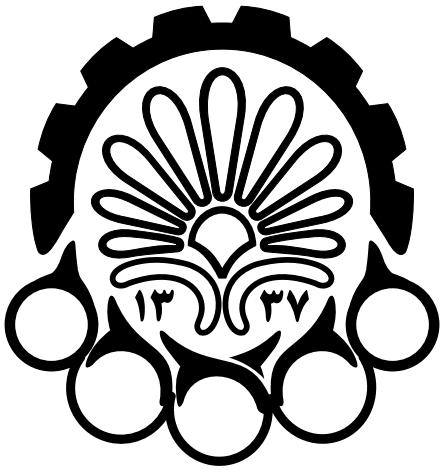


دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

تمر ین اول درس بینایی ماشین

دکتر صفابخش

غلامرضا دار ۴۰۰۱۳۱۰۱۸

بهار ۱۴۰۱

فهرست مطالب

سوال ۱)

نیازی به توضیح نمی‌باشد!

MeeseeksHQ.png

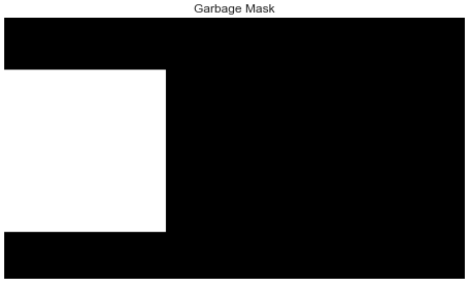
MeeseeksHQ.png (grayscale)

سوال ۲)

برای استخراج کاراکترهای آبی رنگ موجود در تصویر ابتدا تصویر را به فضای HSV می‌بریم و در آن فضا با استفاده از تابع cv2.inRange مقادیر نزدیک به Hue آبی را انتخاب می‌کنیم.

همانطور که دیده می‌شود در سمت راست تصویر نیز مقداری پیکسل آبی وجود دارد. در مرحله اول با کمک تغییر پارامترهای lower و upper تابع cv2.inRange سعی می‌کنیم این نقاط را کاهش دهیم. در ادامه می‌خواهیم با کمک تکنیک Garbage Mask یک ماسک غیر دقیق برای قسمت‌های غیرمهم(Garbage) تصویر درست کنیم. برای آشنایی بیشتر با کاربردهای این تکنیک به این لینک مراجعه کنید.

تصویر ماسک تولید شده را بر روی تصویر مشاهده میکنید.



در تصویر زیر نتیجه اعمال این ماسک بر روی نتیجه مرحله قبل را مشاهده میکنید.

مقایسه نتیجه نهایی و تصویر اولیه:

سوال ۳)

بخش الف)

برای جداسازی تصاویر در این بخش، صرفاً تصاویر را از جهت عمودی به سه بخش مساوی تقسیم کردیم.

