Freeze Frame Effect

פרויקט תוכנה בראיה ממוחשבת, עיבוד תמונה ווידאו

נכתב על ידי לילך הולצמן ויותם יעקב

בהנחיית ד"ר תמר צמח

תוכן עניינים

קצת רקע

ראיה ממוחשבת היא תחום מחקר מרכזי ועשיר בענף מדעי המחשב, העוסק בעיבוד של תמונות מהעולם האמיתי וחילוץ ופענוח מידע הנמצא בהן.

ההשראה העיקרית של הענף היא מערכת הראיה האנושית (בשילוב כוח העיבוד של המוח) המסוגלת לפענח עצמים חזותיים, לפרש אותם ולעבד אותם במהירות ויעילות שאין שנייה לה.

הטכנולוגיות שקיימות כיום אומנם לא מתקרבות ליכולות הזיהוי והלמידה של מערכת הראיה האנושית, אך הן כן עולות ביכולתן להשתמש במסדי נתונים ענקיים ושליפה מזיכרון. למשל, תוכנת מחשב פשוטה תוכל לזהות טביעת אצבע בודדה מתוך מאגר של מיליוני טביעות אצבע בקלות, בעוד שעין אנושית תתקשה לזהות בין זוג טביעות.

שימושים נפוצים של ראיה ממוחשבת: הדמיית תלת-מימד, ניתוח וזיהוי תנועה, זיהוי פנים וזיהוי ביומטרי בכללי, פענוח כתב יד, זיהוי מחלות עור ועוד הרבה.

למה בחרנו בפרויקט

כשהחלנו לחפש נושא לפרויקט, היה לנו ברור שנבחר בפרויקט מתחום הראייה הממוחשבת. הבחירה בעיבוד מסוג זה של הוספת אפקט לסרטון וידאו מאפשרת לנו לעסוק ולהשתמש בכלים רבים הן מתחום עיבוד הוידאו והתמונה, והן מתחום הראייה הממוחשבת. ובנוסף לקבל כתוצאה מיידית סרטון פלט ויזואלי מספק.

את האפקט ראינו לראשונה בסרטון ויראלי באינטרנט. בסרטון נראות מספר בבואות סטטיות של אדם, פזורות לאורך מסלול. בעת התקדמות הסרטון רואים את האדם נע לאורך אותו מסלול, ובכל פעם שהוא מגיע לבבואה של עצמו, הוא "אוסף אותה" והיא ממשיכה איתו.

אימצנו את האפקט לליבנו והחלטנו לממש את האפקט בעצמנו בתוכנה, כך שבאופן אוטומטי כשמשתמש קצה יכניס כקלט סרטון וידאו קצר, התוכנה תזהה את האובייקט הנע בסרטון, תבחר מספר רגעים עתידיים מהסרטון, תזהה את מיקומי האובייקט באותם רגעים עתידיים, ואז תחתוך את האובייקט היכן שימצא במיקומו העתידי ותדביק אותו על כל אחד מהפריימים בסרטון עד הרגע בו האובייקט הנע יגיע לאותה נקודת זמן.

בחרנו לממש את הפרויקט בPython ממספר סיבות. האחת היא שרצינו להתנסות בשפה שלא למדנו בזמן התואר, והשנייה היא שבעת מחקר מקדים ראינו שתכנות אפליקציות בסביבת אנדרואיד מתממשק עם Python והקדמנו לתכנן את האפשרות שנממש גם אפליקציה נגישה למשתמשי אנדרואיד.

קהל היעד

מטרתנו בפרויקט הייתה הנגשה של האפקט למשתמשים ללא רקע בתחום התכנה.

התכנה מתאימה למשתמש הפרטי המעוניין באפקט וידאו מיוחד שייצור עניין בסרטונים קצרים שצילם.

וכמו כן, לשימושים מסחריים. כמו, למשל, סרטוני פרסומת למוצרי ספורט.

2. שלב אחר שלב

a.

סרטון הקלט ובחירת נקודות ההקפאה

על סרטון הקלט:

הסרטון יבחר ויועלה על ידי משתמש הקצה. הסרטון המצופה הוא סרטון קצר שאורכו מספר שניות.

