סיווג ודחיסת תמונות לווין זעיר

תקציר (Abstract):

לוויינים זעירים מוגבלים במשאבים חישוביים, קיבולת אחסון ותקשורת. לפיכך, עיבוד מקדים של תמונות ובחירה חכמה של תמונות לשידור הוא מרכיב חיוני במערכות של לוויינים זעירים. בעבודה זו פותחה מערכת מודולרית אשר מזהה אופק, מזהה כוכבים, מסווג תמונות לפי איכות וערך מדעי , ומבצעת דחיסה מותאמת לחלל .

מבוא:

לווייני CubeSat הפכו נפוצים בשנים האחרונות בזכות עלותם הנמוכה והיכולת לשגרם בקלות. עם זאת, משקלם וגודלם הקטן מגביל את כמות המידע שניתן לאחסן ולשדר. לכן יש צורך במערכת שתעבד תמונות בצורה אוטונומית, תבחר את התמונות החשובות ותדחוס אותן כך שתשמר איכותן לצרכים מדעיים.

מטרות המחקר:

- לאפשר ללוויין לזהות גבולות ברורים בתמונה (כמו קו האופק).
- לזהות כוכבים והבהובים – תוצרים בעלי ערך מדעי.
- לסווג תמונות לפי איכות (חדות, רעש, חשיפה).
- לדחוס תמונות כך שיישמר מידע חשוב תוך חסכון במשאבים.

סקר ספרות:

- **מודול זיהוי קו האופק:**

הפונקציה **`detect_horizon`** אחראית על זיהוי קו האופק בתמונות לוויין באמצעות טכניקות קלאסיות של עיבוד תמונה. להלן תיאור תהליך העבודה:

1. **קדם-עיבוד:**
   - התמונה הקלט מומרת תחילה לגווני אפור בעזרת הפונקציה `cvtColor` של OpenCV.
   - מופעל טשטוש מסוג `GaussianBlur` (Gaussian) להפחתת רעשים ו לחידוד גבולות האזורים הבולטים.
2. **סף בינארי:**
   - מבוצע סף בינארי (threshold) שמבליט את הניגוד בין השמיים לפני השטח של כדור הארץ, מה שמפשט את תהליך זיהוי הקונטורים.
3. **זיהוי קונטורים**:
   - הקונטורים מזוהים באמצעות `findContours`. נבחר הקונטור הגדול ביותר בהנחה שהוא מייצג את הגבול בין השמיים לקרקע – כלומר, קו האופק.
4. **אומדן קו האופק**:
   - הקונטור מותאם לפולינום מדרגה שנייה (`polyfit`) אשר מייצג את עקומת האופק.
   - הפולינום מצויר על גבי עותק של התמונה, כאשר נקודות ירוקות מסמנות את המיקומים המחושבים של קו האופק.
5. **פלט הפונקציה**:
   - הפונקציה מחזירה את ערך ה־y הממוצע של קו האופק (המייצג את מיקומו האנכי בתמונה), ואת התמונה עם הסימון החזותי של קו האופק.

```python
def detect_horizon(image):

    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)
    _, binary = cv2.threshold(blurred, 80, 255, cv2.THRESH_BINARY)

    contours, _ = cv2.findContours(binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    if not contours:
        return None, None

    largest_contour = max(contours, key=cv2.contourArea)

    points = np.squeeze(largest_contour)
    if points.ndim != 2:
        return None, None

    xs = points[:, 0]
    ys = points[:, 1]

    coeffs = np.polyfit(xs, ys, 2)
    poly = np.poly1d(coeffs)

    marked_image = image.copy()
    for x in range(0, image.shape[1], 2):
        y = int(poly(x))
        if 0 <= y < image.shape[0]:
            cv2.circle(marked_image, (x, y), 1, (0, 255, 0), -1)

    avg_y = int(np.mean(ys))
    return avg_y, marked_image
```

- **<u>מודול זיהוי כוכבים בתמונות לוויין:</u>**

פונקציה **`detect_stars`** אחראית על זיהוי כוכבים בתמונות שמגיעות מהלוויין, תוך שימוש בטכניקות עיבוד תמונה קלאסיות. המטרה היא לזהות נוכחות ברורה של מספר כוכבים – לפחות שלושה – כדי לקבוע שהתמונה מתאימה לניווט אסטרונומי.