Extracted Images from 01.jpg

Channel 0 Channel 1 Channel 2

Extracted Images from 02.jpg

Channel 0 Channel 1 Channel 2

Extracted Images from 03.jpg

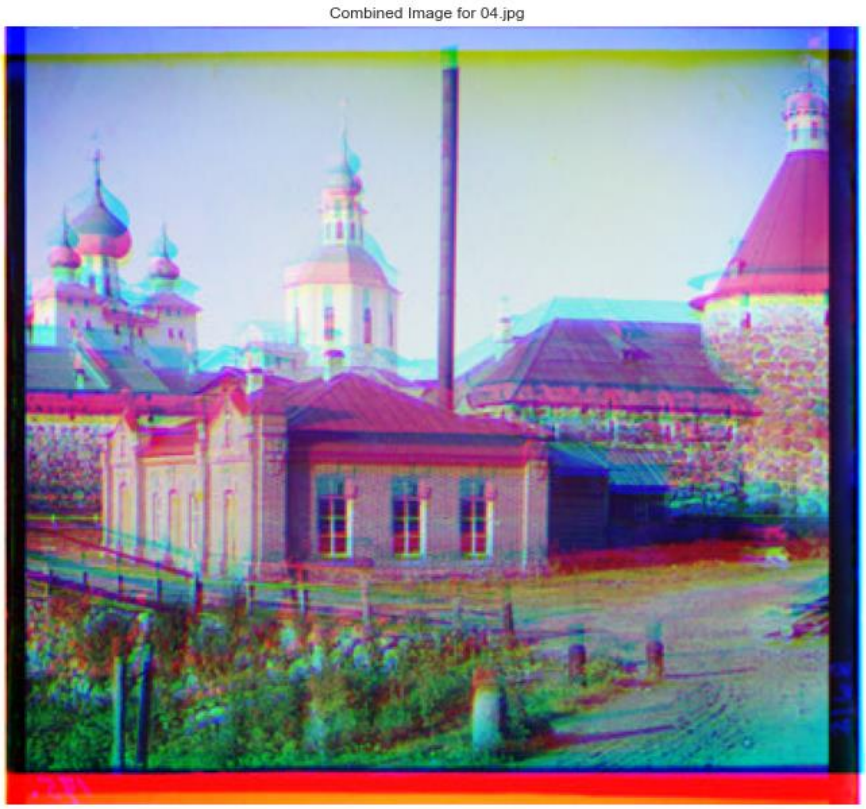
Channel 0 Channel 1 Channel 2

بخش ب)

در این بخش متوجه شدیم که ترتیب رنگی تصاویر جدا شده به صورت (R=2, G=1, B=0) بوده است. با قراردادن این تصاویر درکنار هم تصاویر رنگی زیر ساخته شدند.

Combined Image for 02.jpg

Combined Image for 03.jpg



همان‌طور که مشخص است، این تصاویر به صورت دقیق بر هم منطبق نیستند و این امر باعث شده در تصویر ترکیب، پدیده RGB Shift را مشاهده کنیم. در بخش بعد به حل این مشکل می‌پردازیم.

بخش ج)

در این بخش ابتدا به صورت دستی تصاویر مربوط به کانال های مختلف را کمی جابه‌جا می‌کنیم و به صورت چشمی بهترین ترکیب را برمی‌گزینیم.

Combined+Aligned Image for 01.jpg | PSNR=31.341841800839696

Combined+Aligned Image for 02.jpg | PSNR=25.244971503474698

Combined+Aligned Image for 03.jpg | PSNR=31.738326122525578

Combined+Aligned Image for 04.jpg | PSNR=25.059278767099254

در ادامه یک تابع برای کاهش میزان تارشدگی به صورت اتوماتیک می‌نویسیم. این تابع، یک کانال را ثابت نگه می‌دارد و دو کانال دیگر را در دو جهت x,y مقداری حرکت می‌دهد. پس از هر حرکت، معیار PSNR بین آن کانالی که ثابت بود و نتیجه نهایی ترکیب سه کانال محاسبه می‌شود. این تابع به دنبال بالاترین میزان PSNR است.

```
# Grid search over the movements
for ry in range(ranges[0][0], ranges[0][1]+1):
    for by in range(ranges[1][0], ranges[1][1]+1):
        for rx in range(ranges[2][0], ranges[2][1]+1):
            for bx in range(ranges[3][0], ranges[3][1]+1):
```

