LABORATORIUM PRZETWARZANIA I ANALIZY OBRAZÓW

Data wykonania ćwiczenia:	**06.03.2024**
Rok studiów:	**2023/2024**
Semestr:	**1**
Grupa studencka:	**1**
Grupa laboratoryjna:	**b**

Ćwiczenie nr *2*

Temat: *Analiza tekstur obrazów*

Osoby wykonujące ćwiczenia:

1. Michał Herczyński

Katedra Informatyki i Automatyki

1. Cel ćwiczenia
Celem laboratorium jest zapoznanie się z technikami analizy tekstur obrazów takimi jak funkcje haralickie i filtry Gabora

2. Przebieg ćwiczenia
W celu wypróbowania działania funkcji haralickich i filtra Gabora wybrano trzy obrazy tekstury skał – granitu, bazaltu oraz wapienia.

W celu uzyskania wyników funkcji haralickich utworzono funkcję w języku Python z użyciem bibliotek numpy oraz open-cv

```
# Funkcje haralickie
def calculate_haralick_features(image):
        # Konwersja do obrazu w skali szarości
        gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        # Oblicz macierz współwystępowania szarości (GLCM)
        distances = HARALIC_DISTANCES # Wybrane odległości
        angles = HARALIC_ANGLES # Wybrane kąty
        glcm = graycomatrix(gray_image, distances, angles, symmetric=True, normed=True)
        # Oblicz cechy haralickie: kontrast, korelacja, energia, odchylenie standardowe
        contrast = graycoprops(glcm, 'contrast')
        correlation = graycoprops(glcm, 'correlation')
        energy = graycoprops(glcm, 'energy')
        std_dev = graycoprops(glcm, 'homogeneity')
        return contrast, correlation, energy, std_dev
```

Aby zaaplikować filtr Gabora na wybranym obrazie, utworzono kolejną funkcję o ciele:

```
def apply_gabor_filter(image):
        gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        # Parametry filtru Gabora
        ksize = KSIZE
        theta = THETA
        sigma = SIGMA
        lambd = LAMBDA
        gamma = GAMMA
        gabor_kernel = cv2.getGaborKernel((ksize, ksize), sigma, theta, lambd, gamma)
        # Filtracja obrazu przy użyciu filtru Gabora
        filtered_image = cv2.filter2D(gray_image, cv2.CV_8UC3, gabor_kernel)
        return filtered_image
```

po wywołaniu utworzonych funkcji zwrócone zostają wyniki analizy obrazów:

Analiza tekstur dla bazaltu:

Kontrast: [[292.99140704 233.76568774 280.56379397 509.57958132]
[509.30022727 233.76568774 497.99820707 509.57958132]
[546.38243655 369.39212835 532.43459391 584.57703296]],

Korelacja: [[0.63781053 0.71126315 0.65390668 0.37059689]
[0.36977145 0.71126315 0.38616328 0.37059689]
[0.32348742 0.54362304 0.34393272 0.27784115]],

Energia: [[0.01653492 0.01733657 0.01673586 0.01508827]
[0.01503727 0.01733657 0.01511163 0.01508827]
[0.01492658 0.01587602 0.01492561 0.01475454]],

Odchylenie standardowe: [[0.07457474 0.08373373 0.07697933 0.05617937]
[0.05634068 0.08373373 0.05659777 0.05617937]
[0.05333904 0.06613446 0.0548271 0.05307709]]

Analiza tekstur dla granitu:

Kontrast: [[439.45736181 800.00840888 487.80570352 807.89820964]
[1231.0779798 800.00840888 1337.10015152 807.89820964]
[1924.59111675 1878.83677686 2051.86218274 1890.34759208]],

Korelacja: [[0.9195164 0.85331842 0.91040168 0.85187198]
[0.7749076 0.85331842 0.75423202 0.85187198]
[0.64869908 0.65582221 0.622708 0.6537148]],

Energia: [[0.01173348 0.01027786 0.01141622 0.01031411]
[0.00944448 0.01027786 0.00928194 0.01031411]
[0.00860243 0.00862581 0.00850214 0.00866538]],

Odchylenie standardowe: [[0.09068886 0.06826479 0.08773572 0.06800436]
[0.05509032 0.06826479 0.05384298 0.06800436]
[0.04495129 0.04505439 0.04200242 0.0451251]]

Analiza tekstur dla wapienia:

Kontrast: [[62.34271357 90.40958562 60.72567839 87.30006818]
[113.10128788 90.40958562 115.80775253 87.30006818]
[109.2727665 106.47015611 110.91758883 104.82409958]],

