

POLITECHNIKA WROCŁAWSKA

INTELIGENCJA OBLICZENIOWA I JEJ ZASTOSOWANIA

Ćwiczenie 2
Metody redukcji wymiarowości
– nieujemna faktoryzacja macierzy
i dekompozycje tensorów

Autorzy:

Paweł ANDZIUL 200648
Robert CHOJNACKI 200685
Marcin SŁOWIŃSKI 200638

Prowadzący:

dr hab. inż. Rafał ZDUNEK

9 czerwca 2017

Spis treści

1	Zadanie 1	2
1.1	Algorytm ALS	2
1.2	Algorytm MUE	2
1.3	Algorytm HALS	2
1.4	Realizacja	2
1.5	Wyniki	4
2	Zadanie 3	4
2.1	Opis metody	4
2.2	Algorytm	4
2.3	Realizacja	5
2.4	Wyniki	7
3	Podsumowanie	13

1 Zadanie 1

Wygenerować faktory $A = [a_{ij}] \in R_+^{I \times J}$ i $X = [x_{jt}] \in R_+^{J \times T}$, gdzie $a_{ij} = \max(0, \tilde{a}_{ij})$ i $x_{jt} = \max(0, \tilde{x}_{jt})$ oraz $\tilde{a}_{ij}, \tilde{x}_{jt} \sim N(0, 1)$ (rozkład normalny). Wygeneruj syntetyczne obserwacje $Y = AX$ dla $I = 100, T = 1000, J = 10$. Stosując wybrane algorytmy NMF (ALS, MUE, HALS) wyznacz estymowane faktory \hat{A} i \hat{X} oraz unormowany błąd residualny w funkcji iteracji naprzemiennych. Oceń jakość estymacji stosując miary MSE (ang. Mean-Squared Error) lub SIR (ang. Signal-to-Interference Ratio).

1.1 Algorytm ALS

Algorytm ALS polega na N-krotnym powtórzeniu pętli, której zadaniem jest obliczenie tensorów A oraz X .

W pierwszym etapie tensory A i X inicjalizowane są liczbami losowymi zależnymi od rzędu faktoryzacji. Następnie następuje N-krotne powtórzenie pętli, w której estymowane są faktory stosując wykorzystując pochodną kwadratu odległości euklidesowej.

$$\nabla_A D(Y|AX) = (AX - Y)X^T = 0 \Rightarrow AX X^T = Y X^T \Rightarrow A = Y X^T (X X^T)^{-1}$$

$$\nabla_X D(Y|AX) = A^T(AX - Y) = 0 \Rightarrow A^T A X = A^T Y \Rightarrow X = (A^T A)^{-1} A^T Y$$

Wykorzystując otrzymane wyniki obliczany jest błąd residualny.