על הסרטון להיות מצולם בצילום סטטי (למשל על חצובה), ויימצא בו אובייקט בתנועה מתמדת. תנועתו תוקפא במספר נקודות זמן.

כמה נקודות הקפאה?

לאחר שבחנו מספר סרטונים שונים באורכם, שונים במהירות תנועת האובייקט, ובפריסה המרחבית של התנועה בפריימים, מספר נקודות ההקפאה שהובילו לתוצאה שהיינו מרוצים ממנה מבחינה אסטטית הסתבר שמשתנה מסרטון לסרטון בהתאם לתנועה. בחרנו באופן שרירותי להקפיא 4 נקודות זמן, משום שעל פי רוב היינו מרוצים מתדירות הקפאה שכזו.

אילו רגעים נקפיא?

רגעים בהפרשי פרקי זמן שווים.

אם כן, השלב הראשוני כאמור, יהיה:

- ✓ קליטת סרטון
- ✓ גילוי מספר הפריימים (frames) שהסרטון מכיל.
- ✓ שמירת נקודות ההקפאה. כלומר, האינדקסים של הפריימים שנקפיא מתוך הסרטון.

```
class myVideo():
    """Capture video and create indices"""

    def __init__(self,video):
        self.videoCaptured = cv2.VideoCapture(video)

        videoLength = int(self.videoCaptured.get(cv2.CAP_PROP_FRAME_COUNT))

        self.freezeIndices=[0,0,0,0]

        for i in range(0,4):
            self.freezeIndices[i] = int((videoLength/5)*(i+1))
```

.b .2

סגמנטציה

המשימה המרכזית העומדת בפנינו, היא הפרדת האובייקט מהרקע. או במונח המקצועי בראייה ממוחשבת Segmentation.

עלינו, למצוא את מיקומי כל הפיקסלים השייכים לאובייקט, תוך הפרדתם מכל הפיקסלים שאינם חלק ממנו.

בספר הפרויקט, לצורך הדגמה, נדגים את האתגרים העומדים בפני המתכנת ואת כל השלבים ביצירת האפקט, על סרטון דוגמא של מתעמלת המבצעת סלטה לאחור על מכשיר הקורה.

התמונה הבאה, היא דוגמא לאחד הפריימים שנבחרו לצורך הקפאה.

צופה אנושי יניח בקלות כי העצם בתמונה אותו נרצה להקפיא הוא המתעמלת ולא הקורה או הלוגו (המופיע בצד ימין למעלה).

אולם, בראייה ממוחשבת, קיימים בפריים זה 3 עצמים ברורים.

לכן, האתגר הראשון, יהיה למקד תחילה את הסגמנטציה לאזורים הנמצאים בתנועה בלבד.

בעמודים הבאים נסביר בקווים כלליים את רצף הפעולות שביצענו ובסוף הפרק נסביר באופן מפורט כל שיטה.

.c .2

Motion Detection

השיטה:

על מנת לזהות תנועה נעבוד עם שני פריימים עוקבים. האחד הוא הפריים אותו אנו מעוניינים להקפיא, והשני יהיה הפריים שהופיע בדיוק לפניו. נחפש את ההבדלים בין הפריים הנוכחי לפריים הקודם. נבדוק אילו פיקסלים השתנו.

הקושי:

פיקסלים רבים משתנים מפריים לפריים בגלל שינויי תאורה וצל, ותזוזות מינוריות של אלמנטים ברקע.

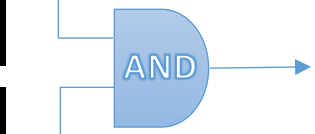
כדי למזער את מספר הפיקסלים שמצאנו כמעידים על תנועה אך אינם שייכים לאובייקט, נחזור על אותה פעולה עם הפריים אותו אנו מקפיאים, אך הפעם נחסר ממנו את הפריים העוקב שמופיע מיד אחריו.

