به نام خدا

گزارش پروژه بینایی کامپیوتر – 99521199 و 99521415

توضیحات به صورت کامل در کد آمده اما برای شرح کوتاه میتوانیم به این اشاره کنیم که ابتدا یک عکس را به عنوان ورودی به تابع main میدهیم و آنجا پس از اینکه با روش گفته شده اسکن و کراب شد چک میکنیم که آیا این کارت از جنس کارت ملیست و یا کارت بانکی است.

برای این چک کردن بررسی میکنیم که آیا کارت تصویر خورشید پایین سمت چپ را دارد یا نه.

```
def is_national_card(image):
    sun = image[700:1000 , 0:300]
    sun_template = cv2.imread('sun.png', 0)
    # template matching
    res = cv2.matchTemplate(sun, sun_template, cv2.TM_CCOEFF_NORMED)
    threshold = 0.5
    # show hit map
    loc = np.where(res >= threshold)
    if len(loc[0]) > 0:
        return True
    else:
        return False
```

وقتی فهمیدیم که کارت از چه جنسی است باید برای هرکدام به طور خاص اطلاعات خواسته شده را استخراج کنیم.

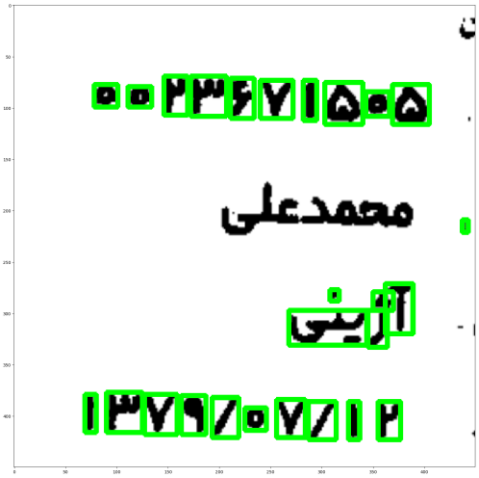
اولین حالت بررسی کارت ملیست.

```
def detect_national_card(image):
    templates = generate_templates('Fa_dataset')
    image = image[200:650, 800:1250]
    clustered_rectangles, matched_image = fa_extract_segments(image)
    for cluster in clustered_rectangles:
        phrase = template_matching(image, matched_image, cluster, templates)
        if "/" in phrase:
            print("expir_date: ", phrase)
            continue
        elif len(phrase) == 10:
            national_id = phrase
            print("national_id: ", national_id)
```

دیتاست `'Fa_dataset'` دیتاست ما برای اعداد فارسی است که درون آن اعداد 0 تا 9 به علاوه / برای استفاده در تمپلیت مچینگ قرار گرفته شده. ابتدا دیتاست خوانده شده و در `templates` قرار میگیرد و سپس عکسی که اسکن و کات کردیم را طوری برش میدهیم که فقط قسمت های شماره ملی و تاریخ انقضا مشخص باشد و بعد آن را به تابع `fa_extract_segment` میهیم تا نواحی مهم اسخراج بشع و بعد از اون اون نواحی رو بررسی میکنیم که آیا با تمپلیت هایی که داریم تشابهی کشف میه و عددی پیدا میشه یا نه.اگر / داشت یعنی تاریخ انقضاست و اگر نداشت یعتی شماره ملیه.

ورودی نمونه:

خروجی:




```
national_id:  0023671505
expir_date:  1379/07/12
```

حال به بررسی کد برای حالت کارت بانکی میرسیم:

```
def detect_credit_card(image):
    templates = generate_templates('dataset')
    clustered_rectangles, matched_image = extract_segments(image)
    for cluster in clustered_rectangles:
        phrase = template_matching(image, matched_image, cluster, templates)
        if "/" in phrase:
            slash_index = phrase.index("/")
            expir_date = phrase[slash_index - 2: slash_index + 3]
            if len(phrase)>=7 and slash_index >= 4:
                print("expir_date: ", expir_date)
        elif len(phrase) == 16:
            card_number = phrase
            print("card_number: ", card_number)
```