همچنین به دلیل وجود اعوجاج در حاشیه های هر تصویر، گوشه های تصاویر همچنان دارای مقداری تاری خواهند بود که به این علت، در محاسبه PSNR از این نواحی صرف نظر میکنیم.

```
# Ignore the border pixels
p = 50
psnr_value = cv2.PSNR(combined_image_gray[p:-p, p:-p], image_g[p:-p, p:-p])
```

```
> Searching 198/3969 (4.99%) | Best PSNR: 33.625244143099245 | Best Movements: [0, -10, 0, 5]
> Searching 396/3969 (9.98%) | Best PSNR: 34.092332119416355 | Best Movements: [0, -9, 0, 5]
> Searching 594/3969 (14.97%) | Best PSNR: 34.092332119416355 | Best Movements: [0, -9, 0, 5]
> Searching 792/3969 (19.95%) | Best PSNR: 34.092332119416355 | Best Movements: [0, -9, 0, 5]
> Searching 990/3969 (24.94%) | Best PSNR: 34.092332119416355 | Best Movements: [0, -9, 0, 5]
> Searching 1188/3969 (29.93%) | Best PSNR: 34.092332119416355 | Best Movements: [0, -9, 0, 5]
> Searching 1386/3969 (34.92%) | Best PSNR: 34.092332119416355 | Best Movements: [0, -9, 0, 5]
> Searching 1584/3969 (39.91%) | Best PSNR: 34.092332119416355 | Best Movements: [0, -9, 0, 5]
> Searching 1782/3969 (44.90%) | Best PSNR: 34.092332119416355 | Best Movements: [0, -9, 0, 5]
> Searching 1980/3969 (49.89%) | Best PSNR: 34.092332119416355 | Best Movements: [0, -9, 0, 5]
> Searching 2178/3969 (54.88%) | Best PSNR: 34.092332119416355 | Best Movements: [0, -9, 0, 5]
> Searching 2376/3969 (59.86%) | Best PSNR: 34.092332119416355 | Best Movements: [0, -9, 0, 5]
> Searching 2574/3969 (64.85%) | Best PSNR: 34.092332119416355 | Best Movements: [0, -9, 0, 5]
> Searching 2772/3969 (69.84%) | Best PSNR: 34.092332119416355 | Best Movements: [0, -9, 0, 5]
> Searching 2970/3969 (74.83%) | Best PSNR: 34.092332119416355 | Best Movements: [0, -9, 0, 5]
> Searching 3168/3969 (79.82%) | Best PSNR: 34.092332119416355 | Best Movements: [0, -9, 0, 5]
> Searching 3366/3969 (84.81%) | Best PSNR: 34.092332119416355 | Best Movements: [0, -9, 0, 5]
> Searching 3564/3969 (89.80%) | Best PSNR: 34.092332119416355 | Best Movements: [0, -9, 0, 5]
> Searching 3762/3969 (94.78%) | Best PSNR: 34.092332119416355 | Best Movements: [0, -9, 0, 5]
> Searching 3960/3969 (99.77%) | Best PSNR: 34.092332119416355 | Best Movements: [0, -9, 0, 5]
```

همان‌طور که مشخص است این روش بسیار کند است و تعداد حالت های مورد نیاز جستجو بسیار زیاد هستند. یک ایده برای بهبود این مشکل، عملکرد حریصانه برای یافتن بهترین ترکیب است. می‌توانیم ابتدا دو کانال دیگر را صرفا در جهت افقی حرکت دهیم و بیشینه PSNR را محاسبه کنیم. سپس با قفل کردن بهترین میزان شیفت افقی، به دنبال بهترین شیفت عمودی بگردیم. این کار معادل جداسازی چهار حلقه به صورت دو دسته حلقه دوتایی است.

```
# Grid search over the movements(vertical)
for ry in range(ranges[0][0], ranges[0][1]+1):
    for by in range(ranges[1][0], ranges[1][1]+1):
```

```
# Grid search over the movements(horizontal)
for rx in range(ranges[2][0], ranges[2][1]+1):
    for bx in range(ranges[3][0], ranges[3][1]+1):
```

با استفاده از این تکنیک، زمان جستجو برای تصویر اول از ۷/۲۵ ثانیه به ۱/۵ ثانیه کاهش پیدا می‌کند. درادامه برای بهبود زمان اجرا میتوان این عمل را بار دیگر تکرار کرد و به طور کلی ۴ حلقه مستقل داشته باشیم ولی این کارباعث می‌شود از بهینه سراسری دور شویم.

در نهایت تصاویر تنظیم شده و برش داده شده را مشاهده می‌کنید.

