושאר הפיקסלים כבויים.

שיטת מסכה חיצונית

עבור מקרים בהם דף מכיל מידע עשיר, דוגמת ציור או שרטוט. הציור עלול להיחשב כשפה השייכת לדף ולשבש את מציאת המסגרת. במצבים שכאלה נרצה לנצל את גוון הרקע בשביל ליצור הפרדה מהמסמך.

נמיר את התמונה ממרחב צבע RGB למרחב צבע HSV (ראה נספח). נחשב את ההיסטוגרמה של ערוץ הרוויה עבור טלאים בחלקים סביב מסגרת התמונה, מתוך הנחה כי הדף ימצא במרכז התמונה ובסביבות המסגרת נוכל למצוא את הרקע עליו מונח הדף. מכיוון שאנו מחשבים היסטוגרמה המבוססת לפי הרקע צפוי כי עיקר הערכים ישקפו את אופי הרקע ולכן הערך עבורו מתקבל מקסימום של ההיסטוגרמה, יתאר את האופי של הרקע. בעזרת ערך זה נבצע תהליך של יצירת מסכה לתמונה, תהליך זה יגרום לכל פיקסל בעל ערך רוויה סביב ערך שחישבנו קודם לקבל ערך 255 וכל פיקסל אחר יקבל ערך 0. בצורה זו קיבלנו תמונת רמות אפור בה בכל מקום הצפוי להכיל את הרקע הפיקסל דלוק ושאר הפיקסלים כבויים.

ניקוי תמונה

לאחר העיבוד המקדים המתאים לפי השיטות השונות, נרצה לבצע ניקוי לתמונה כדאי להקל על מציאת שפות הדף.

נציג מספר מצבים בעייתיים:

1. הטקסט שבתמונה עלול ליצור מספר גבוהה של שפות בתמונה, אשר עלולות לגרום לזמן חישוב ארוך יותר (Canny האלגוריתם רץ על השפות למציאת הרלוונטיות). בנוסף טקסט זה עלול להתחבר למסגרת וליצור מסגרת שגויה.
2. יצירת המסכות עלולה ליצור עצמים שאינם חלק מהתמונה, כלומר נקודות השייכות לאובייקט הדרוש, אך הפיקסל שם אינו נדלק עקב שגיאה קלה בסף הנבחר.
3. רעש הנקלט בתמונה במהלך תהליך רכישת התמונה.

בכל המצבים הללו, נרצה להקטין את השגיאה הנוצרת בעקבות התופעות השונות הנ"ל.

סינון הרעש שבחרנו להשתמש בו הוא מסנן חציון (ראה נספח), משום שבמצבים של טקסט ויצירת מסיכה, המידע לגבי פיקסל ספציפי, קשור ישירות למידע על שכניו. במקרה של טקסט, עובי התו אינו

דק וסביר כי מסנן חציון יגרום לאותיות לעבור סינון ולהוריד את השפעתן בתמונה. עבור יצירת המסכה, פיקסל שנדלק בטעות או לחילופין לא נדלק נוכל להסיק ולתקן את הטעות לפי הפיקסלים השכנים לו. עבור רעש באזורים חלקים בתמונה שיטה של סינון חציון היא שיטה מקובלת.

החיסרון בסינון חציון הוא שהוא עלול להשפיע על השפות שבתמונה ולהסיט אותם במעט ממיקומן המקורי. לכן, נדאג להשתמש בחציון בעל גודל חלון קטן יחסית כדי לא להסיט את השפות יתר על המידה.

מציאת שפות בתמונה

באופן כללי, מציאת שפות בתמונה מתבצעת בעזרת מסנן HPF. מכיוון שבפועל תמונה עלולה להכיל רעש ומתוארת על ידי קירוב דיסקרטי, נהוג להשתמש באלגוריתם מוכר בשם Canny (ראה נספח).

את הספים לאלגוריתם בחרנו בצורה אמפירית, בהתאמה לשיטות העיבוד המקדים השונות. כלומר, בחלק זה אנו מספקים תמונה ומקבלים כפלט תמונה בינארית כאשר פיקלסים השייכים לשפה דלוקים והשאר כבויים.

