עיבוד וניתוח וידאו – פרויקט

בפרויקט זה נדרשנו לבצע מספר פעולות על אובייקט מרכזי בסרטון, לפי השלבים הבאים:

1. Video Stabilization

ייצוב הסרטון המקורי כך שנקבל רקע נקי מרעשים ורעידות וזאת על מנת שנוכל להחסיר את הרקע בשלב הבא באופן אופטימלי. תוצאת שלב זה הינו סרטון מיוצב **stabilize.avi**.

1. Background Subtraction

החסרת הרקע כדי להפריד בין האובייקט לרקע הסרטון. תוצאות שלב זה: **binary.avi** - סרטון טרינארי (בגלל השימוש ב-trimap).

1. Video Matting

השמת רקע שונה מהרקע בו הסרטון המקורי צולם, עליו "יתלבש" האובייקט שהוצאנו בשלב הקודם. תוצאת שלב זה: סרטון שנקרא לו: **matted.avi.**

1. Object Tracking

עקיבה אחר האובייקט בסרטון מהשלב הקודם. תוצאת שלב זה: **OUTPUT.avi**

* לא מימשנו GUI בפרויקט זה. בכל אחד מהשלבים יש Progress bar שנועד להראות את התקדמות התהליך.

1. Video Stabilization  
  
השתמשנו במימוש של video stabilization מתרגיל בית 2.

StabilizedVid = LucasKanadeVideoStabilization(InputVid,WindowSize,MaxIter,NumLevels,NumOfFrames)

קלט:

1. סרטון הקלט – מחרוזת ובה שם סרטון הקלט. ברירת המחדל: INPUT.avi.

הציפייה היא לסרטון הצולם במצלמה לא מיוצבת (ללא חצובה) הכולל אובייקט מרכזי

המבצע תנועה ורקע סטטי.

1. - WindowSize (for LK)גודל הlocal neighborhood מסביב ל. pixel
2. MaxIter (for LK) – מקסימום איטרציות שאנחנו מתירים (לכל שלב בפרמידה).
3. NumLevels (for LK) – מספר השלבים בפרמידת התמונה.
4. NumOfFrames – מספר הפריימים לייצוב.

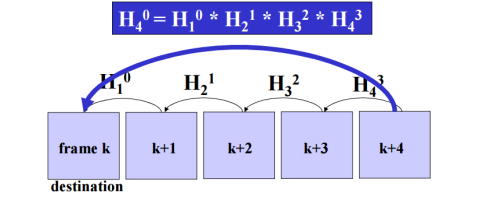
פלט: הפונקציה למעשה לא מחזירה דבר, אלא יוצרת סרטון מקורי מיוצב.

בהינתן וידאו, לכל פריים אנו מחשבים את ה- LK Optic flow בהתחשב ב-pervious warped frame ועושים warp לתמונה כולה.

הרחבה:

For example, if we have warped frame k to frame 1. Now, we compute the warp from frame k+1 to frame k, add the warp computed from frame k to frame 1 and use the combined warp, to warp frame k+1 to the coordinate system of frame 1.

Graphical representation:



בתהליך הייצוב השתמשנו בפונקציה:

(u,v) =LucasKanadeOpticalFlow(I1,I2,WindowSize,MaxIter,NumLevels)

שאותה גם כתבנו בתרגיל בית 2.

הפונקציה מחשבת את ה optical flow בשימוש ב Iterative Pyramid Lucas-Kanade Algorithm.

**Input:**

* I1,I2- images (we want to back-project from I2 to I1)
* WindowSize - size of local neighbourhood around the pixel (in which we assume that the optical flow is constant)
* MaxIter- maximum iterations that we allow (for each level of the pyramid)
* NumLevels- number of levels in the image pyramid

**Output:**

* (u,v) – Optical flow warp parameters (velocity fields)

וכדי ליישם את ה - Iterative Pyramid Lucas-Kanade Algorith התמשנו ב:

(du,dv)= LucasKanadeStep(I1,I2,WindowSize)

This function will compute optical flow using a single iteration of Lucas-Kanade Algorithm.

**Input:**

* I1,I2- images (we want to back-project from I2 to I1)
* WindowSize- size of local neighbourhood around the pixel (in which we assume that the optical flow is constant)

**Output:**

* (du,dv) – Optical flow warp parameters (velocity fields)

וגם השתמשנו ב:

I\_warp = WarpImage(I,u,v)

This function back projects image I2 using u and v. the purpose of this is to see how we go back from I2 to I1 by knowing the motion between both images. The result should be very similar to I1.

