Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Документација за семинарската работа по предметот

Дигитално процесирање на слика

Тема:

Дескриптори за текстура

Изработил:

Стефани Вуксанова, 221218

Содржина:

Вовед

Текстурната анализа е еден од клучните исеци во доменот на компјутерскиот вид, со чија помош може да се издвојат, опишат или класифицираат површинските карактеристики на сликите (шари, повторувања, обрасци, микрорелјеф). За разлика од методите кои се засноваат на детекција на рабови (како Canny или Sobel), текстурните методи се насочени кон суптилните, често репетитивни елементи и грануларни структури во сликата. Овие суптилни обрасци, кога се правилно издвоени, даваат богата информативна основа за понатамошна анализа или класификација.

Зошто е значајна текстурната анализа? Пред сè, во индустриската сфера се користи за навремено препознавање дефекти на производни ленти – како тенки пукнатини, гребаници или недоволна рамномерност. Во медицинските апликации, пак, соодветните текстурни дескриптори можат да помогнат во навремено идентификување на патолошки промени во разни видови медицински слики (томографии, мамографии и сл.). Слично, во археологијата и уметноста, текстурната анализа се користи за да се лоцираат или препознаат специфични материјали, техники на изработка или локации на пронајдена уметност. Дополнително, во дигиталната форензика текстурните карактеристики може да помогнат при верификација на автентичност или откривање потенцијални измени. Конечно, системите за пребарување на слики според содржината (content-based image retrieval) се потпираат на текстурата како една од главните компоненти за да се постигне прецизно издвојување на слични визуелни обрасци.

Овој проект има за цел, преку едноставно корисничко искуство (неколку клика), да овозможи проценка на сличноста на две слики врз основа на нивните текстурни карактеристики. За таа цел, ќе бидат применети четири различни текстурни дескриптори (LBP, GLCM, Gabor, Entropy), при што нивните излезни вредности ќе се споредат за да се добие релевантно растојание помеѓу текстурите. Подоцна, ова растојание се претвора во процентуална мерка која ја одразува нивната сличност. Ваквиот пристап е особено корисен кога бојата и обликот не се клучни за одредено истражување или индустриска апликација, туку фокусот е насочен кон ситните детали и шари во самата слика.

Цели и задачи

1. **Главна цел**:
   - Да се креира проект (демо апликација) кој може автоматски да спореди две слики и да прикаже нумеричка вредност (score) за нивната текстурна сличност.

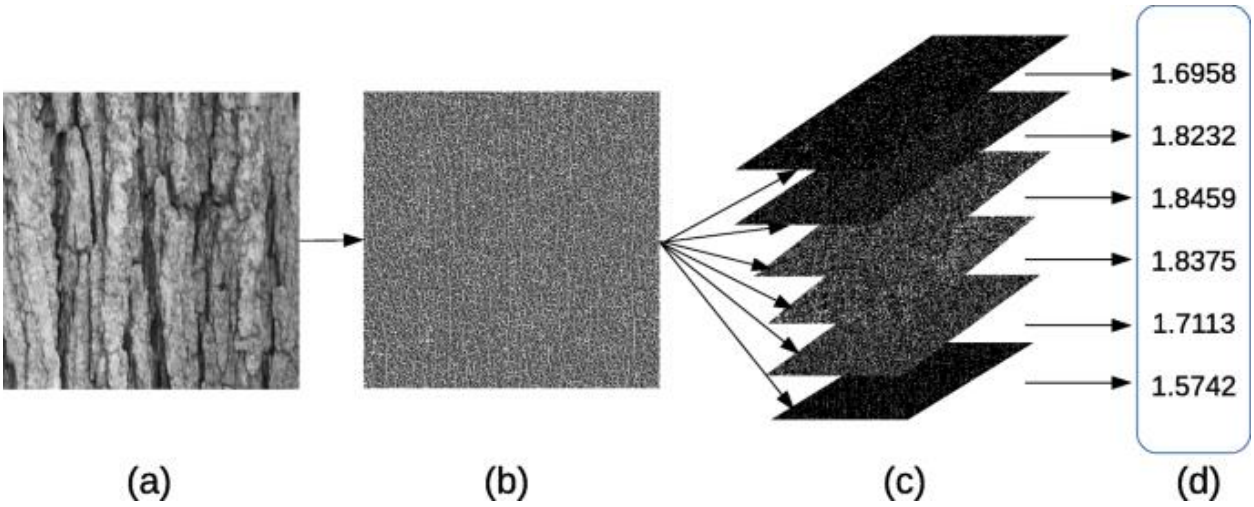
2. **Конкретни задачи**:

   - **Истражување** на LBP, GLCM, Gabor и Entropy како репрезентативни методи за текстурно опишување.
   - **Имплементација** на функциите за екстракција на карактеристики (feature extraction) во Python.

