Технічні науки

УДК 004.93'11, 681.518

Сарапулов Віктор Сергійович
студент
Національний технічний університет України
«Київський політехнічний інститут»
Сарапулов Виктор Сергеевич
студент
Национальный технический университет Украины «Киевский политехнический институт»
Sarapulov V.S.
student
National Technical University of Ukraine «Kyiv Polytechnic Institute»

ДОСЛІДЖЕННЯ EIGENFACE АЛГОРИТМУ РОЗПІЗНАВАННЯ ОБЛИЧЬ ТА ЙОГО РЕАЛІЗАЦІЯ У MATLAB СЕРЕДОВИЩІ

ИССЛЕДОВАНИЕ EIGENFACE АЛГОРИТМА РАСПОЗНАВАНИЯ ЛИЦ И ЕГО РЕАЛИЗАЦИЯ В СРЕДЕ MATLAB

RESEARCH OF EIGENFACE ALGORITHM FOR FACE RECOGNITION AND ITS REALIZATION IN MATLAB

Анотація: Досліджено роботу алгоритму комп'ютерного зору Eigenface. Описано побудову та архітектуру даного алгоритму. Реалізовано його навчання в Matlab середовищі.

Ключові слова: задача розпізнавання обличчя, комп'ютерний зір, власні вектори, системи автоматичного управління.

Аннотация: Исследована работа алгоритма компьютерного зрения Eigenface. Описаны строение и архитектура данного алгоритма. Реализовано его обучение в среде Matlab.
Ключевые слова: задача распознавания лица, компьютерное зрение, собственные вектора, системы автоматического управления.

Summary: Research of Eigenface algorithm of computer vision. Desciption of architecture and structure of algorithm. Realization of its training in Matlab.
Key words: face recognition problem, computer vision, eigen vectors, automatic control systems.

Вступ

В наш час все більшого розповсюдження отримують біометричні системи ідентифікації людини. Традиційні системи ідентифікації потребують знання пароля, наявності ключа, ідентифікаційної картки або іншого ідентифікаційного документа, що можна забути, загубити або підробити. На відміну від них, біометричні системи основані на унікальності біологічних характеристиках, що важко підробити та які однозначно визначають конкретну людину. До таких характеристик відносять відбитки пальців, форма долоні, візерунок радужної оболонки, зображення сітківки ока. Обличчя, голос і запах кожної людини також індивідуальні.

Розпізнавання людини по зображенню обличчя виділяється серед біометричних систем тим, що, по-перше, не потребує спеціального коштовного обладнання. Для більшості додатків достатньо персонального комп’ютера або звичайної відеокамери. По-друге, фізичний контакт людини з приладами відсутній. Не треба ні до чого торкатися або спеціально зупинятися і чекати реакції системи. У більшості випадків

достатньо просто пройти повз або затриматись перед камерою на декілька секунд.

До недоліків розпізнавання людини по зображенню обличчя слід віднести те, зо сама по собі така система не дає 100%-ової гарантії ідентифікації. Там, де необхідна висока надійність, застосовують комбінування декількох біометричних методів.

На даний момент проблемі розпізнаванню людини по зображенню обличчя присвячена велика кількість наукових робіт, однак в цілому вона ще далека від вирішення. Основні труднощі пов'язані з тим, щоб розпізнати людину по зображенню обличчя незалежно від зміни ракурсу та умов освітлення при зйомці, а також при різних змінах, що зв'язані зі зростом, зачіскою і т.д.

Розпізнавання зображень перетинаються з розпізнаванням образів. Такі задачі не мають точного аналітичного розв'язку. При цьому необхідне виділення ключових факторів, що характеризують зоровий образ, визначення відносної важливості факторів шляхом вибору їх вагових коефіцієнтів та врахування взаємозв'язків між ними. Спершу ці задачі вирішувались людиною-експертом вручну, шляхом експериментів, що займало багато часу та не гарантувало якості. У нових методах виділення ключових ознак здійснюється шляхом автоматичного аналізу навчальної вибірки , але тим не менш більша частина інформації щодо ознак вводиться вручну. Для автоматичного застосування таких аналізаторів вибірка має бути досить великою та охоплювати усі можливі ситуації.