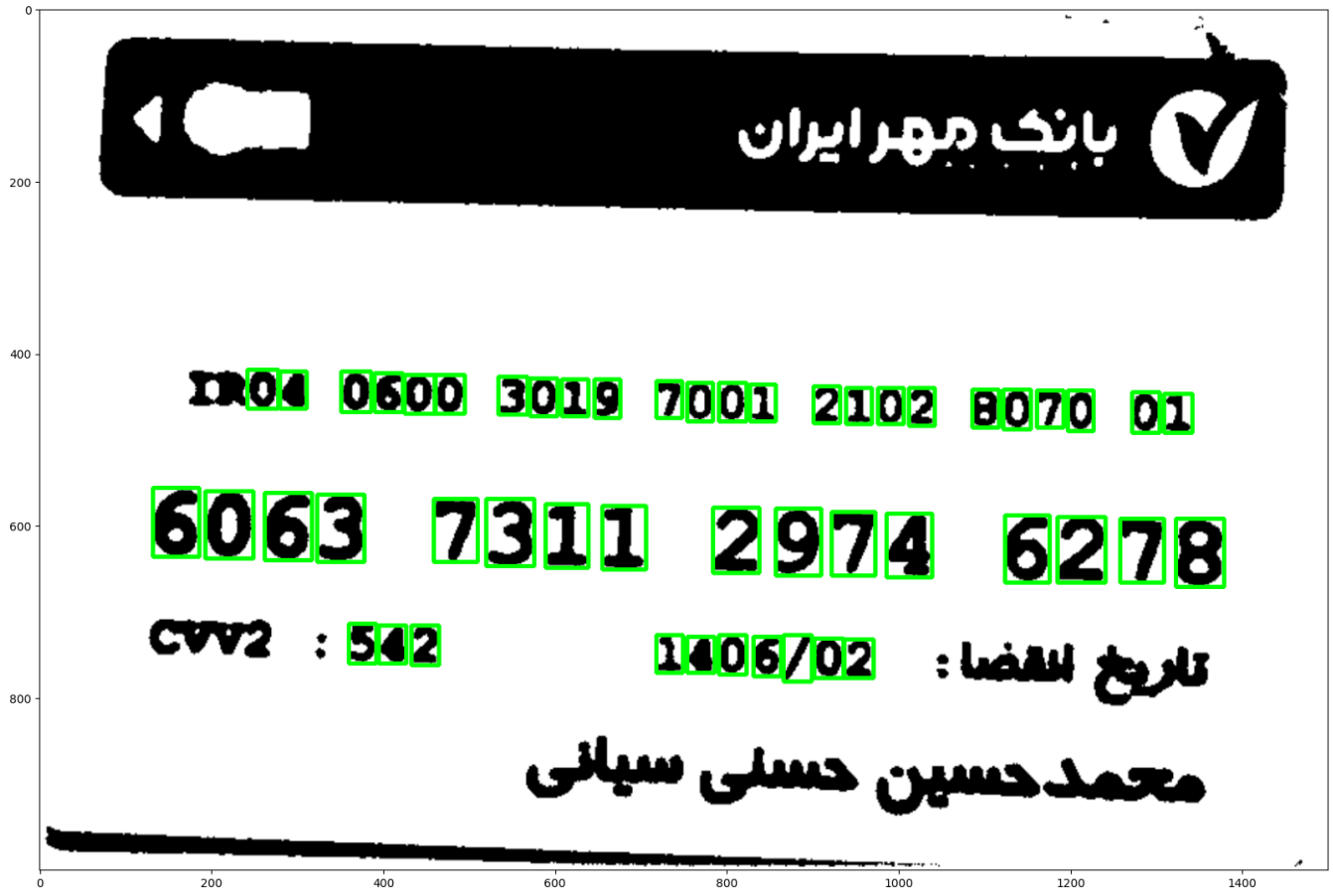
فولدر dataset شامل پند نمونه عدد با فونت های مختلف انگلیسی است که برای تمپلیت مچینگ استفاده میکنیم.

