به نام خالق دانایی

محمد مهدی انصاری (۴۰۲۱۰۱۳۰۶) – سروش مومنی (۴۰۲۱۰۲۵۳۸) گزارش فاز دو پروژه سیگنال و سیستم

بخش تئوری :

سوال ۱-۱ :

همبستگی یا correlation بین دو سیگنال بیانگر میزان شباهت دو سیگنال از نظر شکل و رفتار در طی محور زمان است . اگر دو سیگنال الگو مشابهی داشته باشند و یا با هم به طور همبسته تغییر کنند ؛ همبستگی آن ها زیاد است . اگر مقادیر هر دو سیگنال در طی زمان با هم زیاد شود ، همبستگی مثبت و زیاد است و اگر یکی کم شود و دیگری زیاد شود دو سیگنال همبستگی دارند ولی منفی است و اگر رفتار دو سیگنال در طی زمان نا مربوط باشند ، همبستگی صفر و یا نزدیک به صفر است .

به طور ریاضی همبستگی را به شکل زیر تعریف می کنیم :

$$r = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \times \sum_i (y_i - \bar{y})^2}}$$

X , Y دو سیگنال ورودی می باشند .

$\bar{x}$ و $\bar{y}$ مقدار متوسط دو سیگنال می باشند .

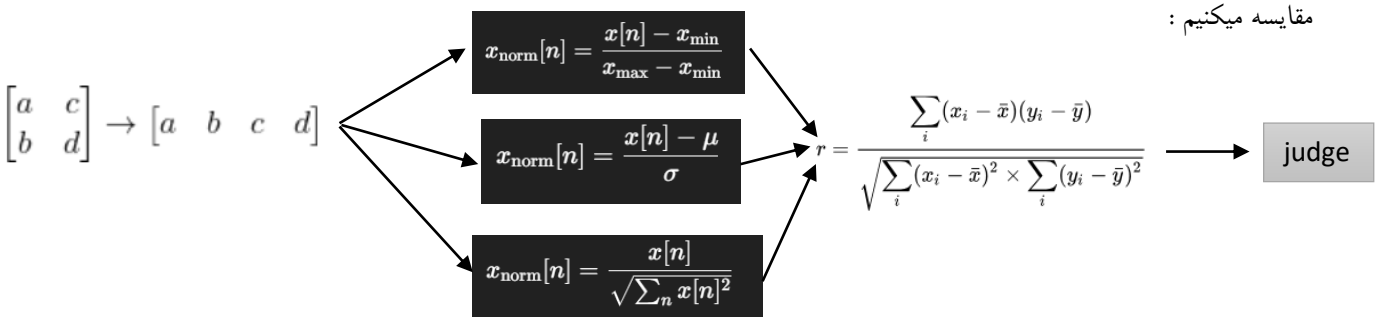
مقادیر i روی طول دو سیگنال به طور N دلخواه پیمایش می شود .

r ضریب همبستگی نام دارد که مقادیر آن بین منفی یک تا یک متغیر است .

- r =1 : همبستگی مثبت، ظاهری کاملا تطابق دارند.
- r = 0 : همبستگی صفر ، نامربوط
- r=-1: همبستگی منفی،سیگنال ها برعکس هم هستند

سوال ۱-۲ :

برای مقایسه دو تصویر ، ابتدا تصویر هارا به صورت ماتریس های دو بعدی فرض می کنیم که هر درایه میتواند روشنایی یا رنگ باشد . فرض میکنیم ماتریس دو بعدی m در n باشد . حال ماتریس دو بعدی را به یک بردار m*n یک بعدی تبدیل میکنیم . سپس نرمالیزه میکنیم و ورودی هارا که به صورت یک سیگنال یک بعدی می باشند با روش همبستگی که در بخش قبلی توضیح داده شده مقایسه میکنیم :



سوال ۲-۱ :

همانطور که میدانیم در هنگام تصویر برداری به دلیل عدم فوکس درست و تکان خوردن لنز دوربین تصویر می تواند blur شود . حال اگر فرض کنیم که این پدیده را با خروجی سیستم روبه رو پیاده سازی کنیم :

$$H_{\text{blur}}(z) = \frac{1-p}{1-pz^{-1}}$$

حال به صورت بازگشتی فیلتر IIR ذکر شده به صورت زیر عنوان می شود :

$$y[n] = (1-p)\cdot x[n] + p\cdot y[n-1]$$

که در آن x[n] سیگنال ورودی (شدت پیکسل ها) می باشد .

و y[n] سیگنال خروجی که تصویر blur شده می باشد . و p ضریب فیلتر است که به طور کلی بین صفر و یک می باشد .

سپس این رابطه بازگشتی به به صورت افقی ، سطر به سطر و عمودی ستون به ستون که به صورت وکتوری از پیکسل ها می باشند ، اعمال می کنیم . این فیلتر به صورت جدا گانه بر هر سطر و ستون عمل می کند . برای جلوگیری از اختلال فاز هم به صورت forward و هم به صورت backward فیلتر را اعمال می کنیم .