Korelacja: [[0.44407519 0.19284345 0.45805954 0.22061313]
[-0.00948425 0.19284345 -0.0346647 0.22061313]
[0.02437697 0.04747891 0.00675169 0.06215865]],

Energia: [[0.04854719 0.04664363 0.04895542 0.04677093]
[0.04639734 0.04664363 0.04628907 0.04677093]
[0.04607133 0.04604904 0.04605913 0.04608834]],

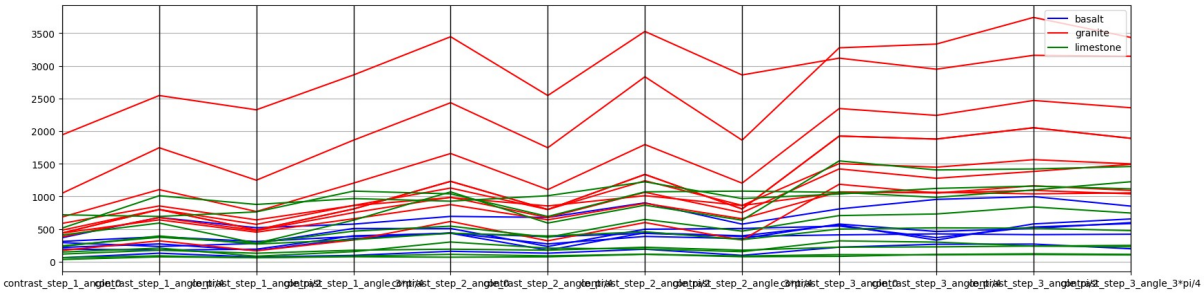
Odchylenie standardowe: [[0.16920695 0.14304264 0.17161328 0.14443412]
[0.13113661 0.14304264 0.13107265 0.14443412]
[0.13610435 0.13423998 0.13558086 0.13472662]]

Po wyniki funkcji haralickich zwracają numeryczne cechy tekstury takie jak kontrast, korelację, energię oraz odchylenie standardowe dla różnych kierunków analizy.

Tego typu zestaw danych może być wykorzystany w celu utworzenia modelu sztucznej inteligencji do klasyfikacji zdjęć wymienionych skał na podstawie tekstury.

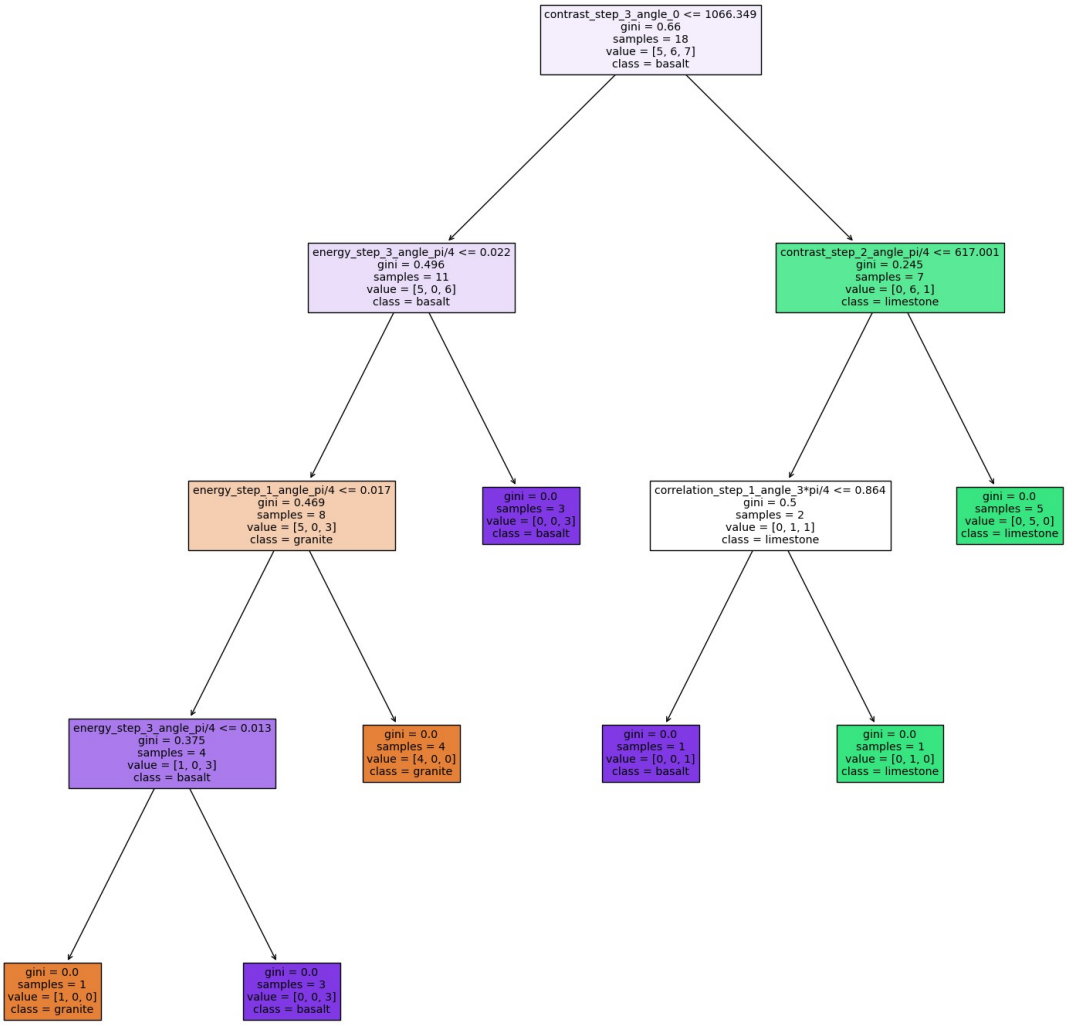
W celach testowych dokonano przekształceń wyników w celu uzyskania tabeli 2d cech objaśniających i klasy. Zwiększono również ilość zdjęć dla każdego z materiałów.

