Turinys

[Turinys 1](#_Toc483523860)

[1. Bag of features 2](#_Toc483523861)

[2. Histogram of oriented gradients 2](#_Toc483523862)

[3. Tyrimas 2](#_Toc483523863)

[3.1 60-40 2](#_Toc483523864)

[3.2 50-50 3](#_Toc483523865)

[3.3 70-30 4](#_Toc483523866)

[3.4 70-30 6](#_Toc483523867)

[3.5 60-40 6](#_Toc483523868)

[3.6 50-50 6](#_Toc483523869)

[4. Kodas 7](#_Toc483523870)

[5. Išvados 10](#_Toc483523871)

# Bag of features

Šio algoritmo esmė paveiksliukuose rasti kažkokius pasikartojančius bruožus ir paskui juos naudojant identifikuoti ar tai yra ieškomas objektas naujose nuotraukose. Algoritmas visų pirma suranda bruožus, tada juos struktūrizuoja sukurdamas tarsi žodyną. Kiekvienai kategorijai ieškoma stipriausių t.y. dažniausiai pasikartojančių bruožų, tik jie ir imami tolesnei nuotraukų analizei. Kai algoritmas apmokomas, jam paduodamos naujos nuotraukos ir jis pagal atrinktus bruožus nusprendžia, kokiai kategorijai priklauso tas paveiksliukas.

# Histogram of oriented gradients

Histogram of oriented gradients algoritmas yra naudojamas vaizdų atpažinimui kompiuteryje tam, kad būtų aptikti nrimi objektai.

Algoritmas skaičiuoja orientuotų gradientų palitimą lokalizuotoje vaizdo dalyje. Algoritmas skaičiuoja kaip kreipiasi vaizde esantys objektai

pasirinktu rėmo dydžiu, taip piešdamas vaizdinį gali pamatyti jo konturus. Vaizdas būna sudalintas į pikselius.

Pirmame žingsnyje algoritmas susivienodina paveiksliukų dydžius bei spalvas, dažniausiai pasiverčia į juodai baltus. Dažniausiai naudojamas gradientų skaičiavimas 1-D

Gradientai gali būti [-1 0 1] taip pat tokia pat matrica tik transformuota.

Antrame žingsnyje yra kuriama histograma. Kiekvienas pikselis turi savo svorį ir gali kreiptis 180 laipsniu kampu. Kad pamatyti pasikeitimus vaizdas suskaidomas į blokus, kurie gali būti 2x2 4x4 8x8 ar didesnio dydžio kuo mažesnis blokas, tuo didesnis tikslumas pasiekiamas.

# Tyrimas

## 60-40

Testavimui naudojau bendrą raidžių katalogą. Maždaug 60% kiekvienos kategorijos raidžių buvo naudotos algoritmo apmokymui, o likę 40% analizei. Testavimo duomenų rinkinys: 259 raidžių nuotraukos. Iš kurių 155 naudotos apmokyti algoritmą, kitos 104 jį testuoti,. Su šiais duomenimis gaunami tokie rezultatai:

Testavimas su apmokymo rinkiniu:

PREDICTED

KNOWN | ą č ė ę į š ū ų ž

-------------------------------------------------------------------------

ą | 0.67 0.17 0.00 0.17 0.00 0.00 0.00 0.00 0.00

č | 0.08 0.75 0.00 0.17 0.00 0.00 0.00 0.00 0.00

ė | 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00

ę | 0.08 0.08 0.00 0.83 0.00 0.00 0.00 0.00 0.00

į | 0.00 0.00 0.00 0.00 0.94 0.00 0.00 0.00 0.06

š | 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00

ū | 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00

ų | 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00

ž | 0.00 0.00 0.00 0.00 0.14 0.00 0.00 0.00 0.86

\* Average Accuracy is 0.89.

Vidutinis tikslumas gautas 89%.