- **Нормализација** на добиените вектори за да се избегнат разлики поради големина или осветленост.
- **Пресметка** на евклидово растојание за секој дескриптор и нивно збирно растојание.
- **Разработка** на едноставна формула (експоненцијална) за претворање на тоа растојание во процент на сличност.
- **GUI интеракција**: Корисникот да може да отвори две слики и да ги види резултатите како pop-up, а исто така резултатите да се снимаат во датотека за подоцнежна анализа.
- **Тестирање** со разни примери на слики (слични/неслични) за да се потврди функционалноста.

Теориска позадина: Текстура и дескриптори

Текстурата претставува **една од клучните карактеристики** во обработката на слика, бидејќи содржи **микроструктурни информации** кои не се очигледни само преку формата или бојата на објектите. За многу апликации (како што се препознавање на материјали, класификација на површини, медицинска дијагностика, форензика, итн.) токму **текстуралните дескриптори** даваат поверодостојни индиции отколку само бојата или рабовите.



Слика 1 . фрактални мерки за анализа на локални карактеристики на слики во препознавање на текстури

Во овој проект употребуваме **четири посебни методи** за да ја опишеме текстурата на една слика:

1. **Локална бинарна шема (LBP)**
2. **GLCM (Gray-Level Co-occurrence Matrix)**
3. **Gabor филтри**
4. **Ентропија (Entropy)**

Секој од овие методи обработува различни аспекти на „шарите“ во сликата и овозможува **количински** (бројчено) да се претстави текстурата.

Слика 1. (различни текстури)

Локална бинарна шема (LBP)

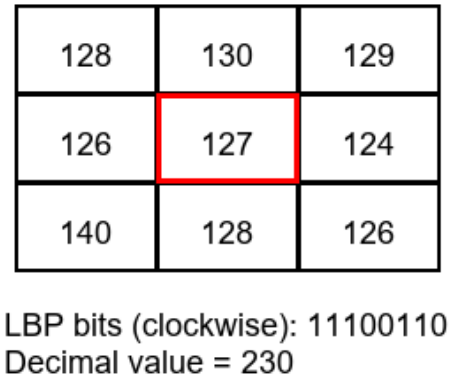
Локалниот бинарен патерн е **едноставен**, но исклучително **моќен** дескриптор за текстура, кој го анализира изгледот на пикселите во рамките на мало соседство околу секој пиксел во сликата. Главната идеја е да се оцени **релативната разлика** помеѓу централниот пиксел и соседните пиксели и да се креира „бинарен потпис“.

1. **Соседство (Neighborhood)**:
   - Обично се користи 3×3 маска (радиус 1) каде 8 пиксели го опкружуваат централниот пиксел.
   - Во понапредни верзии, се применува поголем радиус (пример радиус 2 или 3) и соодветно повеќе соседни точки (16, 24 итн.).
2. **Бинарна шема (Pattern)**:
   - За секој од соседните пиксели, ако вредноста му е **поголема или еднаква** од централниот пиксел, се запишува бита `1`; во спротивно се запишува `0`.
   - Така добиваме **бинарна низа** (на пр. 8-битна, 16-битна), која ја **конвертираме** во децимален број.
   - Пример:

     Ако сите 8 соседи се поголеми од централниот, шемата е `11111111` (binary) = 255 (decimal). Ако сите се помали, добиваме `00000000` = 0.

3. **Хистограм**:
   - Откако ќе го пресметаме LBP за секој пиксел во сликата, се креира **хистограм** на сите добиени вредности.
   - Овој хистограм е **вектор** кој ја репрезентира текстурата на целата слика.
4. **Инвариантност на осветлување**:
   - Затоа што LBP го споредува **локалниот контраст** (сосед vs центар), методот е делумно отпорен на глобални промени во осветлување (пример, ако сликата стане потемна или посветла наеднаш).

- Ова го прави LBP посебно корисен за ситуации каде што интензитетот варира, но структурната (локална) шареност останува.



Слика 3. Демонстрација на LBP во 3×3 соседство

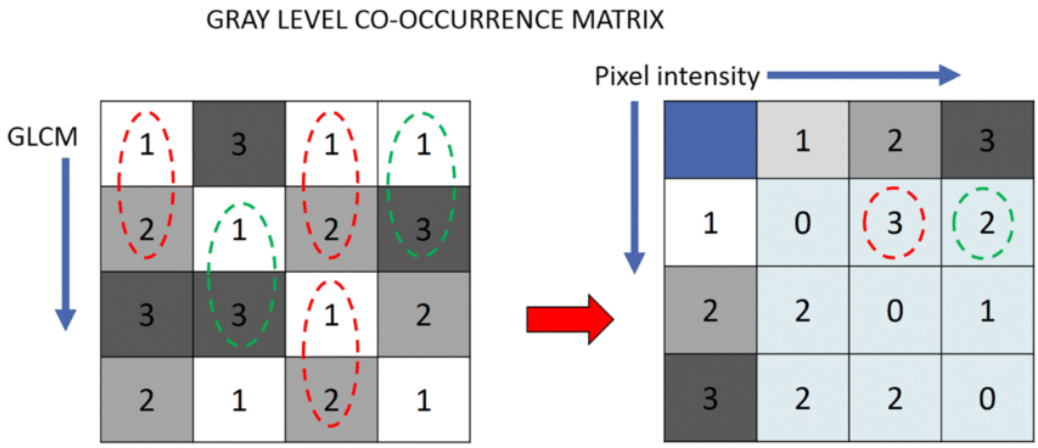
GLCM (Gray-Level Co-occurrence Matrix)

GLCM (Gray-Level Co-occurrence Matrix) е **статистичка** метода којашто ги анализира **паровите на пиксели** и мери колку често тие имаат определени интензитети (i, j) во сликата, на одредено растојание и агол. Идејата е да се разбере **распределбата на тонски релации** во текстурата.

