דחיסת נתונים – פרויקט סופי

מגישים:
דור אביטן
עומר סירפד
עומר אמסלם

- *להורדה של קבצי וידאו ושל קבצי הפרוייקט המלאים ניתן ללחוץ כאן.*
- *הרצת התוכנה: בכדי להריץ את התכנה עצמה, מספיק לפתוח את קובץ הJAR בשם VidCompressionProject.jar. עם זאת, כיוון שהדחיסה מיועדת לקבצי וידאו מסויימים) ברובם סטטיים(, מומלץ להוריד את כל הקבצים ולהשתמש בקבצי הווידאו שבתיקייה videoExamples)בוידאו רגיל, גודל הקובץ המכווץ יהיה גדול יותר מהמקורי(.*
- *הרצת הפרוייקט המלא – כל הקבצים לפרוייקט)כולל ספריות חיצוניות(נמצאים בZIP שבדרייב. רק להוריד, לעשות extract ולהריץ.*

1. הקדמה

בפרויקט זה פיתחנו מערכת לדחיסת וידאו ממצלמות אבטחה .
מצלמות אבטחה מצלמות הרבה וידאו סטטי. רוב הפריימים בווידאו חוזרים על עצמם עם שינויים מזעריים או בלי שינויים כלל .
לכן, חשבנו על הרעיון הבא: במקום לשמור כל פריים בווידאו, ניתן לשמור רק את התמונה הראשונה ולאחריה רק את הפריימים בהם היה שינוי ביחס לפריים השמור האחרון, ויחד איתם את מספר הפריימים בהם לא היה שינוי .
בזמן איחזור הוידאו נוכל לקחת כל פריים סטטי שמור ולהעתיקו לתוך הוידאו כמות פעמים כפי שרשום בקובץ .
כך בקובץ הפרוס נקבל ווידאו זהה למקורי)למעט אולי רעשים קטנים שהתעלמנו מהם בתהליך הדחיסה(.
בנוסף, עבור הפריימים המעטים שנשמרים בקובץ הדחוס, מימשנו דוחס ופורס PNG כדי לצמצם עוד יותר את רמת הדחיסה.

- לצורך עיבוד הווידאו)שליפת פריימים והמרתם למערך פיקסלים, ולאחר מכן ייצור הוידאו הסופי מהפריימים הדחוסים(השתמשנו בספריה החופשית OpenCV אותה ניתן למצוא כאן: https://opencv.org/releases.html
- בבניית דוחס הPNG התבססנו על הגדרות פורמט הPNG מתוך libpng.org :

PNG (Portable Network Graphics) Specification, Version 1.2

- בקידוד הוידאו הסופי השתמשנו בספרייה JCodec אותה ניתן למצוא כאן: http://jcodec.org
- ליצירת GUI להמחשת הדחיסה השתמשנו בWindowBuilder של JAVA.

הפרוייקט נכתב כולו בJava.

2. אופן פעולה

1.2 עיבוד ודחיסת הוידאו

1.1.2 פתיחת הוידאו נעשית באמצעות המחלקה VideoCapture של הספריהOpenCV. באמצעות מחלקה זו והשיטות read. ו
get(propID) , ניתן לשלוף בקלות מידע על הוידאו ופריימים מתוכו בקלות .

1.1.2 כתיבה לקובץ הדחוס נעשית באמצעות FileOutputStream שכותב מערכים של בתים ישירות לקובץ, אחד אחר השני.

דחיסת נתונים – פרויקט סופי

מגישים:
דור אביטן
עומר סירפד
עומר אמסלם

2.1.2 כצעד ראשון, נקרא נתון הFPS (Frame per second) מתוך קובץ המקור. הנתון נשמר בצורת בייט בתחילת הקובץ הדחוס.

2.1.3 הפריים הראשון נקרא ומתוכו מוצאים רוחב וגובה הוידאו. הם נכתבים בצורת integers כנתונים 2 ו3 לתוך הקובץ) 4 בתים כ"א.(

2.1.4 בשלב זה, הפריים הראשון נשלח אל דוחס ה PNG) המחלקה PNGer(לצורך עיבוד .

2.1.5 הפונקציה encodeFrame של המחלקה PNGer מחזירה לאחר תהליך הדחיסה מערך בתים שמהווה תמונת PNG)פירוט בסעיף 2.2 של המסמך(.