Testavimas su naujų raidžių rinkiniu:

PREDICTED

KNOWN | ą č ė ę į š ū ų ž

-------------------------------------------------------------------------

ą | 0.50 0.13 0.00 0.38 0.00 0.00 0.00 0.00 0.00

č | 0.25 0.13 0.00 0.63 0.00 0.00 0.00 0.00 0.00

ė | 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00

ę | 0.38 0.38 0.00 0.25 0.00 0.00 0.00 0.00 0.00

į | 0.00 0.00 0.00 0.00 0.82 0.00 0.00 0.00 0.18

š | 0.00 0.00 0.00 0.00 0.00 0.80 0.00 0.00 0.20

ū | 0.00 0.00 0.00 0.00 0.00 0.00 0.93 0.07 0.00

ų | 0.00 0.00 0.09 0.00 0.00 0.00 0.09 0.82 0.00

ž | 0.00 0.00 0.00 0.00 0.13 0.00 0.00 0.00 0.87

\* Average Accuracy is 0.68.

## 50-50

Rezultatai su duomenų pasiskirstymu: 50% - apmokymui, 50% - analizei. Tai būtų 131 nuotraukos apmokymui, 128 analizei.

Testavimas su apmokymo rinkiniu:

PREDICTED

KNOWN | ą č ė ę į š ū ų ž

-------------------------------------------------------------------------

ą | 0.90 0.00 0.00 0.10 0.00 0.00 0.00 0.00 0.00

č | 0.00 0.90 0.00 0.10 0.00 0.00 0.00 0.00 0.00

ė | 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00

ę | 0.10 0.10 0.00 0.80 0.00 0.00 0.00 0.00 0.00

į | 0.00 0.00 0.00 0.00 0.93 0.00 0.00 0.00 0.07

š | 0.00 0.00 0.00 0.00 0.00 0.94 0.00 0.00 0.06

ū | 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00

ų | 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00

ž | 0.00 0.00 0.00 0.00 0.26 0.05 0.00 0.00 0.68

\* Average Accuracy is 0.91.

Testavimas su naujų raidžių rinkiniu:

PREDICTED

KNOWN | ą č ė ę į š ū ų ž

-------------------------------------------------------------------------

ą | 0.40 0.10 0.00 0.50 0.00 0.00 0.00 0.00 0.00

č | 0.40 0.20 0.00 0.40 0.00 0.00 0.00 0.00 0.00

ė | 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00

ę | 0.40 0.20 0.00 0.40 0.00 0.00 0.00 0.00 0.00

į | 0.00 0.00 0.00 0.00 0.86 0.00 0.00 0.00 0.14

š | 0.00 0.00 0.00 0.00 0.06 0.94 0.00 0.00 0.00

ū | 0.00 0.00 0.06 0.00 0.00 0.00 0.89 0.06 0.00

ų | 0.00 0.00 0.00 0.00 0.00 0.00 0.07 0.93 0.00

ž | 0.00 0.00 0.00 0.00 0.22 0.00 0.00 0.00 0.78

\* Average Accuracy is 0.71.

## 70-30

Rezultatų su duomenų pasiskirstymą 70% - apmokymui, 30% - analizei. Tai būtų 183 nuotraukos apmokymui, 76 analizei.

Testavimas su apmokymo rinkiniu:

KNOWN | ą č ė ę į š ū ų ž

-------------------------------------------------------------------------

ą | 0.71 0.14 0.00 0.14 0.00 0.00 0.00 0.00 0.00

č | 0.07 0.86 0.00 0.07 0.00 0.00 0.00 0.00 0.00

ė | 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00

ę | 0.21 0.14 0.00 0.64 0.00 0.00 0.00 0.00 0.00

į | 0.00 0.00 0.00 0.00 0.95 0.00 0.00 0.00 0.05

š | 0.00 0.00 0.00 0.00 0.00 0.96 0.00 0.00 0.04

ū | 0.00 0.00 0.00 0.00 0.00 0.00 0.96 0.04 0.00

ų | 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00

ž | 0.00 0.00 0.00 0.00 0.15 0.00 0.00 0.00 0.85

\* Average Accuracy is 0.88.