Poniższy wykres wskazuje, że utworzony model będzie w dużym prawdopodobieństwem rozpoznawał granity. Wapienie i bazalty mają podobne cechy teksturalne - drobne uziarnienie i podobna faktura.



Ostatecznie klasyfikacja mogłaby polegać na klasyfikacji uziarnienia skał i kruszyw, lecz w celach edukacyjnych postanowiono pozostać przy wybranym zagadnieniu uzyskania klasyfikacji na podstawie tekstury.

Z uzyskanego zestawu cech, przy użyciu biblioteki scikit-learn drzewo decyzyjne o następującej strukturze:



Uzyskana jakość wg metryki accuracy score wyniósł 85%.

Model testowo został wykorzystany do uzyskania predykcji w procesie który polega na załadowaniu zdjęcia podlegającego klasyfikacji, pozyskaniu cech haralickich oraz podaniu ich do estymatora:

```
image = cv2.imread('images/depositphotos_36517671-stock-photo-basalt-stone-texture.jpg')
# Wybierz obszar obrazu dla analizy tekstur
region_of_interest = image[100:300, 100:300]
# Oblicz funkcje haralickie
contrast, correlation, energy, std_dev = calculate_haralick_features(region_of_interest)

contrast_df=pd.DataFrame([list(contrast.flatten())],columns=create_column_names('contrast'))

correlation_df=pd.DataFrame([list(correlation.flatten())],columns=create_column_names(
'correlation'))
energy_df=pd.DataFrame([list(energy.flatten())],columns=create_column_names('energy'))
std_dev_df=pd.DataFrame([list(std_dev.flatten())],columns=create_column_names('std_dev'))

object = pd.concat([contrast_df,correlation_df,energy_df,std_dev_df], axis=1)

model.predict(object)
```

Następnie do predykcji przekazano dwa obrazy:

array(['granite'], dtype=object) – prawidłowa odpowiedź

array(['limestone'], dtype=object) – odpowiedź jest błędna

Jak można było się spodziewać, predykcje mogą być błędne, szczególnie w przypadku obrazów zawierających niejednoznaczne cechy dla skał których tekstura przełomu jest podobna.

Powyższe zdjęcie bazaltu jest na tyle jasne, że model może pomylić go z wapieniem ze względu na podobieństwo obrazów obu skał.

3. Wnioski

W toku zadań zapoznano się z informacjami jakie można pozyskać przy użyciu funkcji haralickich oraz filtra Gabora.

Filtr Gabora dostarcza użytkownikowi informacji obrazowych na temat tekstury, wykrywając kontury oraz zmiany jasności pikseli na obrazie.

Cechy Haralickie obrazu zostały wykorzystane do implementacji modelu AI, klasyfikującego skały na podstawie tekstury ich przełomu. Uzyskane w ten sposób efekty ocenić można jako poprawne, ale nie wolne od problemów.