В цій статті буде досліджено алгоритм «власних обличь». Зокрема, буде реалізовано навчання алгоритму первинним набором обличь та розпізнавання обличь.

Алгоритм Eigenface

В онову алгоритму покладено використання фундаментальних статичних характеристик: середніх (математичне очікування) та коваріаційної матриці; використання метода головних компонент. Як і будь-який інший алгоритм сфери компьютерного навчання (machine learning), його необхідно спершу навчити первинній виборці (training set), яка складається з певної кількості зображення обличь, які ми хочемо розпізнавати. Як тільки модель стане навченою, слід подати на вхід деяке зображення і в результаті отримаємо відповідь на питання: якому зображенню із загальної виборки з найбільшою вірогідністю відповідає дане та чи належить дане зображення виборці взагалі.

Головний принцип алгоритму – представити вхідні зображення у вигляді однієї спільної матриці, що складатиметься із суми базисних компонент зображень:

$$\Phi_i = \sum_{j=1}^{K} w_j u_j$$

Де Φ_i – відцентроване і-те зображення обличчя, w_j - ваги, u_j – власні вектори.

На рис.1 ми отримуємо вихідне зображення обличчя зваженою сумою власних векторів та додаванням середнього значення. Інакше кажучи, маючи w та u ми можемо відтворити будь-яке вихідне зображення.

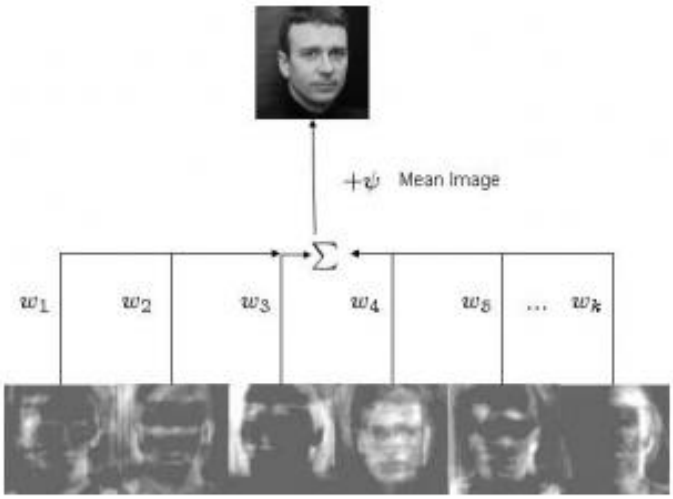

Рис. 1. Отримання вихідного обличчя.

Навчальну виборку необхідно спроектувати у новий простір, де кожна розмірність даватиме певний вклад у спільне представлення. Метод головних компонент дозволяє знайти базис нового простору таким чином, щоб данні у ньому розташовувались, у певному сенсі, оптимально. Щоб це зрозуміти, достатньо уявити, що у новому просторі певні розмірності будуть нести більше спільної інформації, тоді як інші нестимуть лише специфічну інформацію. Як правило, розмірності більш високого порядку несуть набагато менше корисної інформації аніж перші розмірності, що відповідають найбільшим власним значенням. Залишаючи розмірності лише з корисною інформацією, ми отримуємо простір ознак, у якому кожне зображення вихідної виборки представлено у спільному вигляді. У цьому, дуже стисло, і полягає ідея алгоритму.

Надалі, маючи певні зображення, ми можемо відобразити його на створений раніше простір та визначити, до якого зображення навчальної виборки наш приклад відноситься більш за все. Якщо він знаходиться на досить великій відстані від всіх даних, тоді це зображення з найбільшою вірогідністю, не належить до зображень з нашої бази.

Архітектура алгоритму Eigenface. Навчання алгоритму

Алгоритм Eigenface поділяється на певні етапи. На першому етапі всі зображення з бази необхідно представити у вигляді вектору. Так як на вхід подаються напівтонові зображення, значення вектору відповідатимуть значенням рівня сірого кольору відповідних пікселів (див. Рис. 2). Так як зображення має розмір M*K, розмір вектору також дорівнюватиме M*K.

