Techniki Wizyjne i Przetwarzanie Obrazów

Laboratorium 4: Transformata Census

dr inż. Krzysztof Borkowski

Politechnika Świętokrzyska
Wydział Mechatroniki i Budowy Maszyn
Katedra Automatyki i Robotyki

Rok akademicki 2025/2026

Plan laboratorium I

Wprowadzenie teoretyczne

Transformata Census - idea

Definicja

Transformata Census to metoda reprezentacji lokalnych regionów obrazu w postaci ciągów binarnych poprzez porównanie wartości piksela centralnego z jego sąsiadami.

Transformata Census - idea

Definicja

Transformata Census to metoda reprezentacji lokalnych regionów obrazu w postaci ciągów binarnych poprzez porównanie wartości piksela centralnego z jego sąsiadami.

Zastosowania

- **Stereo vision** - dopasowywanie punktów między obrazami
- **Detekcja zmian** - wykrywanie różnic w sekwencjach obrazów
- **Rozpoznawanie tekstur** - analiza struktury powierzchni
- **Tracking obiektów** - śledzenie obiektów w ruchu

Transformata Census - idea

Definicja

Transformata Census to metoda reprezentacji lokalnych regionów obrazu w postaci ciągów binarnych poprzez porównanie wartości piksela centralnego z jego sąsiadami.

Zastosowania

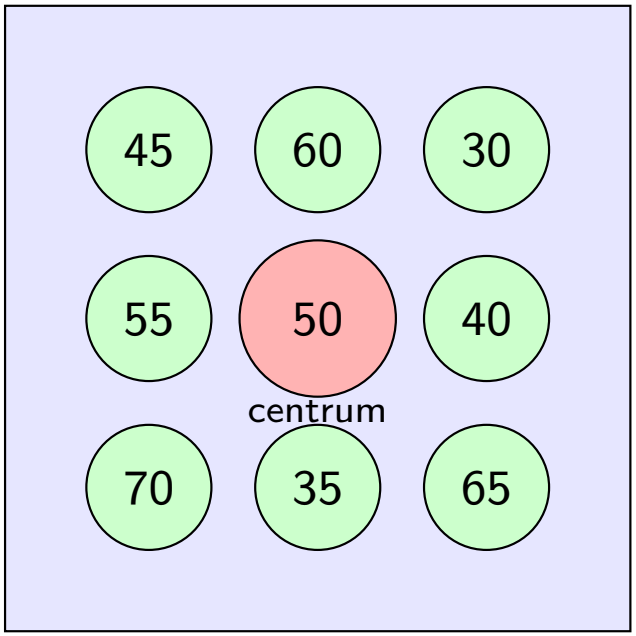
- **Stereo vision** - dopasowywanie punktów między obrazami
- **Detekcja zmian** - wykrywanie różnic w sekwencjach obrazów
- **Rozpoznawanie tekstur** - analiza struktury powierzchni
- **Tracking obiektów** - śledzenie obiektów w ruchu

Kluczowa zaleta

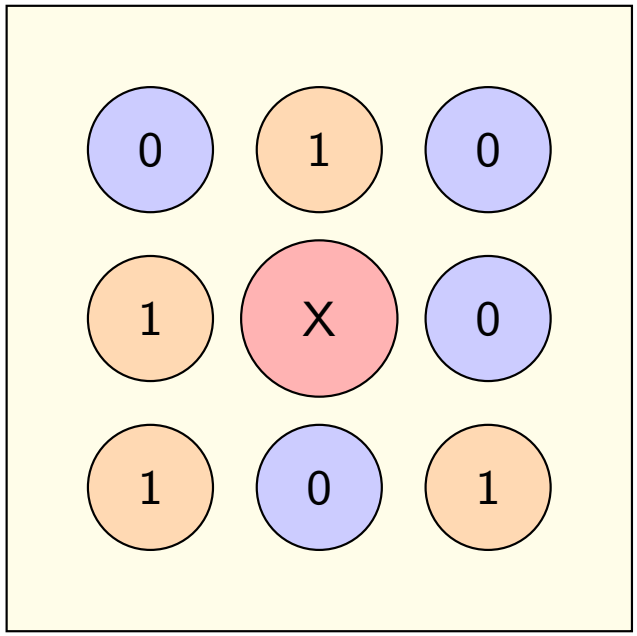
Transformata Census jest **odporna na zmiany oświetlenia**, ponieważ porównuje względne relacje między pikselami, a nie ich bezwzględne wartości.

