

Ćwiczenie 6

ROZPOZNAWANIE OBRAZÓW PRZY UŻYCIU METODY EIGENFACE

Zakres pracy

W ramach ćwiczenia należy do dostarczonego interfejsu, umożliwiającego wyświetlanie obrazów zawartych w plikach graficznych, dodać możliwość rozpoznawania wczytywanych obrazów za pomocą metody *Eigenface*.

Eigenface (Turk & Pentland, 1991) była jedną z pierwszych metod rozpoznawania twarzy. Jest ona w całości oparta na analizie głównych składowych i umożliwia bardzo dużą redukcję informacji, zmniejszając liczbę cech do wartości nieprzekraczającej liczby obrazów uczących m . Podczas realizacji zadania należy wykorzystać **95% informacji** zawartej w zbiorze uczącym.

Stworzony klasyfikator powinien uwzględniać 10 klas obrazów (zdjęcia 10 osób). Rozpoznając zdjęcie program powinien wskazać jedną z osób, podjąć decyzję neutralną (nieznana osoba) lub stwierdzić, że obraz nie przedstawia twarzy. Oprócz tego należy wyświetlić także aproksymację obrazu rozpoznawanego wykonaną przy użyciu wektorów własnych.

Uczenie powinno być realizowane w następujących krokach:

1. zmiana rozdzielczości obrazów (do wymiarów 100×100), przeskalowanie jasności do zakresu $[0;1]$, przekształcenie obrazów w wektory (macierze o jednej kolumnie),
2. utworzenie macierzy kowariancji \mathbf{C} (1) i wyznaczenie średniej ze wszystkich wektorów uczących (ze wszystkich klas),
3. wyznaczenie docelowych unormowanych wektorów własnych \mathbf{u}_i na podstawie wektorów własnych macierzy \mathbf{C} (2) i obliczenie sumy odpowiadających im wartości własnych,
4. przekształcenie wszystkich obrazów uczących wg wzoru (4),
5. wyliczenie wektorów średnich dla poszczególnych klas.

Rozpoznawanie należy przeprowadzić w następujących krokach:

1. zmiana rozdzielczości obrazu, przeskalowanie jasności, przekształcenie w wektor,
2. wyznaczenie \mathbf{w} wg wzoru (4),
3. wyznaczenie \mathbf{y}' wg wzoru (5),
4. wyznaczenie odległości między \mathbf{w} a średnimi klas,
5. wyznaczenie różnicy między \mathbf{y} a \mathbf{y}' .

Jeśli różnica między \mathbf{y} a \mathbf{y}' jest zbyt duża, obraz nie jest twarzą. Jeśli odległość między \mathbf{w} a najbliższą średnią jest zbyt duża, obraz przedstawia nieznaną osobę. W pozostałych przypadkach najbliższa średnia wskazuje rozpoznaną osobę.

Informacje pomocnicze

W metodzie *Eigenface* jako cechy wykorzystuje się bezpośrednio jasności pikseli obrazu, przy czym oryginalną **macierz** (tablicę jasności punktów) \mathbf{Y} przekształca się w **wektor** cech \mathbf{y} poprzez ustawienie wszystkich **kolumn** jedna pod drugą i ich połączenie.

Do utworzonych w ten sposób wektorów cech \mathbf{y}_i stosuje się analizę głównych składowych. Ponieważ najczęściej liczba cech N jest znacznie większa niż liczba przykładów uczących m , zamiast normalnej macierzy kowariancji Σ tworzona jest macierz zmodyfikowana:

$$\mathbf{C} = \Phi^T \Phi \quad (1)$$

gdzie $\Phi = [\phi_1 \ \phi_2 \dots \ \phi_m]$,

$$\phi_i = \mathbf{y}_i - \boldsymbol{\mu},$$

$\boldsymbol{\mu}$ – średnia ze WSZYSTKICH wektorów uczących (ze wszystkich klas).