לבסוף נצליב בין הפיקסלים שנמצאו בכל אחד מן החיסורים האלו, ונתייחס רק לפיקסלים המשותפים לשניהם.

```
for i in range (0,4):
    diff_prev.append(cv2.absdiff(previous_frame[i], current_frame[i]))
    diff_next.append(cv2.absdiff(next_frame[i], current_frame[i]))

mask = cv2.cvtColor(diff_prev[1], cv2.COLOR_BGR2GRAY)
mask2 = cv2.cvtColor(diff_next[1], cv2.COLOR_BGR2GRAY)

th = 1
imask =  mask>th
imask2 =  mask2>th

canvas = np.zeros_like(current_frame[1], np.uint8)
canvas2 = np.zeros_like(current_frame[1], np.uint8)
canvas[imask] = current_frame[1][imask]
canvas2[imask2] = current_frame[1][imask2]

dst = cv2.bitwise_and(canvas,canvas2)
```

חיסור הפריים הקודם

חיסור הפריים הבא



על התמונה המתקבלת, שגם היא אינה נקייה לחלוטין מרעשים נפעיל סדרת פעולות

1. המרה לתמונה בגווני אפור.
2. קביעת ערך סף. Threshold, אשר כל הגוונים בעלי בהירות גבוהה ממנו ייצבעו לבן, ונמוכה ממנו ייצבעו שחור.
3. שימוש במסנן ממצע, Gaussian, במטרה ליצור אפקט של החלקה (טשטוש) ובאמצעות כך לצמצם רעשים.
4. זיהוי קצוות בתמונה, ולאחר מכן, עיבוי אותם קצוות לכדי סקיצה גסה של האובייקט.
5. לבסוף, הצלבת התמונה הצבועה לבן באופן גס, עם תמונת **הקצוות של הפריים המקורי**.

2. d.

Edges Detection

השלבים בזיהוי קצוות:

מתמטית:

$$Edge\_Gradient\ (G) = \sqrt{{G_x}^2 + {G_y}^2}$$

- *Canny edge detection: אלגוריתם רב שלבי לזיהוי קצוות בתמונה, פותח ע"י ג'ון קני ב- 1986. שלבי האלגוריתם בקצרה:*
  - *חישוב פונקציית גרדיאנט, עבור כל נקודה בתמונה נקבל ווקטור נגזרת.*
  - *מציאת גודל וכיוון הווקטור.*
  - *על פי ערך סף שרירותי, נקודות בהן גודל הווקטור קטן מערך הסף נפסלות ומושמטות מהפלט, ונקודות הגדולות ממנו מסומנות כקצה בתמונה.*
  - *כל אחת מהנקודות שנותרו לאחר הסינון, מקורבות לאחד מארבעה כיוונים בדידים: אופקי אנכי או אלכסוני.*
  - *נקודות המסייעות ליצירת רצף של נקודות באותו כיוון מסומנות גם הן כקצוות.*

2. e.

הרכבת Mask

מיסוך בינארי

מיסוך הוא טכניקה בעיבוד תמונה המשמשת למחיקת הרקע של אובייקט מסוים בתמונה, על מנת לבודד את אותו האובייקט לשימושים שונים.

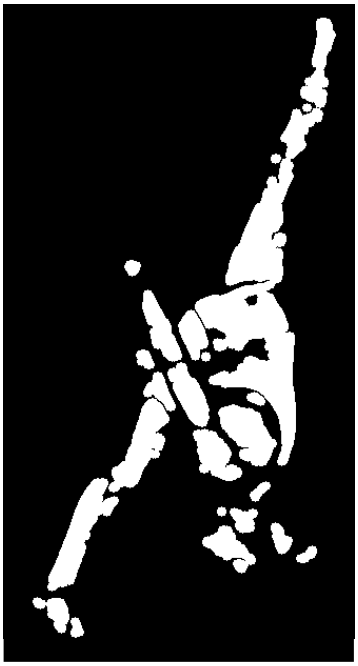
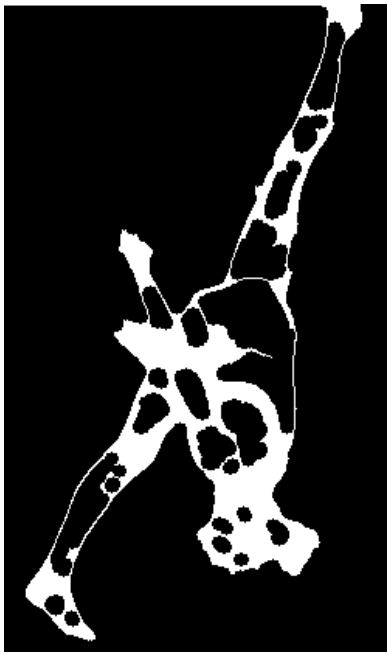
תמונה בינארית מתייחסת לתמונה ב-2 צבעים. שחור ולבן. הצבע הלבן בתמונה בינרית, הוא ה-'1' הלוגי שלנו, והצבע השחור, הוא ה-'0' הלוגי.