Zasada działania

Okno 3×3



Porównanie



Reguła:

- Piksel $<$ centrum $\rightarrow$ bit $=$ **0**
- Piksel $\geq$ centrum $\rightarrow$ bit $=$ **1**

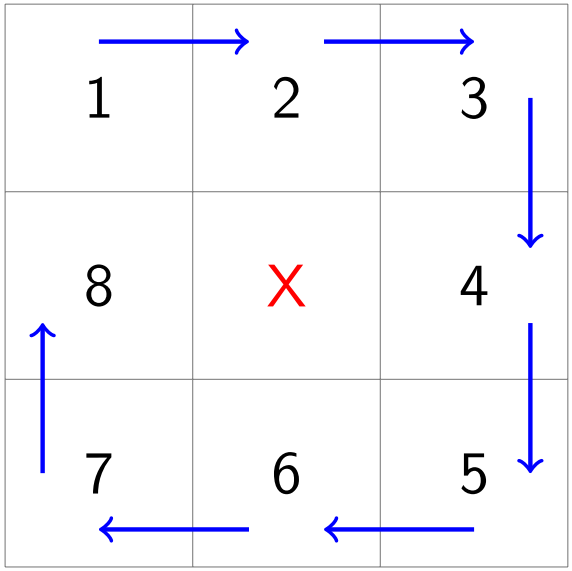
Wynik:

- Ciąg bitów: 01001001
- Wartość dziesiętna: 73

Wprowadzenie teoretyczne

Czym jest transformata Census?

Odczyt



Zalety odczytu spiralnego

- **Niezmienniczość względem obrotów** - łatwiejsza analiza obrazów obróconych
- **Ciągłość przestrzenna** - sąsiednie bity reprezentują sąsiednie piksele
- **Uniwersalność** - działa dla dowolnego rozmiaru okna (3×3, 5×5, 7×7, ...)

Uwaga

Odczyt wierszami jest prostszy w implementacji, ale odczyt „ślimakiem"daje lepsze właściwości dla analizy obrotów obrazu.

Przykład krok po kroku

Obraz wejściowy:

```
45   60   30
55   50   40
70   35   65
```

Centrum: 50

Kolejność odczytu:

1. $45 < 50 \rightarrow 0$
2. $60 \geq 50 \rightarrow 1$
3. $30 < 50 \rightarrow 0$
4. $40 < 50 \rightarrow 0$
5. $65 \geq 50 \rightarrow 1$
6. $70 \geq 50 \rightarrow 1$
7. $55 \geq 50 \rightarrow 1$
8. $35 < 50 \rightarrow 0$

Wynik:

Ciąg bitów: 01001110

Wartość dziesiętna: **78**

Interpretacja

Liczba 78 to **deskryptor** tego fragmentu obrazu. Możemy go porównywać z innymi deskryptorami używając odległości Hamminga.

Odległość Hamminga

Definicja i zastosowanie

Odległość Hamminga

Definicja

Odległość Hamminga to liczba pozycji, na których dwa ciągi binarne o tej samej długości się różnią.

Odległość Hamminga

Definicja

Odległość Hamminga to liczba pozycji, na których dwa ciągi binarne o tej samej długości się różnią.

Przykład

Ciąg 1:	01001001
Ciąg 2:	01101000
Różnice:	2 pozycje (bit 3 i bit 7)

Odległość Hamminga = **2**

Odległość Hamminga

Definicja

Odległość Hamminga to liczba pozycji, na których dwa ciągi binarne o tej samej długości się różnią.

Przykład

Ciąg 1:	01001001
Ciąg 2:	01101000
Różnice:	2 pozycje (bit 3 i bit 7)

Odległość Hamminga = **2**

Zastosowanie w Census

- **Mała odległość** ($\approx$ 0-5) $\rightarrow$ regiony są **podobne**
- **Średnia odległość** ($\approx$ 6-15) $\rightarrow$ regiony **częściowo podobne**
- **Duża odległość** ($>$ 15) $\rightarrow$ regiony są **różne**

Zadania do wykonania

Zadania laboratoryjne

Zadanie 1: Transformata Census dla pojedynczego piksela

Zaimplementuj funkcję `census_pixel(image, x, y, window_size)` obliczającą wartość Census dla jednego piksela.