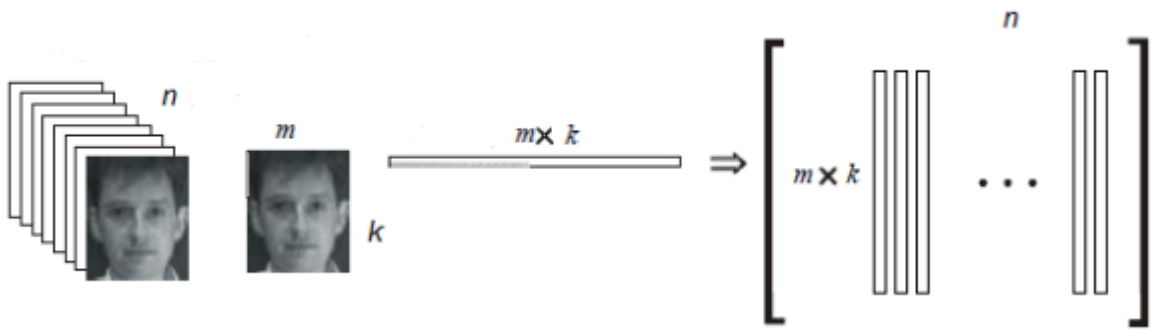

Рис. 2. Представлення вхідного зображення у векторному вигляді.

Отримані вектори у сукупності утворюють матрицю розміром (M*K) x N, де N – кількість вхідних зображень.

Наступним етапом являється знаходження середнього значення матриці та його віднімання від кожного вектору:

$$\boldsymbol{\mu_i} = \frac{\mathbf{1}}{\boldsymbol{n}} \sum_{\boldsymbol{j=1}}^{\boldsymbol{n}} \boldsymbol{x_{ij}}$$

$$\boldsymbol{x_{ij}} = \boldsymbol{x_{ij}} - \boldsymbol{\mu_i}$$

Ця процедура необхідна для того, щоб відкинути спільні ознаки обличь та залишити лише індивідуальні. Якщо зконвертувати вектор середніх значень у зображення, отримаємо «спільне обличчя»:

Рис. 3. Приклад «спільного обличчя».

Рис. 4. Віднімання «спільного обличчя». На другій стрічці ми чітко бачимо особливі ознаки кожного обличчя, що залишилися після віднімання.

Після отримання нормалізованої матриці, вираховуємо матрицю коваріацій C, власні вектори (eigenvectors – від того походить назва алгоритму) u та ваги w для кожного зображення з виборки:

$$C = X * X^T$$

$$w = A * u$$

Алгоритм знаходження власних векторів:

1. Складаємо характеристичний многочлен матриці: $\Delta_A(x) = \det(A - x * E)$
2. Знаходимо всі відмінні корені характеристичного рівняння $\Delta_A(x) = 0$

3. Для кореня x – x_1знайти фундаментальну систему φ_1, $\varphi_{2, \ldots,}$ φ_{n-r} рішень однорідної системи рівнянь $(A - x_1 * E) * x = 0$, де $r = rang(A - x_1 * E)$
4. Записати лінійно незалежні власні вектори матриці A, що відповідають власному значенню x_1: $s_1 = C_1\varphi_1$, $s_2 = C_2\varphi_2$, … , $s_{n-r} = C_{n-r}\varphi_{n-r}$, де C_1, C_2, …, C_{n-r} – відмінні від нуля довільні сталі.
5. Повторити пункти 3, 4 для інших власних значень.

Слід відзначити, що власні значення не мають принципово важливого значення для алгоритму, але вони можуть надати важливу інформацію. Власні значення показують дисперсію щодо кожної осі головних компонент (кожній осі відповідає одна розмірність у просторі). Таким чином ми можемо дізнатись, які з головних компонент являються важливими і формують основну частину інформації, а які – ні і якими можна знехтувати.

Розпізнавання

Після того, як алгоритм було навчено, можна подавати на вхід різні зображення, які в свою чергу будуть проходити майже той самий шлях, що і навчальні зображення: вирахування середнього значення, додавання до спільного простору компонент, знаходження ваги. Далі, коли ваги було розподілено, слід визначити, з яким об’єктом з виборки вхідний об’єкт найбільш схожий. Для цього використовується або пошук евклідової відстані, або відстані Махалонобіса. Той об’єкт, з яким відстань буде мінімальна, і вважається шуканим. Очевидно, що якщо подати на вхід обличчя, що не приймало участі при навчанні, алгоритм його «розпізнає». Для того, щоб алгоритм міг відсіювати зайві об’єкти, що не були у навчальній виборці, слід задати максимальне допустиме значення відстані.