2.1.6 הפריים הנוכחי)הראשון(עובר תהליך של המרה לGrayscale וטשטוש(guassian blur). תהליך זה מונע הפרעת רעשים והופעתם כתנועה בוידאו. גם הפריימים הבאים יעברו תהליך זהה לצורך השוואה .

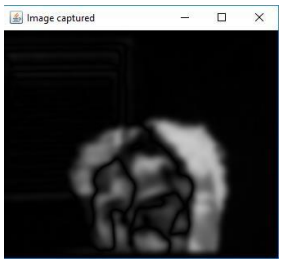
2.1.7 כניסה ללולאה: כל עוד לא הגענו לסוף קובץ הוידאו, תתבצע:

1.7.1.2 שמירת הפריים הקודם .

2.7.1.2 שליפת הפריים הבא .

3.7.1.2 הפיכת הפריים ל grayscale וטשטוש .

4.7.1.2 השוואה בין הפריים הנוכחי לקודם באמצעות השיטה absDiff . תוצאת ההשוואה היא פריים חדש המכיל את הערך המוחלט של ההפרשים בין הפריימים. מידע נוסף על השיטה ניתן למצוא כאן.



תוצאת absDiff על תזוזת ראש במצלמת רשת

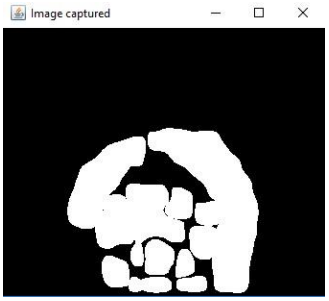
5.7.1.2 הדגשת ההפרשים באמצעות השיטה threshold של OpenCV.

Imgproc.threshold(src Mat, dst Mat,threshold,max_val,typ

מידע על threshold ניתן למצוא כאן.

5.8.1.2 לאחר קבלת פריים התוצאה, נרחיב אותו כדי לקבל תוצאה מדוייקת באמצעות dilate של OpenCV

מידע נוסף על שיטה זו ניתן למצוא כאן. *imgproc.dilate(src,dst,kernel,anchor,iterations)*



תוצאת השיטה Thresh ואחריה Dilate

6.7.1.2 נשתמש בשיטה findContours כדי להפוך כל פוליגון בתמונת הthreshold לאובייקט בתוך מערך שנוכל לנתח:

דחיסת נתונים – פרויקט סופי

מגישים:
דור אביטן
עומר סירפד
עומר אמסלם

מידע נוסף findContours ניתן למצוא כאן. Imgproc.findContours(Mat image, List<MatOfPoint> contours, Mat hierarchy, int mode, int method)
על

7.7.1.2 נעבור על כל אחת מהצורות בלולאה ונבדוק את גודל שטחן. אם מצאנו שגודל שטחן גדול מרמה מסויימת)אותנו
הגדרנו על ידי ניסוי וטעייה כך ששינויים בתמונה יענו על הקריטריון, אך "רעש" רנדומלי לא(– קיימת תנועה בווידאו שקרתה בין הפריים הקודם לנוכחי. אם אכן מצאנו תנועה:

1.7.7.1.2 נכתוב לקובץ את מספר הפריימים מהתמונה האחרונה עד לנוכחית)פריימים שלא הייתה בהן תמונה(– integer בצורת 4 בתים .

2.7.7.1.2 נקודד את הפריים לPNG ונכתוב לקובץ את גודלו) integer בצורת 4 בתים(ואת תוכנו)גודל משתנה, מערך בתים.

8.1.2 אחרי שסיימנו לבצע ניתוח תנועה וכתיבת כל הפריימים הנחוצים, נכתוב לקובץ כמה פעמים הופיע הפריים האחרון.

9.1.2 מבנה הקובץ הסופי:

FPS	Width	Height	Data size	Data	# of times to show frame	.....	Data Size	Data	# of times to show frame
1 byte	*4 bytes*	*4 bytes*	*4 bytes*	*Byte[]*	*4 bytes*		*4 bytes*	*Byte[]*	*4bytes*

2.2 דוחס PNG

1.2.2 דוחס הPNG מקבל מהמחלקה VideoCompressor מערך בתים כאשר כל בית מייצג ערך של צבע וכל פיקסל בתמונה המקורית מהווידאו מיוצג על ידי 3 צבעים בפורמט BGR – Blue,Green,Red כלומר פיקסל=3 בתים.

2.2.2 כצעד ראשון, נשמרים פרטי התמונה – רוחב, גובה ומאותחל מערך המידע הסופי.