**Input:**

* I - image (u,v) – Optical flow warp parameters (velocity fields)

**Output:**

* I\_warp – warped image

2. Background Subtraction

קלט:

סרטון הקלט – מחרוזת ובה שם סרטון הקלט. ברירת המחדל: stabilize.avi.

הציפייה היא לסרטון שעבר ייצוב.

פלט:

הפונקציה מחזירה סרטון בינארי עם מסכת האובייקט על גבי רקע שחור וסרטון בו מוצג האובייקט בלבד.

לביצוע התהליך השתמשנו באלגוריתם בשם: BackgroundSubtractorMOG.

האלגוריתם מבצע: .Gaussian Mixture-based Background/Foreground Segmentation

האלגוריתם ממדל כל background pixel על ידי ערבוב שלK Gaussian distributions .

המשקולות של הגאוסיינים מייצגות את פרופרציות הזמן שצבעים אלו נותרו בתמונה.

הצבעים עם הסתברות גבוה יותר יהיו אלה שהיו יותר סטטיים ונותרו זמן רב יותר.

ליישום האלגוריתם יצרנו אובייקט background בעזרת שימוש בפונקצייה: cv2.createBackgroundSubtractorMOG

ולאחר מכן השתמשנו ב backgroundsubtractor.apply() בתוך לולאה העוברת על כל פריים בודיאו כדי לקבל את ה - foreground mask.

|  |
| --- |
| הפרמטרים עבור :createBackgroundSubtractorMOG   * **history** – Length of the history. * **nmixtures** – Number of Gaussian mixtures. * **backgroundRatio** – Background ratio. * **noiseSigma** – Noise strength (standard deviation of the brightness or each color channel). 0 means some automatic value. |

הפרמטרים עבור backgroundsubtractor.apply:

**Image** - Next video frame

**Fgmask** - The output foreground mask as an 8-bit binary image.

**learningRate** - The value between 0 and 1 that indicates how fast the background model is learnt. Negative parameter value makes the algorithm to use some automatically chosen learning rate. 0 means that the background model is not updated at all, 1 means that the background model is completely reinitialized from the last frame.

3. Matting

קלט:

1. סרטון הקלט – מחרוזת ובה שם סרטון הקלט. ברירת המחדל: stabilize.avi.
2. תמונת רקע – תמונה שתשמש לרקע חדש עליה יושתל האובייקט. ברירת מחדל: background.jpg.
3. סרטון הפלט – מחרוזת ובהה שם סרטון הפלט. ברירת מחדל: matted.avi.
4. מפת אלפא עבור כל פריים (trimap) – path לתיקייה שבה שמורים מפות אלפא.

פלט:

הפונקציה למעשה לא מחזירה דבר, אלא יוצרת סרטון בו נראה האובייקט על גבי הרקע החדש.

עבור מימוש video matting יש צורך ביצירת מפת שקיפות בעלת 3 ערכים. מפה זו מתארת עבור כל פיקסל האם שייך ל: background, foreground או undecided.

לכן מימשנו את הפונקציה: trimap() שמקבלת input video לאחר שעבר background subtraction ומחזירה את ה path לתיקייה שבה נשמרה מפת שקיפות עבור כל אחד מה-frames.

השלבים לביצוע תהליך ה trimap:

1. Converting the image into grayscale, then to binary (using cv2.threshold).
2. Morphological operations.
3. Finding contours, sorting, top to bottom, drawing the topmost contour only as mask.
4. Performing morphological operations to find out sure foreground and unknown area.
5. Saving the final mask of each frame (final\_mask = sure foreground + unknown region).

דוגמא לפריים שמיוצג על ידי מפת שקיפות כפי שקיבלנו שבו ניתן להבחין בחלוקה ל 3 ערכים (foreground, background, undecided):



איור 1

נרחיב על ה - Morphological transformations שבהן השתמשנו לצורך התהליך:

Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images. It needs two inputs, one is our original image, second one is called **structuring element** or **kernel** which decides the nature of operation.

**Erosion:**

The basic idea of erosion is eroding away the boundaries of foreground object. The kernel slides through the image (as in 2D convolution). A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded (made to zero).

So what happens is that, all the pixels near the boundary will be discarded depending upon the size of kernel. So the thickness or size of the foreground object decreases or simply white region decreases in the image.