عملکرد پارامتر p : میزان قدرت فیلتر را در هموار سازی (smoothing) نشان میدهد .

مقدار p به طور کلی بین صفر و یک است و هر چه به یک نزدیک تر می شود فیلتر بهتر و قوی تر blur می کند .

نحوه پیاده سازی فیلتر به صورت سطری و ستونی به شکل زیر است :

```
# Row-wise filtering
   for row in range(filtered.shape[0]):
       for col in range(1, filtered.shape[1]):
           filtered[row, col] =(1-p)*filtered[row,col]+p*filtered[row,col-1]
   # Column-wise filtering
   for col in range(filtered.shape[1]):
       for row in range(1, filtered.shape[0]):
           filtered[row,col] =(1-p)*filtered[row,col]+p*filtered[row-1,col]
```

سوال ۲-۲ :

از نظر تئوری ما با داشتن کرنل بلر می توانیم به صورت معکوس عمل کرده با فیلتر معکوس آن تصویر را ریکاوری کنیم . اما چالش هایی در این مسیر وجود دارد مانند : تقویت نویز هنگام اعمال تابع معکوس به خصوص در فرکانس های بالا و یا ایده آل نبودن کرنل که ممکن است به صورت کامل معکوس آن منطبق بر فیلتر نباشد . پس از نظر تئوری این کار عملی است اما چالش هایی مانند نویز و غیر ایده آل بودن با آن همراه است .

سوال ۳ :

در پردازش تصویر ، نمونه برداری دو بعدی یا 2D sampling فرایندی است که یک تصویر ورودی به شبکه ای گسسته از پیکسل ها تبدیل می شود و هر پیکسل شدت روشنایی و یا رنگ را در یک مکان خاص از تصویر نشان میدهد . این فرایند تعیین کننده وضوح یا رزولوشن تصویر می باشد .

نرخ نمونه برداری یا رزولوشن نشان دهنده تعداد پیکسل ها در واحد سطح است ، بدیهی است با بالا رفتن این نرخ به تصویری با وضوح بیشتری دست پیدا می کنیم . برای بازسازی صحیح تصویر ، نرخ نمونه برداری باید حداقل دو برابر بیشترین فرکانس موجود در تصویر باشد . این گزاره به قضیه نمونه برداری نایکوئیست – شانون نسخه 2D معروف است و اگر این شرط رعایت نشود ، اعوجاج یا (aliasing) رخ می دهد و باعث تحریف و یا ایجاد الگو های غلط در تصویر می شود . از طرفی در نرخ نمونه برداری پایین ، جزئیات مهم از بین می روند و درک و تشخیص تصویر دچار اشکال می شود .

بخش عملی :

- تصاویر ایده آل : ابتدا در این بخش می خواهیم تصاویر ایده آل پلاک را پردازش کنیم و شماره آن را به صورت عدد تشخیص دهیم ، سپس با کاهش نرخ کیفیت عکس یا همان down sampling تاثیر کیفیت بر تشخیص درست و حداقل کیفیت لازم و نرخ نمونه برداری را برای تشخیص درست شماره پلاک بررسی کنیم :

ابتدا به بررسی سه تابع مورد نیاز می پردازیم :

- برای لود کردن عکس ها و توابع مورد نیاز از آدرس های مورد نظر از این توابع استفاده می کنیم :

```
def load(address):
patterns = {}
    for name in sorted(os.listdir(address)):
            num = os.path.splitext(name)[0]
            path = os.path.join(address, name)
            image_data = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
            if image_data is not None:
                patterns[num] = image_data
    return patterns
```

برای پردازش بهتر هنگام خواندن عکس ها آن هارا احتیاطا به grayscale می بریم .

- بعد از دریافت عکس های پلاک باید شماره هارا از آن ها جدا کرده و تک تک کنار هم به صورت ارقامی مجزا در قالب یک عدد که همان شماره پلاک باشد آماده پردازش کنیم برای این از تابع زیر استفاده میکنیم :

```
-   def split_func(input, show_plots=True):
```

ابتدا عکس های دریافت شده را از نظر سایز بررسی و سپس از فیلتر های افزایش کیفیت عبور می دهیم ، بعد از آن به وسیله threshold معین پس زمینه را سیاه و حروف را سفید میکنیم و بعد از ترمیم این تصویر آماده یافتن اشکال در آن می باشد :

```
 # we first change color to gray and then apply blur filter to improve quality
    grayed = cv2.cvtColor(real_image, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(grayed, (5, 5), 0)
    # now we Apply an inverted binary threshold. This makes characters white and
the background black.
    _, black_white = cv2.threshold(blurred, 0, 255, cv2.THRESH_BINARY_INV +
cv2.THRESH_OTSU)
    # we  fill small gaps or holes and make them solid and easier to detect .
    filling_elements = np.ones((3,3), np.uint8)
    clean_image = cv2.morphologyEx(black_white, cv2.MORPH_CLOSE,
filling_elements)
```

