עיבוד תמונה – תרגיל בית 1

אוריין חסידים | oryan.hassidim | 31931579

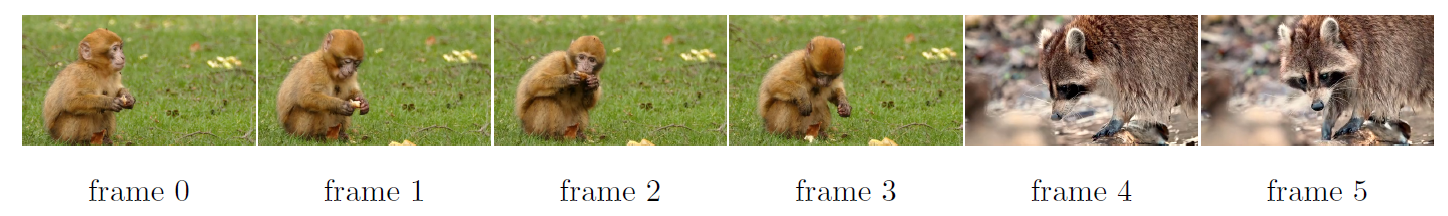
# מבוא

1. Introduction

(a) In your own words, state the goal of the exercise and what was the main technique (i.e. an idea or concept you've learned in class, not a technical tool like numpy) you've used to solve it.

(b) Briefly specify the differences between the two categories of videos (category 1 and category2) and how these differences may affect the approach.

בתרגיל זה ניישם שימוש בהיסטוגרמה של רמות בהירות של תמונות כפי שלמדנו בהרצאה 1 כדי לזהות מעבר בין סצנות.



באופן המתבקש מההרצאה הנ"ל, נשתמש בשתי גישות – נורמת ההפרש בין ההיסטוגרמות ונורמת ההפרש בין הסכומים המצטברים של ההיסטוגרמות. כפי שהיה ניתן לנחש עוד לפני שרואים את הסרטונים – זהו ההבדל שנדרש במימוש בין שתי הקטגוריות...

# אלגוריתם

2. Algorithm

(a) Clearly describe the steps involved in your scene cut detection algorithm (i.e. describe the conceptual steps and building blocks, e.g., if the algorithm is how to make coffee, the steps could be 1) Heat water, 2) Put coffee in glass, 3) Pour water, 4) Add sugar, 5) Add milk).

(b) Clearly describe any modifications or adjustments made between the first video category and the second one.

1. טעינה של הוידאו frame by frame
2. עבור כל פריים:
   1. עיבוד לגווני אפור
   2. חישוב היסטוגרמה
   3. **עבור קטגוריה 2 בלבד –** חישוב הסכומים המצטברים
3. חישוב נורמת ההפרש בין כל שני וקטורים עוקבים
4. מציאת האינדקס בעל ההפרש המקסימלי

# פרטי מימוש

השתמשתי בספרייה [OpenCV](https://opencv.org/) כדי לקרוא מקובץ הוידאו – שזו הייתה ההמלצה ברשת ואכן נראה שסיפקה לי את הפונקציונליות הנדרשת בקוד פשוט וקצר, וכמובן בספרייה [NumPy](https://numpy.org/).

מהאספקט התכנותי, השתמשתי בכלים מעולם ה-Functional Programming.

השתמשתי ב-Pipeline כדי לממש את רצף הפעולות שפורטו לעיל באלגוריתם.