3.2.2 הבתים נהפכים כך שBGR)הפורמט בו OpenCV שומרת את מידע התמונה(הופך ל RGB)פורמט הצבע הנפוץ יותר.(

4.2.2 ע"פי הגדרות פורמט PNG, הקובץ הסופי צריך להיות בנוי ממספר בלוקים, או chunks, כאשר לכל בלוק תפקיד שונה בקובץ. מבנה הבלוקים קבוע ובנוי מ: <גודל הבלוק>,<כותרת הבלוק><מידע הבלוק><ערך CRC לבלוק>
הבלוקים נכתבים אחד אחר השני לתוך מערך הבתים הסופי PNG Array :

1.4.2.2 חתימת קובץ הPNG – מערך בתים מוגדר מראש.

2.4.2.2 בלוק הHEADER)פירוט על כל פרמטר שנכתב ניתן למצוא בPNG Specs, סעיף 1.41.(.

3.4.2.2 בלוק הIDAT – במימוש שלנו נכתבים)בהתאם לגודל התמונה(מספר בלוקים של IDATכך שכל אחד מהם שוקל עד
32kb – שיטה זו חוסכת עומס לדוחס ולפורס. מידע הIDAT הוא למעשה מידע התמונה ונכתב בצורה הבאה:

דחיסת נתונים – פרויקט סופי

מגישים:
דור אביטן
עומר סירפד
עומר אמסלם

1.3.4.2.2 כל שורת פיקסלים)רוחב התמונה*3 בתים(נשלחת לשיטה filterSelector. שיטה זו מעבירה את השורה ל5 פילטרים שונים – None,Sub,Up,AVG,Paeth)פירוט על הפילטרים בPNG Specs סעיף 61.(. השיטה מבצעת חישוב של סכום ההפרשים בין בית לבית בערך מוחלט , עבור כל אחת משורות הפלט של כל אחד מהפילטרים ,וכך מחליטה מה הפילטר הטוב ביותר לאותה שורה)הפילטר בו ההפרש הוא הכי קטן(. פעולה זו נעשית כדי לשפר את יכולת דחיסת המידע)הקטנת האנטרופיה(. לאחר החלטה על פילטר מתאים, סוג הפילטר נשמר כבית בתחילת השורה)בין 0 ל4(ואחריו נכנס מידע השורה.

2.3.4.2.2 דחיסת המידע נעשית על ידי המחלקה Deflater של ג'אווה. מחלקה זו מכווצת את המידע על ידי שימוש באלגוריתם שמשלב LZ77 וקידוד הופמן)ספריית ZLIB(. מידע נוסף כאן.

3.3.4.2.2 בלוק ה IDAT נכתב לקובץ בהתאם לפירוט שנמצא בPNG Specs, סעיף 1.43..

4.4.2.2 לאחר כתיבת הIDAT)אחד או יותר(, נכתב בלוק הIEND בהתאם לפירוט בPNG Specs, סעיף 1.44.

5.2.2 לבסוף, המערך עובר RESIZE כדי להוריד בתים מיותרים בסופו, ומוחזר לVideoCompressor לצורך כתיבה לקובץ הווידאו הדחוס.

מבנה קובץ\מערך הPNG

PNG Signature	IHDR	IDAT	…	IEND
8 bytes	13 bytes	Varying size	Varying # of IDATs	12 bytes

3.2 פורס PNG

1.3.2 פורס ה PNG הוא מחלקה שמקבלת בבנאי שלה מערך בתים)קובץ PNG(.

2.3.2 הפורס מחלץ את גובה ורוחב התמונה)שני integers שמיוצגים כ4 בתים שנמצאים בבתים 16,20 במערך(וממיר אותם לint בעזרת המחלקה bytes2int .

3.3.2 החל מבית 33 מתחילה חתיכת הIDAT של התמונה)או מספר חתיכות, בהתאם לתמונה(. הפורס משתמש בשיטה extractIDAT כדי לחלצן:

1.3.3.2 השיטה מקבלת מערך בתים וסמן בו מתחילות חתיכות הIDAT.

2.3.3.2 השיטה מחלצת מהמערך את גודל חבילת הIDAT הבאה) int שמיוצג כ4 בתים ומתורגם על ידי bytes2int(.

3.3.3.2 השיטה מחלצת את מידע החבילה לפי הגודל בסעיף 2.3.3.2 ומוסיפה אותו לווקטור.

4.3.3.2 השיטה חוזרת על הצעדים עד שמגיעה לגודל חבילה של 0.