1. **Растојание и агол**:
   - Обично distances=[1] и angles=[0°] значи дека гледаме **хоризонтални** парови пиксели кои се соседни (лево-десно).
   - За поголема робустност, се додаваат и агли 45°, 90°, 135°, па се зема просек од добиените GLCM матрици.
2. **Сместување во матрица**:
   - GLCM е 2D матрица со големина (Level × Level), каде Level е бројот на можни сиви нивоа (на пр. 256 за 8-битна слика).
   - GLCM[i, j] расте секогаш кога во сликата постои пиксел со вредност i којшто е до пиксел со вредност j (според дефинираното растојание/агол).
3. **Haralick мерки**:
   - Од GLCM се вадат неколку **стандардни** статистички карактеристики, меѓу кои:
     - **Contrast** (мерка за интензитетски разлики),
     - **Dissimilarity** (колку се разликуваат пикселите),
     - **Homogeneity** (колку е рамномерна распределбата),
     - **Energy** (колку матрицата е униформна),
     - **Correlation** (поврзаност меѓу парови пиксели).
   - Овие карактеристики понатаму формираат **вектор**, што ја опишува текстурата во „GLCM просторот“.
4. **Значење**:

- Хомогена текстура (пример, рамномерна) → висока Homogeneity, висока Energy, ниска Contrast.
- Хаотична текстура (пример, шумовита или шарена) → висока Contrast/Dissimilarity, пониска Homogeneity.

5. **Примена**:
   - GLCM е многу популарна во медицински слики (препознавање ткива), геонаука (сателитски снимки), и индустриски инспекции, бидејќи дава **„глобална статистика“** за каков тип на тонски транзиции се доминантни



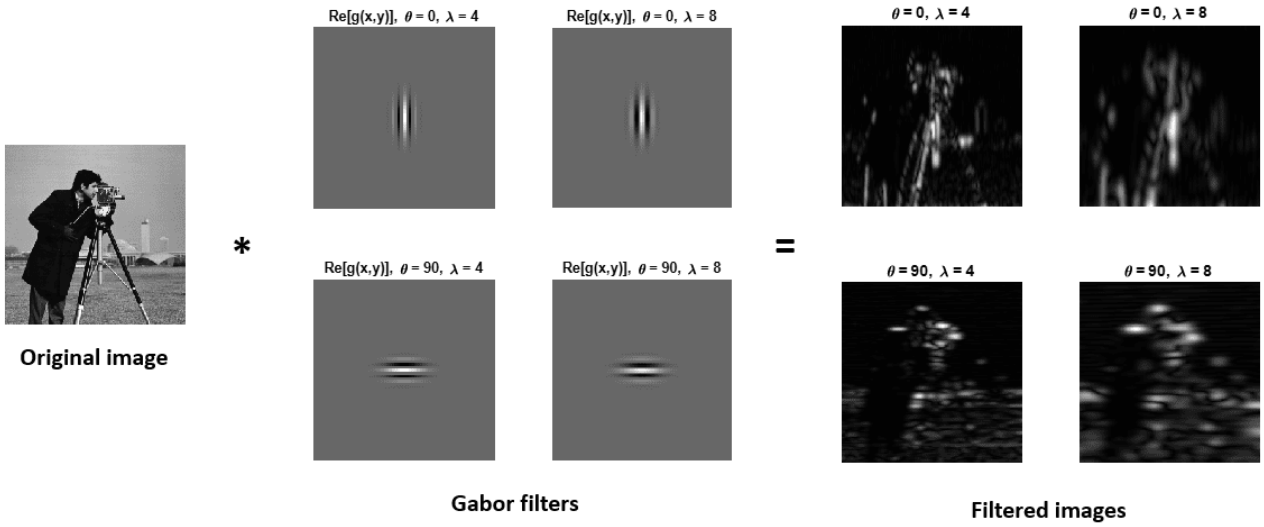
Слика 4.

Gabor филтри

Gabor филтрите **го комбинираат просторниот и фреквенцискиот** пристап за анализа на шари во слика. Замислете синусен бран (кој детектира периодична структура) „спакуван“ во Гаусова обвивка (за локализирање во простор).