Після того, як алгоритм було навчено, можна подавати на вхід різні зображення, які в свою чергу будуть проходити майже той самий шлях, що і навчальні зображення: вирахування середнього значення, додавання до спільного простору компонент, знаходження ваги. Далі, коли ваги було розподілено, слід визначити, з яким об'єктом з виборки вхідний об'єкт найбільш схожий. Для цього використовується або пошук евклідової відстані, або відстані Махалонобіса. Той об'єкт, з яким відстань буде мінімальна, і вважається шуканим. Очевидно, що якщо подати на вхід обличчя, що не приймало участі при навчанні, алгоритм його «розпізнає». Для того, щоб алгоритм міг відсіювати зайві об'єкти, що не були у навчальній виборці, слід задати максимальне допустиме значення відстані.

Реалізація у середовищі Matlab

Для навчання виборки було розроблено таку функцію:

```
function [Imgs,w,h,Vecs,Vals,Psi] = load_images(filelist,downscale_f)
%
%     Imgs - матриця зображень
%     w    - ширина зображення
%     h    - довжина зображення
%     Vecs - власні вектори зображень
%     Vals - власні значення зображень
%     Psi  - середнє значення
%
  if nargin < 1 || nargin > 2
    error('usage: load_Imgs(filelist[,downscale_f]');
  end;
  if nargin == 1
    downscale_f = 1.0;
  end;
  Imgs = []; old_w = 0; old_h = 0; w=0; h=0;
  numimgs = linecount(filelist);
  fid = fopen(filelist,'r');
  if fid < 0 || numimgs < 1
    error(['Cannot get list of Imgs from file "' filelist, '"']);
```

```
  end;
  for i = 1:numimgs
    imgname = fgetl(fid);
    if ~isstr(imgname)            %
      break;                      % вихід з циклу, якщо EOF
    end;
    fprintf(1,'loading PGM file %s\n',imgname);
    Img = readpgm(imgname);       % Окрема функція зчитування у
двопросторовий              масив
    if i==1                       % Якщо це перше зображення,
      old_w = size(Img,2);        % тоді визначити довжину, ширину та
масштаб
      old_h = size(Img,1);
      if downscale_f <= 1.0
    w = old_w; h = old_h;
      else
    w = round(old_w/downscale_f); h = round(old_h/downscale_f);
      end;
      Imgs = zeros(w*h,numimgs);    % - передвстановлення значень матриці
    end;
    if downscale_f > 1.0
      Img = im_resize(Img,w,h);       % зміна масштабування
    end;
    Imgs(1:w*h,i) = reshape(Img',w*h,1);    % зі стрічки зробити стовбець
  end;
  fclose(fid);                    % закрити список зображень після
закінчення
  fprintf(1,'Read %d Imgs.\n',numimgs);
[Vecs,Vals,Psi] = pc_evectors(Imgs,numimgs); % визначення власних векторів.
                                             % Окрема функція
```

Функція підрахунку кількості зображень (linecount):

```
function lc = linecount(filename)
  fid = fopen(filename,'r');
  if fid < 0
    lc = 0;
  else
    lc = 0;
    while 1
```

```
      ln = fgetl(fid);
      if ~isstr(ln) break; end;
      lc = lc + 1;
    end;
    fclose(fid);
  end;
```

Функція зчитування зображення (readpgm):

```
function image = readpgm(filename)
fid = fopen(filename,'r');
A = fgets(fid);
if strcmp(A(1:2),'P5') ~= 1
  error('File is not a raw PGM');
end;
A = fgets(fid);
sizes = sscanf(A,'%d');
w = sizes(1);
h = sizes(2);
A = fgets(fid);
max = sscanf(A,'%d');
tlength = w*h;
if max ~= 255
 error('Cannot handle anything but 8-bit graymaps');
end;
[v,count] = fread(fid,inf,'uchar');
if count ~= tlength
 error('File size does not agree with specified dimensions.');
end;
image = reshape(v,w,h)';
fclose(fid);
```