Następnie wyznaczamy wektory własne \mathbf{v}_i (jest ich m) i odpowiadające im wartości własne λ_i macierzy \mathbf{C} . Wartości własne \mathbf{C} są takie same jak wartości własne właściwej macierzy kowariancji Σ . Natomiast wektory własne macierzy Σ wyznaczamy na podstawie \mathbf{v}_i ze wzoru:

$$\mathbf{u}_i = \sum_{j=0}^m v_{ij} (\mathbf{y}_j - \boldsymbol{\mu}) \quad (2)$$

gdzie v_{ij} oznacza j -ty element wektora \mathbf{v}_i .

Wektory własne \mathbf{u}_i należy unormować:

$$\mathbf{u}_i \leftarrow \frac{\mathbf{u}_i}{\|\mathbf{u}_i\|}$$

oraz poustawiać jako kolumny macierzy \mathbf{U} :

$$\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_K] \quad (3)$$

Wartość $K < m$ (liczbę wykorzystywanych wektorów własnych) wyznaczamy na podstawie wartości własnych λ_i . Jeżeli chcemy wykorzystać 95% informacji, musimy użyć tylu wektorów własnych, aby $\sum_{k=1}^K \lambda_k \approx 0,95 * \sum_{i=1}^m \lambda_i$

Każdy wektor wejściowy \mathbf{y} przekształcamy w następujący sposób:

$$\mathbf{w} = \mathbf{U}^T (\mathbf{y} - \boldsymbol{\mu}) \quad (4)$$

Ostatecznie z wektorów \mathbf{w} tworzymy klasyfikator minimalnoodległościowy jednomodalny z metryką Euklidesa.

Obrazy rozpoznawane również należy zamienić na wektory, a następnie przekształcić wg wzoru (4).

Aby umożliwić podejmowanie decyzji neutralnej, należy dla każdej klasy wyznaczyć próg odległości (np. na podstawie największej w danej klasie odległości między obrazem uczącym a modą).

Aproksymację \mathbf{y}' obrazu \mathbf{y} utworzoną przy użyciu wektorów własnych \mathbf{u}_i wyznaczamy ze wzoru:

$$\mathbf{y}' = \sum_{i=0}^K w_i \mathbf{u}_i \quad (5)$$

Wartość różnicy między \mathbf{y}' a \mathbf{y} służy do stwierdzenia, czy obraz w ogóle przedstawia twarz.

Wskazówki implementacyjne

OBRAZY NALEŻY ZMNIĘSZYĆ DO ROZDZIELCZOŚCI **100x100** PIKSELI. Jasność obrazu powinna być sprowadzona do przedziału **[0,1]**.

Zamiana macierzy **A** na macierz **B** w formacie CV_64F:

```
A.convertTo(B, CV_64F);
```

Przeskalowanie macierzy **MatSrc** do rozmiarów (*szerokość* × *wysokość*) i umieszczenie wyniku w macierzy **MatDst**, której typ jest taki sam jak **MatSrc**:

```
resize(MatSrc, MatDst, cvSize(szerokosc, wysokosc));
```

Zamiana macierzy **Y** na wektor kolumnowy **y** stanowiący połączenie WIERSZY macierzy **Y**:

```
y = Y.reshape(1, Y.rows*Y.cols);
```

Następująca funkcja wyliczy zmodyfikowaną macierz kowariancji **C** o rozmiarach $m \times m$ (wzór 1) i wektor średni **mi** na podstawie macierzy **A**, która zawiera dane uczące w kolejnych kolumnach:

```
calcCovarMatrix(A, C, mi, CV_COVAR_SCRAMBLED|CV_COVAR_COLS|CV_COVAR_SCALE);
```

Następująca funkcja wyliczy wartości własne (wektor **lambda**) i wektory własne (wiersze macierzy **V**) macierzy kowariancji **C**:

```
eigen(C, lambda, V);
```

Wyznaczenie normy $\|\mathbf{u}\|$ (czyli długości) wektora **u**:

```
d = norm(u);
```

Wydzielenie z macierzy kolumn z zakresu $\langle a; b \rangle$:

```
Mat B = A.colRange(Range(a, b));
```

Wyświetlenie zawartości macierzy **X**:

```
namedWindow("macierz X");
```

```
imshow("macierz X ", X);
```

```
waitKey(1000);
```