1. **Како работат**:
   - Се дефинира Gabor kernel со одредена **фреквенција** и **ориентација** (пример, 0°, 45°, 90°, итн.).
   - Кога ќе се свитка (convolve) со сликата, се добиваат два излеза: **real** и **imag**.
   - Овие излези покажуваат колку „силно“ сликата резонира со таа просторна фреквенција и ориентација.
2. **Features (карактеристики)**:
   - Најчесто се земаат **средина (mean)** и **варијација (variance)** од real и imaginary деловите → вкупно 4 бројки.
   - Ако има потреба, се комбинираат повеќе Gabor филтри (повеќе честоти и агли) за поцеловито покривање на текстурата.
3. **Примери**:
   - **Ткаенина**: Има редовни линии и шари, па Gabor може да ги детектира тие периодични структури.
   - **Фасади, архитектура**: Ако има цигли наредени, Gabor ќе препознае ориентирани линии.

4. **Важност**:
    - Овој метод често се споредува со начинот на кој визуелниот кортекс во човечкото око одговара на линии и ориентации.
    - Корисен е кога бараме **„насочени“** или **„рифли“** (ridges), повторувачки текстури, и слично.



Слика 5.

Ентропија (Entropy)

Ентропијата е мерка што ја користиме за да ја оцениме **неизвесноста** или **хаотичноста** во распределбата на пиксел вредностите во една слика. Оваа мерка потекнува од **теоријата на информации** (Shannon Entropy) и се користи за да се процени колку информации содржи сликата во однос на нејзината текстура.

1. **Дефиниција**:

$$H = -\sum_{i} p_i \log_2(p_i)$$

каде што:

- $(\boldsymbol{p_i})$е **веројатноста** (или фреквенцијата) за појава на пиксел вредност во сликата.
- **Σ (сума)** значи дека ги сумираме сите можни нивоа на сивило (0 до 255 за 8-битна слика).
- $\boldsymbol{\log_2(p_i)}$ ја мери количината на информации што ја носи пиксел вредноста .

Пример: Замислете едноставна слика со 4 пиксели со вредности [100, 100, 150, 150].

Пресметка на веројатностите: Вредност 100 се јавува 2 пати → $p(100) = 2/4 = 0.5$; Вредност 150 се јавува 2 пати → $p(150) = 2/4 = 0.5$

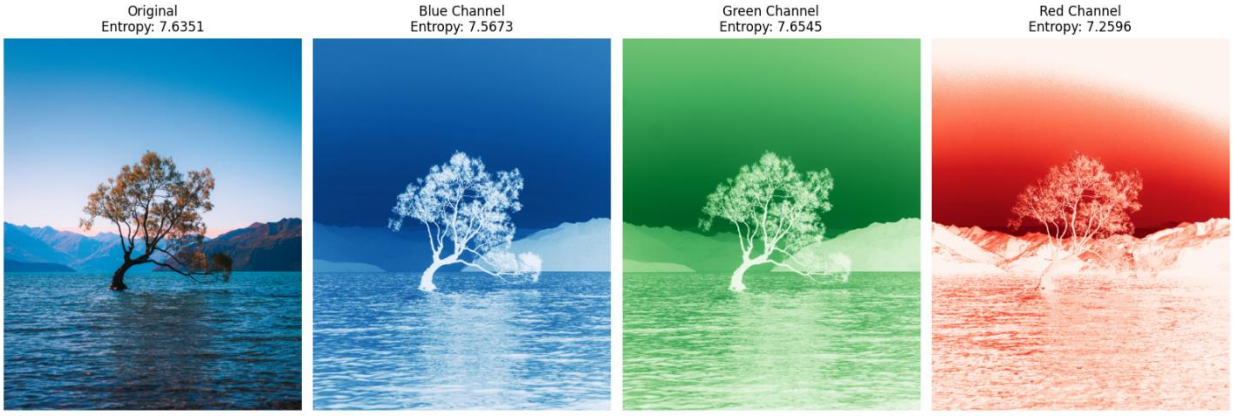
Применување на формулата:

$$H = -[(0.5 \cdot log2(0.5)) + (0.5 \cdot log2(0.5))]$$

$$H = -[(0.5 \cdot -1) + (0.5 \cdot -1)]$$

$$H = -[-0.5 - 0.5] = 1$$

Интерпретација: Ентропијата е 1, што значи дека сликата има умерена комплексност. Ако сите пиксели беа исти, ентропијата ќе беше 0 (нема неизвесност). Ако имавме повеќе различни вредности, ентропијата ќе беше повисока.



Слика 6. Ентропија на слика заедно со нејзините индивидуални RGB канали.

1. **Ниска ентропија:**
   - Ако сликата е **многу еднолична**, на пример, целосно сива слика без детали, сите пиксели имаат **исто ниво на сивило**.
   - Веројатноста за тоа ниво на сивило ќе биде **1**, а за сите други нивоа ќе биде **0**.
   - Во овој случај, ентропијата ќе биде **многу мала или нула**, што значи дека нема многу информации или варијации во сликата.
2. **Висока ентропија:**
   - Ако сликата е **многу сложена**, со многу детали и различни текстури (на пример, шума или сложен мозаик), има **голем број различни нивоа на сивило**.
   - Веројатноста е распоредена помеѓу многу вредности, што создава поголема "неизвесност" за тоа каков ќе биде следниот пиксел.
   - Ентропијата ќе биде **висока**, што укажува на голема разновидност и сложеност во текстурата.