שלבי העיבוד:

1. **שימוש בגווני אפור והגברת ניגודיות**:
   - התמונה מומרת לגווני אפור (`cv2.cvtColor`) כדי לפשט את ניתוח הבהירות.
   - עשה שימוש באלגוריתם CLAHE (Contrast Limited Adaptive Histogram Equalization) לשיפור הניגודיות באופן מקומי – מה שמבליט נקודות בהירות קטנות (כלומר כוכבים).
2. **סף בינארי וסינון רעשים**:
   - מבוצע סף של ערכי בהירות גבוהים (220–255) כדי לבודד את האזורים הבהירים מהתמונה.
   - מופעלת סגירה מורפולוגית (`morphologyEx`) להסרת רעשי רקע ולאיחוד אזורים סמוכים.
3. **איתור קונטורים ואימות עיגוליות**:
   - הקונטורים בתמונה מזוהים, באמצעות חישוב שטח והיקף נבדקת ה"עיגוליות" של כל קונטור (`circularity`).
   - אם צורתו של האובייקט עגולה מספיק והוא קטן (שטח מתחת ל־30 פיקסלים), הוא נחשב למועמד להיות כוכב.
4. **שיפור נוסף עם מעגלי האף (HoughCircles)**:
   - מתבצעת בדיקה נוספת לזיהוי כוכבים קטנים באמצעות אלגוריתם זיהוי מעגלים של האף – מה שמאפשר זיהוי מדויק של כוכבים קטנים במיוחד.
5. **פלט הפונקציה**:
   - הפונקציה מחזירה:
   - תמונה מסומנת עם מיקום הכוכבים.
   - ערך בוליאני האם נמצאו לפחות שלושה כוכבים – תנאי סף לשימוש בתמונה לניווט.

```python
def detect_stars(image):

    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    enhanced = clahe.apply(gray)
    _, thresh = cv2.threshold(enhanced, 220, 255, cv2.THRESH_BINARY)
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3, 3))
    clean = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel, iterations=2)
    contours, _ = cv2.findContours(clean, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    marked_image = image.copy()
    stars_count = 0

    for c in contours:
        area = cv2.contourArea(c)
        if area < 30:
            perimeter = cv2.arcLength(c, True)
            if perimeter == 0:
                continue
            circularity = 4 * np.pi * area / (perimeter * perimeter)
            if 0.6 < circularity <= 1.2:
                (x, y), radius = cv2.minEnclosingCircle(c)
                center = (int(x), int(y))
                cv2.circle(marked_image, center, 3, (255, 0, 0), 1)
                stars_count += 1

    circles = cv2.HoughCircles(enhanced, cv2.HOUGH_GRADIENT, dp=1.2, minDist=5,
                               param1=50, param2=15, minRadius=1, maxRadius=5)

    if circles is not None:
        for circle in circles[0, :]:
            center = (int(circle[0]), int(circle[1]))
            cv2.circle(marked_image, center, 3, (255, 0, 0), 1)
        stars_count += circles.shape[1]

    return (stars_count >= 3), marked_image
```

- **מודול זיהוי הבהובים (Flicker Detection):**

הפונקציה **`detect_flickering`** נועדה לזהות שינויים פתאומיים בתמונות שמגיעות מהלוויין בהשוואה לתמונה הקודמת. שינוי כזה, המכונה **הבהוב** (Flicker), עלול להעיד על בעיה בחיישן המצלמה, הפרעה בשידור, או הבהוב אור רגעי בסביבה. זיהוי של הבהוב מאפשר למערכת לסנן תמונות פגומות שאינן מתאימות לעיבוד נוסף.