השאיפה שלנו היא ליצור תמונת שחור לבן שכל פיקסל לבן בה יסמל מיקום בתמונה המקורית השייך לאובייקט ואת הפיקסל באותו מיקום נרצה להעתיק, ואילו פיקסל שחור יסמל פיקסל שאין לנו עניין בהעתקתו.

שלבים ביצירת הMask

משכבר יש בינו מסגרת של התחום אותו אנו מעוניינים להקפיא, נותר לנו לצבוע את פנים התחום בלבן.

- נשתמש בשיטה *morphologyEx* על מנת לסגור תחומים פתוחים.
- נמלא את התחומים הסגורים של תמונת הקצוות בצבע לבן (בהירות 255).


- לבסוף, בשימוש בשער OR נחבר בין 2 התוצאות שקיבלנו. זה יהיה הMask הסופי.

.f .2

שימוש ב-Mask שיצרנו, וייצור סרטון הפלט

הדגמנו את אופן היצירה של Mask בודד. השלב הבא יהיה שימוש ב4 הMasks וארבעת הפריימים התואמים להם,

והדבקתם במקומות המתאימים בסרטון הפלט החדש.

נדביק את האובייקטים על הסרטון, בכל נקודת זמן שקודמת לרגע ההקפאה. וכך, כשהסרטון יתקדם, הדמות "תאסוף בדרך" את הבבואות שלה, בזו אחר זו.

```
##########################################################################
def pasteMasks(video,freezedFrames,masks):

    freeze1 = cv2.bitwise_not(freezedFrames[0])
    freeze2 = cv2.bitwise_not(freezedFrames[1])
    freeze3 = cv2.bitwise_not(freezedFrames[2])
    freeze4 = cv2.bitwise_not(freezedFrames[3])

    frame_list = []     # Initialize: output frames list

    # paste masks on video
    frameCount = 0
    while(video.videoCaptured.isOpened()):
        ret, frame = video.videoCaptured.read()
        if ret==True:

            frameCount += 1

            if frameCount < video.freezeIndices[0]:
                dst = cv2.bitwise_not(freeze1,frame, mask=masks[0])
            if frameCount < video.freezeIndices[1]:
                dst = cv2.bitwise_not(freeze2,frame, mask=masks[1])
            if frameCount < video.freezeIndices[2]:
                dst = cv2.bitwise_not(freeze3,frame, mask=masks[2])
            if frameCount < video.freezeIndices[3]:
                dst = cv2.bitwise_not(freeze4,frame, mask=masks[3])
            if frameCount > video.freezeIndices[3]:
                dst = frame

            frame_list.append(dst) # Insert output video frame by frame

            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
        else:
            break

    frame_list.pop()

    return frame_list
##########################################################################
```

בשלב זה, לפני שנייצא את סרטון הפלט נוסיף עוד אלמנטים אחרונים לעיצוב הוידאו.

במהלך התקדמות הוידאו, הדמות אספה את הבבואות של עצמה. כשתסיים לאסוף את כולן ותגיע לסוף הוידאו המקורי, נוסיף קטע וידאו קצר נוסף. בקטע הוידאו הנוסף נשתמש בוידאו שיצרנו עד כה, ונריץ את הפריימים מהסוף להתחלה, כך שהדמות תזוז בהילוך לאחור, ותפרוש את הבבואות בחזרה על פני מסלול התנועה שבוידאו.

את קטע הוידאו שנע לאחור, נריץ במהירות גבוהה פי 3.