3. **Интуитивен пример**
    - **Едноставна слика (низок H):**
      Замисли бел лист хартија. Сите пиксели се бели. Нема разлика, нема изненадување → ентропијата е многу ниска.

    - **Комплексна слика (висок H):**
      Замисли слика од толпа луѓе на концерт, со различни бои, осветлување и текстури. Секој дел е различен → ентропијата е висока.

4. **Зошто е важна ентропијата?**

Ентропијата е важна бидејќи обезбедува мерка за тоа колку информација или комплексност постои во една слика. Таа игра клучна улога во различни апликации за обработка на слики и анализа на податоци. Со мерење на распределбата на вредностите на пикселите, ентропијата овозможува:

- **Откривање на сложеност:** Високата ентропија означува слики со богати детали и разновидни текстури, додека ниската ентропија укажува на едноставни, еднолични слики.
- **Оптимизација на алгоритми:** Во системите за препознавање слики, ентропијата помага да се идентификуваат региони со повеќе информации кои се клучни за анализа.
- **Квалитет на компресија:** Во компресија на податоци, ентропијата се користи за проценка на тоа колку е можно да се намали големината на сликата без загуба на важни информации.
- **Детекција на аномалии:** Во безбедносни системи, може да се користи за да се откријат необични промени во слики, што може да укажува на потенцијални закани.

Со други зборови, ентропијата не само што ја мери хаотичноста, туку и ја открива суштината на информациите скриени во визуелните податоци.

Детален опис на алгоритмите

Екстракција на LBP карактеристики

- **Конверзија на сликата во grayscale формат:**

  Ова е важно бидејќи LBP алгоритмот работи на сивински вредности на пикселите.

- **Примена на local_binary_pattern:**
  Се користи од библиотеката skimage.feature со параметри како број на точки (n_points), радиус (radius), и метод (често method='uniform').
  Овој метод споредува вредности на пикселите во однос на нивниот центарен пиксел и создава бинарен модел.

- **Генерирање на LBP вредности:**
  Добиената слика (lbp) содржи LBP вредност за секој пиксел што претставува текстурален потпис.
- **Пресметка на хистограм:**
  Хистограмот ги брои сите LBP вредности.
- **Нормализација:**
  Се користи за да се избегнат разлики во големината на сликите.

```
def extract_lbp_features(image, radius=3, n_points=24, method='uniform'):  2
    """Extract Local Binary Pattern (LBP) features from an image."""
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    lbp = local_binary_pattern(gray, n_points, radius, method)
    # Use a basic histogram (no double normalization), then safely normalize
    hist, _ = np.histogram(
        lbp.ravel(),
        bins=np.arange(0, n_points + 3),  # n_points+2 bins, +1 for range
        density=False
    )
    return safe_normalize(hist)
```

Слика 7.

Екстракција на GLCM карактеристики

- **Конверзија во grayscale:**
  Прво, сликата се претвора во grayscale за да се анализираат нивоата на сивило.

- **Генерирање на Gray-Level Co-occurrence Matrix (GLCM):**
  Оваа матрица ги мери врските помеѓу парови на пиксели.
- **Екстракција на карактеристики:**
  Се пресметуваат статистички мерки.
- **Формирање на вектор:**
  Овие вредности се комбинираат во еден вектор за полесна анализа.
- **Нормализација:**
  Со цел да се добие униформност во анализата на различни слики.

```
# For simplicity, just take the first distance/angle combination
contrast = graycoprops(glcm, prop: 'contrast')[0, 0]
dissimilarity = graycoprops(glcm, prop: 'dissimilarity')[0, 0]
homogeneity = graycoprops(glcm, prop: 'homogeneity')[0, 0]
energy = graycoprops(glcm, prop: 'energy')[0, 0]
correlation = graycoprops(glcm, prop: 'correlation')[0, 0]

features = np.array([
    contrast, dissimilarity, homogeneity, energy, correlation
])
return safe_normalize(features)
```

Слика 8.

```
def extract_glcm_features(image, distances=None, angles=None):  2 usages
    """
    Extract Gray-Level Co-occurrence Matrix (GLCM) features from an image.
    Using default distance=[1], angle=[0].
    """
    if distances is None:
        distances = [1]
    if angles is None:
        angles = [0]

    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    glcm = graycomatrix(
        gray,
        distances=distances,
        angles=angles,
        levels=256,
        symmetric=True,
        normed=True
    )
```

Слика 9.

Екстракција на Gabor карактеристики

- **Примена на Gabor филтер:**
  Филтерот се користи за анализа на фреквенциски информации во сликата.