سپس مانند قبل نواحی مهم را استخراج میکنیم و برای هر کدام چک میکنیم که آیا مچ پیدا میشه یا نه

در صورتی که پیدا کرد و 16 رقم بود شماره کارت هست و اگر / داشت تاریخ انقضا.

نمونه ورودی:

خروجی:




```
card_number:  6063731129746278
expir_date:  06/02
```

این نمونه خروجی ما برای دو حالت خواسته شده است.

حال به توضیح توابع استفاده شده در هر بخش می پردازیم:

1. Reorder :

```
def reorder(poly):
    # Find the index of the point with the lowest y-coordinate
    p_lowest = np.argmax(poly[:, 1])

    # Check if there are two points with the same lowest y-coordinate
    if np.count_nonzero(poly[:, 1] == poly[p_lowest, 1]) == 2:
        # Find the index of the point with the lowest sum of x and y coordinates
        p0_index = np.argmin(np.sum(poly, axis=1))

```

```
        # Determine the indices of the other points in clockwise order
        p1_index = (p0_index + 1) % 4
        p2_index = (p0_index + 2) % 4
        p3_index = (p0_index + 3) % 4

        # Reorder the points and return the new polygon
        return poly[[p1_index, p0_index, p3_index, p2_index]]
    else:
        # Find the indices of the points to the right and left of the lowest point
        p_lowest_right = (p_lowest - 1) % 4
        p_lowest_left = (p_lowest + 1) % 4

        # Calculate the angle between the lowest point and the point to the right
        angle = np.arctan2(poly[p_lowest_right][1] - poly[p_lowest][1], poly[p_lowest_right][0] - poly[p_lowest][0])

        # Check if the angle is negative
        if angle < 0:
            print(angle, poly[p_lowest], poly[p_lowest_right])

        # Check if the angle is greater than 45 degrees (pi/4 radians)
        if np.degrees(angle) > 45:
            # Determine the indices of the points in counterclockwise order
            p2_index = p_lowest
            p1_index = (p2_index - 1) % 4
            p0_index = (p2_index - 2) % 4
            p3_index = (p2_index + 1) % 4
            return poly[[p1_index, p0_index, p3_index, p2_index]]
        else:
            # Determine the indices of the points in counterclockwise order
            p3_index = p_lowest
            p0_index = (p3_index + 1) % 4
            p1_index = (p3_index + 2) % 4
            p2_index = (p3_index + 3) % 4
            return poly[[p1_index, p0_index, p3_index, p2_index]]

```

تابع `reorder` به عنوان ورودی یک چندضلعی محدب (polygon) به شکل یک آرایه دو بعدی دریافت می‌کند و آن را مرتب می‌کند. فرض می‌شود که چندضلعی ورودی دارای چهار راس است .

ابتدا، این تابع نقطه‌ای را پیدا می‌کند که کمترین مختصات `y` را دارد. برای این کار، اندیس نقطه با بیشترین مقدار `y` را محاسبه می‌کند و آن را در متغیر `p_lowest` ذخیره می‌کند.

سپس، بررسی می‌شود که آیا دو نقطه با کمترین مختصات `y` دارای مقدار یکسان هستند یا خیر. اگر دو نقطه دارای مختصات یکسان باشند، این به معنی وجود دو نقطه با کمترین مختصات `y` است. در این صورت، تابع محاسباتی را انجام می‌دهد تا نقطه‌ای که مجموع مختصات `x` و `y` کمتری داشته باشد را پیدا کند. سپس، نقاط را به ترتیب مخالف عقربه‌های ساعت مرتب می‌کند و چندضلعی جدید را برمی‌گرداند.

در غیر این صورت(هنگامی که دو نقطه با کمترین مختصات `y` مختصات یکسان ندارند)، ابتدا اندیس نقطه‌ای که به سمت راست نقطه کمترین مختصات `y` قرار دارد را محاسبه می‌کند و به ترتیب مقدار عقربه‌های ساعت از نقطه کمترین مختصات `y` به نقطه‌ای که به سمت راست آن قرار دارد را محاسبه می‌کند. سپس، تابع زاویه محاسبه می‌کند که آن را به عنوان زاویه بین خطی که از نقطه کمترین مختصات `y` به سمت راست آن کشیده می‌شود و خط افقی در نقطه کمترین مختصات `y` در نظر می‌گیرد .

