Текст программы «Анализ изображения керна нефтегазовой скважины»

Описание:

Данная программа предназначена для анализа изображения керна нефтегазовой скважины и для анализа керна скважины рудного месторождения. Она включает в себя функции для визуализации кластеров, построение графиков статистик, евклидова расстояния и спектральных углов, а также тренда для графиков; получение выборок яркостей пикселей как с использованием скользящего окна, так и для всего изображения в целом.

Импорт необходимых бибилотек:

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import DBSCAN

from scipy.signal import convolve

**Листинг 1. Функция для построения выборок яркостей пикселей изображения с помощью скользящего окна, вычисления статистик и построения графиков статистик.**

Возвращает словарь со списками статистик и выборками яркостей пикселей. (sliding\_window\_sampling). Исходные изображения ориентированы по вертикали.

Параметры:

1) image\_path: Путь к изображению керна (в градациях серого)

2) window\_size: Размер скользящего окна

3) step: Шаг перемещения окна

4) column: Столбец для анализа (по умолчанию центральный (None))

5) plot\_stats: Визуализировать графики статистик

:return:

def sliding\_window\_sampling(image\_path, window\_size, step=1, column=None, plot\_stats=True):

# Чтение изображения и преобразование его в полутоновое

image = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)

if image is None:

raise FileNotFoundError("Не удалось загрузить изображение.")

# Определение параметра column (None – середин по умолчанию)

if column is None:

column = image.shape[1] // 2

pixel\_values = image[:, column]

samples = []

# Объявление массивов статистик

stats = {

'mean': [],

'median': [],

'mode': [],

'std': [],

'range': []

}

# Подсчет статистик

for i in range(0, len(pixel\_values) - window\_size + 1, step):

window = pixel\_values[i:i + window\_size]

samples.append(window)

stats['mean'].append(np.mean(window))

stats['median'].append(np.median(window))

counts = np.bincount(window)

stats['mode'].append(np.argmax(counts))

stats['std'].append(np.std(window))

stats['range'].append(np.max(window) - np.min(window))

# Построение графиков статистик

if plot\_stats:

plt.figure(figsize=(15, 4))

        plt.plot(x, stats['mean'], label='Среднее', color='blue')

        plt.title('Средняя яркость')

        plt.xlabel('Номер окна')

        plt.ylabel('Среднее значение')

        plt.xticks(x\_ticks, x\_labels)

        plt.grid(True)

        plt.tight\_layout()

        plt.show()

        plt.figure(figsize=(15, 4))

        plt.plot(x, stats['mean'], label='Среднее', color='blue', alpha=0.5, linewidth=1)

        plt.plot(x, mean\_trend, label='Тренд', color='red', linewidth=2)

        plt.title('Средняя яркость с трендом')

        plt.xlabel('Номер окна')

        plt.ylabel('Среднее значение')

        plt.xticks(x\_ticks, x\_labels)

        plt.grid(True)

        plt.legend()

        plt.tight\_layout()

        plt.show()

        plt.figure(figsize=(15, 4))

        plt.plot(x, stats['mode'], label='Мода', color='red')

        plt.title('Мода яркости')

        plt.xlabel('Номер окна')

        plt.ylabel('Мода')

        plt.xticks(x\_ticks, x\_labels)

        plt.grid(True)

        plt.tight\_layout()

        plt.show()

        plt.figure(figsize=(15, 4))

        plt.plot(x, stats['std'], label='Стандартное отклонение', color='purple')

        plt.title('Стандартное отклонение яркости')

        plt.xlabel('Номер окна')

        plt.ylabel('Стандарт. отклон.')

        plt.xticks(x\_ticks, x\_labels)

        plt.grid(True)

        plt.tight\_layout()

        plt.show()

return {'samples': samples, 'stats': stats\_dict}

**Листинг 2. Функция выделения тренда на графике.**

**Возвращает усредненное значение для каждого окна усреднения.**

**Параметры:**

**1) data: массив значений (напр. средние яркости)**

**2) window\_size: размер окна усреднения**

**def moving\_average\_trend(data, window\_size):**

**window = np.ones(int(window\_size)) / float(window\_size)**

**return convolve(data, window, mode='same')**

**Листинг 3. Функция построения графиков евклидова расстояния и спектральных углов.**

**Строит графики расстояний от начала координат до векторов признаков и спектральных углов до фиксированного вектора (255, 0, 0).**