בחירת הקונטור המייצג את מסגרת הדף

לפי ההנחות אשר אנו מניחים על התמונה, תחת שימוש נכון באפליקציה, האובייקט המצולם אמור להיות מוכל בתוך השפה הסגורה בעלת השטח הגדול ביותר מכל הקונטורים שהתקבלו לאחר מציאת השפות שבתמונה.

נבהיר כי אנו מתמקדים מציאת הקונטור המכיל את השטח הגדול ביותר ולא הארוך ביותר, משום שקיימים מקרים בהם יהיה קונטור ארוך יותר לדוגמה אם המסמך מכיל כתב מחובר (בעל עובי שימנע ממסנן החציון להעלימו) או אם יש בדף ציור.

נשתמש בפונקציות המובנות בOPENCV לקבלת מערך לתיאור הקונטורים השונים בתמונה. אנו עוברים על הקונטורים השונים ומחשבים לכל קונטור את השטח שהוא מכיל (בעזרת פונקציה מוכנה בOPENCV). בסופו של התהליך אנו מחזירים את הקונטור בעל השטח הגדול ביותר, קונטור זה "חשוד" להיות הקונטור המייצג את מסגרת הדף.

בסיום שלב זה אנו מחזיקים באוסף נקודות המייצגות קונטור בעל סבירות גבוהה ליצג את מסגרת הדף אותו אנו מבקשים לסרוק.

בחירת ארבע הנקודות ליצוג פינות הדף

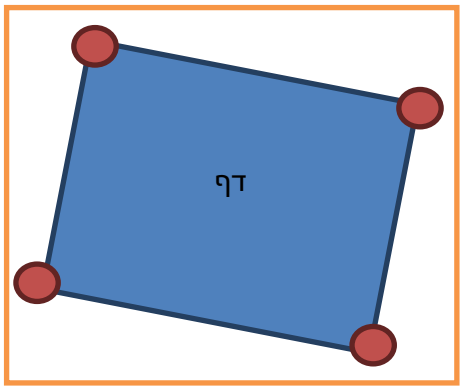
לאחר מציאת אוסף נקודות המייצג את מסגרת הדף. אנו נדרשים למצוא את 4 הנקודות אשר משוייכות לפינות הדף.

באופן כללי נעבור על אוסף נקודות זה, כאשר קיימות 4 נקודות בעלות עניין.

1) הנקודה בעלת ערך הX המקסימאלי מבין כל הנקודות באוסף.
2) הנקודה בעלת ערך הX המינימלי מבין כל הנקודות באוסף.
3) הנקודה בעלת ערך הY המקסימאלי מבין כל הנקודות באוסף.
4) הנקודה בעלת ערך הY המינימלי מבין כל הנקודות באוסף.

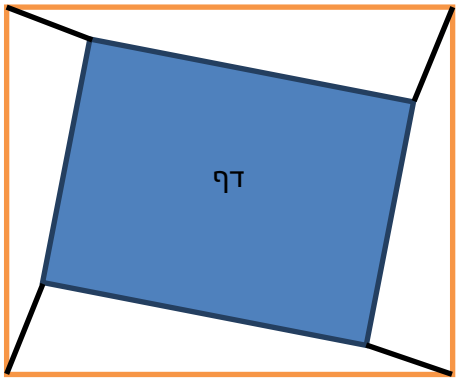
מובטח כי קיימת לפחות נקודה אחת עבור כל תנאי (אך לא מובטח כי הן בהכרח שונות). במידה ונמצאו 4 נקודות שונות נבדוק האם הם מתאימות (סבירות) לייצוג מלבן, על ידי חישוב יחסי צלעות וזוויות. במידה ואכן הן מתאימות ליצוג מלבן, במקרה הכללי, בו הדף הנסרק אינו מיושר באופן

מושלם, צפוי כי 4 פינות המלבן יתוארו ע"י 4 הנקודות המתארות את התנאים המצויינים. כלומר במקרה זה מצאנו את 4 פינות המסגרת המבוקשות.



