პიქსელთა ნაკადის ამოცნობა და მისი შედარებითი ანალიზი ჩაშენებულ ბაზასთან და ცალკეულ ინდექსებთან

დალი მოდრეკელიძე, თენგიზ ბახტაძე
საქართველოს ტექნიკური უნივერსიტეტი

რეზიუმე

მანქანური დასწავლა არის თანამედროვე და მარტივი, მაგრამ შრომატევადი ტექნოლოგია, რომელიც დღითიდღე იზრდება და ვრცელდება მსოფლიო. აღსანიშნავი კი ის ფაქტია, რომ არსებობს მრავალი ბაზა, რომლების გამოყენება და მოხმარება შესაძლებელია პირადი მიზნებისთვის. სტატიაში განხილულია ჩვენს მიერ შემუშავებული მარტივი ალგორითმები, რომელთა საშუალებითაც ადვილად ხდება ობიექტების ამოცნობა და შედარებულია ის პიქსელურ ამოცნობასთან. მათი მეშვეობით უფრო იოლდება ამ სფეროში მოღვაწეობა, რადგან შესაძლებელია არსებული პროგრამული უზრუნველყოფის განვითარება და შესაბამისი ფუნქციონალის დამატება, რათა უფრო ღირებული და გამოყენებითი გახდეს ჩვეულებრივი თუ გამოცდილი მომხმარებლისთვის.

საკვანძო სიტყვები: მანქანური დასწავლა. ობიექტების ამოცნობა. პროგრამული უზრუნველყოფა. ხელოვნური ინტელექტი.

1. შესავალი

სულ უფრო და უფრო აქტუალური ხდება მანქანური სწავლების სფერო, რაც გავლენას ახდენს ჩვენს ყოველდღიურ ცხოვრებაზე. თანამედროვე ტექნოლოგიები უკვე იძლევა ხელოვნური ინტელექტის შექმნის საშუალებას. ხელოვნური ნეირონული ქსელების მეშვეობით არაერთი მანქანა გარდაიქმნა მოაზროვნე მოწყობილობად. ახლა უკვე შესაძლებელია საკუთარი რობოტის ყოლა, რომელიც დაგვეხმარება სხვადასხვა ამოცანის მაქსიმალური ეფექტურობით შესრულებაში, რაც მათი მრავალფეროვნების ზრდასთან ერთად დღითიდღე ამატებს მათ სახეობებს.

ყველა ნაკადში მდებარე ობიექტის ვიზუალიზაცია ეყრდნობა KITTI გერმანული ორგანიზაციის საერთაშორისო სტანდარტებს, რომელიც დაფუძნდა გერმანიის კარლსრუეს ტექნოლოგიურ უნივერსიტეტში 2011 წლის ზაფხულში [1]. იგი მოიცავს მრავალი მეცნიერის ნაშრომთა კრებულებს, როგორებიცაა კალიფორნიის ტექნოლოგიური უნივერსიტეტის მონაცემებისა და ავსტრიის გრაცის ტექნოლოგიური უნივერსიტეტის მონაცემების გაერთიანება [2,3]. მათი გადმოწერა, გამოყენება და გავრცელება ნებადართულია და საჯაროა ყველასთვის. იგი სავსებით საკმარისია ნორმალური ზომის პროექტის დასაწყებად. სტატიაში განხილული ჩვენს მიერ შემუშავებული ალგორითმები კი წარმოადგენს ზემოთ ხსენებული ალგორითმების განვითარებულ ვარიანტს, რომლებიც კონკრეტულ სიტუაციაში მეტად სწრაფად მუშაობს, ვინაიდან გამოყენებულია თანამედროვე მიდგომები, რისი საშუალებითაც მეტად ხარისხიანად ამუშავებს მონაცემებს.