- **Пресметка на статистики:**
- **Нормализација:**
  Овие вредности се нормализираат за конзистентни резултати.

```python
def extract_gabor_features(image, frequency=0.6):  2 usages
    """Extract Gabor filter features from an image."""
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gabor_real, gabor_imag = gabor(gray, frequency=frequency)
    # Create a feature vector and normalize
    features = np.array([
        gabor_real.mean(),
        gabor_real.var(),
        gabor_imag.mean(),
        gabor_imag.var()
    ])
    return safe_normalize(features)
```

Слика 10.

Екстракција на ентропија

- **Конверзија во grayscale:**
  За да се анализира дистрибуцијата на пиксел вредности.
- **Генерирање на хистограм:**
  Се брои појавувањето на секоја сивинска вредност.
- **Пресметка на ентропија**
- **Нормализација:**
  Се става во низа и се нормализира за споредливост со другите карактеристики.

```python
def extract_entropy(image):  2 usages
    """Extract entropy feature from an image."""
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    # Use a simple histogram (no double normalization), then compute entropy
    hist, _ = np.histogram(gray.ravel(), bins=256, density=False)
    entr = entropy(hist)
    return safe_normalize(np.array([entr]))
```

Слика 11.

Пресметка на сличност

По екстракцијата на четирите видови на дескриптори (LBP, GLCM, Gabor, Entropy) за двете слики, следува процесот на пресметка на сличност. Овој процес се базира на мерење на разликите помеѓу векторите што ги претставуваат карактеристиките на двете слики.

- **Пресметка на Евклидово растојание за LBP дескрипторите**
  Евклидовото растојание е метрика што ја мери "должината" на векторот што ги поврзува двата точки (вектори) во просторот.
  За LBP дескрипторите се пресметува како:

$$d_{LBP} = \sqrt{\sum_{i=1}^{n} (LBP_{1i} - LBP_{2i})^2}$$

- **Евклидово растојание за GLCM, Gabor и Entropy**
  Истиот пристап се користи и за останатите дескриптори.
  Секој од овие пресметува колку се разликуваат соодветните карактеристики за двете слики:
  - **GLCM**: Разлика во структурни текстури.
  - **Gabor**: Разлика во фреквенциски и ориентациски текстури.
  - **Entropy**: Разлика во хаотичноста и распределбата на информациите.

- **Вкупно растојание**
  За да добиеме единствена метрика за сличност, сите растојанија се сумираат:

$$D_{total} = d_{LBP} + d_{GLCM} + d_{Gabor} + d_{Entropy}$$

  Колку е поголемо D_total, толку се поголеми разликите меѓу сликите.

- **Пресметка на сличност**
  За да го претвориме вкупното растојание во процентуална мерка на сличност, се користи експоненцијална функција:

$$Similarity = e^{-D_{total}} \times 100\%$$

- **Објаснување:**

  - **Ако D_total е 0** (што значи дека сликите се идентични), тогаш:

$$\exp(0) = 1 \Rightarrow Similarity = 100\%$$

- **Колку повеќе расте D_total**, експоненцијалната функција експоненцијално опаѓа кон 0.
  Ова значи дека многу различни слики ќе имаат сличност блиску до 0%.

Имплементациски детали

- **Јазик:** Python 3 (препорачана верзија 3.7 или понова).
- **Користени библиотеки:**
  - **`opencv-python (cv2)`** – за вчитување и зачувување на слики, како и работа со BGR формат.
  - **`numpy`** – за математички операции со низи и вектори.
  - **`scikit-image`** – за пресметка на LBP, GLCM и Gabor дескриптори.
  - **`scipy.stats`** – за пресметка на ентропија.
  - **`scipy.spatial.distance`** – за пресметка на Евклидови растојанија.
  - **`matplotlib`** – за графички приказ и визуелизација на резултатите.
  - **`tkinter`** – за отворање на дијалози (file chooser) и пораки (message boxes).
- **Логирање:** Секое споредување се запишува во фајлот `texture_comparison_log.txt`, кој содржи:
  - Датум и време на споредбата (на пример: 2025-01-30 14:05:12).
  - Имиња на споредуваните слики (basename, без целосната патека).
  - Вредности на растојанија за LBP, GLCM, Gabor и Entropy.
  - Финален процентуален сличносен скор.

Дополнително, за напредни проекти може да се користи `logging` модулот на Python за покомплексно и структуирано логирање, но за оваа апликација едноставното запишување во текстуален фајл е доволно.

Упатство за користење на демо апликацијата

- **Инсталација на потребни библиотеки:**
  - pip install opencv-python numpy scikit-image matplotlib scipy
- **Покренување на апликацијата:**
  - python main.py

Оваа команда се извршува во терминалот (Command Prompt) во соодветната папка каде што се наоѓа скриптата.

- **Избор на слики:**

- Се отвора прозорец со наслов „Select two images“.
- Изберете точно две слики (поддржани формати: JPG, JPEG, PNG).
- Ако изберете помалку или повеќе слики, ќе добиете порака за грешка.