4.3.2 כעת הווקטור מכיל את כל חבילות הIDAT של התמונה)מידע התמונה(במצב דחוס. כדי לפענח, המחלקה קוראת לשיטה decodeIDAT:

1.4.3.2 השיטה מקבלת את הגודל הצפוי של המידע המפוענח לפי החישוב גובה*רוחב*3)כמות פיקסלים * בית לפיקסל(+ גובה)לכל שורה יש בית נוסף שמציין את סוג הפילטר.(

2.4.3.2 השיטה מחברת את כל המערכים בווקטור לתוך מערך בתים אחד גדול כהכנה לפענוח .

3.4.3.2 השיטה מפענחת את המידע בשימוש בשיטה java.util.zip.Inflater ומחזירה את המידע המפוענח.

5.3.2 כעת המידע מפוענח אך ערכי הפיקסלים עדיין מופיעים בצורה מפולטרת, כל שורה לפי הפילטר שלה. לצורך חזרה לצורה המקורית קוראים למחלקה unfiltered ולשיטה unfilter:

1.5.3.2 השיטה מקבלת את מערך המידע, גובה ורוחב התמונה.

2.5.3.2 השיטה מתחילה לעבור על התמונה בלולאה עבור השורות:

1.2.5.3.2 ראשית מחולץ הבית הראשון שמסמן את סוג הפילטר.

2.2.5.3.2 לאחר מכן, ע"פ סוג הפילטר שנקרא, מתבצעת הפעולה ההופכית לפילטר)כך לדוגמא עבור פילטר sub נעתיק את שלושת הבתים הראשונים, ועבור כל צבע בבתים הבאים) RGB(נחבר את הבית התואם בפיקסל הקודם יחד עם ההפרש כדי להגיע לבית הנוכחי)ההפרש הוא המידע בתא הנוכחי במערך data((.

3.2.5.3.2 לאחר הפענוח נמיר את המערך לאובייקט MAT של OpenCV כדי שנוכל לעבודה עם הספרייה בהמשך.

דחיסת נתונים – פרויקט סופי

מגישים:
דור אביטן
עומר סירפד
עומר אמסלם

6.3.2 לבסוף המחלקה מחזירה את אובייקט הMAT שנוצר.

4.2 <u>פורס וידאו</u>

1.4.2 פורס הווידאו מקבל נתיב ושם לקובץ דחוס)סיומת odo(. 2.4.2 הפורס פותח את הקובץ על ידי שימוש בRandomAccessFile והשיטות readInt,readFully,readByte)מידע נוסף בתיעוד של RandomAccessFile(.

3.4.2 כל עוד לא הגענו לסוף הקובץ:

1.3.4.2 הפורס קורא את בית הFPS, ואת גובה ורוחב התמונה) 4 בתים כל אחד(.

2.3.4.2 הפורס קורא את גודל התמונה הבאה) 4 בתים.(

3.3.4.2 הפורס קורא את מידע התמונה הבאה)לפי הגודל מהסעיף הקודם(

4.3.4.2 הפורס קורא את מס' הפעמים לשכפול התמונה בווידאו הסופי) 4 בתים(

5.3.4.2 הפורס שולח את מידע התמונה לפורס הPNG לצורך הפיכה לאובייקט MAT שניתן לשלוח לVideowriter של

OpenCV

6.3.4.2 האובייקט VideoWriter של OpenCV פותח קובץ וידאו חדש ומתחיל ליצור בו פריימים לפי המידע שחולץ מהקובץ הדחוס ונתון מוגדר מראש - FourCC)4 בתים של קידוד הווידאו הרצוי( )תיעוד VideoWriter(.

7.3.4.2 חזרה על סעיף 3.3.2 לתמונות הבאות.

3. <u>מסקנות</u>

- דחיסת הוידאו עם האלגוריתם שמימשנו נותן תוצאות טובות יותר ככל שיש פחות זמן תנועה מתוך הזמן הכולל של הוידאו.
- פריסת הוידאו איטית מאוד - אלגוריתם הקידוד של JCodec מאוד איטי.
- בחרנו לדחוס את התמונות בפורמט PNG לפי המלצת המרצה לפורמט עדכני ולא מיושן. בפועל היה עדיף לדחוס בJPG מה שהיה נותן יחס דחיסה טוב הרבה יותר.
- קיים איבוד של מידע מסויים)תנועות קטנות יותר מתנאי הסף שהגדרנו(– מידע זה זניח בהתחשב בעובדה שמדובר בווידאו ממצלמות אבטחה .