حال اشکال پیدا شده با سایز خیلی کوچک و سایز خیلی بزرگ را حذف میکنیم و بعد از سورت کردن اشکال پیدا شده که هفت یا کمتر است آن هارا مرتب و آماده ارسال به بخش تشخیص و شناسایی اعداد و حروف میکنیم :

```
# finding
    shapes, _ = cv2.findContours(clean_image, cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)
    possible_shapes = []
    for shape in shapes:
        shape_area = cv2.contourArea(shape)
        # we  filter out shapes that are too small or too big (for canceling
noise or errors ).
        if 50 < shape_area < (w * h / 5):
            possible_shapes.append(shape)
    if len(possible_shapes) >= 7: # our license has seven number but if system
found less , it would keep it all
        possible_shapes.sort(key=cv2.contourArea, reverse=True)
        final_shapes = possible_shapes[:7]
    else:
        final_shapes = possible_shapes
    # now we sort them and stick them together
    shape_list = sorted(final_shapes, key=lambda c: cv2.boundingRect(c)[0])
```

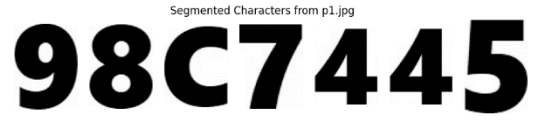
- حال که عکس پلاک ما به یک لیست از عکس های جدا شده کنار هم تبدیل شده باید تک به تک آن عکس هارا با محاسبه همبستگی با تصاویر الگو شناسایی کنیم ، ما بر اساس ضریب همبستگی حاصل مقایسه را با هر الگو ذخیره و بیشترین اندازه ضریب را به عنوان بهترین عکس و در اصل کاراکتر شناسایی شده در نظر می گیریم :

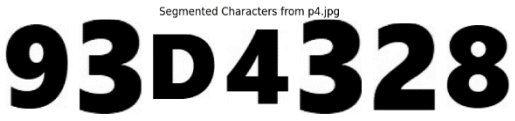
```
# now we compare each segment with all patterns in this loop
        for name, image in patterns.items():
           # resize to avoid any mismatch
            template_resized = cv2.resize(image, (standard_w, standard_h))
            # calculate the score (correlation ratio which is between -1 and
1)
            match_result = cv2.matchTemplate(prepared, template_resized,
cv2.TM_CCOEFF_NORMED)
            _, similarity_score, _, _ = cv2.minMaxLoc(match_result)
            # consider the best one through loop with choosing the max rate
            if similarity_score > best:
                best = similarity_score
                choice = name
        # add the best matched one
        ans += choice
        correlations_ratios.append(best)
```

قبل از اجرای کد های بالا کمی تغییرات روی سایز و مکان تصویر و کمی فیلتر برای افزایش کیفیت و بهتر پردازش شدن تصاویر اعمال میکنیم .

خروجی ها :

تصاویر سمت راست عکس پلاک ها و تصاویر سمت چپ کاراکتر های جدا شده و چسبانده شده که آماده ورودی به بخش مقایسه و پردازش هستند ، می باشند :





خروجی ترمینال به صورت زیر است که توانسته درست تشخیص دهد و امتیاز در اصل همان ضریب همبستگی است که توانسته برای آن کاراکتر مورد نظر بیشترین امتیاز را کسب کند و انتخاب شود :

```
Processing: p1.jpg
Result: 98 C 7445
Scores: ['0.69', '0.68', '0.69', '0.79', '0.87', '0.87', '0.77']

Processing: p2.jpg
Result: 56 A 7495
Scores: ['0.77', '0.70', '0.69', '0.79', '0.81', '0.69', '0.77']

Processing: p3.jpg
Result: 79 B 1208
Scores: ['0.79', '0.69', '0.78', '0.72', '0.74', '0.67', '0.68']

Processing: p4.jpg
Result: 93 D 4328
Scores: ['0.69', '0.78', '0.70', '0.87', '0.78', '0.74', '0.68']
```

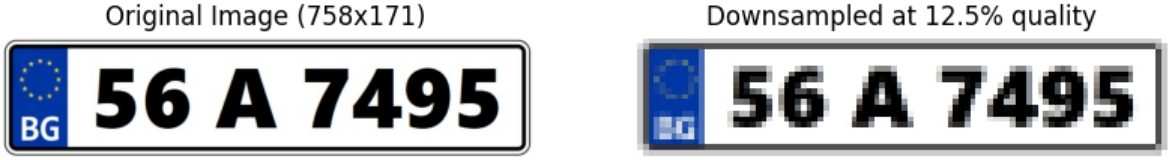
- خواسته بعدی فایل پروژه بررسی اثر کاهش کیفیت یا همان down sampling با نرخ های مختلف بر روی عملکرد سیستم بود :

ابتدا اثر این عمل را با نرخ های تعیین شده بر روی کیفیت تصاویر پلاک مشاهده می کنیم :