**Dilation:**

It is just opposite of erosion. Here, a pixel element is ‘1’ if at least one pixel under the kernel is ‘1’. So it increases the white region in the image or size of foreground object increases. Normally, in cases like noise removal, erosion is followed by dilation. Because, erosion removes white noises, but it also shrinks our object. So we dilate it. Since noise is gone, they won’t come back, but our object area increases.

cv2.findContours:

Finds contours in a binary image.

cv2.drawContours:

The function draws contour outlines in the image if thickness≥0 or fills the area bounded by the contours if thickness<0

כדי להסיר רעשים בפריים השתמשנו בפונקציה: cv2.morphologyEx

opening = cv2.morphologyEx(image1, cv2.MORPH\_OPEN, kernel, iterations=1)

כאשר ‘Opening’זה פשוט שם לביצוע של erosion followed by dilation**.**

לאחר הסרת הרעשים השתמשנו ב ,Opening )הפלט של הפונקציה הנ"ל( כדי לבצע עליו את הפעולות הבאות:

הכנסנו אותו לפונקציה cv2.dilate וקיבלנו כפלט את ה sure background area.

הכנסנו אותו לפונקציה cv2.erode וקיבלנו כפלט את ה sure foreground area.

לאחר מכן מצאנו את ה unknown region שכולל את ערכי ה undecided.

עשינו זאת על ידי שימוש בפונקציה cv2.subtract, כאשר הכנסנו לפונקציה את הsure background area - , וה - ,sure foreground areaובתמורה הפונקציה ביצעה החסרה של האזורים החופפים וקיבלנו את האזור האפור שעוטף את הדמות כפי שניתן לראות באיור 1 בעמוד הקודם.

unknown = cv2.subtract(sure\_bg, sure\_fg)

**לבסוף יצרנו final\_mask ושמרנו אותו לתוך תיקייה מיועדת, עבור על פריים. לבסוף התיקייה תכיל את מפות השקיפות עבור כל אחד מהפריימים.**

ה- matting :  
ולבסוף שילבנו את האובייקט עם תמונת הרקע ע"י:

כאשר בחרנו ערך קבוע:

4. Object Tracking

קלט:

1. סרטון הקלט – מחרוזת ובה שם סרטון הקלט. ברירת המחדל: matted.avi.

הציפייה היא לסרטון בעל אובייקט מרכזי ורקע יחסית סטטי.

1. סרטון הפלט – מחרוזת ובה שם סרטון הפלט. ברירת מחדל: OUTPUT.avi.
2. N – מספר החלקיקים בו יעשה שימוש בתהליך העקיבה. ברירת מחדל: 100.

פלט: הפונקציה למעשה לא מחזירה דבר, אלא יוצרת סרטון בעל מסגרת מלבנית סביב האובייקט אחרו רוצים לבצע עקיבה.

חלק זה הסתמך על מימוש particle filter מתרגיל בית 3.

בהתחלה נקבל מהמשתמש את פרמטרי מלבן העקיבה (ע"י סימון ריבוע על גבי ה-Frame הראשון) ונאתחל את Sבהתאם לערכים אלו.  
**על-מנת להריץ את התכנית על סרטון ארביטררי, יש לשנות את ערכי S כך שיכיל את מרכז האובייקט וגודלו בפריים הראשון.**

כעת ניצור את החלקיקים המשוערים עבור אותו S התחלתי, ניצור את ההיסטוגרמה q אותה נשווה בכל פעם לאפשרויות השונות של המסנן, נבנה את המשקלים ואת פונקציית הצפיפות המצטברת.

על סמך האתחול הנ"ל, בוצעו הפעולות הבאות על כל ה-Frames בסרטון:

נעדכן את Sprev להיות ה-S האחרון.

S החדשה חושבה ע"י דגימת החלקיקים, שערוך המיקום החדש, חישוב משקלים ופונקציית צפיפות מצטברת ואז דגימה מחדש בהתאם לחישובים שביצענו. תהליך זה נעשה ע"י שימוש בפונקציות: sampleParticles, predictParticles.

לבסוף, האפשרות הסבירה ביותר על גבי ה-Frame מתקבל בעזרת הפונקציה showParticles ונכתב ל-Frame בסרט הפלט.

**מסקנות כלליות:**

1. כדי שפעולת העקיבה תתבצע בצורה מייטבית ונקייה יותר (מתקבלת תוצאה רציפה יחסית) רצוי שהרקע החדש שנבחר יהיה שונה יחסית מהאובייקט.
2. מתקבלת תוצאה טובה יותר בשלב הייצוב ככל שיש יותר נקודות עניין בסרטון.
3. נוכל לסכם שזו משימה לא פשוטה לפתח את הפונקציות הנדרשות כך שיתאימו להרבה מצבים אפשריים של סרטוני קלט שונים, אך אנו מאמינים שהשגנו תוצאות טובות בעיקר במצבים בהם לסרטון יש רקע סטטי והאובייקט נע בקו אופקי מתמיד.
4. קבצי הפלט יימצאו בתיקיית CODE לאחר סיום ריצת התכנית.
5. לינק לגיטהאב של הפרויקט שלנו:  
   <https://github.com/Izzy90/video_processing_project>