**Параметры:**

**stats: словарь с признаками**

**def plot\_distance\_and\_spectral\_angles(stats):**

**X = np.column\_stack([**

**stats['mean'],**

**stats['mode'],**

**stats['std'],**

**])**

**ref = np.array([255, 0, 0])**

**ref = ref / np.linalg.norm(ref)**

**norms = np.linalg.norm(X, axis=1, keepdims=True)**

**X\_normed = X / np.where(norms == 0, 1, norms)**

**dot\_products = np.clip(np.dot(X\_normed, ref), -1.0, 1.0)**

**spectral\_angles = np.arccos(dot\_products)**

**distances = np.linalg.norm(X, axis=1)**

**x = np.arange(len(distances))**

**# График евклидова расстояния**

**fig, ax1 = plt.subplots(figsize=(12, 5))**

**ax1.plot(x, distances, 'r-', label='Расстояние')**

**ax1.set\_ylabel('Евклидово расстояние', color='r')**

**ax1.set\_xlabel('Номер окна')**

**# График спектральных углов**

**ax2 = ax1.twinx()**

**ax2.plot(x, spectral\_angles, 'b--', label='Спектральный угол')**

**ax2.set\_ylabel('Спектральный угол', color='b')**

**plt.title('Расстояние и спектральный угол')**

**fig.tight\_layout()**

**plt.show()**

**Листинг 4. Визуализация результата кластеризации изображения.**

**Визуализирует вертикальные участки кластеров на фоне изображения керна.**

**Параметры:**

**image: полутоновое изображение керна**

**stats: словарь со статистиками**

**labels: список меток кластеров**

**column: номер столбца, по которому получена выборка**

**window\_size: высота одного окна**

**def visualize\_clusters(image, stats, labels, column=None, window\_size=9):**

**height, width = image.shape**

**cluster\_mask = np.zeros((height, 1, 3), dtype=np.uint8)**

**# Палитра для визуализации кластеров**

**colors = [**

**[255, 0, 0],**

**[0, 255, 0],**

**[0, 0, 255],**

**[255, 255, 0],**

**[0, 255, 255]**

**]**

**# Визуализация**

**for i, label in enumerate(labels):**

**start = i \* window\_size**

**end = start + window\_size**

**cluster\_mask[start:end, 0] = colors[label % len(colors)]**

**cluster\_mask\_wide = np.repeat(cluster\_mask, 300, axis=1)**

**plt.figure(figsize=(15, 5))**

**plt.subplot(1, 2, 1)**

**plt.imshow(image, cmap='gray')**

**if column is not None:**

**plt.axvline(column, color='cyan', linestyle='--')**

**plt.title('Изображение керна')**

**plt.subplot(1, 2, 2)**

**plt.imshow(cluster\_mask\_wide)**

**plt.title('Кластеры')**

**plt.tight\_layout()**

**plt.show()**

**Листинг 5. Извлечение статистик для изображений керна скважины рудного месторождения.**

**Функция process\_single\_image обрабатывает одно изображение, извлекая статистики по центральному вертикальному столбцу. Возвращает словарь со статистиками. Вызывается внутри функции process\_all\_images**

**Параметры:**

**image\_path: путь к изображению**

**column: номер столбца (по умолчанию центральный)**

**Функция process\_all\_images обрабатывает все изображения в указанной папке и собирает статистики. Возвращает наименования изображений и собранные статистики.**

**Параметры:**

**folder\_path: путь к папке с изображениями**

**def process\_single\_image(image\_path, column=None):**

**image = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)**

**if image is None:**

**raise FileNotFoundError(f"Не удалось загрузить изображение: {image\_path}")**

**column = image.shape[1] // 2 if column is None else column**

**pixels = image[:, column]**

**return {**

**'mean': np.mean(pixels),**

**'mode': np.argmax(np.bincount(pixels)),**

**'std': np.std(pixels)**

**}**

def process\_all\_images(folder\_path):

    all\_stats = {'mean': [], 'mode': [], 'std': []}

    filenames = []

    for filename in sorted(os.listdir(folder\_path)):

        if filename.lower().endswith(('.png', '.jpg', '.jpeg')):

            try:

                stats = process\_single\_image(os.path.join(folder\_path, filename))

                for key in all\_stats:

                    all\_stats[key].append(stats[key])

                filenames.append(filename)

            except Exception as e:

                print(f"Ошибка при обработке {filename}: {e}")

    return filenames, all\_stats

**Руководство по использованию**

**Импортирование необходимых библиотек:**

Убедитесь, что у вас установлены следующие библиотеки: numpy, matplotlib, scikit-learn, scipy.

1) Использование функции **sliding\_window\_sampling()**

Вызовите функцию sliding\_window\_sampling, указав:

* путь к изображению,
* размер скользящего окна (),
* номер столбца (column) — обычно это середина изображения,
* шаг окна (step),
* plot\_stats=True, если вы хотите автоматически построить графики.

**Пример использования:**

result = sliding\_window\_sampling(

image\_path='core.jpg',

window\_size=15,

step=1,

column=350,

plot\_stats=True

)

2) Использование функции **moving\_average\_trend()**

Вызовите функцию, указав выборку для усреднения (data) и размер окна усреднения (window\_size).

**Пример использования:**

mean\_trend = moving\_average\_trend(result['stats']['mean'], window\_size=25)

plt.plot(mean\_trend)

plt.title('Сглаженный тренд среднего значения')

plt.show()

3) Использование функции **plot\_distance\_and\_spectral\_angles()**

Вызовите функцию, указав словарь с признаками (stats).

**Пример использования:**

plot\_distance\_and\_spectral\_angles(result['stats'])

**4) Использование функции visualize\_clusters()**

Выполните кластеризацию с помощью DBSCAN, сформировав метки для кластеров в переменной labels. Затем вызовите функцию для визуализации кластеров, указав изображение (image), словарь статистик (stats), метки (labels) и номер столбца (column)

**Пример использования:**

image = cv2.imread('core.jpg', cv2.IMREAD\_GRAYSCALE)

labels = dbscan.fit\_predict(X) # где X — матрица признаков

visualize\_clusters(image, result['stats'], labels, column=350, window\_size=15)

5) Использование функции process\_all\_images

Вызовите функцию для обработки изображений керна скважины рудного месторождения для извлечения статистических признаков. Укажите имя папки (folder\_path), где хранятся изображения

**Пример использования:**

filenames, all\_stats = process\_all\_images(folder\_path)