2. ძირითადი ნაწილი

მთავარი დავალებაა განვასხვავოთ ობიექტი არაობიექტისგან ამ ბიბლიოთეკების გამოყენებით მოვიყვანოთ სურათების დახარისხების წყობის მაგალითი:

```
def extract_features(imgs, color_type='RGB', main_size=(64, 64),
                        hist_bins=64, orient=9,
                        cell_pixel=8, cell_block=2, hog_channel=0,
                        spatial_feat=True, hist_feat=True, hog_feat=True):
    for file in imgs:
        file_features = []
        image = mpimg.imread(file)
        if color_type != 'RGB':
            if color_type == 'HSV':
                pixel_object = cv2.cvtColor(image, cv2.COLOR_RGB2HSV)
            elif color_type == 'LUV':
                pixel_object = cv2.cvtColor(image, cv2.COLOR_RGB2LUV)
            elif color_type == 'HLS':
                pixel_object = cv2.cvtColor(image, cv2.COLOR_RGB2HLS)
            elif color_type == 'YUV':
                pixel_object = cv2.cvtColor(image, cv2.COLOR_RGB2YUV)
            elif color_type == 'YCrCb':
                pixel_object = cv2.cvtColor(image, cv2.COLOR_RGB2YCrCb)
        else: pixel_object = np.copy(image)

        if spatial_feat == True:
            spatial_features = bin_spatial(pixel_object, size=main_size)
            file_features.append(spatial_features)
        if hist_feat == True:
            hist_features = color_hist(pixel_object, nbins=hist_bins)
            file_features.append(hist_features)
        if hog_feat == True:
            if hog_channel == 'ALL':
                hog_features = []
                for channel in range(pixel_object.shape[2]):

hog_features.append(get_hog_features(pixel_object[:,:,channel],
                                        orient, cell_pixel, cell_block,
                                        vis=False, feature_vec=True))
            hog_features = np.ravel(hog_features)
         else:
            hog_features = get_hog_features(pixel_object[:,:,hog_channel], orient,
                            cell_pixel, cell_block, vis=False, feature_vec=True)

            file_features.append(hog_features)
        features.append(np.concatenate(file_features))
    return features

def get_hog_features(img, orient, cell_pixel, cell_block,
                        vis=False, feature_vec=True):
        if vis == True:
        features, hog_image = hog(img, orientations=orient,
```

```
                                    pixels_per_cell=(cell_pixel, cell_pixel),
                                    cells_per_block=(cell_block, cell_block),
                                    transform_sqrt=False,
                                    visualise=vis, feature_vector=feature_vec)
        return features, hog_image
    else:
        features = hog(img, orientations=orient,
                       pixels_per_cell=(cell_pixel, cell_pixel),
                       cells_per_block=(cell_block, cell_block),
                       transform_sqrt=False,
                       visualise=vis, feature_vector=feature_vec)
        return features

def bin_spatial(img, size=(64, 64)):
    color1 = cv2.resize(img[:,:,0], size).ravel()
    color2 = cv2.resize(img[:,:,1], size).ravel()
    color3 = cv2.resize(img[:,:,2], size).ravel()
    return np.hstack((color1, color2, color3))

def color_hist(img, nbins=64):
    channel1_hist = np.histogram(img[:,:,0], bins=nbins)
    channel2_hist = np.histogram(img[:,:,1], bins=nbins)
    channel3_hist = np.histogram(img[:,:,2], bins=nbins)
    hist_features = np.concatenate((channel1_hist[0], channel2_hist[0],
channel3_hist[0]))
    return hist_features
```

ხოლო მონაცემების ამკითხვისთვის და დადარებისთვის მოცემული გვაქვს კოდის შემდეგი ფრაგმენტი:

```
images = glob.glob('img.png')
obj1 = []
notobj1 = []
all_obj1 = []
all_notobj1 = []

for image in images:
    if 'nonvehicle' in image:
        all_notobj1.append(image)
    else:
        all_obj1.append(image)
for ix, notcar in enumerate(all_notobj1):
    if ix % 5 == 0:
        notobj1.append(notcar)

for ix,objcet1in enumerate(all_obj1):
    if ix % 5 == 0:
        obj1.append(car)

car_image = mpimg.imread(obj1[5])
notcar_image = mpimg.imread(notobj1[0])
```

```
def compare_images(image1, image2, image1_exp="Image 1", image2_exp="Image 2"):
    f, (ax1, ax2) = plt.subplots(1, 2, figsize=(6, 3))
    f.tight_layout()
    ax1.imshow(image1)
    ax1.set_title(image1_exp, fontsize=20)
    ax2.imshow(image2)
    ax2.set_title(image2_exp, fontsize=20)
    plt.subplots_adjust(left=0., right=1, top=0.9, bottom=0.)

compare_images(car_image, notcar_image, "Car", "Not Car")
```

შედეგი: პროგრამას შეუძლია განასხვავოს ობიექტი არა ობიექტისგან.