1.2 Algorytm MUE

1.3 Algorytm HALS

1.4 Realizacja

Listing 1: Skrypt z realizacją w środowisku MATLAB

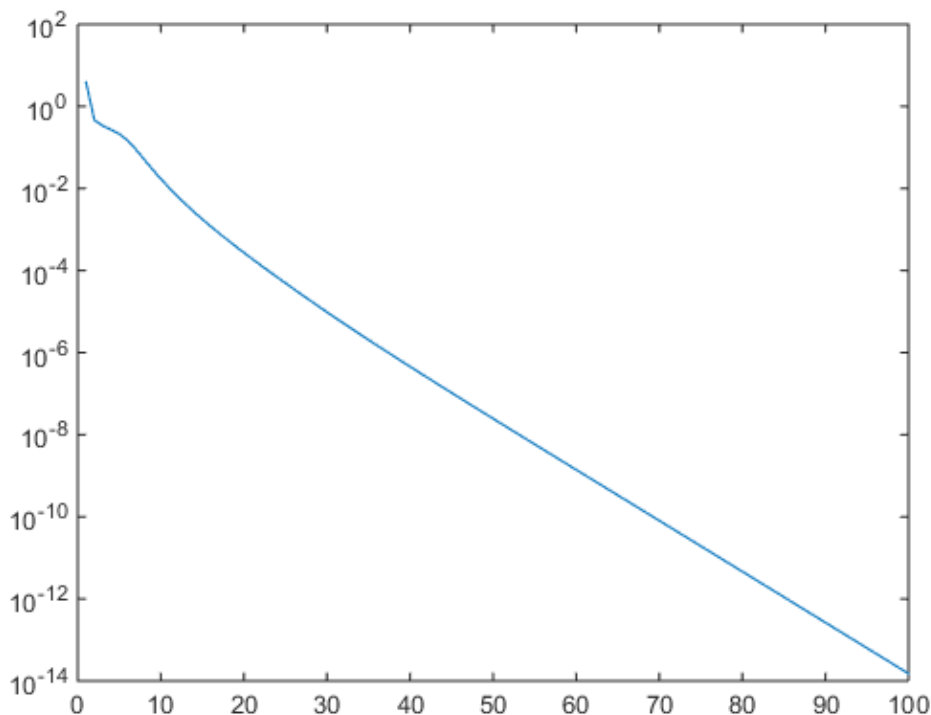
```
1 clc;
2 clear all;
3 close all;
4
5 % dane oryginalne
6 I = 100; T = 1000; J = 10;
7 Aw = max(0, randn(I, J));
8 Xw = max(0, randn(J, T));
9
10 Y = Aw*Xw;
11
12 % inicjalizacja
13 A = rand(size(Y, 1), J);
14 X = rand(J, size(Y, 2));
15
16 A1 = A;
17 A2 = A;
18 %A3 = A;
19
20 X1 = X;
21 X2 = X;
```

```

22 %X3 = X;
23
24 MaxIter = 100;
25
26 [A1,X1,res1] = NMF_ALS(A1,X1,Y,MaxIter);
27 [A2,X2,res2] = NMF_MUE(A2,X2,Y,MaxIter);
28 %[A3,X3,res3] = NMF_HALS(A3,X3,Y,J,MaxIter);
29
30 figure
31 hold on;
32 semilogy(res1)
33 semilogy(res2)
34 %semilogy(res3)
35
36 % to samo ale z wykorzystaniem wbudowanej funkcji
37
38 rng(1) % for reproducibility
39 [W,H] = nnmf(Y,10);
40 D = norm(Y-W*H,'fro')/sqrt(I*T); % blad residualny (resztowy)
41
42
43 SIR = CalcSIR(Aw,A);
44
45 % ---- to co pisal na konsoli ----
46
47 Aws = Aw*diag(1./sum(Aw,1));
48 grid on
49 eps
50 Aws(1:5,:)

```

1.5 Wyniki



Rysunek 1: Wykres ilustrujący przebieg optymalizacji naprzemiennej

2 Zadanie 3

Obrazy twarzy z bazy ORL (lub podobnej) przedstaw za pomocą tensora $Y \in R^{I_1 \times I_2 \times I_3}$, gdzie I_3 jest liczbą obrazów. Rozdziel obrazy na zbiory trenujący i testujący według odpowiedniej zasady, np. 5-folds CV i utwórz odpowiednie tensory trenujący Y_r i testujący Y_t . Tensor trenujący poddaj dekompozycji CP (np. algorytmem ALS) oraz HOSVD dla $J = 4, 10, 20, 30$. Pogrupować obrazy stosując metodę k-średnich dla faktora $\hat{U}^{(3)}$. Badania przeprowadzić dla różnej liczby grup. Porównać dokładność grupowania z metodą PCA (z poprzedniego ćwiczenia). Następnie dokonaj projekcji obrazów z tensora Y_t na podprzestrzeń cech generowaną faktorem otrzymanymi z Y_r . Dokonaj klasyfikacji obrazów w przestrzeni cech w $\hat{U}^{(3)}$ za pomocą klasyfikatora k-NN. Porównać efekty klasyfikacji różnymi metodami (np. PCA, CP, HOSVD).