שלבי העבודה:

1. **בדיקות תקינות ראשוניות**:
   - אם אחת מהתמונות היא `None` או שגודל התמונות אינו תואם – הפונקציה מחזירה `False` (לא זוהה הבהוב).
2. **המרה למרחב צבעים YUV**:
   - תמונות מומרות מ־BGR ל־YUV, ומתבצע ניתוח רק על הערוץ **Y** – שהוא ערוץ הבהירות (Luminance), כדי להתמקד בשינויי תאורה.
3. **חישוב השונות המנורמלת**:
   - מחושבת ההפרש המוחלט בין ערכי הבהירות של שתי התמונות.
   - נרמול נעשה ע"י חלוקה בערך המקסימלי בין הפיקסלים (`np.maximum(prev_y, 1)`) כדי למנוע חלוקה באפס ולהפוך את ההפרשים למשמעותיים יחסית לעוצמה המקורית.
4. **סינון שינויים זניחים**:
   - אם ממוצע ההבדלים נמוך מסף מינימלי (`min_change`), הפונקציה מחזירה `False`.
5. **חישוב אחוז פיקסלים משתנים**:
   - סופרים את מספר הפיקסלים שבהם ההפרש חצה את סף ההבהוב (`threshold=0.25`).
   - אם יחס הפיקסלים המשתנים גדול מ־15% – הפונקציה מחזירה `True` (זוהה הבהוב).

```python
def detect_flickering(prev_img, current_img, threshold=0.25, min_change=0.05):
    if prev_img is None or current_img is None:
        return False
    if prev_img.shape != current_img.shape:
        return False

    prev_yuv = cv2.cvtColor(prev_img, cv2.COLOR_BGR2YUV)
    curr_yuv = cv2.cvtColor(current_img, cv2.COLOR_BGR2YUV)

    prev_y = prev_yuv[:,:,0].astype(np.float32)
    curr_y = curr_yuv[:,:,0].astype(np.float32)

    diff = np.abs(prev_y - curr_y)
    normalized_diff = diff / np.maximum(prev_y, 1)

    mean_diff = np.mean(normalized_diff)
    if mean_diff < min_change:
        return False

    changed_pixels = np.sum(normalized_diff > threshold)
    total_pixels = prev_y.size
    ratio = changed_pixels / total_pixels

    return ratio > 0.15
```

- **מודול הערכת איכות תמונה (Image Quality Assessment):**

הפונקציה `assess_image_quality` נועדה לקבוע האם תמונה שהגיעה מהלוויין נחשבת **טובה מספיק לעיבוד** ולשידור. הערכה זו מתבצעת על בסיס מאפיינים כמו חדות, ניגודיות, בהירות, רעש חזותי ואיכות כללית לפי מדדים סטנדרטיים.

שלבי עבודה:

1. **בדיקות תקינות**:
   - אם התמונה ריקה או לא קיימת – הפונקציה מחזירה תוצאה שלילית לכל המדדים.

2. **המרה לגווני אפור**:
   - כל המדדים מחושבים על גרסה בגווני אפור של התמונה (Gray-scale), כדי להתרכז במידע מבני.
3. **מדד חדות (Laplacian Variance)**:
   - מחושב השונות של המסנן לפלסיאני על התמונה – מדד נפוץ לקביעת חדות.
   - ערך גבוה מ־80 נחשב "חד".

4. **מדד ניגודיות (RMS Contrast)**:
   - מחושב סטיית התקן של ערכי הפיקסלים ומנורמל ל־[0–1].
   - ערך גבוה מ־0.3 מצביע על ניגודיות טובה.

5. **בהירות ממוצעת (Brightness)**:
   - מחושב ממוצע הבהירות של התמונה.
   - טווח סביר הוא בין 60 ל־220.

6. **מדד רעש (Noise Estimate)**:
   - התמונה מחולקת לתאי 50×50, ולכל אחד מחושבת השונות הפנימית (Variance).
   - רעש כללי גבוה מ־500 נחשב בעייתי.

7. **מדד איכות BRISQUE :**
   - Perceptual Image Quality, מבצע חישוב של מדד BRISQUE (Blind/Referenceless Image Spatial Quality Evaluator) בעזרת PyTorch.
   - ציון נמוך יותר משמעותו איכות גבוהה יותר (0 = מושלם, 100 = רע מאוד).

יתרונות:

- מאפשר בחירה אוטומטית של **תמונות איכותיות בלבד**, כדי לחסוך נפח שידור ולשפר תוצרי עיבוד.
- מוכן לפעולה גם אם מודול BRISQUE לא מותקן – עם fallback חכם.
- מבוסס על שיטות מדידה סטנדרטיות בתחום עיבוד תמונה.