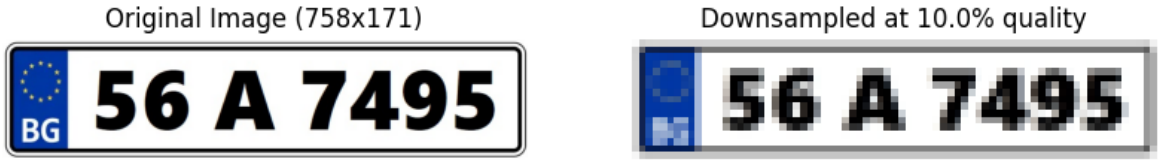
Rate=1/2

Rate=1/4

Rate=1/6



Rate=1/8



Rate=1/10

حال شماره تشخیص داده شده را به ازای کیفیت های و نرخ های مختلف بررسی می کنیم :

خروجی ترمینال به شکل زیر است :

```
 Starting Downsampling Analysis

Analyzing quality effects on: p2.jpg

Results Table: Accuracy vs. Quality
Quality | Avg Score        | Correct/wrong | Recognized Text
-------------------------------------------------------------
1.000   | 0.7459           | True          |56 A 7495
0.500   | 0.7687           | True          |56 A 7495
0.250   | 0.7388           | True          |56 A 7495
0.167   | 0.4923           | False         |80A76
0.125   | 0.2990           | False         |60A9
0.100   | 0.3311           | False         |A09
```

آزمایش بر روی تصویر دوم انجام شد با تغییر کد می توانید نمونه را عوض کنید ، همانطور که مشاهده می کنید کمترین نرخی که توانسته سیستم تشخیص به درستی در آن عمل کند ¼ است و در کمتر از آن متوسط ضریب همبستگی کاراکتر ها افت قابل توجهی میکند و همانطور که در خروجی مشخص است نمی تواند تعداد مورد نظر و حروف درستی را تشخیص دهد و شناسایی کند .

پس کمترین نرخی که برنامه به درستی کار می کند در این حالت ۲۵٪ است .

انجام آزمایش down sampling توسط این بخش از کد انجام شد :

```
# diffrent rates of down sampling :
    rates = [1.0, 0.5, 0.25, 1/6, 1/8, 1/10]
    results = []
    # check the effect in diffrent rates through looping :
    for rate in rates:
        h, w, _ = original_plate_image.shape
        low_quality_image = cv2.resize(original_plate_image, (int(w*rate),
int(h*rate)), interpolation=cv2.INTER_AREA)
        # now we want to run again the process of identification with low qaulity
images
        parts = split_func(low_quality_image, show_plots=False)
        text_result, rates_list = recognize_func(parts, samples)
        # checking the results
        check = (text_result == correct_text)
        average_rate = np.mean(rates_list) if rates_list else 0
        results.append((rate, text_result, check, average_rate))
```

- **بخش عملی : حالت واقعی و نویز دار**

۱)استفاده از تابع استخراج کاراکتر بخش قبل بدون اعمال تغییر در عکس ها:

همانطور که انتظار میرود با توجه به محوشدگی وعدم وضوح کاراکترهای پلاک،خواندن پلاک شکست میخورد و تابع splitfunc هیچ کاراکتری را نمیتواند تشخیص دهد و در هر ۴ مورد خروجی تهی است:

```
Processing: p1.jpg
Result:
Scores: []

Processing: p2.jpg
Result:
Scores: []

Processing: p3.jpg
Result:
Scores: []

Processing: p4.jpg
Result:
Scores: []
```

۲و۳)تخمین p بهینه:

تبدیل z داده شده معادل با سیستم زیر است:

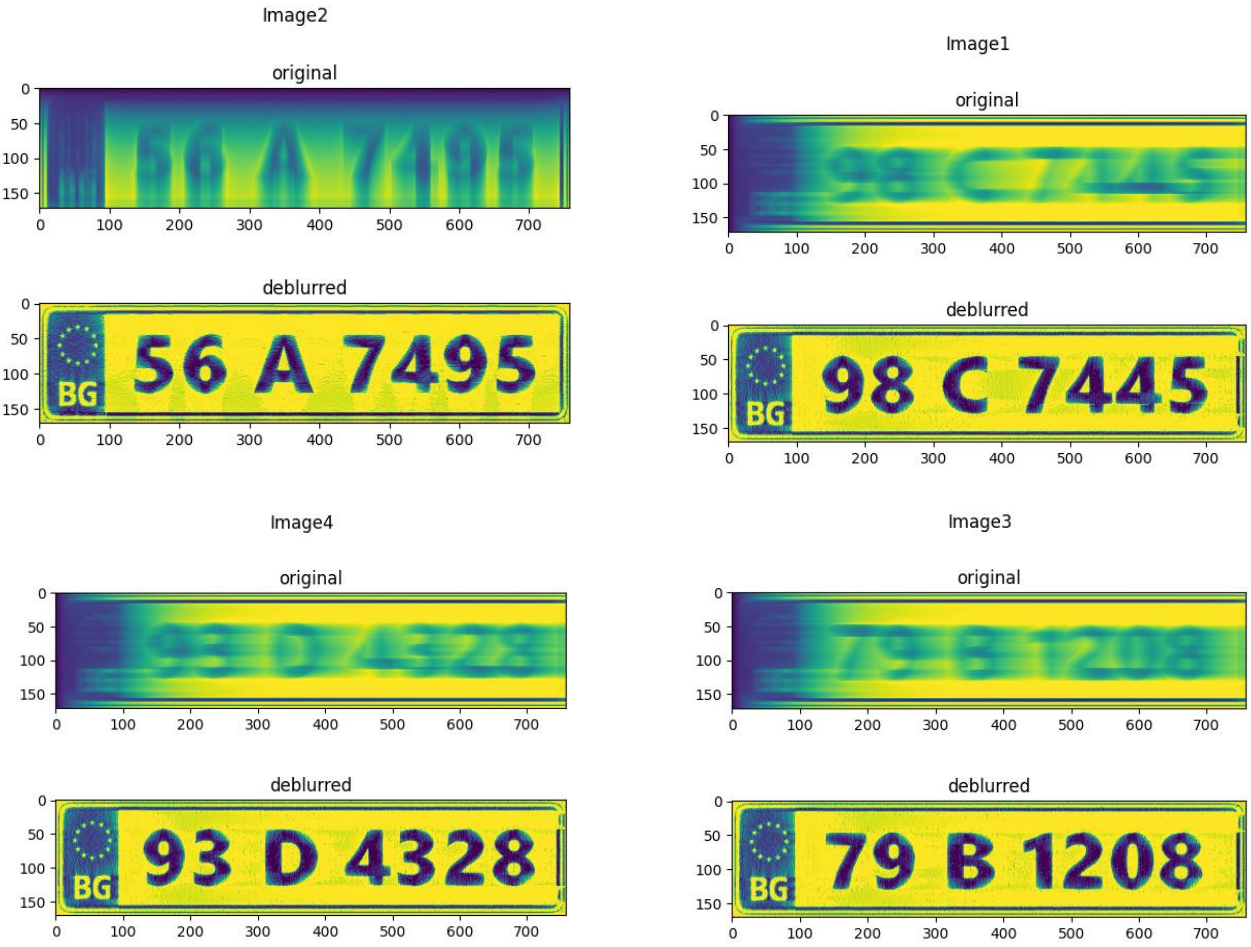
$$y[n] = (1-p)x[n] \; + \; py[n-1]$$

تصویر سیاه و سفید معادل آرایه ۲ بعدی است،و هر سطر یا ستون این عکس یک سیگنال گسسته است و این فیلتر روی آن قابل اجراست.ما در ابتدا عکس فیلترشده را داریم،یعنی خروجی این سیستم y[n] را داریم و از روی آن میتوانیم x[n] یعنی مقادیر قبل از اعمال فیلتر را با رابطه $x[n] = \frac{y[n]-p \cdot y[n-1]}{1-p}$ به دست آوریم.با توجه به اینکه 1-p در مخرج است p را کوچکتر از یک انتخاب میکنیم و هرچقدرp بیشتر به یک نزدیک باشد چنانکه در پاسخ سوال ۲ تئوری آمده فیلتر قویتر بوده است و بیشتر عکس را محو کرده،و از آنجا که عکس های پلاکها بسیارناخوانا هستند در مقایسه با عکس قرار داده شده در سوال ۲ متوجه میشویم p باید بسیار به یک نزدیک باشد و p بهینه را p=0.9 تخمین میزنیم.برای اعمال فیلتر به صورت نرم افزاری یک تابع deblurFilter_horizontal نوشته شده که با حلقه ای عمل فوق را روی هر سطر عکس انجام میدهد و یک تابع deblurFilter_vertical که همین کار را روی هر ستون عکس انجام میدهد.(تعداد سطر و ستون های هر عکس با دستور grayedImage.shape به دست می آید)

با آزمون و خطا به دست می آید که پلاک های ۱و۳و۴ با اعمال پادفیلتر افقی با پارامتر p=0.99 به عکس قابل قبولی میرسند و پلاک ۲ با اعمال پادفیلتر عمودی با همان پارامتر p=0.99 قابل قبول میشود:

```
plate_path_raw = "C://Users//LENOVO//Downloads//PHASE2//realistic//"
for i in range(1,5):
 plate_path=plate_path_raw+"p"+ str(i)+".jpg"
 real_image = cv2.imread(plate_path)
 grayed = cv2.cvtColor(real_image, cv2.COLOR_BGR2GRAY)
 grayed=np.array(grayed)
 if(i==2):
  grayed2=deblurFilter_vertical(grayed,0.99)
 else:
  grayed2=deblurFilter_horizontal(grayed,0.99)
```

خروجی:



۴)تشخیص کاراکترهای پلاک:

برای این قسمت تابع Optimumdeblur نوشته شده تا p بهینه را پیدا کند و همچنین این قابلیت را دارد که روی عکس هم فیلتر افقی و هم به طور ترکیبی فیلتر عمودی با p های نه لزوما یکسان اعمال کند.روش کار این تابع به این گونه است که ابتدا فیلتر افقی روی تمام سطور عکس با پارامترهای p=±0.51,0.53,..0.99 اعمال میکند و بین عکس های حاصل شده که شرط لازم دارا بودن ۷ کاراکتر(طبق تابع recognize) را داشته باشند،ملاک تشخیص بهترین فیلتر،میانگین ضرایب correlation است.سپس روی بهترین نتیجه حاصل شده تا اینجا(که میتواند همان عکس اولیه باشد) دوباره فیلترعمودی با همان پارامتر های p=±0.51,0.53,..0.99 اعمال میشود و بهترین نتیجه بین نتایج دارای ۷ کاراکتر طبق میانگین ضرایب correlation انتخاب میشود و به تابع splitfunc و سپس recognizefunc داده میشود تا خروجی را نمایش دهد.کد این بخش:

```
letters_path = "C://Users//LENOVO//Downloads//PHASE2//letters"
numbers_path = "C://Users//LENOVO//Downloads//PHASE2//numbers"
plates_path = "C://Users//LENOVO//Downloads//PHASE2//realistic"
#loading sources
number_samples = load(numbers_path)
letter_samples = load(letters_path)
samples = {**number_samples, **letter_samples}
# at the end we want to verify if our system works or not so we save the correct
answers
plates = ['p1.jpg', 'p2.jpg', 'p3.jpg', 'p4.jpg']
correct_answers = {
        'p1.jpg': '98 C 7445', 'p2.jpg': '56 A 7495',
        'p3.jpg': '79 B 1208', 'p4.jpg': '93 D 4328'
}
# 4. Loop through each license plate file for testing.
for plate_name in plates:
    plate_path = os.path.join(plates_path, plate_name)
    print(f"\nProcessing: {plate_name}")
    output, rates = OptimumDeblur(plate_path,samples)
    print(f"Result: {output}")
    print(f"Scores: {[f'{s:.2f}' for s in rates]}")
```

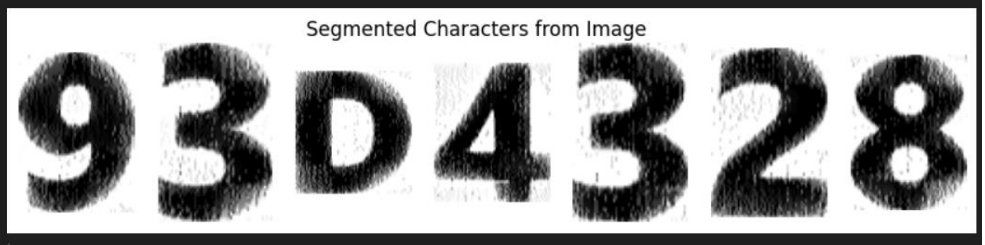
نتایج:

```
Processing: p1.jpg
```


```
Result: 98 C 7445
Scores: ['0.82', '0.77', '0.74', '0.79', '0.87', '0.79', '0.73']

Processing: p2.jpg
```


```
Result: 56 A 7495
Scores: ['0.71', '0.78', '0.77', '0.74', '0.73', '0.78', '0.71']

Processing: p3.jpg
```


```
Result: 79 B 1208
Scores: ['0.80', '0.79', '0.76', '0.72', '0.77', '0.76', '0.73']

Processing: p4.jpg
```