2.1 Opis metody

2.2 Algorytm

2.3 Realizacja

Listing 2: Podstawowy skrypt z realizacją w środowisku MATLAB

```
1  clc;
2  clear;
3  close all;
4  load('FaceData_56_46.mat');
5
6  Persons = 8;
7  ImagesPerPerson = 10;
8  nOfImages = Persons*ImagesPerPerson;
9
10 % wczytanie danych do tensora
11 P = zeros(1,nOfImages);
12 Y=zeros(56,46,nOfImages);
13 img_index = 1;
14 for p=(1:Persons)
15     for i=(1:ImagesPerPerson)
16         P(img_index) = p;
17         Y(:,:,img_index) = FaceData(p, i).Image;
18         img_index = img_index + 1;
19     end
20 end
21 P = P';
22
23 figure;
24 suptitle('Twarze oryginalne');
25 for i=(1:nOfImages)
26     subplot(Persons, ImagesPerPerson, i);
27     imagesc(Y(:,:,i));
28     title(i)
29     colormap gray;
30     set(gca,'XtickLabel',[], 'YtickLabel', []);
31 end
32
33 % rozdzielenie na dwa zbiory (5-folds CV)
34 CV = cvpartition(P,'kfold',5);
35 train_idx = CV.training(1);
36 test_idx = CV.test(1);
37
38 % utworzenie tensorow trenujacego i testowego
39 Y_train = Y(:,:,train_idx);
40 Y_test = Y(:,:,test_idx);
41 Class_train_idx = P(train_idx);
42 Class_test_idx = P(test_idx);
43
44 J_serie = [4 10 20 30];
45
46 res_time_hosvd = zeros(1,length(J_serie));
47 res_time_kmeans = zeros(1,length(J_serie));
48 res_time_knn = zeros(1,length(J_serie));
49 res_acc_kmeans = zeros(1,length(J_serie));
50 res_acc_knn = zeros(1,length(J_serie));
51 res_rands_kmeans = zeros(1,length(J_serie));
52 res_rands_knn = zeros(1,length(J_serie));
```

```

53 res_delta = zeros(1,length(J_serie));
54 res_groups_kmeans = [];
55
56 for J_current=(1:length(J_serie))
57     J(1:3) = J_serie(J_current);
58
59     % dekompozycja hosvd (pod kmeansa)
60     tic
61     [A, B, C, G, Y_hat] = skrypt_zad3_hosvd(Y, J);
62     res_time_hosvd(J_current) = toc;
63
64     figure;
65     subplot(sprintf('Twarze zredukowane J=%d (HOSVD)', J_serie(J_current)));
66     for i=(1:nOfImages)
67         subplot(Persons, ImagesPerPerson, i);
68         imagesc(Y_hat(:,:,i));
69         title(i)
70         colormap gray;
71         set(gca, 'XtickLabel', [], 'YtickLabel', []);
72     end
73
74     % grupowanie metoda ksrednich dla faktora U^(3) - stala liczba grup (ilosc
       osob)
75     tic
76     kmeans_result = kmeans(C, Persons);
77     res_time_kmeans(J_current) = toc;
78     res_groups_kmeans = [res_groups_kmeans kmeans_result];
79     [res_acc_kmeans(J_current), res_rands_kmeans(J_current), ~] = AccMeasure(P,
       kmeans_result');
80
81     % dekompozycja hosvd
82     [Ar, Br, Cr, Gr, Yr_hat] = skrypt_zad3_hosvd(Y_train, J);
83
84     % projekcja
85     Y3 = reshape(permute(Y_test, [3 1
       2]), size(Y_test,3), size(Y_test,1)*size(Y_test,2));
86     G3 = reshape(permute(Gr, [3 1 2]), [J(3), J(1)*J(2)]);
87     Ct = Y3*pinv(double(G3)*(kron(Br,Ar))');
88     Ct = Ct.*repmat(1./sqrt(sum(Ct.^2,2)+eps), 1, size(Ct,2));
89
90     % klasyfikacja w przestrzeni cech U^(3)
91     tic
92     mdl_class = fitcknn(Cr, Class_train_idx, 'NumNeighbors', 1);
93     prediction = predict(mdl_class, Ct);
94     res_time_knn(J_current) = toc;
95     [res_acc_knn(J_current), res_rands_knn(J_current), ~] =
       AccMeasure(prediction, Class_test_idx');
96
97     % dokladnosc klasyfikacji (podobnie jak w AccMeasure)
98     res_delta(J_current) = 100*(length(find((prediction -
       Class_test_idx)==0))/length(Class_test_idx));
99
100 end
101 %%
102 figure;
103 hold on

```

```

104 title('Czas przetwarzania dla roznych wartosci J');
105 plot(J_serie, res_time_hosvd, J_serie, res_time_kmeans, J_serie, res_time_knn);
106 xlim([4 30]);
107 legend('hosvd','kmeans','knnclassify',
108        'Location','southeast','Orientation','vertical')
108 xlabel('J')
109 ylabel('czas przetwarzania [s]')
110 hold off