מודל דחיסה:

1.מודול דחיסה רגילה (JPEG) :

שיטה זו משתמשת בדחיסת תמונות סטנדרטית מסוג JPEG או JPEG2000 באמצעות הספרייה OpenCv

עקרון פעולה:

- השיטה פועלת על תמונות RGB.
- ניתן לבחור בין שני פורמטים:
  - `"jpeg"` — דחיסה לפי איכות ניתנת לשליטה (לדוג׳ quality=90).
  - `"jpeg2000"` — דחיסה יעילה יותר אך איטית, עם פרמטר קבוע.

2. מודול דחיסה באמצעות Autoencoder:

שיטה זו משתמשת במודל רשת עצבית (Autoencoder) כדי לבצע דחיסה לא לינארית של תמונות. במקום להקטין את איכות התמונה על ידי קידוד סטנדרטי, היא לומדת לשמר את המידע החשוב ביותר ולשחזר את התמונה בצורתה הקרובה למקור.

שלבי הפעולה:

1. **הכנה**: שינוי גודל התמונה ל־256×256, נירמול לערכים בין 0–1.
2. **הדחיסה**: שימוש במודל Autoencoder מאומן (אם לא קיים — נבנה חדש).
3. **השחזור**: הפלט של המודל נשמר כקובץ תמונה חדש.
4. **חישוב יחס דחיסה**: מחושב גודל חדש מול המקורי.

מבנה המודל:

- Encoder: שלוש שכבות Conv2D עם MaxPooling ו-BatchNorm.
- Decoder: שלוש שכבות UpSampling עד לגודל המקורי.
- Activation סופי: `sigmoid` (בין 0 ל-1).

- **<u>שיפורים ופיתוח עתידי</u>**

שיפורים טכנולוגיים :

1. שיפור סיווג כוכבים באמצעות למידת מכונה

- אנחנו השתמשנו ב אלגוריתמים קלאסיים. נוכל לאמן מודל קטן שיסווג כוכבים על בסיס תכונות חזותיות של נקודות בתמונה.(עשינו קוד עם למידת מכונה שעושה דחיסה לתמונות שעובד עם שיטה של autoencoder אבל אפשר לשפר איכות הדחיסה הזה).