איור 1 המחשה מציאת פינות לפי 4 נקודות

אחרת, במקרה בו 4 הנקודות אינן מתאימות לייצוג מלבן, כנראה בעקבות שגיאת מדידה או בעקבות המצב התיאורטי בו הדף מיושר באופן מושלם והפתרון הכללי כושל בו. נחפש את 4 הפינות בקונטור אשר הזווית שלהן מתאימה לייצוג פינה וגם הן הכי קרובות במטריקה אוקלידית לפינה המתאימה במסגרת התמונה. לדוגמא הפינה השמאלית עליונה בקונטור צריכה להיות קרובה לפינה השמאלית עליונה של מסגרת הדף. בפתרון זה קיבלנו כי עבור תמונה בה הדף המצולם נמצא בזווית קרובה לזווית הדרושה (ישרה), ניתן לזהות את פינות הדף לפי נקודות שיכולות ליצג פינה לפי הזוויות שהן יוצרות עם שכנותיה ונמצאות קרוב לפינת מסגרת התמונה המתאימה. נדגיש כי יש צורך בתנאי כפול לגבי זווית וקירבה לפינה עקב אי דיוקים ורעש בתמונה.



איור 2 המחשה מציאת פינות לפי זווית ומרחק מפינה

חישוב להזזת המצלמה

לאחר זיהוי מסגרת הדף, ברצוננו לחשב את מיקום המצלמה לעומת האובייקט. נעשה זאת לפי ההטלה של העולם התלת מימדי על גבי העולם הדו מימדי המתואר במצלמה. נשתמש במודל הטלה פרספקטיבית (Perspective Projection) המקובל לתיאור ההטלה המתקבלת בתהליך הצילום.

התמרה פרספקטיבית

מודל מופשט להתמרה מעולם התלת מימדי לדו מימד המיוצג במצלמה.

עבור קואורדינאטות לא הומוגניות ההטלה מתוארת ע"י

$$\bar{x} = P_z(p) = \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix}$$

עבור קואורדינאטות הומוגניות ההטלה מתוארת ע"י

$$x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} p$$

תכונות מלבן תחת התמרה פרספקטיבית

נראה כי תחת הטלה פרספקטיבית מלבן שומר על תכונותיו אם ורק אם הוטל בזווית אנכית.

$$\begin{aligned}
&Given: \\
&\text{Rectangle } p_i; i \in \{0,1,2,3\} \\
&P(p_i) \text{ is Rectangle IFF} \\
&\vec{A} = P(p_0) - P(p_1) \\
&\vec{B} = P(p_0) - P(p_2) \\
&\vec{A} \perp \vec{B} \Leftrightarrow \vec{A} \cdot \vec{B} = 0 \\
&\Leftrightarrow \overbrace{(x_0 - x_1)(x_0 - x_2) + (y_0 - x_1)(y_0 - y_2)}^{=0} + (z_0 - z_1)(z_0 - z_2) = 0 \\
&\ldots \Rightarrow \\
&(z_0 - z_1)(z_0 - z_2) = 0 \wedge (z_1 - z_0)(z_1 - z_2) = 0 \\
&\wedge (z_2 - z_3)(z_2 - z_0) = 0 \wedge (z_3 - z_2)(z_3 - z_1) = 0 \\
&\Rightarrow \left\{ \begin{array}{c} \{\{z_0 = z_1\} \vee \{z_0 = z_2\}\} \wedge \{\{z_0 = z_1\} \vee \{z_1 = z_2\}\} \\ \wedge \\ \{\{z_2 = z_3\} \vee \{z_0 = z_2\}\} \wedge \{\{z_1 = z_3\} \vee \{z_3 = z_2\}\} \end{array} \right\}
\end{aligned}$$

$$A)\{z_0 = z_2\} \wedge \{z_1 = z_2\} \Rightarrow \{z_1 = z_3\} \vee \{z_3 = z_2\} \Rightarrow \boxed{z_0 = z_1 = z_2 = z_3}$$