იმისათვის, რომ დავრწმუნდეთ, შესაბამისი ობიექტია თუ არა, უნდა შევისწავლოთ მისი მახასიათებლები, რაც მოცემულია კოდის შემდეგ ფრაგმენტზე:

```
color_space = 'YUV'
orient = 7  pix_per_cell = 8
block_cell = 2
hog_channel = "ALL"
spatial_size = (64, 64)
hist_bins = 64    spatial_feat = True
hist_feat = True hog_feat = True
converted_car_image = cv2.cvtColor(car_image, cv2.COLOR_RGB2YUV)
car_ch1 = converted_car_image[:,:,0]
car_ch2 = converted_car_image[:,:,1]
car_ch3 = converted_car_image[:,:,2]

converted_notcar_image = cv2.cvtColor(notcar_image, cv2.COLOR_RGB2YUV)
notcar_ch1 = converted_notcar_image[:,:,0]
notcar_ch2 = converted_notcar_image[:,:,1]
notcar_ch3 = converted_notcar_image[:,:,2]

car_hog_feature, car_hog_image = get_hog_features(car_ch1,
                                        orient, pix_per_cell, block_cell,
                                        vis=True, feature_vec=True)

notcar_hog_feature, notcar_hog_image = get_hog_features(notcar_ch1,
                                        orient, pix_per_cell, block_cell,
                                        vis=True, feature_vec=True)

car_ch1_features = cv2.resize(car_ch1, spatial_size)
car_ch2_features = cv2.resize(car_ch2, spatial_size)
car_ch3_features = cv2.resize(car_ch3, spatial_size)
notcar_ch1_features = cv2.resize(notcar_ch1, spatial_size)
notcar_ch2_features = cv2.resize(notcar_ch2, spatial_size)
notcar_ch3_features = cv2.resize(notcar_ch3, spatial_size)

def show_images(image1, image2, image3, image4,  image1_exp="Image 1",
image2_exp="Image 2", image3_exp="Image 3", image4_exp="Image 4"):
    f, (ax1, ax2, ax3, ax4) = plt.subplots(1, 4, figsize=(24, 9))
    f.tight_layout()
```

```
    ax1.imshow(image1)
    ax1.set_title(image1_exp, fontsize=20)
    ax2.imshow(image2)
    ax2.set_title(image2_exp, fontsize=20)
    ax3.imshow(image3)
    ax3.set_title(image3_exp, fontsize=20)
    ax4.imshow(image4)
    ax4.set_title(image4_exp, fontsize=20)
    plt.subplots_adjust(left=0., right=1, top=0.9, bottom=0.)

show_images(car_ch1, car_hog_image, notcar_ch1, notcar_hog_image, "Car ch 1", "Car
ch 1 HOG", "Not Car ch 1", "Not Car ch 1 HOG")
show_images(car_ch1, car_ch1_features, notcar_ch1, notcar_ch1_features, "Car ch
1", "Car ch 1 features", "Not Car ch 1", "Not Car ch 1 features")
show_images(car_ch2, car_ch2_features, notcar_ch2, notcar_ch2_features, "Car ch
2", "Car ch 2 features", "Not Car ch 2", "Not Car ch 2 features")
show_images(car_ch3, car_ch3_features, notcar_ch3, notcar_ch3_features, "Car ch
3", "Car ch 3 features", "Not Car ch 3", "Not Car ch 3 features")
```

ანუ ვიღებთ ობიექტს და ვშლით მას შემადგენელ ნაწილებად – ვატარებთ სხვადასხვა ფილტრის ქვეშ, ვაკვირდებით და გამოგვაქვს დასკვნა. ამაში გვეხმარება სურათების დამუშავების ჩაშენებული ზომის შეცვლის ფუნქცია – ვიღებთ ობიექტს და ვასუფთავებთ მას ზედმეტი ელემენტებისგან, ანუ ვაძლევთ სიმკვეთრეს, რათა შედარება მოხდეს მაქსიმალური სიზუსტით.