2 . דחיסה אדפטיבית על בסיס תוכן

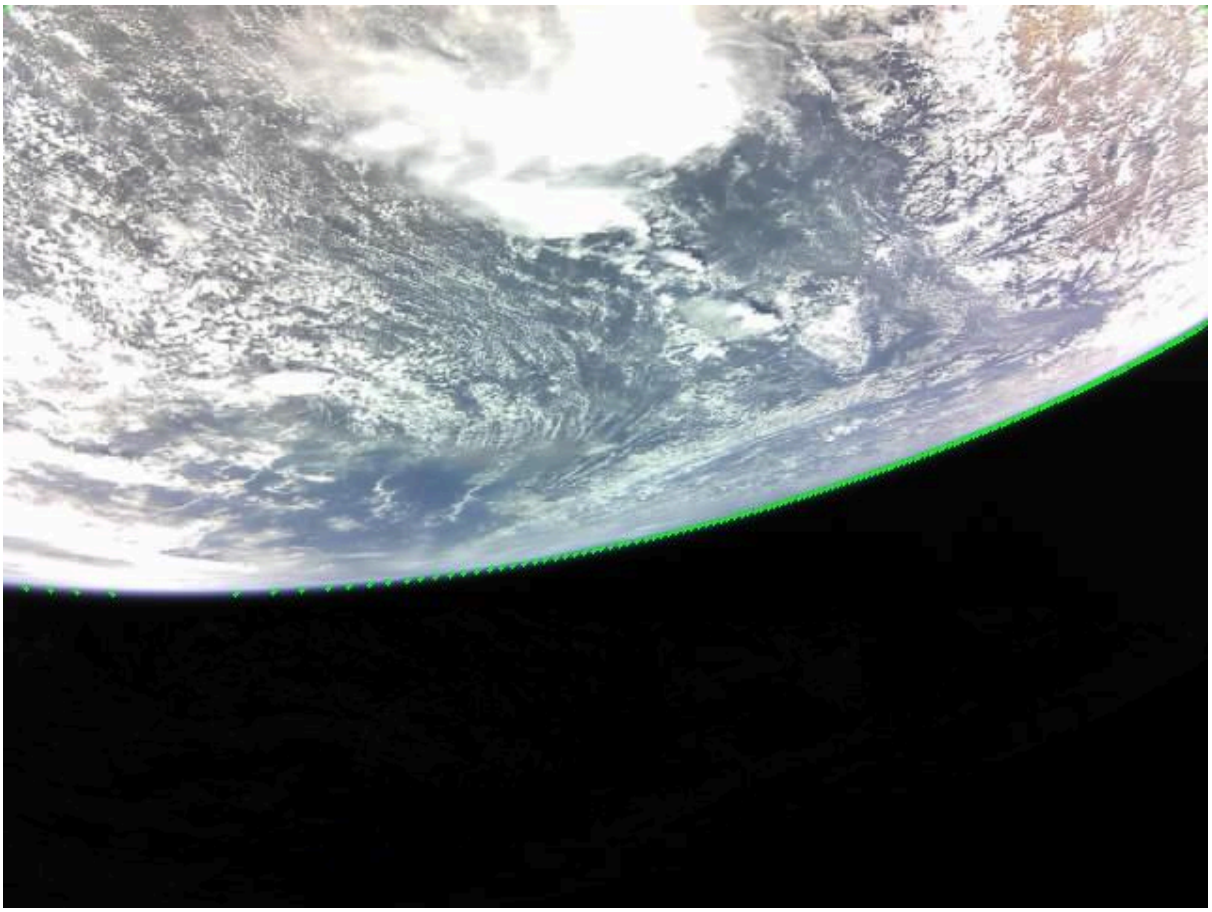
- להשתמש בשיטות כמו **Region of Interest (ROI) encoding**, לדוגמה:
  אזור עם כוכבים מקבל פחות דחיסה.
  רקע/שמיים מקבלים דחיסה גבוהה.