1. טענתי כאיטרטור את הפריימים של התמונה (ככה שלא העמסתי יותר מדי על הזיכרון, שהיה צריך להחזיק כל פעם פריים אחד בלבד בזיכרון ולא את כולם יחד)
2. את האיטרטור הנ"ל מיפיתי (map) לפונקציית המרה לגווני אפור מ-BGR (הפורמט התמוה של OpenCV) שימפה פריים פריים. השתמשתי בפונקצייה המובנית של OpenCV.
3. מיפוי להיסטוגרמה באמצעות NumPY.
4. **עבור קטגוריה 2 –** מיפוי לפונקציית cumsum של NumPY.
5. שימוש בפונקצייה המובנית pairwise ממודול itertools (מובנה, החל מגרסה 3.10) – כדי לקבל כל פעם זוג היסטוגרמות לפונקצייה הבאה.
6. מיפוי כל זוג להפרש הנורמות. שוב NumPY.
7. מציאת argmax – פונקצייה שמימשתי בעצמי. לא השתמשתי בפונקצייה המובנית של NumPY כי היא עובדת על מערכים שיושבים בזיכרון אך אני רציתי לעבוד על איטרטור ולא שהכל יהיה בזיכרון במקביל.
8. המרה מאינדקס בודד לתצורת tuple של האינדקס והאינדקס העוקב.

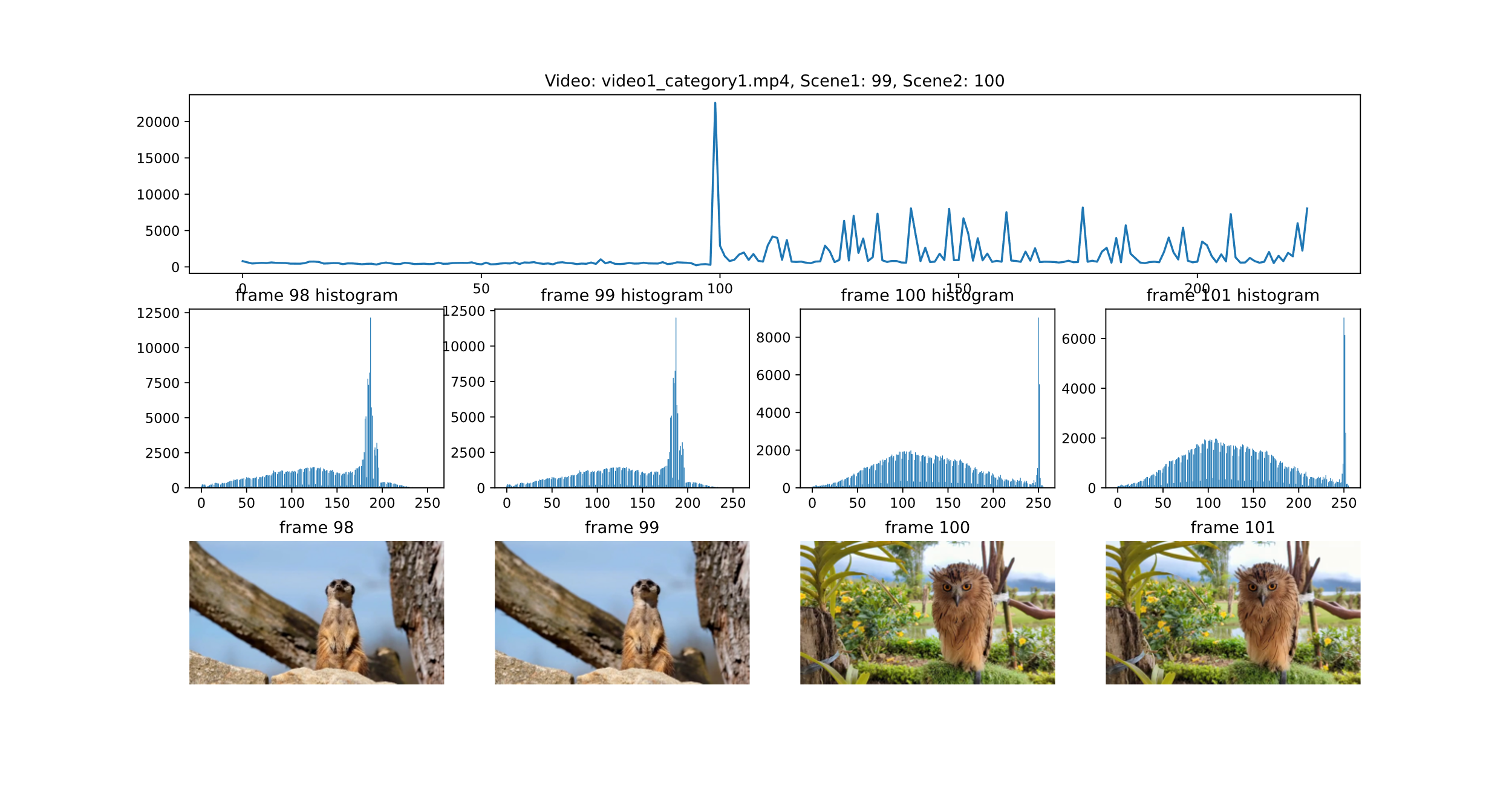
אחרי כתיבה של הקוד באופן ישיר הקוד עבד אך לא החזיר את התשובה המצופה (הצגתי את שני הפריימים אחד ליד השני וראיתי שאין מעבר סצינה). לאחר כמה דקות הסתבר שמשום מה בדיעבד בטעות השתמשתי בפונקצייה min במקום max 🙄. חוץ מזה כל הקוד עבד חלק ישר מהפעם הראשונה.