Zadania laboratoryjne

Zadanie 1: Transformata Census dla pojedynczego piksela

Zaimplementuj funkcję `census_pixel(image, x, y, window_size)` obliczającą wartość Census dla jednego piksela.

Zadanie 2: Transformata Census dla całego obrazu

Zaimplementuj funkcję `census_transform(image, window_size)` obliczającą Census dla wszystkich pikseli obrazu (z pominięciem brzegów).

Zadania laboratoryjne

Zadanie 1: Transformata Census dla pojedynczego piksela

Zaimplementuj funkcję `census_pixel(image, x, y, window_size)` obliczającą wartość Census dla jednego piksela.

Zadanie 2: Transformata Census dla całego obrazu

Zaimplementuj funkcję `census_transform(image, window_size)` obliczającą Census dla wszystkich pikseli obrazu (z pominięciem brzegów).

Zadanie 3: Odległość Hamminga

Zaimplementuj funkcję `hamming_distance(a, b)` obliczającą liczbę różniących się bitów.

Zadania laboratoryjne

Zadanie 1: Transformata Census dla pojedynczego piksela

Zaimplementuj funkcję `census_pixel(image, x, y, window_size)` obliczającą wartość Census dla jednego piksela.

Zadanie 2: Transformata Census dla całego obrazu

Zaimplementuj funkcję `census_transform(image, window_size)` obliczającą Census dla wszystkich pikseli obrazu (z pominięciem brzegów).

Zadanie 3: Odległość Hamminga

Zaimplementuj funkcję `hamming_distance(a, b)` obliczającą liczbę różniących się bitów.

Zadanie 4: Mapa podobieństwa

Zaimplementuj funkcję `similarity_map(image1, image2, window_size)` tworzącą mapę różnic między dwoma obrazami.

Zadania laboratoryjne

Zadanie 1: Transformata Census dla pojedynczego piksela

Zaimplementuj funkcję `census_pixel(image, x, y, window_size)` obliczającą wartość Census dla jednego piksela.

Zadanie 2: Transformata Census dla całego obrazu

Zaimplementuj funkcję `census_transform(image, window_size)` obliczającą Census dla wszystkich pikseli obrazu (z pominięciem brzegów).

Zadanie 3: Odległość Hamminga

Zaimplementuj funkcję `hamming_distance(a, b)` obliczającą liczbę różniących się bitów.

Zadanie 4: Mapa podobieństwa

Zaimplementuj funkcję `similarity_map(image1, image2, window_size)` tworzącą mapę różnic między dwoma obrazami.

Zadanie 5: Wyszukiwanie wzorca

Zaimplementuj funkcję `find_pattern(image, template, window_size, threshold)` wyszukującą wzorzec na obrazie.

Zadania do wykonania

Przegląd zadań

Wskazówki implementacyjne

Zadanie 1: Wskazówki

Struktura funkcji census_pixel

```python
def census_pixel(image, x, y, window_size=3):
    # TODO: Zaimplementuj obliczanie transformaty Census dla pojedynczego piksela
    # 1. Oblicz promień okna: radius = window_size // 2
    # 2. Pobierz wartość piksela centralnego
    # 3. Zainicjuj census_value = 0
    # 4. Dla każdego sąsiada w oknie:
    #    - Przesuń census_value o 1 bit w lewo
    #    - Jeśli sąsiad >= centrum, dodaj 1 (census_value |= 1)
    # 5. Zwróć census_value
    offset = window_size // 2
    center_value = image[y,x]
    census_value = 0

    for dy in range(-offset, offset +1):
        for dx in range(-offset, offset +1):
            if dy == 0 and dx == 0:
                continue

            census_value <<= 1

            if image[y+dy, x+dx] >= center_value:
                census_value |= 1
    return census_value
```

Kluczowe operacje bitowe

- « przesunięcie bitowe w lewo (zrobienie miejsca na nowy bit)