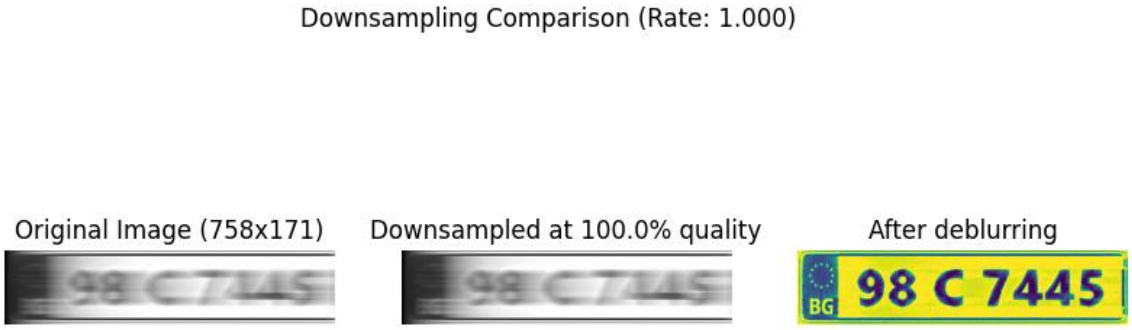


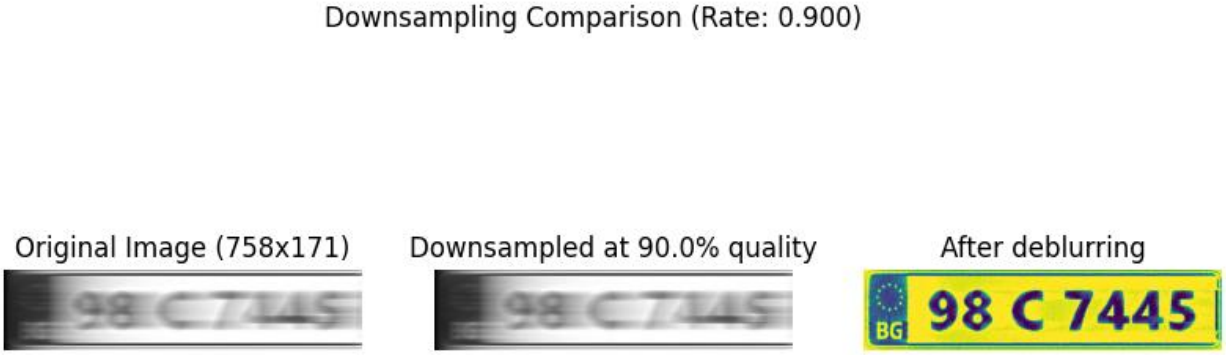

```
Result: 93 D 4328
Scores: ['0.81', '0.75', '0.66', '0.85', '0.76', '0.77', '0.74']
```

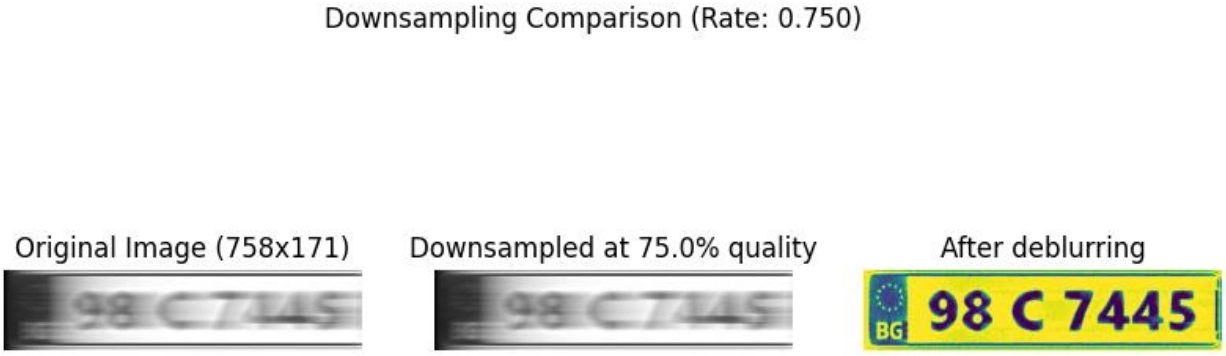
(این نتایج در Jupiter notebook ارسالی در پوشه کدها نیز نمایش داده شده اند)

۵)downsampling:

کد این قسمت دقیقا مانند کد قسمت downsampling بخش ایده ال است،با این تفاوت که اینبار در کنار دو عکس اصلی و عکس پس از downsampling،یک عکس دیگر نیز نشان میدهیم که عکس Grayscale حاصل از اعمال فیلتر deblur_horizontal(برای پلاک های ۱و۳و۴) و یا فیلتر deblur_vertical(برای پلاک ۲) است.همچنین تابع splitfunc مقدار عکس آخر یعنی عکس پادفیلتر شده را ورودی میگیرد.همه نتایج را برای پلاک اول نشان میدهیم:







Downsampling Comparison (Rate: 0.500)

Original Image (758x171)

Downsampled at 50.0% quality

After deblurring

Downsampling Comparison (Rate: 0.250)

Original Image (758x171)

Downsampled at 25.0% quality

After deblurring

Downsampling Comparison (Rate: 0.167)

Original Image (758x171)

Downsampled at 16.7% quality

After deblurring

Downsampling Comparison (Rate: 0.100)

Original Image (758x171)

Downsampled at 10.0% quality

After deblurring

مشاهده میشود تا downsampling تا 25 درصد و حتی با ارفاق تا ۱۶,۷ درصد،پلاک برای چشم انسان قابل تشخیص است اما تابع recognize در کاهش نمونه ۱۰۰ درصد(بدون تغییر) و ۹۰ درصد کاملا درست شناسایی میکند،در کاهش نمونه ۷۵ و ۵۰ درصد با مقداری اشتباه حدود نصف کاراکترها را به درستی شناسایی میکند و در زیر ۵۰ درصد به کل شکست میخورد(تعداد کاراکترهای شناسایی شده توسط split function صفر بوده لذا مجموعه rates تهی و میانگین امتیاز همبستگی صفر است)میانگین امتیاز همبستگی نیز با افزایش downsampling نزولی اکید است.