02.jpg Auto Combined & Cropped

03.jpg Auto Combined & Cropped

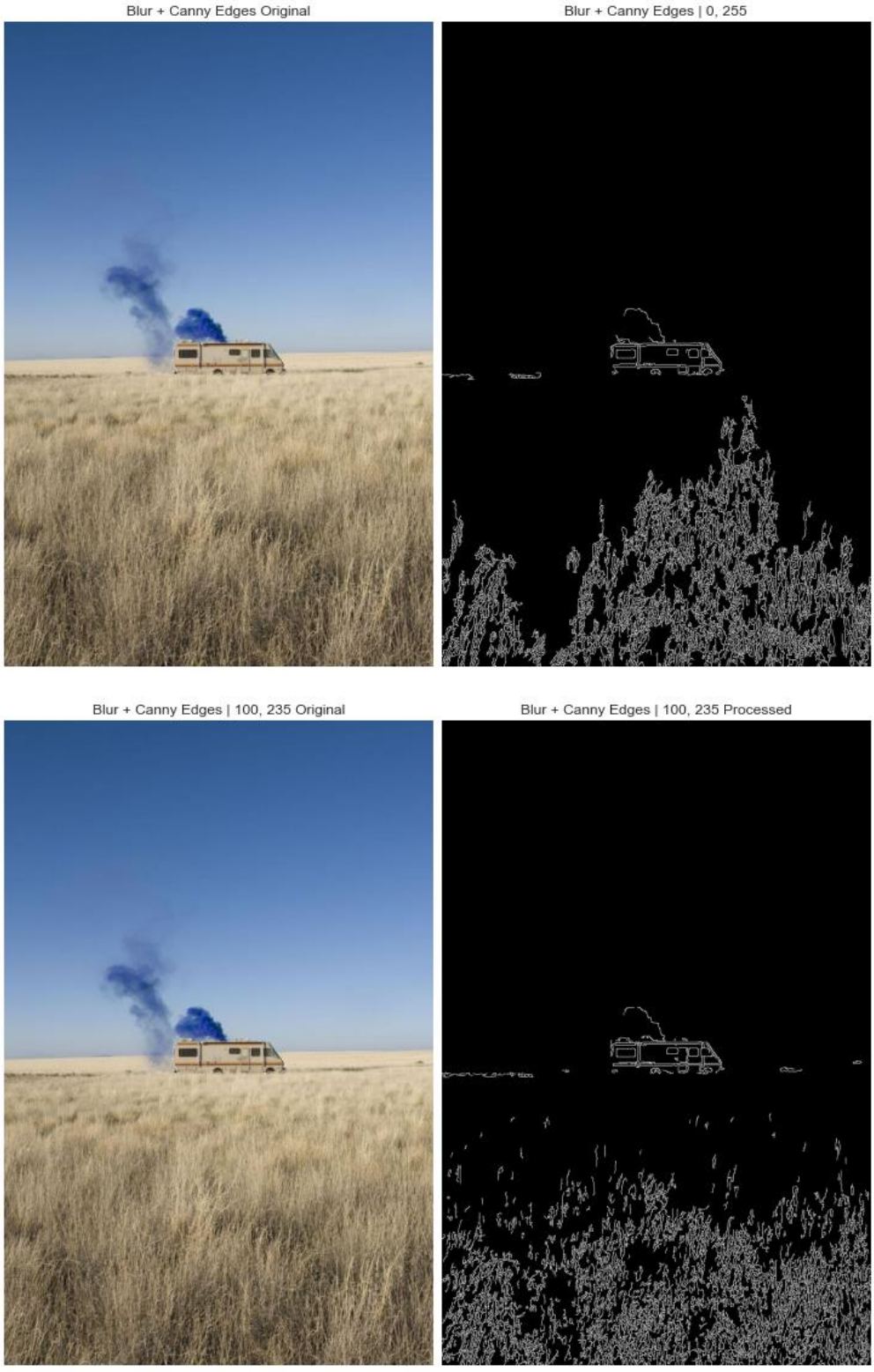
04.jpg Auto Combined & Cropped

سوال ۴)

در این سوال می‌خواهیم لبه‌های مربوط به ماشین والتر وایت را به طور مستقل داشته باشیم. ابتدا بدون کار اضافه ای با استفاده از الگوریتم Canny لبه های تصویر ورودی را استخراج می‌کنیم.

همانطورکه مشاهده می‌شود به دلیل وجود جزییات بسیار بالا در زمین زراعی، لبه های اضافه زیادی در تصویر نهایی مشاهده می‌شوند.

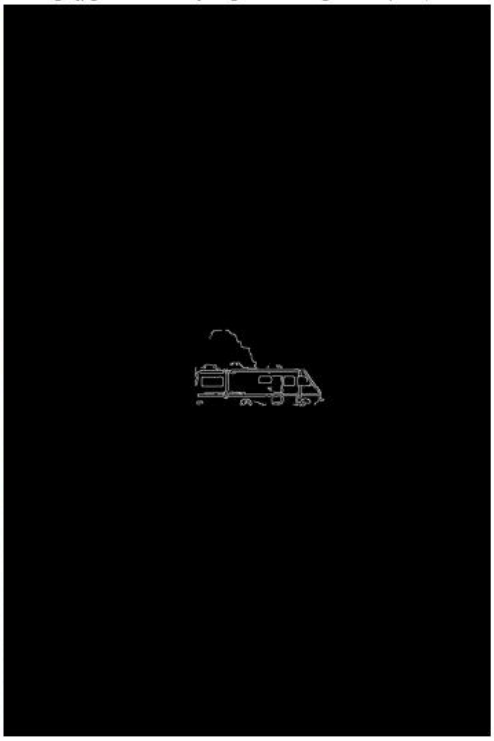
یکی از تکنیک‌ها برای کاهش این لبه‌های اضافه هموارسازی تصویر قبل از انجام لبه یابی است.