Функція зчитування власних векторів (pc_evectors):

```
function [Vecs,Vals,Psi] = pc_evectors(A,numvecs)
  if nargin ~= 2
    error('usage: pc_eVecs(A,numvecs)');
  end;
  nexamp = size(A,2);
  fprintf(1,'Computing average vector and vector differences from
avg...\n');
```

```
  Psi = mean(A')';
  % Вирахування середнього значення з усіх векторів
  for i = 1:nexamp
    A(:,i) = A(:,i) - Psi;
  end;
  % Знаходження матриці коваріацій
  fprintf(1,'Calculating L=A''A\n');
  L = A'*A;
  % Підрахунок власних векторів(стовбці Vecs) та власних значень (диагоналі
Vals)
  fprintf(1,'Calculating eigenVecs of L...\n');
  [Vecs,Vals] = eig(L);
  % Сортування Джеймса Джавурек-Хаміга по зменшенню значень власних значень
  fprintf(1,'Sorting eVecs/Vals...\n');
  [Vecs,Vals] = sortem2(Vecs,Vals);
  % конвертування власних векторів з A'*A у власні вектори A*A'
  fprintf(1,'Computing eigenVecs of the real covariance matrix..\n');
  Vecs = A*Vecs;
  Vals = diag(Vals);
  Vals = Vals / (nexamp-1);
  % Нормалізація власних веторів та відкидання «поганих»
  num_good = 0;
  for i = 1:nexamp
    Vecs(:,i) = Vecs(:,i)/norm(Vecs(:,i));
    if Vals(i) < 0.00001
      % Set the vector to the 0 vector; set the value to 0.
      Vals(i) = 0;
      Vecs(:,i) = zeros(size(Vecs,1),1);
    else
      num_good = num_good + 1;
    end;
  end;
  if (numvecs > num_good)
    fprintf(1,'Warning: numvecs is %d; only %d exist.\n',numvecs,num_good);
    numvecs = num_good;
  end;
  Vecs = Vecs(:,1:numvecs);
```

Функція сортування Джеймса Джавурек-Хаміга:

```
function [vectors values] = sortem2(vectors, values)
if nargin ~= 2
 error('Must specify vector matrix and diag value matrix')
end;
vals = max(values);
[svals inds] = sort(vals,'descend');
vectors = vectors(:,inds);
values = max(values(:,inds));
values = diag(values);
```

На вхід функції load_images подаємо список адрес зображень at.txt. Після виклику функції, у терміналі з'являється інформація щодо кількості зображень та інформацію, на якому етапі проходить навчання в даний момент (Див. Рис. 5)

```
>> [Imgs,w,h,Vecs,Vals,Psi] = load_images('at.txt');
loading PGM file C:/test/at/s13/2.pgm
loading PGM file C:/test/at/s13/7.pgm
loading PGM file C:/test/at/s13/6.pgm

          • • •

loading PGM file C:/test/at/s38/8.pgm
loading PGM file C:/test/at/s38/1.pgm
Read 399 Imgs.
Computing average vector and vector differences from avg...
Calculating L=A'A
Calculating eigenVecs of L...
Sorting eVecs/Vals...
Computing eigenVecs of the real covariance matrix..
Warning: numvecs is 399; only 398 exist.
```

Рис.5. Термінал MatLab після виклику функції load_images.

Після закінчення роботи функції, всі дані зберігаються у відповідних їм змінних (Див. рис. 6).