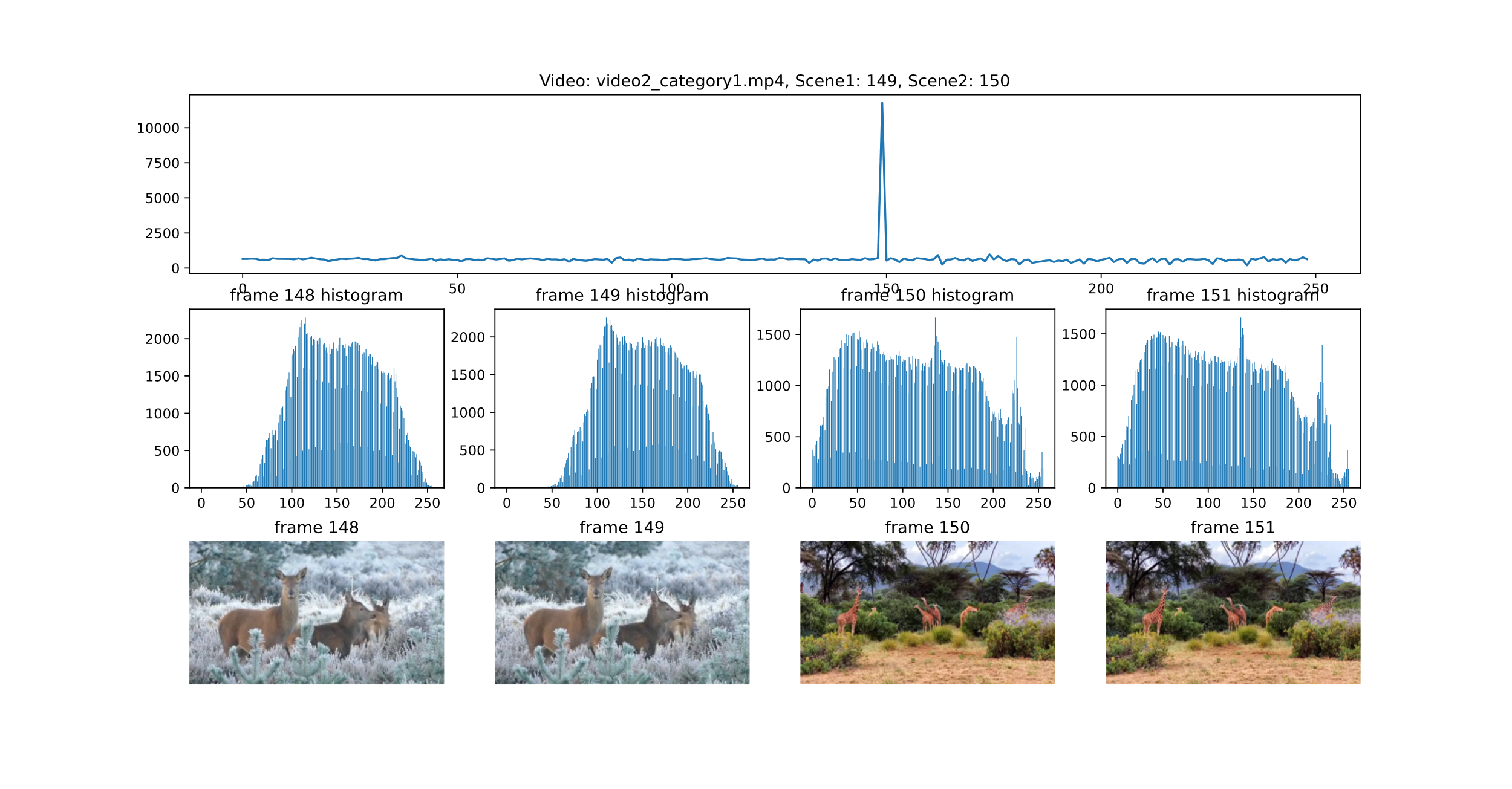
אתגר מעניין שהיה הוא לממש piping ב-python, שהיא לא שפה שזה טבעי לה וזה קונספט מעולם ה-FP... כתבתי מימוש בעצמי שענה לי על הצרכים ועשה עבודה נהדרת.