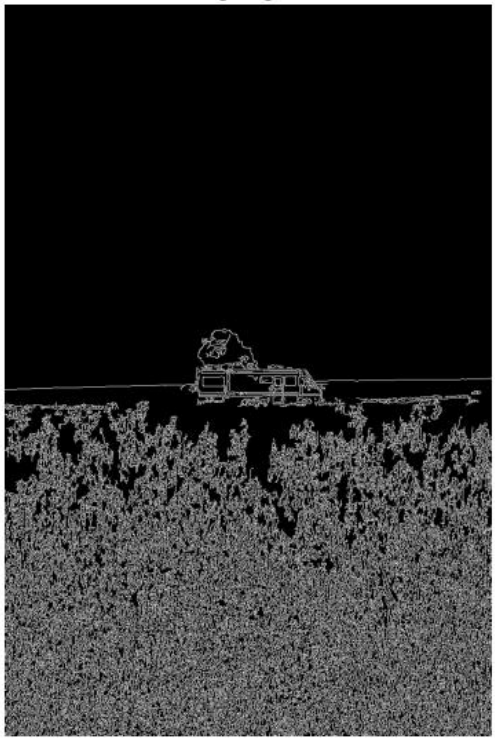
هر چه تصویر را هموارتر کنیم، لبه‌های اضافه تصویر کاهش می‌یابند اما با زیاده روی در اعمال Blur لبه‌های مربوط به سوژه اصلی(ماشین) را نیز از بین می‌بریم. در این قسمت مانند سوال ۲ از تکنیک Garbage Mask استفاده می‌کنیم. به دلیل اینکه لبه های اضافی اطراف ماشین را توانستیم به کمک مقدار کمی Blur حذف کنیم، مشخص کردن یک Mask تقریبی اطراف ماشین کار ساده ای است.

"edge.jpg" Original

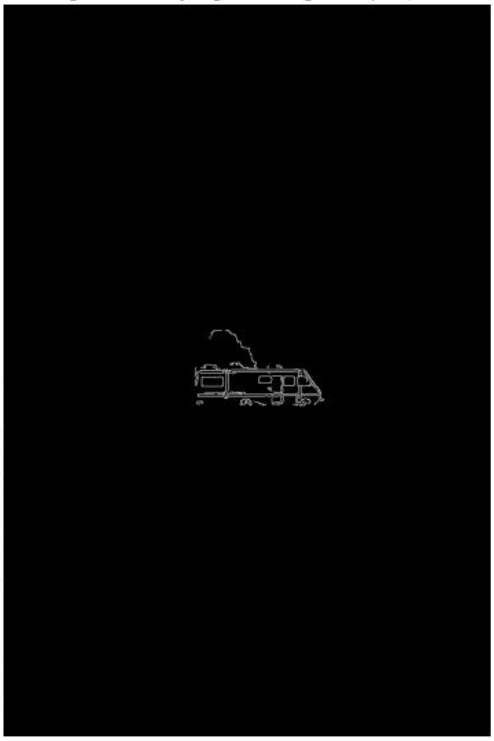
"edge.jpg" Blur + Canny Edges + Garbage Mask | 100, 235