$$B)\{z_0 = z_1\} \wedge \{z_2 = z_3\} \Rightarrow \boxed{z_0 = z_1, z_2 = z_3}$$

$$\left(\frac{x_0 - x_1}{z_0}\right)^2 + \left(\frac{y_0 - y_1}{z_0}\right)^2 = \left(\frac{x_2 - x_3}{z_2}\right)^2 + \left(\frac{y_2 - y_3}{z_2}\right)^2$$

$$\Rightarrow z_0 = z_2$$

$$\Rightarrow \boxed{z_0 = z_1 = z_2 = z_3}$$

$$Q.E.D$$

כלומר בהנתן מלבן בעולם התלת מימדי, לאחר ההטלה פרספקטיבית המלבן משמר על תכונותיו אמ"מ כל קודקודי המלבן צולמו ממרחק זה, קרי מישור המלבן צולם בזווית אנכית.

הטייה

נשתמש בתוצאות הנ"ל על מנת לחשב את מיקום המצלמה ביחס למישור הצילום ולהנחות את המשתמש כיצד עליו להזיז את המצלמה.

כפי שהראנו צורת המלבן תשמר תחת ההטלה, בצילום אנכי למישור המבוקש. אחרת נקבל כי צלע אחת קטנה מהצלע שמולה. במקרה שכזה נדע כי יש להטות את המצלמה כלפי הצלע הקטנה יותר כדי לקרב את הנקודות הללו עד לזווית הרצויה.

אנו נמנעים מחישובים מסובכים כדי לחסוך בפעולות מעבד וכן כדי לשמור על פשטות שתמנע סיבוך ובלבול המשתמש.

זום

כאשר נזהה כי התמונה אינה מנצלת אחוז גבוהה מספיק משטחה ללכידת שטח ופרטי הדף המבוקש. נבקש מהמשתמש לקרב את המצלמה עד לניצול השטח כראוי.

הזזה

במקרה זה נחשב את מרחק כל צלע מהמסגרת הקרובה לה, במידה והמרחק בין הצלעות מקבילות מהמסגרת המתאימה להן אינו קרוב אחד לשני. נבקש מהמשתמש להזיז את המצלמה בכיוון המתאים.

ביבליוגרפיה

[1] Matlab documentation

[2] Opencv documentation

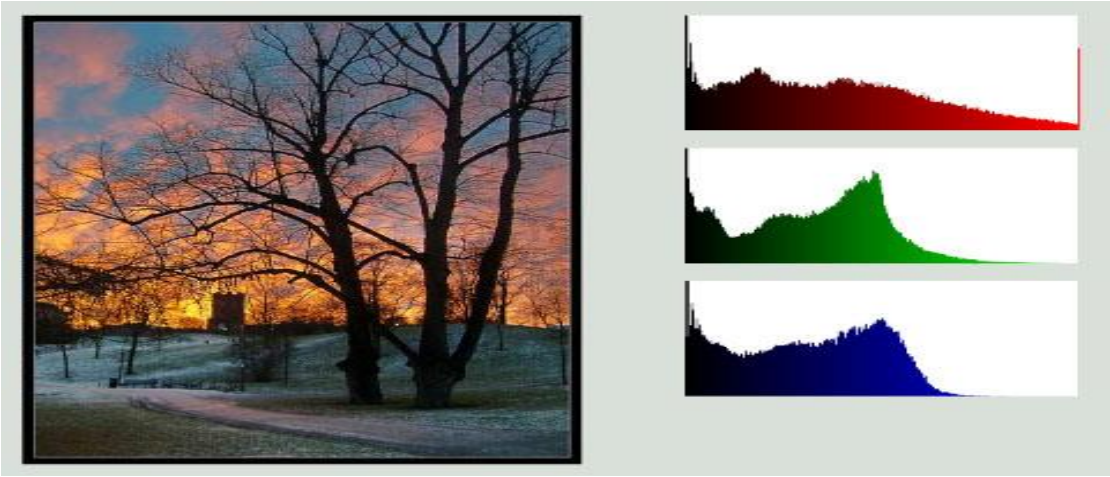
[3] Android developer

[4] Computer Vision: Algorithms and Applications; Richard Szeliski

נספחים

היסטוגרמה

הינה צורת הצגה גרפית של נתונים, בעזרת תרשים עמודות. בהקשר של תמונות, נהוג להציג גרף לכול ערוץ בתמונה. כאשר הגרף מיצג כמה פיקסלים הם בעלי ערך מסויים.