- **Приказ на резултатите:**

  - Апликацијата ги чита сликите, ги екстрахира дескрипторите и ги пресметува растојанијата.
  - Резултатот (процент на сличност) се прикажува во конзолата и во pop-up прозорец.
  - Се отвора графички прозорец (matplotlib) со двете слики прикажани една до друга и наслов „Texture Similarity: XX.XX%“.
  - Прозорецот автоматски се затвора по околу 10 секунди.
  - Резултатите се запишуваат во texture_comparison_log.txt.

Пример за извршување и резултати

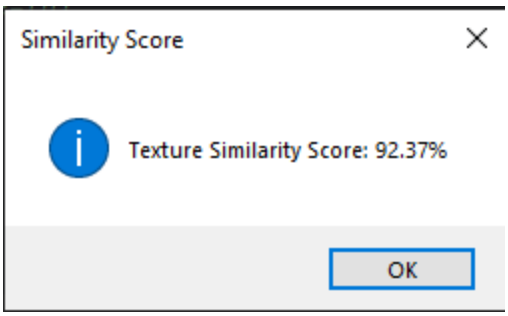
- Како пример, земаме две тест-слики, image1.jpg и image2.jpg.

1. **Старт**:
   a. python main.py
2. **Фајл дијалог**:
   a. Избираме image1.jpg и image2.jpg.
3. **Конзолен излез (пример)**
4. **Pop-up:**

   a. „Texture Similarity Score: 92.37%“.

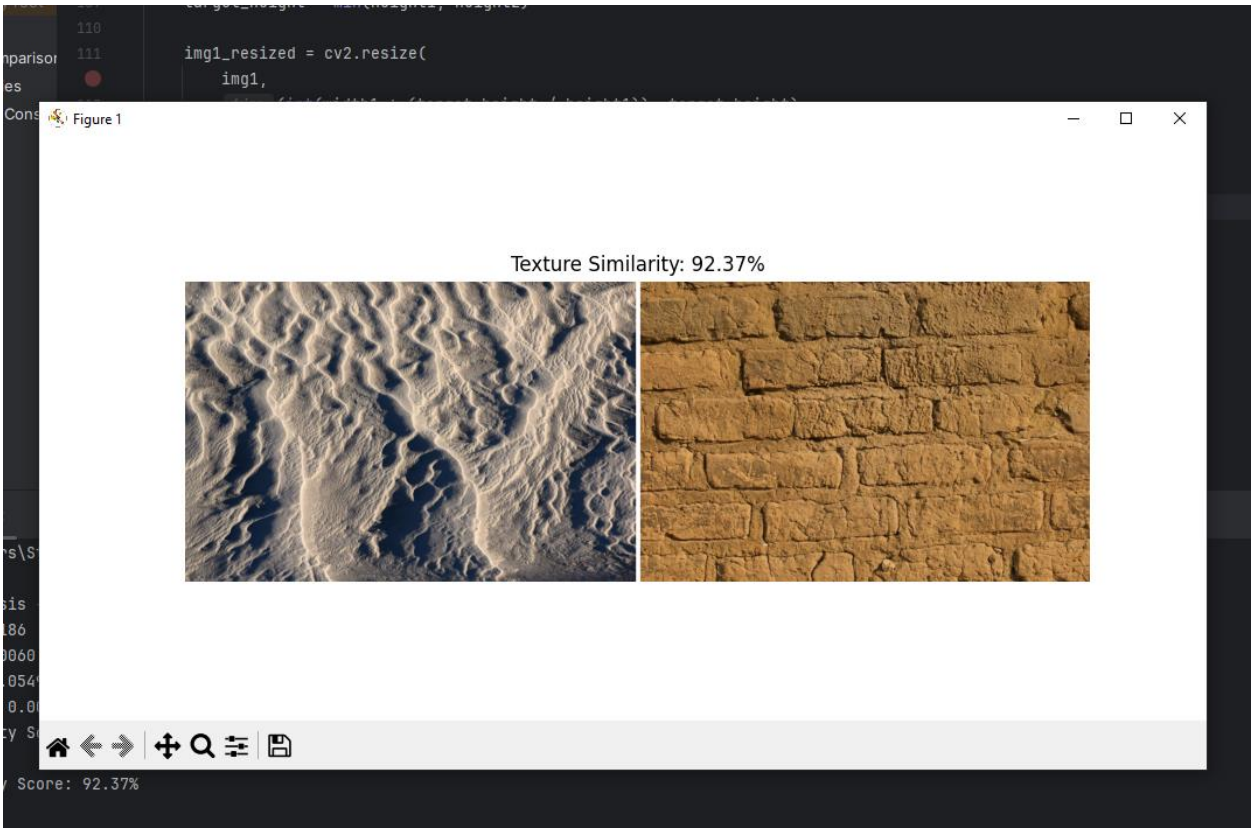
```
--- Texture Analysis ---
LBP Distance: 0.0186
GLCM Distance: 0.0060
Gabor Distance: 0.0549
Entropy Distance: 0.0000
Computed Similarity Score: 92.37%

Texture Similarity Score: 92.37%
```



Слика 12. и 13.