הפרוייקט נמצא ב-[github שלי](https://github.com/Oryan-Hassidim/67829-Image-Processing/tree/main/ex1).

# תוצאות קטגוריה 1

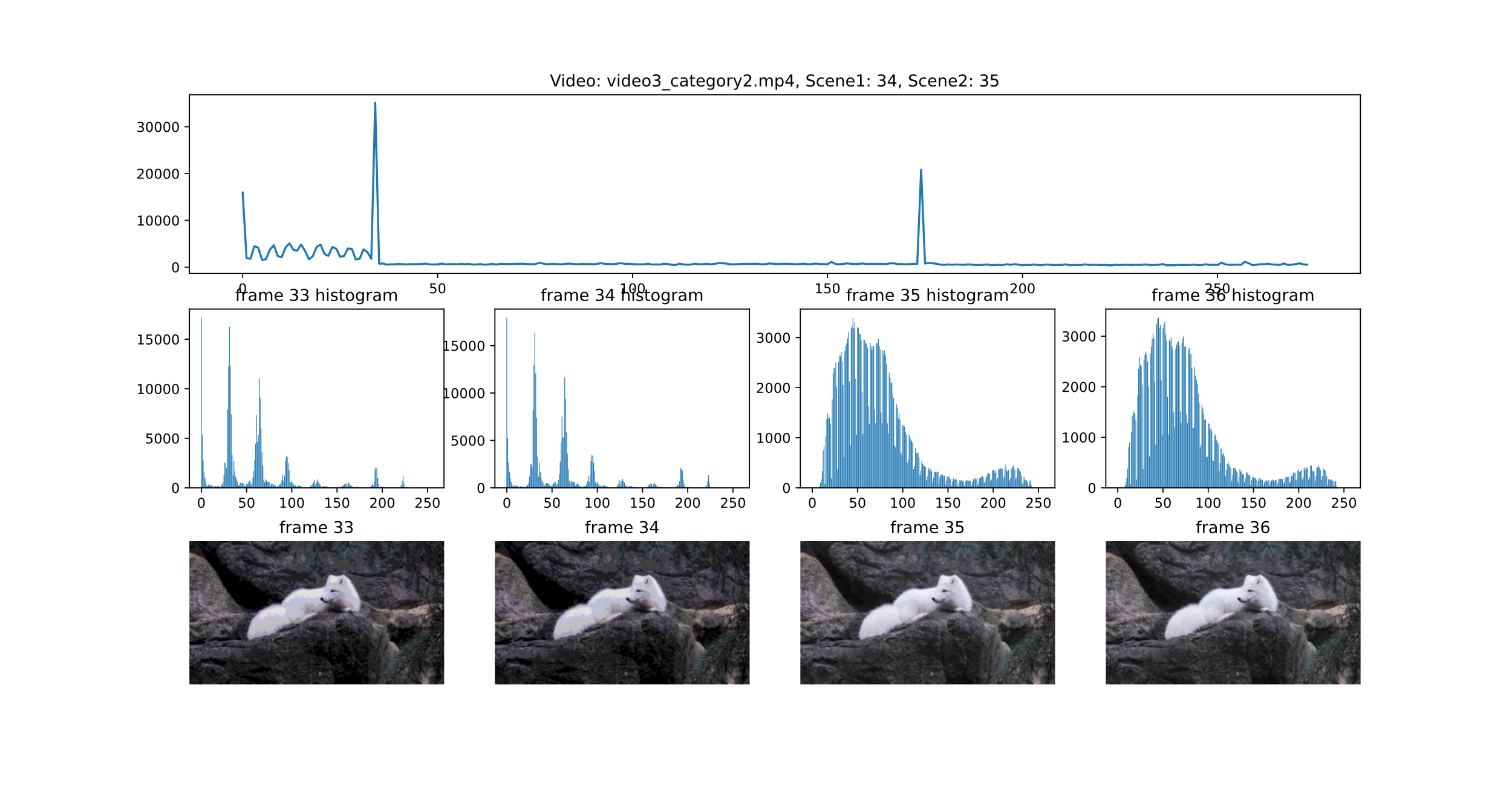
עבור כל וידאו: בכותרת שם הוידאו, והפריימים בהם היה החיתוך. מתחתיה גרף הנורמות של הפרשי ההיסטוגרמות לאורך הסרטון, מתחתיהם שני הפריימים האחרונים בסצנה הראשונה ושני הראשונים בשנייה, עם ההיסטוגרמה המתאימה מעל כל פריים.