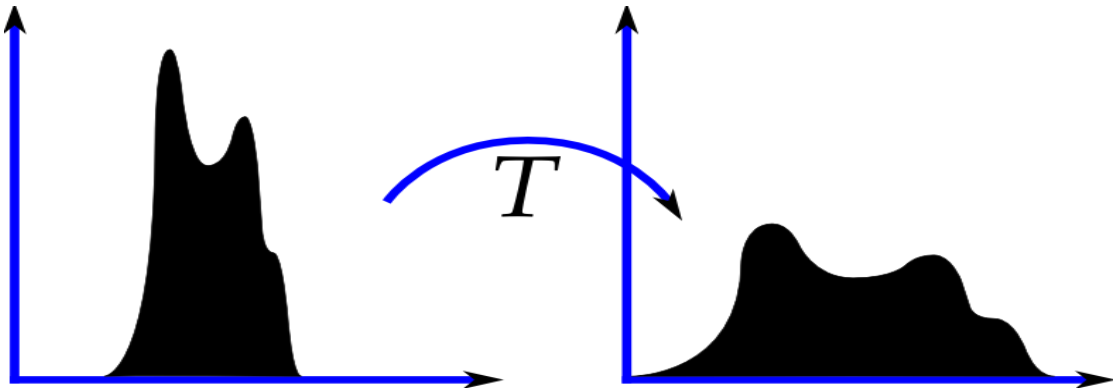
שיוויון היסטוגרמות

עבור תמונות רמות אפור, ניתן למצוא טרנספורמציה אשר תביא לניצול מירבי של התחום הדינמי בתמונה

$$y = T(x) = \sum_{j=0}^{x} \frac{n_j}{n}$$

כאשר n מייצג את כמות הפקסלים הכללית, n_j מייצג כמות הפיקסלים בעלי הערך j וx מייצג את ערך רמת האפור לה מחשבים טרנספורמציה. התוצאה המתקבלת היא בין [0,1] לכן יש לבצע לה הזזה מתאימה לפי:

$$y' = y\left(\max\{x\} - \min\{x\}\right) + \min\{x\}$$



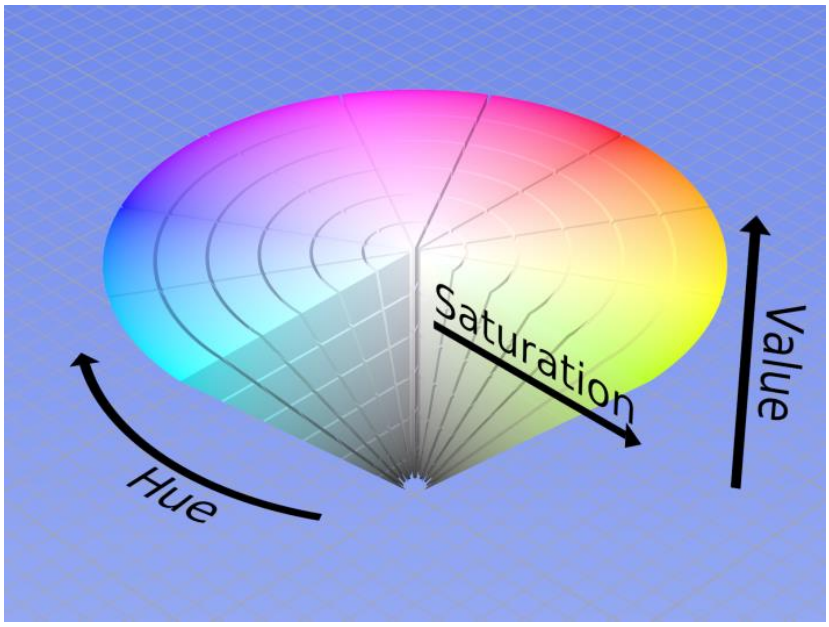
מרחב צבע HSV

בנוסף למרחב הצבע המקובל אדום-ירוק-כחול. קיים מרחב צבע נוסף HSV.