5. **Matplotlib прозорец**:
   a. Двете слики една до друга, наслов: „Texture Similarity: 92.37%“. По 10 сек. се затвора.



Слика 14.

6. **Лог-фајл**

```
--- Log Entry: 2025-02-01 22:49:33 ---
Image 1: img5.jpg
Image 2: img6.jpg
LBP Distance: 0.0186
GLCM Distance: 0.0060
Gabor Distance: 0.0549
Entropy Distance: 0.0000
Computed Similarity Score: 92.37%
```

Слика 15.

Дискусија за резултатите и анализа

- **Интерпретација на резултатите:**
  Самиот сличносен скор (на пример 92.37%) не значи дека двете слики се „идентични“ или „целосно различни“. Тоа претставува приближна мера на текстурната сличност заснована на пресметаните дескриптори.

- **Врска помеѓу растојание и сличност:**
  Кога вкупното растојание е блиску до нула, добиваме скор близу 100%, што укажува на висока сличност помеѓу текстуралните обрасци и распределбата на сивилата во двете слики.
- **Праг за одлука:**
  Во одредени апликации, може да се дефинира праг (на пример 70%) за да се утврди дали две текстури се сметаат за исти. Овој праг зависи од контекстот, како што се медицински слики, природни пејзажи или индустриски материјали.
- **Осетливост на секој дескриптор:**

  - **LBP:** Робустен на промени во осветлувањето и одличен за микротекстури, но може да биде чувствителен на шум.
  - **GLCM:** Обезбедува статистички информации за тонските односи, но може да биде помалку робустен на промени во осветлувањето и контрастот.
  - **Gabor:** Ефикасен за откривање на насочени шари и фреквенции, но бара прилагодување на параметрите за различни текстури.
  - **Ентропија:** Мери степен на неуредност во сликата и нуди корисни дополнителни информации, иако не ја анализира директно структурата.

- **Тестирање со различни типови слики:**

  - Слики со слична текстура (на пр. две слики од иста површина на дрво).
  - Слики со слични бои, но различни шари (на пр. сино небо vs. сина вода).
  - Црно-бели во споредба со колор слики.
  - Слики со ротации, зум и други трансформации за тестирање на ротациска инваријантност.

Заклучок и идна работа

Овој систем претставува основен, но ефикасен пристап за текстурална анализа и споредба на слики. Користењето на четири класични дескриптори (LBP, GLCM, Gabor, Entropy) во комбинација со Евклидово растојание и експоненцијална функција овозможува пресметка на текстурната сличност.

Идна работа и можности за унапредување:

1. **Мултиаголен GLCM:**
   - Наместо мерење на растојанија само на агол од 0°, да се вклучат и 45°, 90°, 135° за подобра ротациска робустност.
2. **Повеќе Gabor конфигурации:**
   - Комбинирање на различни фреквенции (0.2, 0.4, 0.6, 0.8) и агли за пообемна анализа.
3. **Тежинско комбинирање на дескрипторите:**

   - Доделување на различни тежини на секој дескриптор, во зависност од нивната важност за специфичната апликација.
4. **Интеграција на Deep Learning техники:**
   - Користење на конволутивни невронски мрежи (CNN) за екстракција на feature vectors, што може да ја зголеми точноста на споредбата.
5. **Работа со повеќе канали и висока динамичка резолуција:**
   - Поддршка за 16-bit слики, HDR формати и алтернативни колор простори (Lab, HSV) за побогата анализа на текстури.

Овие подобрувања можат значително да го зголемат квалитетот и стабилноста на системот. Сепак, и со моменталните четири дескриптори и едноставниот модел на споредба, системот обезбедува солидна основа за анализа на текстурната сличност помеѓу слики.

Референци

[1] Gonzalez, R. C., & Woods, R. E. (2018). *Digital Image Processing* (4th Edition). Pearson.

[2] Ojala, T., Pietikäinen, M., & Harwood, D. (1994). *Performance Evaluation of Texture Measures with Classification Based on Kullback Discrimination of Distributions*. *Pattern Recognition*, 29(1), 51-59.

[3] Haralick, R. M., Shanmugam, K., & Dinstein, I. (1973). *Textural Features for Image Classification*. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6), 610–621.

[4] Jain, A. K., & Farrokhnia, F. (1991). *Unsupervised Texture Segmentation Using Gabor Filters*. *Pattern Recognition*, 24(12), 1167-1186.

[5] **Scikit-Image Documentation:** https://scikit-image.org/docs/stable/
Covers Python implementations of image processing algorithms, including LBP, GLCM, and Gabor filters.

[6] **OpenCV Documentation:** https://docs.opencv.org/
Official documentation for OpenCV, including image loading, color conversions, and feature extraction techniques.

[7] **SciPy Documentation (Entropy Functions):**
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.entropy.html
Explains how entropy is calculated using Python's SciPy library.

[8] **IEEE Xplore Digital Library:** https://ieeexplore.ieee.org/
A great source for academic papers on image processing, machine learning, and texture analysis.