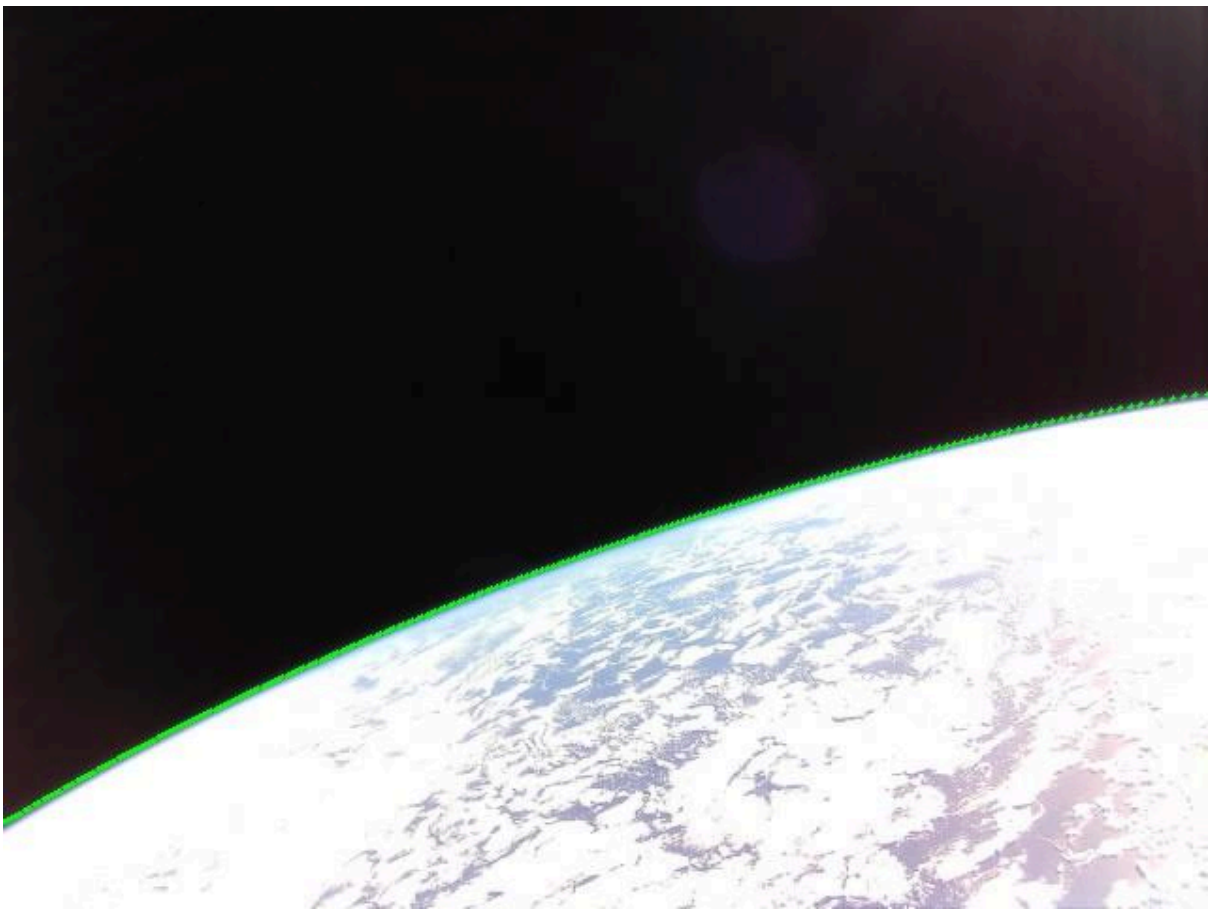
שיפורים מערכתיים / תפעוליים

ניהול זיכרון ואחסון בלוויין

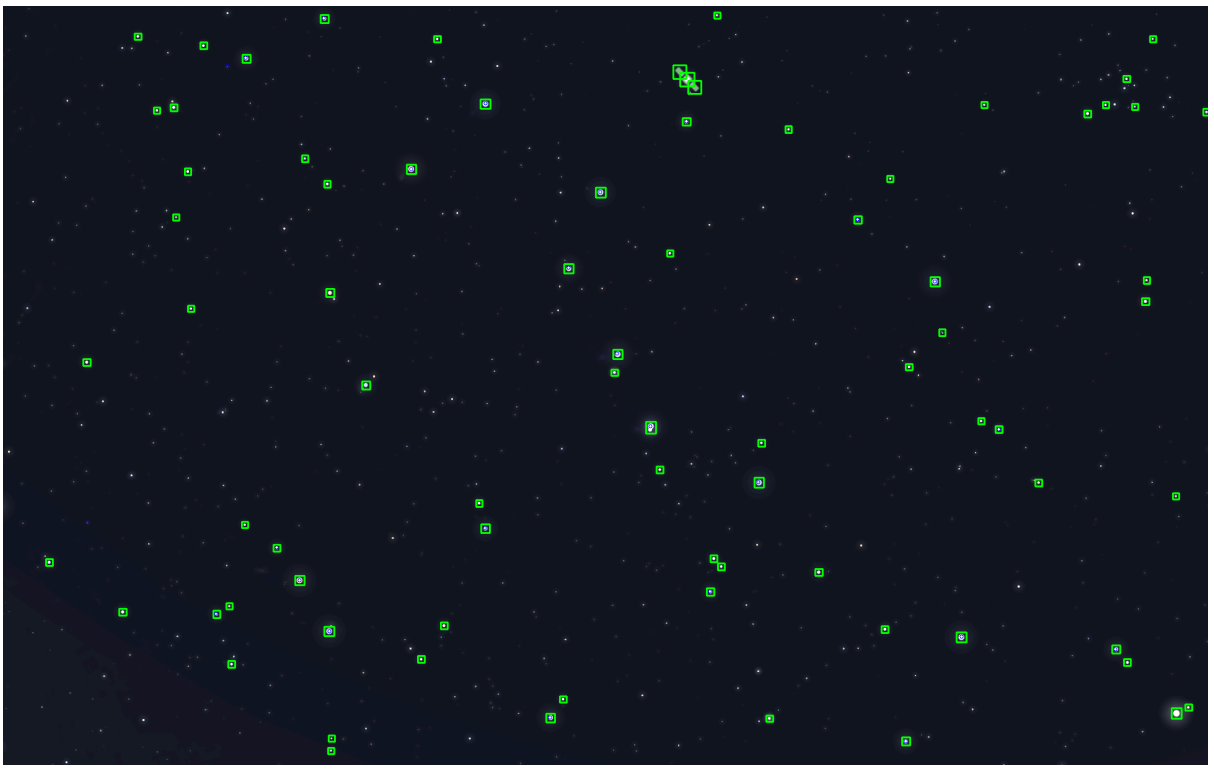
- תכנן מערכת ניהול תור/Buffer:
  תיעדוף תמונות לפי ערך מדעי.
  מחיקה אוטומטית של תמונות ישנות או בעלות איכות נמוכה.