როცა მონაცემები გამზადებული და სრულყოფილია, შეგვიძლია დავიწყოთ ჩვენი ხელოვნური ინტელექტის დატრენინგება. მოვიყვანოთ ამ პროცესის კოდის ბლოკის მაგალითი:

```
car_features = extract_features(obj1, color_space=color_space,
                        spatial_size=spatial_size, hist_bins=hist_bins,
                        orient=orient, pix_per_cell=pix_per_cell,
                        block_cell=block_cell,
                        hog_channel=hog_channel, spatial_feat=spatial_feat,
                        hist_feat=hist_feat, hog_feat=hog_feat)
notcar_features = extract_features(notobj1, color_space=color_space,
                        spatial_size=spatial_size, hist_bins=hist_bins,
                        orient=orient, pix_per_cell=pix_per_cell,
                        block_cell=block_cell,
                        hog_channel=hog_channel, spatial_feat=spatial_feat,
                        hist_feat=hist_feat, hog_feat=hog_feat)

X = np.vstack((car_features, notcar_features)).astype(np.float64)
X_scaler = StandardScaler().fit(X)
scaled_X = X_scaler.transform(X)

y = np.hstack((np.ones(len(car_features)), np.zeros(len(notcar_features))))

rand_state = np.random.randint(0, 100)
X_train, X_test, y_train, y_test = train_test_split(
```

```
    scaled_X, y, test_size=0.2, random_state=rand_state)

print('Using:',orient,'orientations',pix_per_cell,
    'pixels per cell and', block_cell,'cells per block')
print('Feature vector length:', len(X_train[0]))
svc = LinearSVC()
t=time.time()
svc.fit(X_train, y_train)
t2 = time.time()
print(round(t2-t, 2), 'Seconds to train SVC...')
print('Test Accuracy of SVC = ', round(svc.score(X_test, y_test), 4))
t=time.time()
```

ალგორითმის ძირითადი პრინციპი მდგომარეობს შემდეგში – შემომავალი ობიექტი სათითაოდ დარდება მონაცემთა ბაზას და დგინდება თითოეულთან მსგავსების კოეფიციენტი. ბოლოს ითვლება ამ ყველა კოეფიციენტის საშუალო არითმეტიკული, რაც არის ამოცანის საბოლოო პასუხი – პროცენტული შეფარდებითი იდენტურობა საძიებო ობიექტის მიმართ.

ცოტათი რთულდება ამოცანა, როცა ობიექტის ამოცნობა გვინდა არა სურათიდან, არამედ ვიდეოდან. მაგრამ არც ამ პრობლემის გადაწყვეტაა რთული. ვიდეო არის სურათების ნაკადი. აქ საჭიროა მოძრავი ობიექტის ამოცნობა, რაშიც გვეხმარება შემდეგი კოდი:

```
from scipy.ndimage.measurements import label

def add_heat(heatmap, bbox_list):
    for box in bbox_list:
        heatmap[box[0][1]:box[1][1], box[0][0]:box[1][0]] += 1
    return heatmap
def apply_threshold(heatmap, threshold):
    heatmap[heatmap <= threshold] = 0
        return heatmap
def draw_labeled_bboxes(img, labels):
        for car_number in range(1, labels[1]+1):
        nonzero = (labels[0] == car_number).nonzero()

        nonzeroy = np.array(nonzero[0])
        nonzerox = np.array(nonzero[1])
        bbox = ((np.min(nonzerox), np.min(nonzeroy)), (np.max(nonzerox),
np.max(nonzeroy)))
                cv2.rectangle(img, bbox[0], bbox[1], (0,0,255), 6)
    return img

heat = np.zeros_like(output_image[:,:,0]).astype(np.float)
heat = add_heat(heat, bboxes)

threshold = 1
heat = apply_threshold(heat, threshold)
heatmap = np.clip(heat, 0, 255)
```

```
labels = label(heatmap)
draw_img = draw_labeled_bboxes(np.copy(image), labels)

def show_images(image1, image2,  image1_exp="Image 1", image2_exp="Image 2"):
    f, (ax1, ax2) = plt.subplots(1, 2, figsize=(24, 9))
    f.tight_layout()
    ax1.imshow(image1)
    ax1.set_title(image1_exp, fontsize=20)
    ax2.imshow(image2, cmap='hot')
    ax2.set_title(image2_exp, fontsize=20)
    plt.subplots_adjust(left=0., right=1, top=0.9, bottom=0.)
show_images(output_image, heatmap, "Car Positions", "Result")
```