```

Listing 3: Funkcja realizująca algorytm HOSVD

```

1 function [ A, B, C, G, Y_hat ] = skrypt_zad3_hosvd( Y, J )
2
3     DimY = size(Y);
4
5     % unfolding
6     Y1 = reshape(Y,DimY(1),DimY(2)*DimY(3));
7     Y2 = reshape(permute(Y,[2 1 3]),DimY(2),DimY(1)*DimY(3));
8     Y3 = reshape(permute(Y,[3 1 2]),DimY(3),DimY(1)*DimY(2));
9
10    % dekompozycja tensorow
11    [E1,~] = eig(Y1*Y1');
12    A = fliplr(E1(:,DimY(1)-J(1)+1:DimY(1)));
13
14    [E2,~] = eig(Y2*Y2');
15    B = fliplr(E2(:,DimY(2)-J(2)+1:DimY(2)));
16
17    [E3,~] = eig(Y3*Y3');
18    C = fliplr(E3(:,DimY(3)-J(3)+1:DimY(3)));
19
20    G = ntimes(ntimes(ntimes(Y,A',1,2),B',1,2),C',1,2); % core tensor
21    Y_hat = ntimes(ntimes(ntimes(G,A,1,2),B,1,2),C,1,2); % tensor 3-way
22
23    C = C.*repmat(1./sqrt(sum(C.^2,2)+eps),1,size(C,2));
24
25 end

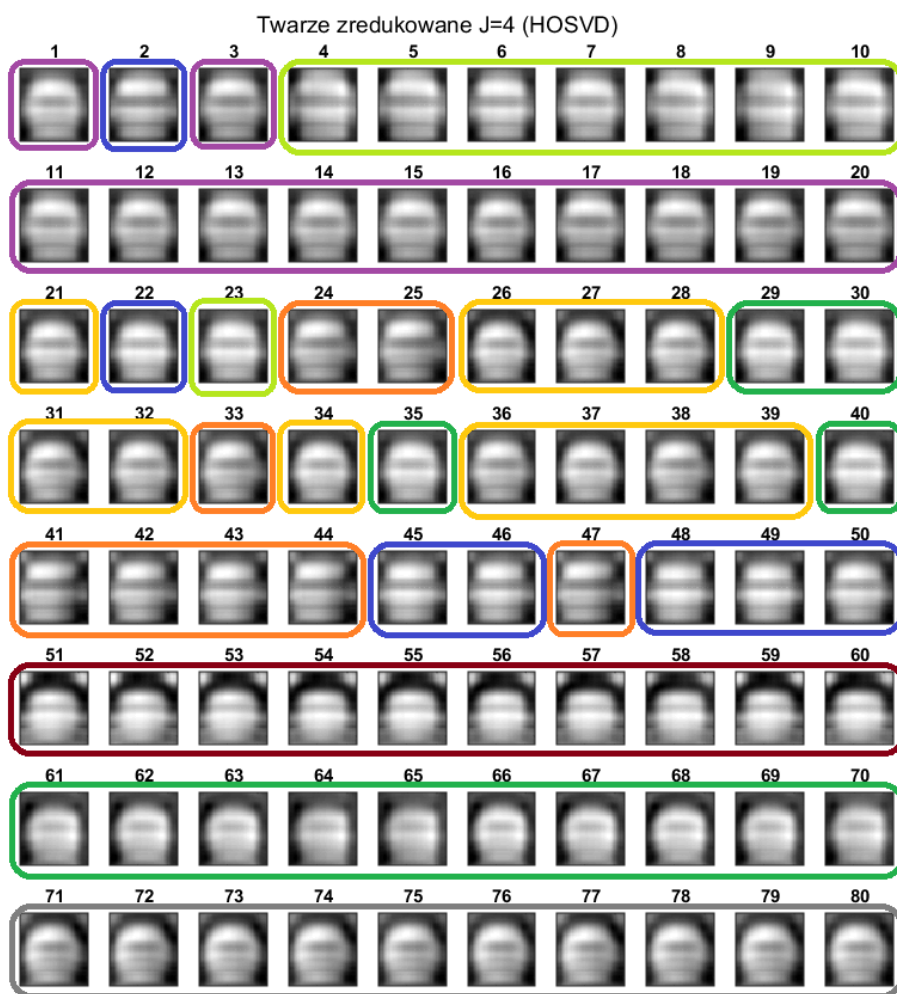
```