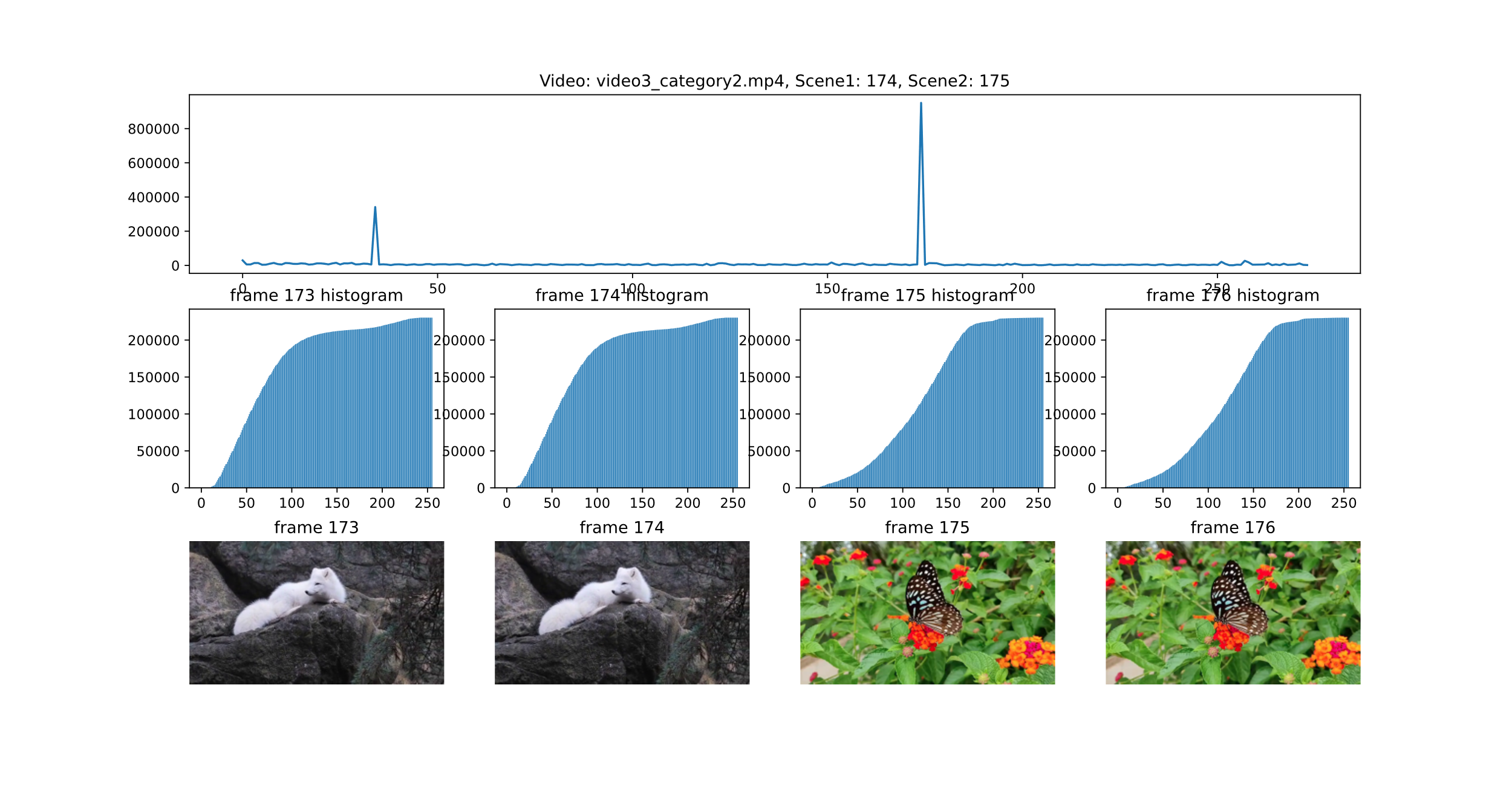


# תוצאות קטגוריה 2

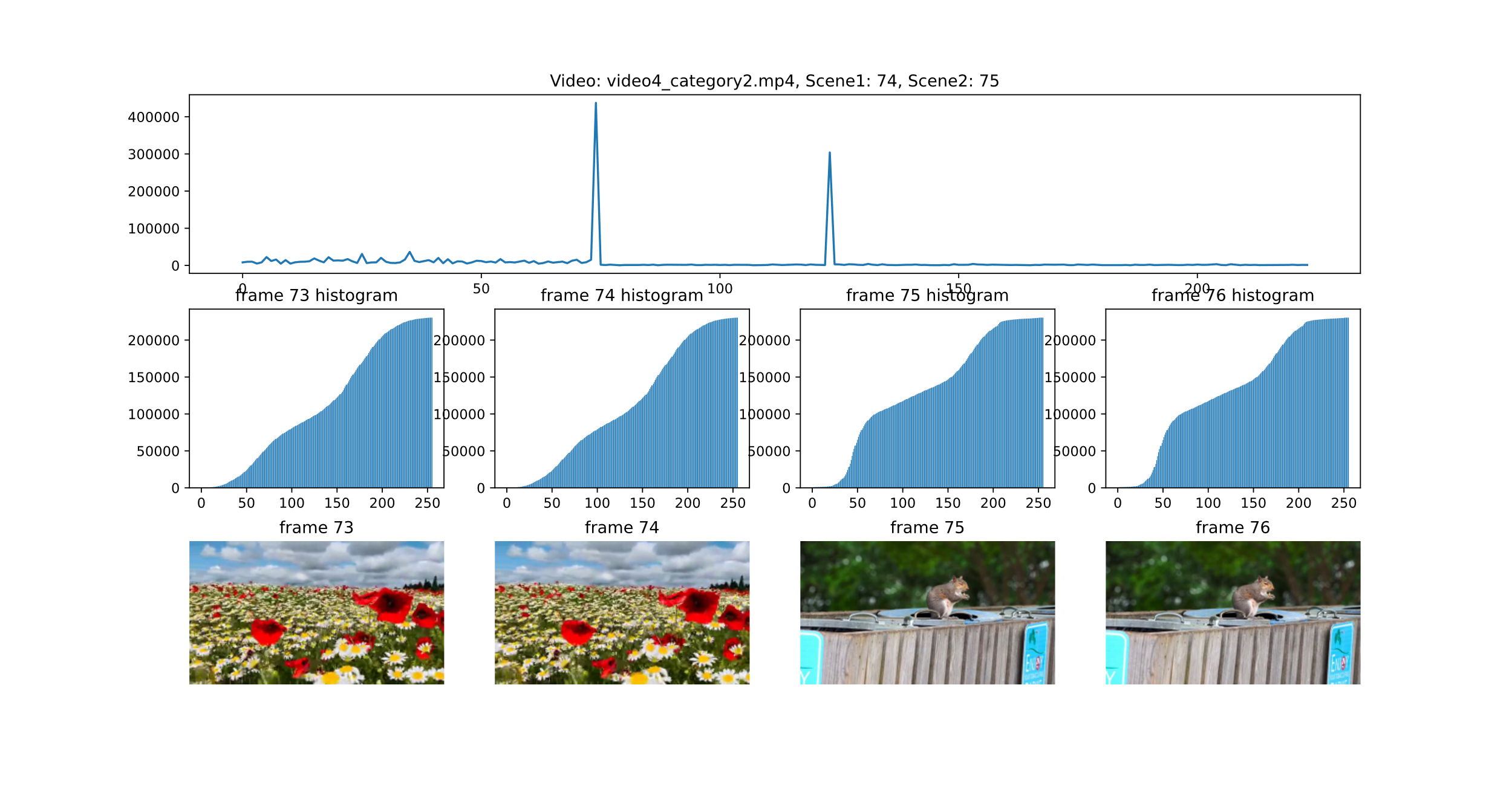
אכן ללא חישוב של הסכומים המצטברים, הקוד לא מצא את הפריים הנכון. לדוגמא, עבור וידאו 3 עם שימוש בקטגוריה 1:



מצער. אכן רואים בין שני הפריימים שינוי יחסית קיצוני באור – רוב הצבע של הפרווה השתנה מ-פריים 34 לפריים 35, מה שוודאי עשה הבדל כה משמעותי בין ההיסטוגרמות, אע"פ שהוא הבדל לא גדול כי הם רק זזו כמה עמודות, אך לכל הפחות הוא הבדל יותר גדול מהשינוי האמיתי בין שתי הסצנות שגם אותו ניתן לראות בגרף. אך עם שימוש בקטגוריה 2, שמשתמשת בסכומים מצטברים (גרפי ההיסטוגרמה עבור הסכומים המצטברים):



ניתן לראות שעדיין יש משמעות לפריים 34, אך הוא נהיה פחות משמעותי מפריים 174.



# מסקנות

בתרגיל זה השתמשנו בהיסטוגרמה של תמונה בגווני אפור כדי לזהות מעבר בין סצנות. ראינו כמה חישוב סכום מצטבר של היסטוגרמה הוא משמעותי לביצועים של האפליקציה לזהות נכון ולא להיות מושפע משינוי אור יחסית מינורי.

בזמני הפנוי כשעברתי על מצגת 1 אז גם כתבתי קוד לשיפור תמונה ע"י שיווי היסטוגרמה. מגניב רצח.

כמו כן היה אתגר מהנה מהפן התכנותי, ותרגיל מרענן לחזרה מהמילואים. תודה רבה!