Testavimas su naujų raidžių rinkiniu:

PREDICTED

KNOWN | ą č ė ę į š ū ų ž

-------------------------------------------------------------------------

ą | 0.67 0.17 0.00 0.17 0.00 0.00 0.00 0.00 0.00

č | 0.33 0.00 0.00 0.67 0.00 0.00 0.00 0.00 0.00

ė | 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00

ę | 0.17 0.50 0.00 0.33 0.00 0.00 0.00 0.00 0.00

į | 0.00 0.00 0.00 0.00 0.75 0.00 0.00 0.00 0.25

š | 0.00 0.00 0.00 0.00 0.00 0.90 0.00 0.00 0.10

ū | 0.00 0.00 0.00 0.00 0.00 0.00 0.91 0.09 0.00

ų | 0.00 0.00 0.13 0.00 0.00 0.00 0.00 0.88 0.00

ž | 0.00 0.00 0.00 0.00 0.18 0.00 0.00 0.00 0.82

\* Average Accuracy is 0.69.

Histogram of oriented gradients algoritmas

Testavimui naudojau bendrą raidžių katalogą. Maždaug 60% kiekvienos kategorijos raidžių buvo naudotos algoritmo apmokymui, o likę 40% analizei. Testavimo duomenų rinkinys: 259 raidžių nuotraukos. Iš kurių 155 naudotos apmokyti algoritmą, kitos 104 jį testuoti,. Su šiais duomenimis gaunami tokie rezultatai:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |

## 70-30

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Column1** | **ą** | **č** | **ė** | **ę** | **į** | **š** | **ū** | **ų** | **ž** |
| 'ą' | 50 | 33,3333 | 0 | 16,66667 | 0 | 0 | 0 | 0 | 0 |
| 'č' | 33,3333 | 16,66667 | 0 | 33,33333 | 16,66667 | 0 | 0 | 0 | 0 |
| 'ė' | 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 10 |
| 'ę' | 50 | 33,33333 | 0 | 16,66667 | 0 | 0 | 0 | 0 | 0 |
| 'į' | 0 | 0 | 0 | 0 | 87,5 | 0 | 0 | 0 | 12,5 |
| 'š' | 0 | 0 | 10 | 0 | 20 | 70 | 0 | 0 | 0 |
| 'ū' | 0 | 0 | 0 | 0 | 0 | 0 | 100 | 0 | 0 |
| 'ų' | 0 | 0 | 12,5 | 0 | 0 | 12,5 | 37,5 | 37,5 | 0 |
| 'ž' | 0 | 0 | 0 | 0 | 18,18182 | 0 | 0 | 0 | 81,81818 |

## 60-40

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Column1** | **ą** | **č** | **ė** | **ę** | **į** | **š** | **ū** | **ų** | **ž** |
| 'ą' | 37,5 | 12,5 | 0 | 37,5 | 12,5 | 0 | 0 | 0 | 0 |
| 'č' | 25 | 12,5 | 0 | 50 | 12,5 | 0 | 0 | 0 | 0 |
| 'ė' | 0 | 0 | 64,28571 | 0 | 0 | 21,42857 | 0 | 0 | 14,28571 |
| 'ę' | 62,5 | 25 | 0 | 12,5 | 0 | 0 | 0 | 0 | 0 |
| 'į' | 0 | 0 | 0 | 0 | 90,90909 | 0 | 0 | 0 | 9,090909 |
| 'š' | 0 | 0 | 6,666667 | 0 | 26,66667 | 60 | 0 | 0 | 6,666667 |
| 'ū' | 0 | 0 | 0 | 0 | 0 | 0 | 100 | 0 | 0 |
| 'ų' | 0 | 0 | 9,090909 | 0 | 0 | 9,090909 | 27,27273 | 54,54545 | 0 |
| 'ž' | 0 | 0 | 6,666667 | 0 | 13,33333 | 6,666667 | 0 | 0 | 73,33333 |