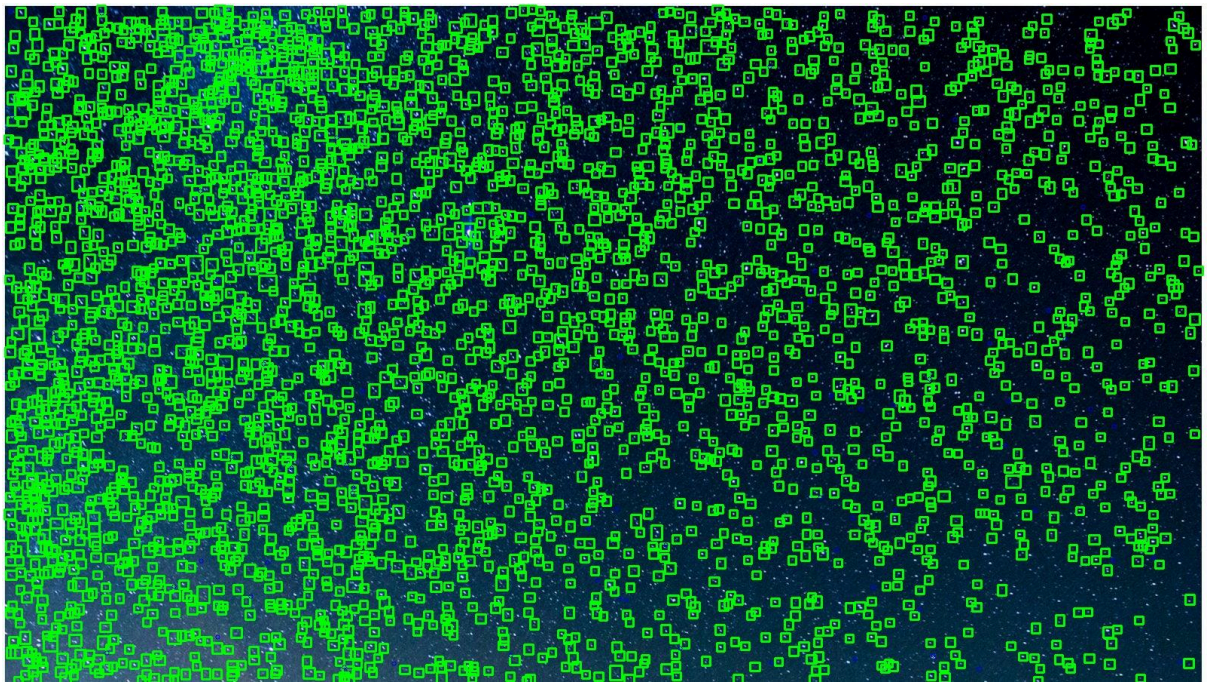
תוצאות זיהוי אופק:





תוצאות זיהוי כוכבים:





התוצאות עבור בתמונות קובץ csv:

https://github.com/IbrahemHurani/Space-Project/blob/main/summary.csv

קישורים:

- https://data.mendeley.com/datasets/5kygfmfdmr/1
- https://www.kaggle.com/datasets/divyansh22/dummy-astronomy-data
- קישורים לקוד של למידת מכונה לדחיסת תמונות בשיטת autoencoder:

https://github.com/IbrahemHurani/Space-Project/blob/main/train_autoencoder.py

https://github.com/IbrahemHurani/Space-Project/blob/main/core/autoencoder_compressor.py

- קישור לפרויקט שלנו:

https://github.com/IbrahemHurani/Space-Project/tree/main