H- Hue גוון, צבע

S- Saturation רוויה, מושג בקולומטריה המתאר את האינטנסיביות של צבע מסויים.

V- Value ערך, בהירות.



ניתן להמיר ממרחב RGB למרחב HSV ע"י אוסף נוסחאות

$$V \leftarrow \max(R, G, B)$$

$$S \leftarrow \begin{cases} \frac{V - \min(R,G,B)}{V} & \text{if } V \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$H \leftarrow \begin{cases} 60(G - B)/(V - \min(R, G, B)) & \text{if } V = R \\ 120 + 60(B - R)/(V - \min(R, G, B)) & \text{if } V = G \\ 240 + 60(R - G)/(V - \min(R, G, B)) & \text{if } V = B \end{cases}$$

מסנן חציון

מסנן לא לינארי, המקובל בעיבוד תמונה לסינון רעש. עבור כל פיקסל לוקחים את הפיקסל ואת שכניו לפי גודל המסנן הנבחר (גודל המסנן צריך להיות אי זוגי). מסדרים הערכים הללו בסדר עולה\יורד ובוחרים להחליף את הערך הפיקסל המקורי בערך החציון קרי הערך שנמצא באמצע הסידור (בהכרח יש אחד כזה משום שגודל המסנן אי זוגי).

אלגוריתם Canny

אלגוריתם רב שלבי בעיבוד תמונה לזיהוי קצוות בתמונה, פותח ב1986 ע"י ג'ון קני.

האלגורתמים מקבל תמונת רמות אפור ומוציא כפלט תמונת שחור לבן בה הקצוות מסומנות בלבן והשאר שחור. האלגורתמים מקבל בנוסף לתמונה שני פרמטרים נוספים סף עליון וסף תחתון.

שלבי האלגורתמים:

1) ניקוי רעש בעזרת מסנן גאוסי.
2) הפעלת אופרטור SOBEL לחישוב גראדיאנטים בצירים x,y המהווים קירוב לפעולת הנגזרת, כך שמתקבל לכל נקודה בתמונה קירוב לנגזרת וקטורית.
3) חישוב גודל וכיוון הנגזרת לפי מטריקה אוקלידית.
4) בחירת נקודות שיחשבו חלק מהשפה:
   i. אם ערך הגראדיאנט גדול מהסף העליון.
   ii. אם ערך הגראדיאנט בין הסף העליון ותחתון וכיוון הנגזרת הוקטורית ממשיך ומקיים רצף יחד עם נקודות אחרות אשר סומנו כבר להיות חלק מהשפה.

תיעוד הקוד

בחלק זה נסביר את הקוד בקווים כלליים ונרד לפרטים לצורך הקלה בהבנתו ובתפעולו בעתיד.

תיאור על

הקוד מורכב משני מודולים עיקריים, הראשון כתוב בשפת java תוך שימוש ב-sdk של אנדרואיד והשני כתוב ב-C++.החלק הראשון אחראי על האפליקציה עצמה, ההתממשקות לאנדרואיד והאינטראקציה עם המשתמש בעוד החלק השני אחראי על עיבוד התמונה ומציאת המלבן (המסמך, אובייקט הצילום).

המנשק בין שני המודולים האלה נכתב ב-JNI, שפה המיועדת לשילוב קוד native בתוכנה כתובה ב-java. המנשק אצלנו פשוט מאוד והוא כולל פונקציה בודדת שמעבירה תמונה שנרכשה במכשיר האנדרואיד למודול עיבוד התמונה ומחזירה מאותו מודול ארבע נקודות המהוות ארבע פינות המלבן שזוהה.