2.4 Wyniki

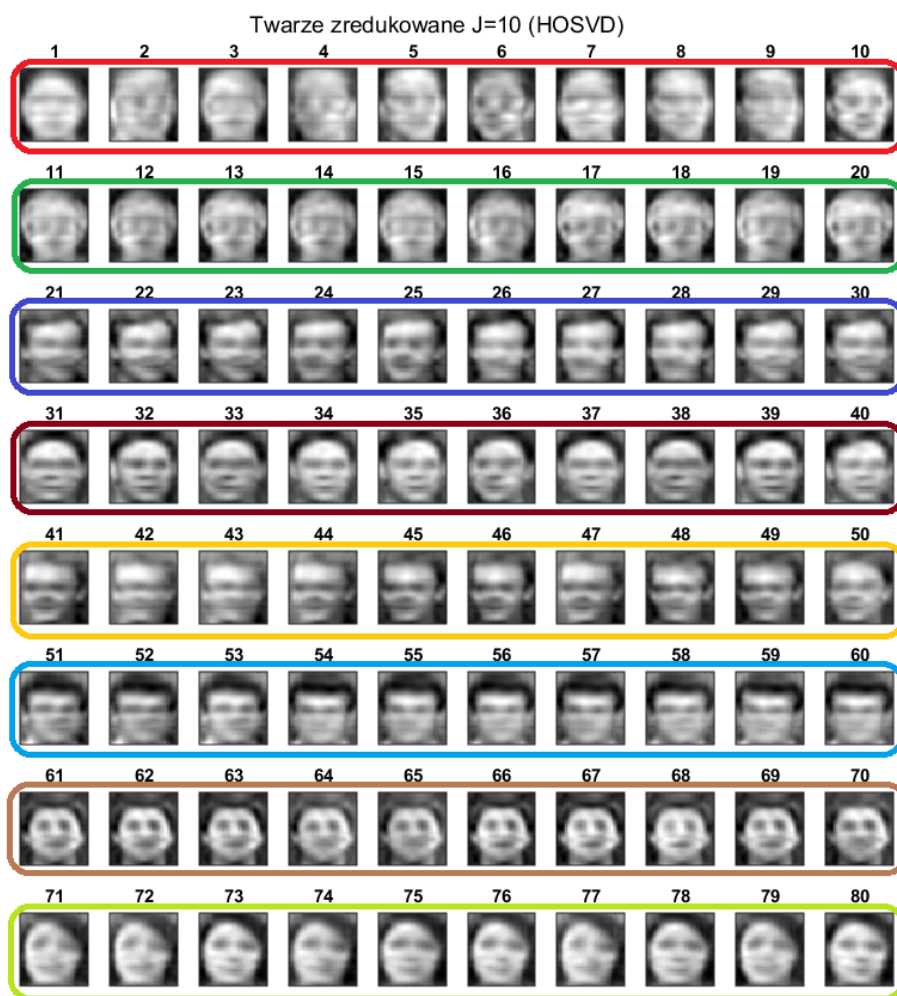
W niniejszym punkcie zamieszczono wyniki dla dekompozycji z wykorzystaniem metody HOSVD. Badaniu poddano 80 obrazów twarzy (ilustracja 2) z bazy Uniwersytetu Cambridge [7].



Rysunek 2: Twarze wykorzystane podczas testów



Rysunek 3: Twarze zredukowane J=4 (HOSVD)



Rysunek 4: Twarze zredukowane J=10 (HOSVD)



Rysunek 5: Twarze zredukowane J=20 (HOSVD)



Rysunek 6: Twarze zredukowane J=30 (HOSVD)

W tabeli 1 zamieszczono otrzymane wartości metryk. Dla algorytmu k-średnich zostały policzone Acc (dokładność) oraz Rand's index.

Tabela 1: Otrzymane metryki dla różnych wartości parametru J (k-średnich)

	4	10	20	30
Acc (dokładność)	76,25	100	80,00	63,75
Rand's index	92,56	100	93,04	84,56

Jak możemy zauważyć metoda k-średnich daje najlepsze rezultaty dla J=10, zarówno mniej jak i więcej szczegółów w obrazie negatywnie wpływa na rezultat grupowania. W przypadku metody najbliższych sąsiadów jest inaczej – tutaj im bardziej szczegółowy obraz otrzyma ta metoda na wejściu tym lepsze będą rezultaty. Trzeba również pamiętać, że metoda k-średnich nie wymaga zbioru treningowego i uczącego.