## 50-50

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Column1** | **ą** | **č** | **ė** | **ę** | **į** | **š** | **ū** | **ų** | **ž** |
| 'ą' | 0 | 20 | 0 | 80 | 0 | 0 | 0 | 0 | 0 |
| 'č' | 30 | 20 | 0 | 30 | 20 | 0 | 0 | 0 | 0 |
| 'ė' | 0 | 0 | 70,5882 | 0 | 5,88235 | 11,7647 | 0 | 0 | 11,7647 |
| 'ę' | 40 | 30 | 0 | 30 | 0 | 0 | 0 | 0 | 0 |
| 'į' | 0 | 0 | 0 | 0 | 92,85714 | 0 | 0 | 0 | 7,142857 |
| 'š' | 0 | 0 | 5,882353 | 0 | 35,29412 | 41,17647 | 0 | 0 | 17,64706 |
| 'ū' | 0 | 0 | 0 | 0 | 0 | 0 | 100 | 0 | 0 |
| 'ų' | 0 | 0 | 7,142857 | 0 | 0 | 7,142857 | 28,57143 | 57,14286 | 0 |
| 'ž' | 0 | 0 | 16,66667 | 0 | 11,11111 | 5,555556 | 0 | 0 | 66,66667 |

# Kodas

|  |
| --- |
| *BoF.m* |
| close all;    rootFolder = fullfile('C:\Users\vaitk\Desktop\IntelektikosProjektas\BoF', 'Raides');  testFolder = fullfile('C:\Users\vaitk\Desktop\IntelektikosProjektas\BoF', 'TestuojamosRaides');  categories = {'ą', 'č', 'ę', 'ė', 'į', 'š', 'ų', 'ū', 'ž'};  imds = imageDatastore(fullfile(rootFolder, categories), 'LabelSource', 'foldernames');  imdts = imageDatastore(fullfile(testFolder, categories), 'LabelSource', 'foldernames');    trainingSet = imds;  validationSet = imdts;  %[trainingSet, validationSet] = splitEachLabel(imds, 0.6);  tsc = countEachLabel(trainingSet)  vsc = countEachLabel(validationSet)    % Tiesiog paziureti ar geroje vietoje iesko raidziu  aNosine = find(validationSet.Labels == 'ą', 1);  cSuVarnele = find(validationSet.Labels == 'č', 1);  eNosine = find(validationSet.Labels == 'ę', 1);  subplot(1,3,1);  imshow(readimage(validationSet,aNosine))  subplot(1,3,2);  imshow(readimage(validationSet,cSuVarnele))  subplot(1,3,3);  imshow(readimage(validationSet,eNosine))      % Apmokome algoritma  bag = bagOfFeatures(trainingSet);    img = readimage(imds, 1);  featureVector = encode(bag, img);  categoryClassifier = trainImageCategoryClassifier(trainingSet, bag);    % Visu pirma isbandome savo apmokyta algoritma su testavimo setu,  % kad paziuretume jog gerai veiki. Tikslumas turi buti pakankamai didelis  confMatrix = evaluate(categoryClassifier, trainingSet);    % Testuojame su naujais duomenimis ir ziurime kokias reiksmes gausim  confMatrix = evaluate(categoryClassifier, validationSet);    % Vidutinis tikslumas  mean(diag(confMatrix));    img = imread(fullfile(rootFolder, 'č', '1.jpg'));  [labelIdx, scores] = predict(categoryClassifier, img);    % Spejama raide  categoryClassifier.Labels(labelIdx) |