קוד האפליקציה

MainScannerActivity – הפעילות היחידה של האפליקציה, זהו האובייקט הראשי של האפליקציה אשר מורץ עם הפעלת האפליקציה.

CameraPreview – אובייקט שאחראי על תצוגת תמונות ה-preview וכן שליפת פריימים מתוכו.

ScannerGraphics – אובייקט שאחראי על קבלת הנקודות שמייצגות את המלבן והצגת האנוטציות המתאימות על התמונה.

FrameContour – המודל שמייצג את המרובע, מתבסס על ארבע נקודות שנמצאו באמצעות האלגוריתם. הוא אחראי על מציאת האוריינטציה של המצלמה אל מול האובייקט המצולם על סמך הנקודות הנ"ל.

ContourView – האובייקט אחראי על הצגת הריבוע שנמצא באופן דינאמי.

TiltDown/Up/Left/RightView – אובייקטים האחראים על הצגת האנוטציה המכוונות את המשתמש כיצד להזיז את המצלמה כך שהיא תמוקם במדויק מעל המסמך המצולם.

TurnView – אובייקט שאחראי על הצגת אנוטציה המכוונת את המשתמש כיצד לסובב את המצלמה כך שתוצב בזווית הנכונה מעל המסמך המצולם.

ButtonView – אנוטציה שמהווה כפתור לצילום, אדומה כאשר המצלמה לא מכוונת טוב מעל המסמך וירוקה כאשר ניתן לצלם.

TakePictureSeries – thread שאחראי על שליפת הפריימים מתוך ה-preview ושליחתם לעיבוד תמונה.

ImageProcessor – thread האחראי על עיבוד התמונה. קורא למודול השני באמצעות ה-JniWrapper.

JniWrapper – אובייקט מעטפת לקריאה לפונקציות ה-Jni.

MainScannerActivity

OnCreate – נקרא עם פתיחת האפליקציה, מתפקד כמו c'tor לצורך אתחול המשתנים והשדות הפנימיים. אנחנו קובעים בו גם את הדגלים המשמשים להרצת האפליקציה (למשל קביעת אוריינטציה, נעילת המסך וכו').

OnCreateOptionsMenu, **OnOptionsItemSelected** – מתודות שנועדו לטיפול בתפריט האפליקציה.

המשך הקוד מסביר את עצמו, נפנה תשומת הלב לכך שמוגדרים שם מספר "מאזינים" לאירועים שמתרחשים באנדרואיד כמו צילום תמונה ולאירועים באפליקציה שאנחנו הגדרנו לצורך פעפוע המידע מ-threads אחרים.

TakePictureSeries

PreviewCallback – המתודה שמופעלת על כל פריים שנתפס. אחראי על שליחת התמונה ל-thread שאחראי על עיבוד התמונה וכן קבלת ארבע הנקודות בחזרה. במתודה זו אנחנו שולטים גם כן באיזו מהשיטות לעיבוד התמונה נבחר.

עיבוד התמונה – native code

<הסבר כללי קצר על המודול מה הוא מקבל ומה הוא מוציא>

פרמטרים:

`Offset`- המרחק בין נקודות בזמן מדידת זווית

`angleLimit`- הטווח במעלות סביב 90 מעלות להחשיב זווית כחשודה לפינה

`reasonableFactor`- אינדקס יתכנות, סף קירבה מתי להחשיב מרובע כקרוב מספיק למלבן

`sizeRatioToFrame`- קריטריון יחס בין גודל התמונה לגודל אובייקט מתי להחשיב אובייקט כמעניין

`CANN_THRESH_low_xxx`- פרמטר סף תחתון לאלגורתמים Canny, הXXX מוחלף בשם השיטה שבשימוש

`CANN_THRESH_high_xxx`- פרמטר סף עליון לאלגורתמים Canny, הXXX מוחלף בשם השיטה שבשימוש