W tabeli 2 zamieszczono metryki dla klasyfikacji metodą najbliższych sąsiadów. Zostały policzone dokładność (Acc) oraz Rand's index dla klasyfikacji w przestrzeni cech $\hat{U}^{(3)}$ przy pomocy metody najbliższych sąsiadów.

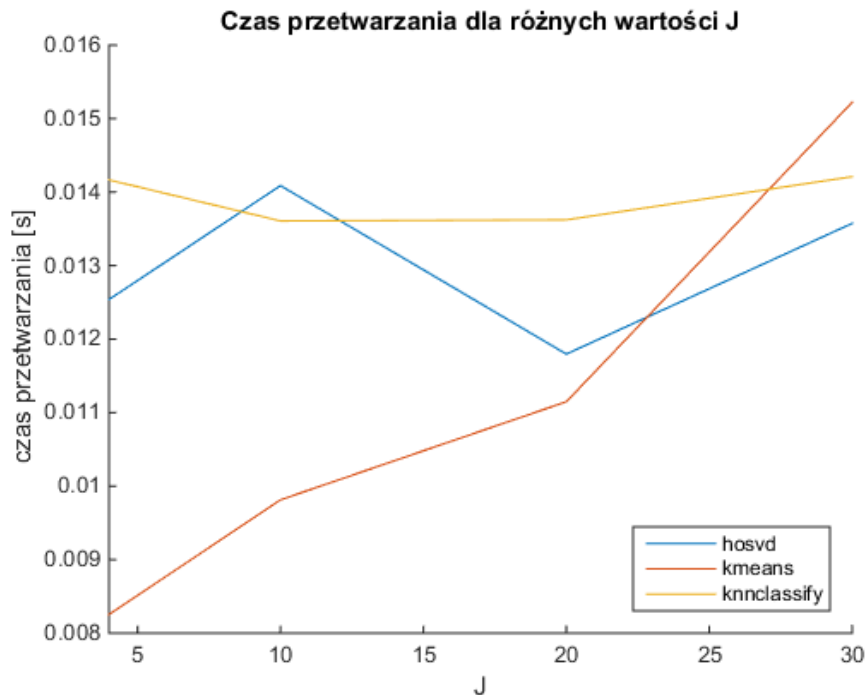
Tabela 2: Otrzymane metryki dla różnych wartości parametru J (k-najbliższych sąsiadów)

	4	10	20	30
Acc (dokładność)	81,25	93,75	100	100
Rand's index	94,17	97,50	100	100

Natomiast zależności czasowe przedstawiono w tabeli 3. Grupowanie przeprowadzono dla ilości grup równej liczbie osób aby móc jednoznacznie zinterpretować wyniki. Na ilustracji 7 zamieszczono graficzne porównanie.

Tabela 3: Czas przetwarzania [ms] w zależności od parametru J

	4	10	20	30
HOSVD	12,54	14,09	11,80	13,58
grupowanie k-średnich	8,25	9,81	11,15	15,22
klasyfikacja k-NN	14,17	97,50	100	100



Rysunek 7: Czas przetwarzania w zależności od wartości J

3 Podsumowanie

Podczas prac nad zadaniami zapoznano się z metodami redukcji wymiarowości przy pomocy nieujemnej faktoryzacji macierzy i dekompozycji tensorów.

Literatura

- [1] Dokumentacja środowiska MATLAB, <https://www.mathworks.com/>
- [2] Zdunek, Rafał, „Nieujemna faktoryzacja macierzy i tensorów : zastosowanie do klasyfikacji i przetwarzania sygnałów”, Oficyna Wydawnicza Politechniki Wrocławskiej, 2014
- [3] <http://www.sandia.gov/~tgkolda/TensorToolbox/index-2.6.html>
- [4] <http://www.esat.kuleuven.be/sista/tensorlab/>
- [5] <http://www.bsp.brain.riken.jp/TDALAB/>
- [6] <http://www.bsp.brain.riken.jp/~phan/>
- [7] <http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html>