سپس، بررسی می‌شود که آیا زاویه به دست آمده منفی است یا خیر. اگر زاویه منفی باشد، این به معنی وجود یک زاویه بیشتر از ۱۸۰ درجه است. در این صورت، تابع نقاط را به ترتیب مقدار عقربه‌های مخالف عقربه‌های ساعت مرتب می‌کند و چندضلعی جدید را برمی‌گرداند.

در غیر این صورت (زاویه کمتر یا مساوی ۱۸۰ درجه)، تابع نقاط را به ترتیب مقدار عقربه‌های عقربه‌های ساعت مرتب می‌کند و چندضلعی جدید را برمی‌گرداند.

در هر دو حالت، چندضلعی جدید که نقاط را به ترتیب صحیح مرتب کرده است را برمی‌گرداند.

2. reflect 101

```
def Reflect101(img,filter_size):
    image = np.pad(img, pad_width=((filter_size//2, filter_size//2), (filter_size//2,
filter_size//2)), mode='reflect', reflect_type='odd')
    return image
```

کد مانند یکی از تمرین ها

.2Averaging_Blurring

```
def Averaging_Blurring(img, filter_size):
    image = Reflect101(img, filter_size)
    result = np.zeros((img.shape))
    avg_filter = np.ones((filter_size , filter_size))/ (filter_size * filter_size)
    for i in range(img.shape[0]):
      for j in range(img.shape[1]):
        result[i,j] = np.convolve(image[i :i + filter_size , j : j +
filter_size].flatten(), avg_filter.flatten() , mode='valid')
    return result
```

مانند یکی از تمرینها

.3distance

```
def calculate_distance(point1, point2):
    x1, y1 = point1
    x2, y2 = point2
    distance = math.sqrt((x2 - x1)**2 + (y2 - y1)**2)
    return distance
```

مانند یکی از تمرینها

4. scanning:

```
def scanner(img , kernel_size ):
  preprocessed_image = preprocess_image(img, kernel_size)
  card_contour = find_card_contour(preprocessed_image)
  perspective_transformed = perspective_transform(img, card_contour)
  plt.imshow(perspective_transformed , cmap = 'gray')
  cv2.imwrite('output.jpg', perspective_transformed)
```

اول با تابع نوشته شده تصویر را preprocess میکنیم (gray , blur , canny , dilate) سپس گوشه ها و ضالع هارا در میاریم و بعد presprctive transform را پیاده سازی میکنیم (مانند یکی از تمرین ها)

5.make templates :

```
def make_templates(template):
    list_candid_temps = []
    # Define the desired number of resize steps
    resize_steps = 10
    # Calculate the resize ratios for both width and height
    width_ratios = np.linspace(1, 0.4, resize_steps)
    height_ratios = np.linspace(1, 0.4, resize_steps)
    # Iterate over the resize ratios
    for width_ratio in width_ratios:
        for height_ratio in height_ratios:
            if abs(width_ratio - height_ratio) == 0:
                # Resize the template
                resized_template = cv2.resize(template, (int(template.shape[1] *
width_ratio), int(template.shape[0] * height_ratio)))
                # erosion
                kernel = np.ones((5,5),np.uint8)
                erosion = cv2.erode(resized_template,kernel,iterations = 1)
                # Rotate the resized template
```

```
                for angle in range(-5, 5):
                    rotated_template = cv2.warpAffine(resized_template,
cv2.getRotationMatrix2D((resized_template.shape[1] / 2, resized_template.shape[0] / 2),
angle, 1), (resized_template.shape[1], resized_template.shape[0]))
                    # Add the rotated template to the list of candidates
                    list_candid_temps.append(rotated_template)
                # Rotate the resized template
                for angle in range(-5, 5):
                    rotated_template = cv2.warpAffine(erosion,
cv2.getRotationMatrix2D((erosion.shape[1] / 2, erosion.shape[0] / 2), angle, 1),
(erosion.shape[1], erosion.shape[0]))
                    # Add the rotated template to the list of candidates
                    list_candid_temps.append(rotated_template)

    return list_candid_temps
```

تابع `make_templates` یک قالب (template) تصویر را به عنوان ورودی دریافت می‌کند و قالب‌های جدید را ایجاد می‌کند. در این تابع، ابتدا یک لیست خالی به نام `list_candid_temps` تعریف می‌شود که قالب‌های جدید را در آن ذخیره می‌کنیم.