`Hue_range`- פרמטר לקביעת תחום בו נבחר הערכים להדלקה בזמן שימוש במסכה

`RESIZE_IMAGE`- פרמטר לדגימה מחדש של התמונה הנרכשת המצלמה

`FRAME_SIZE`- רוחב הטלאי בשימוש עבור שיטת מסכה חיצונית

`INNER_SIZE`- גודל טלאי בשימוש עבור שיטת מסכה פנימית

פונקציות:

```
double euclideanDist(Point p1, Point p2):
```

פונקציה לחישוב מטריקה אוקלידית בין נקודות

קלט: שתי נקודות

פלט: המרחק האוקלידי בניהן

```
double getAngle(Point p1, Point p2):
```

פונקציה לחישוב זווית בין שתי נקודות

קלט: שתי נקודות

פלט: הזווית אשר הקו יוצר עם ציר הX

```
int FindBigestClosedContour(vector<vector<Point> > &contours, double area):
```

פונקציה מוצאת את אינדקס הקונטור הצפוי להיות מסגרת הדף

קלט: מערך הקונטורים, השטח של התמונה הכוללת

פלט: מפסר האינדקס של הקונטור החשוד להיות מסגרת הדף או 1- במקרה של אי מציאת קונטור מתאים

```
double CalcAngleOfSegment(vector<vector<Point> > &contours, int index, int maxPosition)
```

פונקציה מחשבת את הזווית בין נקודה לשכניה (בOffset)

קלט: מערך הקונטורים, אינדקס הנקודה, ואינדקס הקונטור הרלוונטי

פלט: זווית הנקודה ביחס לשכניה בקונטור

```
bool CoordinatesMakeSense(Point topLeft, Point bottomLeft, Point
topRight,Point bottomRight) :
```

פונקציה בודקת את ארבעת הנקודות החשודות לפינות

קלט: 4 נקודות

פלט: אמת אם הן עומדות בקריטריונים לתאר מסגרת מלבן, שקר אחרת

```
vector<vector<Point> > RetrieveContours(Mat &mSrc, int optionEQ, int
optionMask):
```

פונקציה מוצאת את הקונטורים שבתמונה, בהתאם לעיבוד המקדים שנבחר

קלט: התמונה ואופציות שיטות העיבוד המקדים אופצית איקולייזר ואופציית מסכה

פלט: וקטור של וקטורי נקודות לציון מערך הקונטורים

```
void UpdateRectangleCorners(double angle,Point ref, Point &topLeft, Point
&bottomLeft, Point &topRight,Point &bottomRight,int height, int width):
```

פונקציה את הנקודות החשודות לפינות במידת הצורך

קלט: נקודה והזווית שלה ביחס לשכנותיה

פלט\קלט: ארבעת הנקודות החשודות ליצוג פינות המסגרת

```
int* GetRectCorners(Mat &mSrc, int optionEQ,int optionMask):
```

פונקציה מחשבת את פינות מסגרת הדף לפי תמונה נתונה בהתאמה לשיטת העיבוד המקדים הנבחרת

קלט: התמונה ושיטת העיבוד המקדים

פלט: במקרה של מציאת 4 פינות; מוחזר מערך בגודל 8 המתאר את קורדיאנטות פינות המסגרת. אחרת מוחזר מערך בגודל 1 המכיל 1-

```
int* match_points(vector<vector<Point> > contours , int maxPosition, int
height, int width):
```

פונקציה מחשבת עבור קונטורים נתונים את 4 הפינות ליצוג פינות המסגרת

קלט: וקטור של וקטורי נקודות (מערך הקונטורים), אינדקס הקונטור הרלוונטי, ומימדי הרוחה והגובהה של התמונה

פלט: במקרה של מציאת 4 פינות; מוחזר מערך בגודל 8 המתאר את קורדיאנטות פינות המסגרת. אחרת מוחזר מערך בגודל 1 המכיל 1-

```
void calcMask(Mat &mSrc, Mat &result, int optionMask):
```

פונקציה המפעילה פעולת מסיכה מתאימה על התמונה

קלט: התמונה ואופצית המסכה המתאימה

פלט: התמונה לאחר הפעלת המסכה (התמונה מועברת לפי כתובת)