|  |
| --- |
| *HOG.m* |
| close all;  tic  % Load training and test data using |imageDatastore|.  syntheticDir = fullfile('C:\Users\Dovydas\Desktop\kurs','TestuojamosRaides2');  handwrittenDir = fullfile('C:\Users\Dovydas\Desktop\kurs','Raides2');    % |imageDatastore| recursively scans the directory tree containing the  % images. Folder names are automatically used as labels for each image.  trainingSet = imageDatastore(handwrittenDir, 'IncludeSubfolders', true, 'LabelSource', 'foldernames');  testSet = imageDatastore(syntheticDir, 'IncludeSubfolders', true, 'LabelSource', 'foldernames');    countEachLabel(trainingSet)  countEachLabel(testSet)      figure;    subplot(2,3,1);  imshow(trainingSet.Files{13});    subplot(2,3,2);  imshow(trainingSet.Files{16});    subplot(2,3,3);  imshow(trainingSet.Files{21});    subplot(2,3,4);  imshow(testSet.Files{5});    subplot(2,3,5);  imshow(testSet.Files{7});    subplot(2,3,6);  imshow(testSet.Files{12});      % Show pre-processing results  exTestImage = readimage(testSet,1);  processedImage = imbinarize(rgb2gray(exTestImage));    figure;    subplot(1,2,1)  imshow(exTestImage)    subplot(1,2,2)  imshow(processedImage)    img = readimage(trainingSet, 26);  processedImage = imbinarize(rgb2gray(img));    % Extract HOG features and HOG visualization  [hog\_2x2, vis2x2] = extractHOGFeatures(processedImage,'CellSize',[2 2]);  [hog\_4x4, vis4x4] = extractHOGFeatures(processedImage,'CellSize',[4 4]);  [hog\_8x8, vis8x8] = extractHOGFeatures(processedImage,'CellSize',[8 8]);    % Show the original image  figure;  subplot(2,3,1:3); imshow(processedImage);    % Visualize the HOG features  subplot(2,3,4);  plot(vis2x2);  title({'CellSize = [2 2]'; ['Feature length = ' num2str(length(hog\_2x2))]});    subplot(2,3,5);  plot(vis4x4);  title({'CellSize = [4 4]'; ['Feature length = ' num2str(length(hog\_4x4))]});    subplot(2,3,6);  plot(vis8x8);  title({'CellSize = [8 8]'; ['Feature length = ' num2str(length(hog\_8x8))]});    cellSize = [2 2];  hogFeatureSize = length(hog\_2x2);    numImages = numel(trainingSet.Files);  trainingFeatures = zeros(numImages, hogFeatureSize, 'single');    for i = 1:numImages  img = readimage(trainingSet, i);    img = rgb2gray(img);    % Apply pre-processing steps  img = imbinarize(img);  img = imresize(img, [64 64]);  trainingFeatures(i, :) = extractHOGFeatures(img, 'CellSize', cellSize);  end    % Get labels for each image.  trainingLabels = trainingSet.Labels;    classifier = fitcecoc(trainingFeatures, trainingLabels);    [testFeatures, testLabels] = helperExtractHOGFeaturesFromImageSet(testSet, hogFeatureSize, cellSize);    % Make class predictions using the test features.  predictedLabels = predict(classifier, testFeatures);    % Tabulate the results using a confusion matrix.  [confMat,order] = confusionmat(testLabels, predictedLabels)  confMat = bsxfun(@rdivide,confMat,sum(confMat,2))\*100;    letters = order  colHeadings = arrayfun(@(x)sprintf('%s',x), order,'UniformOutput',false);  format = repmat(' %-7s',1,11);  header = sprintf(format,'letters |',colHeadings{:});  fprintf('\n%s\n%s\n',header,repmat('-',size(header)));  for idx = 1:numel(letters)  fprintf(' %-7s', [letters(idx) ' |']);  fprintf(' %-9.2f', confMat(idx,:));  fprintf('\n')  end    toc |

# Išvados

Galiu teigti, kad algoritmas veikia, teisingai, bet ne ypač tiksliai. Priklausomai nuo duomenų kiekio pamokymui ir analizei rezultatai per daug nesikeitė, buvo 68-71% tikslumo tarpe. Tokius rezultatus labiausiai įtakoja duomenys ir jų paruošimas, kadangi raštas ranka turi daug skirtingų savybių ir sunku parašyti panašias raides, tai ir tikslumas gavosi pusėtinas. Vis dėlto Bag of Features galima naudoti teksto rašyto ranka atpažinimui. Histogram of oriented gradients algoritmas veikia geriau su didesniu apmokymu duomenų kiekių, buvo kad algoritmas raidę prognozavo 100%, buvo, kad prognozės nepasitvirtino ir raidės kokrečiai prognozuoti nepavyko, tačiau ją palaikė panašiomis