سپس، تعداد مراحل تغییر اندازه را تعیین می‌کنیم. در اینجا، تعداد مراحل تغییر اندازه را برابر با ۱۰ قرار داده‌ایم.

سپس، با استفاده از توابع `linspace` در numpy، ضرایب تغییر اندازه را برای عرض و ارتفاع محاسبه می‌کنیم. این ضرایب به صورت یک آرایه توزیع شده از ۱ تا ۰.۴ هستند و نشان دهنده میزان تغییر اندازه قالب نسبت به اندازه اولیه است.

سپس، با استفاده از دو حلقه `for` به ترتیب بر روی ضرایب تغییر اندازه عرض و ارتفاع، عملیات تغییر اندازه و چرخش را انجام می‌دهیم. در هر مرحله، قالب را با استفاده از تابع `cv2.resize` تغییر اندازه می‌دهیم و سپس با استفاده از تابع `cv2.warpAffine` آن را به اندازه تغییر یافته چرخانده و در لیست `list_candid_temps` قرار می‌دهیم.

قبل از افزودن قالب چرخانده به لیست `list_candid_temps`، از تابع `cv2.erode` برای اعمال عملیات فرسایش (erosion) بر روی قالب تغییر یافته استفاده می‌کنیم. این عملیات می‌تواند به کاهش نویز و تقویت حواشی قالب کمک کند. سپس، قالب چرخانده و فرسایش یافته را به لیست `list_candid_temps` اضافه می‌کنیم.

در نهایت، لیست `list_candid_temps` که حاوی قالب‌های جدید است را برمی‌گردانیم.

```
    def extract_segments(image):
```

```
def fa_extract_segments(image):
```

در ارایه به صورت کامل توضیح داده خواهد شد.

```
def template_matching(image, matched_image, filtered_rectangles, templates):
    # crop rectangles and compare with candidates
    scores = {(x, y, w, h): {template: 0 for template in templates} for (x, y, w, h) in
filtered_rectangles}
    for (x, y, w, h) in filtered_rectangles:
        crop_img = image[y:y+h, x:x+w]
        for template in templates:
            max_score = 0
            for candid_template in templates[template]:
                # make size equal
                candid_template = cv2.resize(candid_template, (crop_img.shape[1],
crop_img.shape[0]))
                # compare in CCOEFF_NORMED
                res = cv2.matchTemplate(crop_img, candid_template,
cv2.TM_CCOEFF_NORMED)
                min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
                if max_val > max_score:
                    max_score = max_val
            scores[(x, y, w, h)][template] = max_score

    # find max score
    max_scores = {}
    for (x, y, w, h) in filtered_rectangles:
        # set key of max_scores
        number = -1
        max_score = 0
        for template in templates:
            if scores[(x, y, w, h)][template] > max_score:
                max_score = scores[(x, y, w, h)][template]
                number = template
        max_scores[(x, y, w, h)] = number if max_score > 0.1 else None

    # sort by x
    filtered_rectangles = sorted(filtered_rectangles, key=lambda x: x[0])
```

```
    data = ""
    # draw number
    for (x, y, w, h) in filtered_rectangles:
        char = max_scores.get((x, y, w, h))
        if char is None:
            continue
        cv2.putText(matched_image, max_scores[(x, y, w, h)], (x, y),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv2.LINE_AA)
        if char == 'slash.png':
            char = '/.png'
        data += char.split('.')[0]

    return data
```

به علت اینکه ارائه داده میشه دیگه بقیه کدها رو اونجا توضیح میدیم :))