რეალურ რეჟიმში მისი შედეგი გამოიყურება შემდეგნაირად:

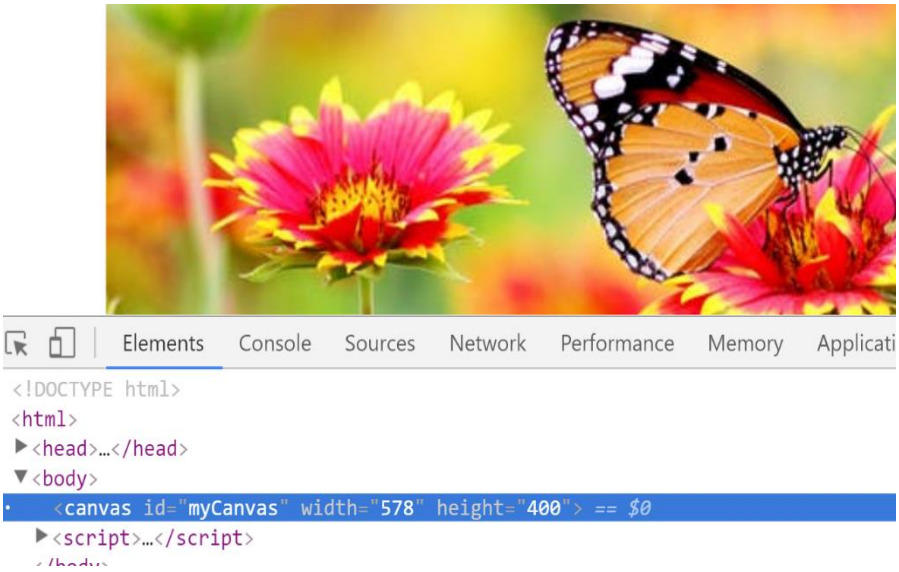
ვიყენებთ სითბურ ეფექტს, რაც აადვილებს ამოცანის გადაწყვეტას. მისი მეშვეობით შესაძლებელია ნებისმიერი ვიდეო ნაკადიდან მოძრავი ობიექტების დაჭერა [4].

შემდეგი, რაც უნდა გავაკეთოთ, არის ყოველი შემხვედრი ფრაგმენტის ამოღება და ჩვენ მონაცემთა ბაზასთან დადარება.

ჩვენი ალგორითმით ობიექტის ამოცნობის სიზუსტე უფრო დიდია ვიდრე სტანდარტული ალგორითმებით, რომლებიც გასაგებადაც უფრო რთულია და უფრო ნელაც მუშაობს.

მეორე ვარიანტში ობიექტების შედარება ხდება კონკრეტულ პიქსელებზე დაყრდნობით და შესაბამისად ობიექტების ამოცნობა ხდება ფერების კომბინაციის ტრაექტორიის საშუალებით.

1-ელი სურათიდან ფერის ამოღება ხდება შემდეგნაირად: თავდაპირველად ვიგებთ სურათის კიდეების x და y კოორდინატებს ზომის დასადგენად. ვიწყებთ უკიდურესი x[i] პიქსელის ამოღებას და შედარებას სხვა პიქსელთან, არსებულს ვინახავთ და პიქსელ-პიქსელ ვხატავთ ჩანაცვლებულ ობიექტს.