Zadanie 2: Wskazówki

Struktura funkcji census_transform

```
    # TODO: Zaimplementuj transformatę Census dla całego obrazu
    # 1. Utwórz macierz wynikową wypełnioną zerami (dtype=np.uint64)
    # 2. Oblicz promień okna
    # 3. Dla każdego piksela (z pominięciem brzegów):
    #     - Wywołaj census_pixel i zapisz wynik
    # 4. Zwróć macierz wynikową
    rows,cols = image.shape
    census = np.zeros((rows, cols), dtype=np.uint64)
    offset = window_size // 2

    for y in range(offset, rows - offset):
        for x in range(offset, cols - offset):
            census[y, x] = census_pixel(image, x, y, window_size)

    return census
```

Zadanie 3: Wskazówki

```python
def hamming_distance(a, b):
    """
    Oblicza odległość Hamminga między dwoma liczbami.

    Parametry:
    a, b - liczby całkowite (reprezentacje Census)

    Zwraca:
    int - liczba różniących się bitów
    """
    xor_result = a ^ b  # XOR - różnice bitów
    return bin(xor_result).count('1')  # Zlicz jedynki

# Przykład użycia
census1 = 0b01001001  # 73
census2 = 0b01101000  # 104
distance = hamming_distance(census1, census2)
print(f"Odległość Hamminga: {distance}")  # Wynik: 2
```

Listing 1: Prosta implementacja w Pythonie

Kluczowa operacja

Operator `XOR` (`^`) zwraca 1 tam, gdzie bity się różnią, i 0 tam, gdzie są takie same.

Zadanie 4: Mapa podobieństwa

Idea

Dla każdego piksela oblicz odległość Hamminga między wartościami Census z dwóch obrazów. Im większa odległość, tym większa różnica.

```python
def similarity_map(image1, image2, window_size=3):
    # 1. Oblicz Census dla obu obrazów
    census1 = census_transform(image1, window_size)
    census2 = census_transform(image2, window_size)

    # 2. Utwórz mapę różnic
    similarity = np.zeros_like(census1, dtype=np.uint8)

    # 3. Dla każdego piksela oblicz odległość Hamminga
    for y in range(census1.shape[0]):
        for x in range(census1.shape[1]):
            similarity[y, x] = hamming_distance(
                census1[y, x], census2[y, x]
            )

    return similarity
```

Listing 2: Szkielet funkcji

Wizualizacja

Jasne piksele = duże różnice, ciemne piksele = małe różnice

Zadanie 5: Wyszukiwanie wzorca

Algorytm sliding window

1. Oblicz Census dla obrazu i wzorca
2. Przesuń wzorzec po całym obrazie
3. Dla każdej pozycji oblicz średnią odległość Hamminga
4. Zwróć pozycje, gdzie odległość $<$ threshold

```
# Wytnij fragment obrazu o rozmiarze wzorca
region = census_img[y:y+tpl_h, x:x+tpl_w]

# Oblicz średnią odległość Hamminga
total_distance = 0
count = 0
for ty in range(tpl_h):
    for tx in range(tpl_w):
        total_distance += hamming_distance(
            region[ty, tx], census_tpl[ty, tx]
        )
        count += 1

avg_distance = total_distance / count
if avg_distance < threshold:
    matches.append((x, y, avg_distance))
```

Listing 3: Fragment kluczowy

Co dziś osiągniesz?

Wiedza teoretyczna

- Rozumiesz ideę transformaty Census
- Znasz różnicę między odczytem wierszowym a spiralnym
- Wiesz czym jest odległość Hamminga i jak ją stosować
- Rozumiesz zastosowania Census w przetwarzaniu obrazów

Umiejętności praktyczne

- Potrafisz zaimplementować transformatę Census
- Umiesz obliczać odległość Hamminga
- Potrafisz tworzyć mapy podobieństwa między obrazami
- Umiesz wyszukiwać wzorce na obrazach

Kluczowa zaleta Census

Transformata Census jest **odporna na zmiany oświetlenia**, co czyni ją idealną do porównywania obrazów w różnych warunkach.

Pytania i odpowiedzi

Pytania?

Powodzenia w implementacji!

Materiały dostępne na: `https://github.com/kbor89/TWiPO`