نتیجه برای پلاک اول:

```
Results Table: Accuracy vs. Quality
Quality | Avg Score        | Correct/wrong | Recognized Text
-------------------------------------------------------------
1.000   | 0.7769           | True          |98 C 7445
0.900   | 0.7608           | True          |98 C 7445
0.750   | 0.7599           | False         |98C75
0.500   | 0.6757           | False         |A98C75
0.250   | 0.0000           | False         |
0.167   | 0.0000           | False         |
0.100   | 0.0000           | False         |

Conclusion: The system works down to approx. 0.900 quality.
```

پلاک دوم:

```
Results Table: Accuracy vs. Quality
Quality | Avg Score        | Correct/wrong | Recognized Text
-------------------------------------------------------------
1.000   | 0.7451           | True          |56 A 7495
0.900   | 0.7405           | True          |56 A 7495
0.750   | 0.7400           | True          |56 A 7495
0.500   | 0.6318           | False         |C5 6 A495
0.250   | 0.0000           | False         |
0.167   | 0.0000           | False         |
0.100   | 0.0000           | False         |

Conclusion: The system works down to approx. 0.750 quality.
```

پلاک سوم:

```
Results Table: Accuracy vs. Quality
Quality | Avg Score        | Correct/wrong | Recognized Text
-----------------------------------------------------------------
1.000   | 0.7483           | True          |79 B 1208
0.900   | 0.7422           | True          |79 B 1208
0.750   | 0.7475           | True          |79 B 1208
0.500   | 0.6888           | False         |A9B208
0.250   | 0.0000           | False         |
0.167   | 0.0000           | False         |
0.100   | 0.0000           | False         |

Conclusion: The system works down to approx. 0.750 quality.
```

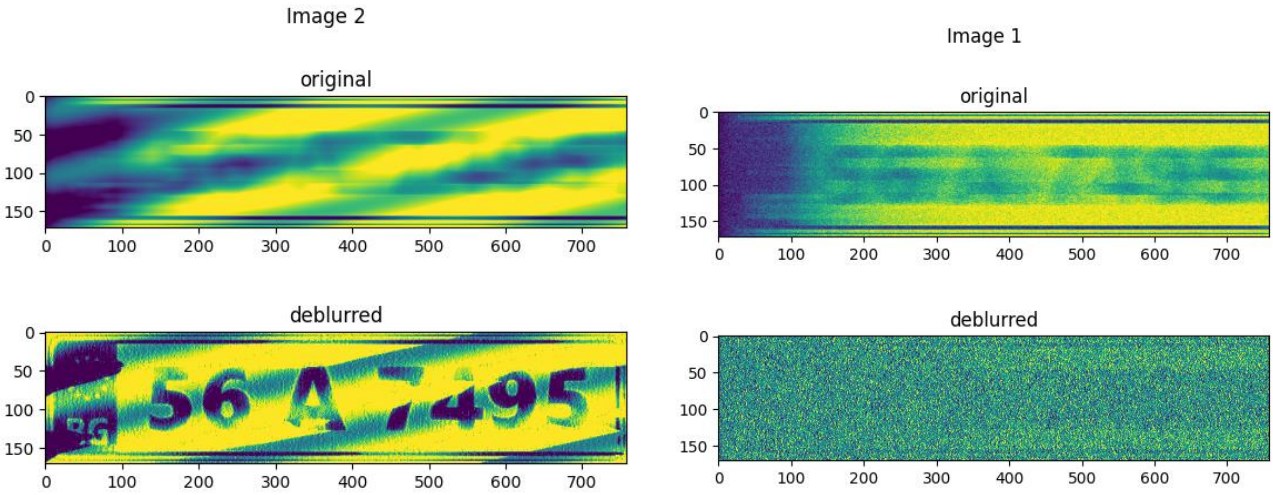
پلاک چهارم:

```
Results Table: Accuracy vs. Quality
Quality | Avg Score        | Correct/wrong | Recognized Text
-----------------------------------------------------------------
1.000   | 0.7542           | True          |93 D 4328
0.900   | 0.7434           | True          |93 D 4328
0.750   | 0.7556           | True          |93 D 4328
0.500   | 0.7075           | False         |A93D4328
0.250   | 0.0000           | False         |
0.167   | 0.0000           | False         |
0.100   | 0.0000           | False         |

Conclusion: The system works down to approx. 0.750 quality.
```

پلاک های دوم تا چهارم تا downsampling با نرخ 75 درصد درست کار میکنند

۶)عکس های نویزدار:

با اعمال فیلتر ضد محوشدگی روی دو عکس داریم:



عکس دوم نویز با فرکانس کمتری دارد،زیرا نویز آن به این صورت است که در سطر،در دو یا سه بازه نسبتا کوتاه عدد بزرگی دارد و در سایر n ها تقریبا صفر است،لذا دوره تناوب نویز آن در حدود $\frac{700(\text{پیکسل به عکس طول})}{3}$ است،اما در عکس اول تقریبا از هر دو پیکسل متوالی یکی تغییر رنگ داده که یعنی دوره تناوب نویز آن در حدود ۲(یا هر عدد به اندازه کافی کوچکی که باعث شود چشم انسان نقاط زرد و آبی را چه در راستای افقی چه عمودی تقریبا یک پیکسل درمیان ببیند)