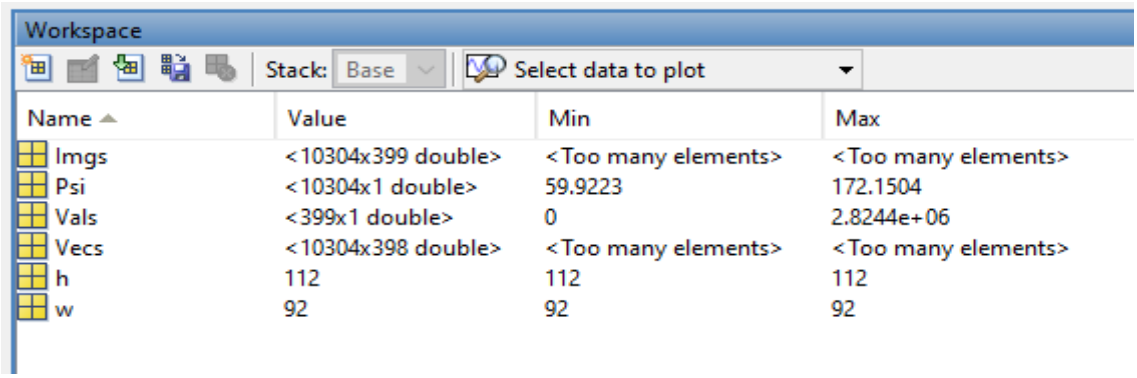

Рис. 6. Дані навченого алгоритму.

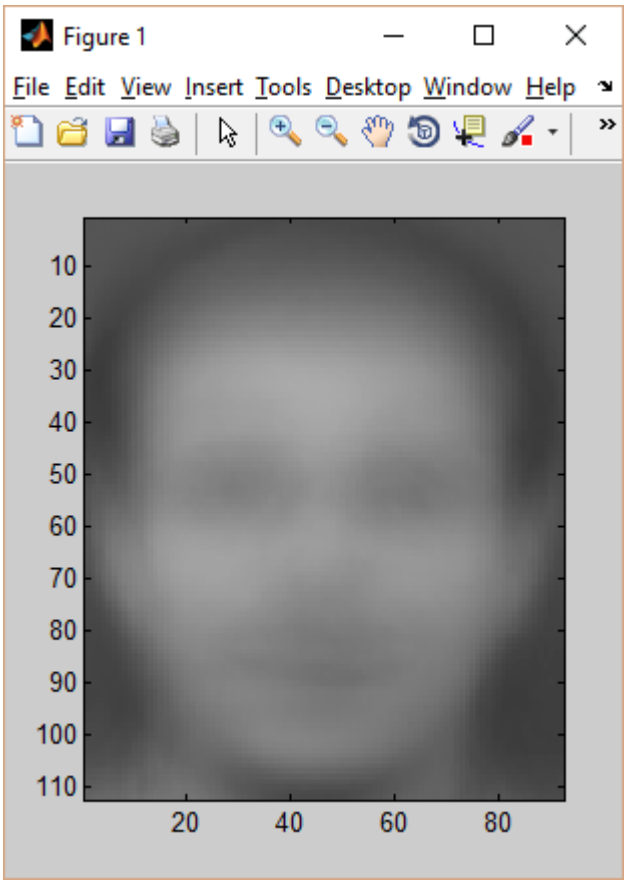

Рис.7. Отримане «спільне обличчя»

Також, була розроблена додаткова функція spectrum, що візуалізувала значення власних значень. Вона була розроблена саме для того, щоб зрозуміти, якими головними компонентами ми можемо знехтувати. Лістинг функції:

```
plot(Vals);
CVals = zeros(1,length(Vals));
```

```
CVals(1) = Vals(1);
for i = 2:length(Vals)
CVals(i) = CVals(i-1) + Vals(i);
end;
CVals = CVals / sum(Vals);
plot(CVals);
ylim([0 1]);
```

Результат роботи цієї функції можемо побачити на рис. 8.