המהירות הגבוהה מתאפשרת כשמוותרים על 2/3 מהפריימים. לכן, נשאיר פריים אחד, ועל שני הבאים אחריו נוותר. בצורה מחזורית.

```
#################################################################
def getReversedSpeedUpVideo(frame_list):

    reversed_frame_list = []
    cnt=0
    # reverse the order of the element present in the list
    frame_list.reverse()
    # looping in the List of frames.
    for frame in frame_list:
        cnt+=1
        if cnt%3==0:
            reversed_frame_list.append(frame)

    return reversed_frame_list
#################################################################
```

.g .2

המסננים והכלים בהם השתמשנו

```
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(9,9))
closed = cv2.morphologyEx(edged, cv2.MORPH_CLOSE, kernel)
th, im_th = cv2.threshold(closed, 220, 255, cv2.THRESH_BINARY);
```

- *cv2.morphologyEx: על ידי שימוש בגרעין שהגדרנו, הפונקציה הזו מסייעת לסגור חורים ונקודות שחורות על רקע לבן, בתמונת שחור לבן שלנו.*
- *cv2.threshold: מייצר תמונה בינארית מהתמונה הסגורה, על פי ערכי הסף שהגדרנו.*

```
mask = np.zeros((h+2, w+2), np.uint8)
cv2.floodFill(im_floodfill, mask, (0,0), 255);
im_floodfill_inv = cv2.bitwise_not(im_floodfill)
im_out = im_th | im_floodfill_inv
```

- *mask: נאתחל מטריצה ריקה בגודל המסגרת התוחמת את האובייקט בתנועה.*
- *cv2.floodFill: הפונקציה floodFill מציפה חורים סגורים בתמונה בצבע שמוגדר על ידי המשתמש.*
- *cv2.Bitwise_not: אופרציה בינארית על התמונה שלנו, הופכת את כל הביטים הדלוקים לכבויים ולהפך.*
- *בשורה האחרונה אנחנו מאחדים את שתי התמונות, לכדי המסכה הסופית.*

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
blurred = cv2.GaussianBlur(gray, (3, 3), 0)
```

- *מטשטש גאוסיאני: שימוש בחלון (kernel) בגודל 3x3 למיצוע פיקסלים שכנים וסינון רעשים גאוסיאניים, מה שיוצר אפקט של טשטוש והחלקה של קצוות. קצוות חלקים יותר קלים לזיהוי במזהה קצוות.*

- *Canny edge detection: אלגוריתם רב שלבי לזיהוי קצוות בתמונה, פותח ע"י ג'ון קני ב- 1986. שלבי האלגוריתם בקצרה:*
  - *חישוב פונקציית גרדיאנט, עבור כל נקודה בתמונה נקבל ווקטור נגזרת.*
  - *מציאת גודל וכיוון הווקטור.*
  - *על פי ערך סף שרירותי, נקודות בהן גודל הווקטור קטן מערך הסף נפסלות ומושמטות מהפלט, ונקודות הגדולות ממנו מסומנות כקצה בתמונה.*
  - *כל אחת מהנקודות שנותרו לאחר הסינון, מקורבות לאחד מארבעה כיוונים בדידים: אופקי אנכי או אלכסוני.*
  - *נקודות המסייעות ליצירת רצף של נקודות באותו כיוון מסומנות גם הן כקצוות.*

הקוד:

```
def auto_canny(image, sigma=0.95):
    """ User handled Canny function """
    # compute the median of the single channel pixel intensities
    v = np.median(image)

    # apply automatic Canny edge detection using the computed median
    lower = int(max(0, (1.0 - sigma) * v))
    upper = int(min(255, (1.0 + sigma) * v))
    edged = cv2.Canny(image, lower, upper)