ნახ.1

```
function repainin() {
        var el2 = document.getElementById('viewport2');
        var el = document.getElementById('viewport'),
            context = el.getContext('2d'),
            context2 = el2.getContext('2d'),
            width = el.width,
            height = el.height,
            imageData2,
            imageData,
            pixels,i=0,
                x,y;
```

სურათიდან ამოღებული პიქსელი საბოლოოდ გადადის string ფორმატში. სისწრაფისთვის ვახდენთ 4 პიქსელის ერთდროულ ამოღებას და ჩანაცვლებას. გამომდინარე იქიდან, რომ თითოეული ფერი წარმოადგენს წითლის, მწვანის და ლურჯის კომბინაციას, სურათიდან ფერის ამოღება ხდება შემდეგნაირად:

```
for (i = 0; i < pi.length; i+=4) {
        pi[i] = 255 - pi[i];
    // ფერთა სპექტრის პირველი ინდექსი
    pi[i + 1] = 255 - pi[i + 1];
    // ფერთა სპექტრის მეორე  ინდექსი
    pi[i + 2] = 255 - pi[i + 2];
   }ფერთა სპექტრის მესამე ინდექსი
            context.putImageData(imageData, 0, 0);
        }
```

3. დასკვნა

საბოლოო ჯამში ობიექტების ამოცნობა პიქსელურ დონეზე დადის, მაგრამ ვინაიდან და რადგანაც ყოველთვის არაა საჭირო მისი პიქსელური გარდაქმნა, ამოცანიდან გამომდინარე შეგვიძლია ჩავანაცვლოთ ის უკვე არსებული ობიექტებით, რაც შედარების დონეზე საქმეს გვიმარტივებს. მაგრამ კონკრეტულ სიტუაციებში, როდესაც ბაზაში არსებული მონაცემები არ არის საკმარისი, მაშინ საჭირო ხდება ობიექტის ამოცნობა პიქსელურ დონეზე და ახლა უკვე საჭიროა სურათიდან ტრაექტორიის განსაზღვრა და შესაბამისი შეტყობინების გამოტანა. აღსანიშნავია, რომ დროის ფაქტორი მნიშვნელოვან როლს ასრულებს მონაცემების დამუშავებისას, განხილულ მეთოდებში კი დამუშავების სისწრაფე გაუმჯობესებულია. ვფიქრობთ, რომ სტატიაში გამოყენებული ჩვენს მიერ შემუშავებული ალგორითმები გაამარტივებს ობიექტების ამოცნობას და მეტად მოქნილს გახდის შესაბამის პროგრამას კონკრეტული ამოცანების გადაწყვეტისას.

ლიტერატურა – References – Литература:

1. http://www.gti.ssr.upm.es/data/Vehicle_database.html
2. Fergus R., Perona P., Zisserman A. (2003). In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Madison, Wisconsin, 16–22 June-2003.

3. https://matplotlib.org/users/image_tutorial.html

4. Arróspide J., Salgado L., Nieto M. (2012). Video analysis based vehicle detection and tracking using an MCMC sampling framework. EURASIP Journal on Advances in Signal Processing, vol. 2012, Article ID 2012:2, Jan. (doi: 10.1186/1687-6180-2012-2).

IDENTIFY THE FLOW OF PIXELS AND ITS COMPARATIVE ANALYSIS WITH THE DATABASE AND INDIVIDUAL INDEXES

Modrekelidze Dali, Bakhtadze Tengiz
Georgian Technical University

Summary

Machine learning is a modern and simple, but compromised technology that grows and spreads around the world. It is noteworthy that there are many databases that can be used and implemented for personal purposes. The article discusses simple algorithms that can easily identify objects and compare them with the recognition of pixels. Because of this, they are more likely to work in this area, as they can develop existing software and add related features to make it more valuable and useful to ordinary or experienced users.

РАСПОЗНАВАНИЕ ПОТОКА ПИКСЕЛЕЙ И ЕГО СРАВНИТЕЛЬНЫЙ АНАЛИЗ СО ВСТРОЕННОЙ БАЗОЙ И ИНДИВИДУАЛЬНЫМИ ИНДЕКСАМИ

Модрекелидзе Д., Бахтадзе Т.
Грузинский Технический Университет

Резюме

Машинное обучение - это современная и простая, но скомпрометированная технология, которая растет и распространяется по всему миру. Примечательно, что существует множество баз данных, которые можно использовать и применять в личных целях. В статье рассматриваются простые алгоритмы, которые могут легко идентифицировать объекты и сравнить их с распознаванием пикселей. Благодаря этому они с большей вероятностью будут работать в этой области, поскольку они могут разрабатывать существующее программное обеспечение и добавлять соответствующие функции, чтобы сделать его более ценным и полезным для обычных или опытных пользователей.