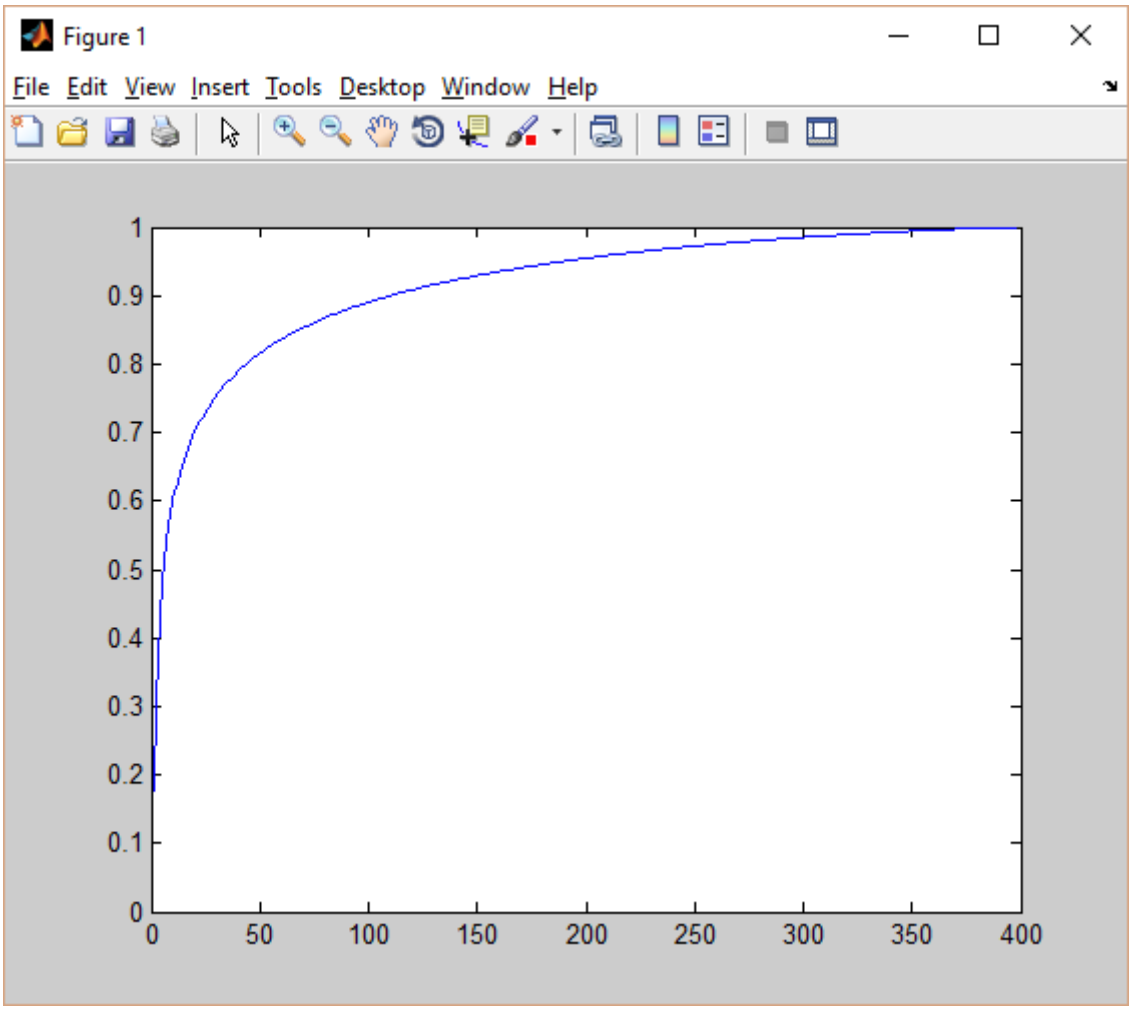

Рис. 8. Нормалізовані значення власних значень.

Згідно з графіком, ми чітко бачимо, що перші 150 компонентів містять 90% всієї інформації. Іншими компонентами можна знехтувати, адже на точність розпізнавання вони мають занадто низький вплив, а швидкість роботи алгоритму при цьому збільшиться.

Висновки

У даній роботі було описано побудову та архітектуру алгоритму розпізнавання Eigenface та його навчання у середовищі Matlab. Було застосовано аналіз головних компонент (PCA) для навчання первинної виборки алгоритму.

Література:

1. D. Pissarenko, *Eigenface-based facial recognition*, 2003
2. T. M. Mitchell, *Machine Learning*, McGraw-Hill International Editions, 1997.
3. M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Proc. of Computer Vision and Pattern Recognition*, IEEE, June 1991.
4. Delac, K., Grgic, M., Liatsis, P., *Appearance-based Statistical Methods for Face Recognition*, Proceedings of the 47th International Symposium ELMAR-2005 focused on Multimedia Systems and Applications, Zadar, Croatia, 08-10 June 2005