Edge Original



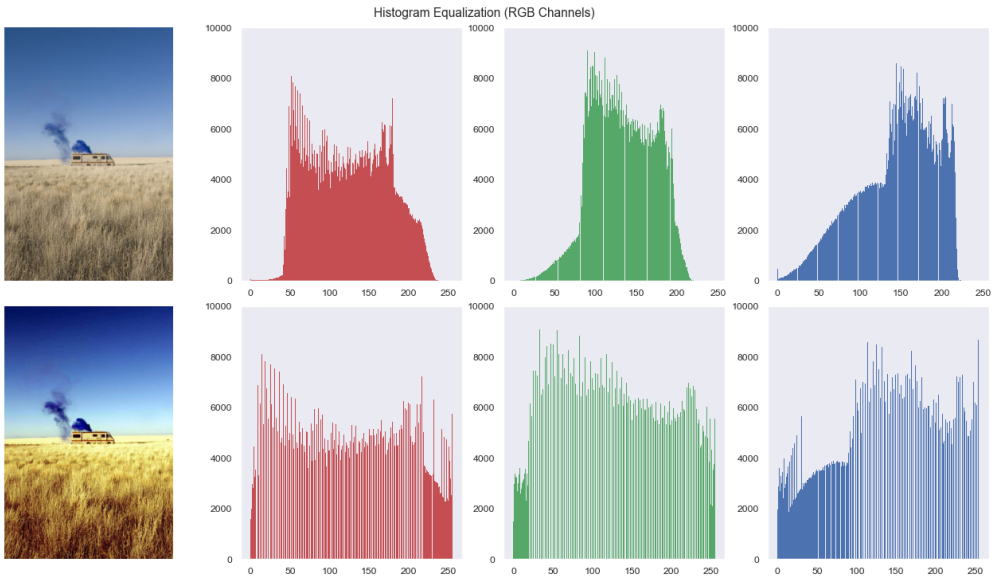
Edge Blur + Canny Edges + Garbage Mask | 100, 235



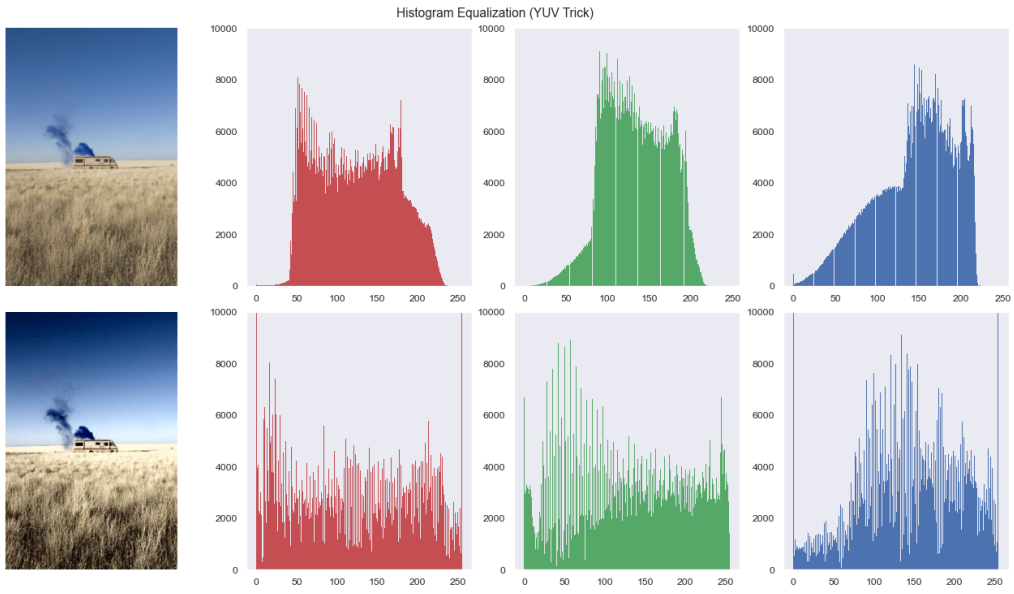
سوال ۵)

برای انجام Histogram Equalization بر روی تصاویر رنگی نمی‌توانیم به صورت مستقل این کار را برای هر کانال رنگی انجام دهیم.

این کار باعث می‌شود هر کانال رنگی بدون توجه به اینکه چه نسبتی با سایر کانال‌ها دارد، سعی کند خود را در بازه ۰ ـ ۲۵۵ بکشاند.



راه حل بهتر برای این کار این است که ابتدا تصویر را به فضای رنگی YUV ببریم. در این فضا، اطلاعات روشنایی تصویر در کانال Y انباشته می‌شود و اطلاعات رنگی تصویر در کانال‌های U,V جای می‌گیرد. برای انجام Histogram Equalization در این فضا، کافی‌است کانال Y را Equalize کنیم. با برگرداندن تصویر به فضای RGB، اطلاعات روشنایی Equalize شده به تصویر اعمال می‌شود.



پایان