    # return the edged image
    return edged
```

3. <u>קשיים ואתגרים</u>

a. מפני שביצירת המאסק הסתמכנו על צביעת תחומים סגורים, לעיתים אנחנו צובעים קטעים קטנים ששייכים לרקע ומוקפים באובייקט.
b. בפריימים בהם גוון הרקע דומה לקו המתאר של האובייקט, אנחנו לא מצליחים למצוא שם קו קצה.
c. שנינו חדשים בעולם הראיה הממוחשבת, נדרש זמן רב ללימוד ספריות ומתודות חדשות.
d. בהמשך לסעיף הקודם, כל פעם שלמדנו על שיטה חדשה נדרשנו לבצע שינויים בקוד, הרבה ניסוי וטעיה שגרמו לעיכובים בלוחות הזמנים.

הצעות לשיפור:

a. בניית אפליקציה שתאפשר למשתמש לבחור את נקודות ההקפאה, כולל בחירה של אזור ההקפאה הרצוי, ובעזרת הכלים שבנינו יהיה אפשר לדייק את גבולות האובייקט הנבחר.
b. מימוש הפרויקט ב-CPP, מאפשר שימוש במגוון רחב יותר של מתודות ושיטות עיבוד תמונה.

4. נספחים: Python Code

```
################################################################################
def createMasks(myVideo):

    frameCount = 0
    iterations = 0
    freezedFrames=[0,0,0,0]
    masks=[0,0,0,0]

    while(myVideo.videoCaptured.isOpened()):
        ret, frame = myVideo.videoCaptured.read()
        if ret==True:
            frameCount = frameCount+1
            if iterations < 4:
                if frameCount == myVideo.freezeIndices[iterations]:

                    iterations = iterations+1
                    """PREPROCCESSING"""
                    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  #create
grayscale image
                    blurred = cv2.GaussianBlur(gray, (3, 3), 0)      #Use Gaussian
filter for blurring the image

                    edged=auto_canny(blurred)           #auto canny edge detector
                    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(9,9))
                    closed = cv2.morphologyEx(edged, cv2.MORPH_CLOSE, kernel)
                    th, im_th = cv2.threshold(closed, 220, 255, cv2.THRESH_BINARY);
                    im_floodfill = im_th.copy() # Copy the thresholded image.
                    h, w = im_th.shape[:2]

                    mask = np.zeros((h+2, w+2), np.uint8) # Mask used to flood
filling.
                    cv2.floodFill(im_floodfill, mask, (0,0), 255); # Floodfill from
point (0, 0)
                    im_floodfill_inv = cv2.bitwise_not(im_floodfill) # Invert
floodfilled image
                    im_out = im_th | im_floodfill_inv # Combine the two images to
get the foreground.
                    """END PREPROCCESSING"""
                    if iterations==1:
                        freezedFrames[0]=frame
                        masks[0]=im_out
                    elif iterations==2:
                        freezedFrames[1]=frame
                        masks[1]=im_out
                    elif iterations==3:
                        freezedFrames[2]=frame
```

```
                        masks[2]=im_out
                    elif iterations==4:
                        freezedFrames[3]=frame
                        masks[3]=im_out
            else:
                break
        else:
            break
    return freezedFrames, masks
################################################################################
def pasteMasks(video,freezedFrames,masks):
    """after generating masks, paste them on the frames and generate output video"""
    freeze1 = cv2.bitwise_not(freezedFrames[0])
    freeze2 = cv2.bitwise_not(freezedFrames[1])
    freeze3 = cv2.bitwise_not(freezedFrames[2])
    freeze4 = cv2.bitwise_not(freezedFrames[3])

    # Define the codec and create VideoWriter object
    out = cv2.VideoWriter('output.avi', get_video_type('output.avi'), 25,
get_dims(video.videoCaptured, '720p'))

    # paste masks on video
    frameCount = 0
    while(video.videoCaptured.isOpened()):
        ret, frame = video.videoCaptured.read()
        if ret==True:
            frameCount = frameCount+1
            """In this section we invert the frames with the relevant masks"""
            if frameCount < video.freezeIndices[0]:
                cv2.bitwise_not(freeze1,frame, mask=masks[0])
            if frameCount < video.freezeIndices[1]:
                cv2.bitwise_not(freeze2,frame, mask=masks[1])
            if frameCount < video.freezeIndices[2]:
               cv2.bitwise_not(freeze3,frame, mask=masks[2])
            if frameCount < video.freezeIndices[3]:
               cv2.bitwise_not(freeze4,frame, mask=masks[3])

            # Write output video
            out.write(frame)
            cv2.imshow('output',frame)
           if cv2.waitKey(1) & 0xFF == ord('q'):
                break
        else:
            break

    out.release()
################################################################################
```