American University of Armenia

College of Science and Engineering

Optimization Project

Point-Pattern Matching

Authors
Syuzanna Loretsyan
Karlos Muradyan
David Shadunts

Supervisor
Lusine Poghosyan

Spring 2018

Contents

1 Introduction **3**

2 Application **3**

3 Similarity transformation **4**

4 Problem formulation **4**

5 Solution of the problem in Eq.(4) **5**

6 Alternative measure of dissimilarity **6**

7 Handwritten character recognition **8**

8 Implementation **11**
8.1 dataManage.m 12
8.2 ppm.m 15

9 Conclusion **19**

Abstract

The problem of the Point-Pattern Matching is the following: you have a database of point patterns (say, different handwritten letters), you observe a new pattern, and your task is to identify the one from the database which is the closest one to this new pattern. Of course, people used to use Neural Networks for this task, but this project will be devoted to the transformation-minimization method for the solution.

Keywords: Point-pattern matching,vectors,optimization,handwritten characters, error,Hessian,Similarity transformation,rotation.

1 Introduction

Point pattern matching is more efficient then other techniques of image recognition in case when the image is large and we have few points.In such cases its speed and flexibility helps it to be the most optimal algorithm.

A problem that we are going to solve in this project we would call point-pattern matching problem. In point-pattern matching problem a pattern such as a printed or handwritten character, numeral, symbol, or even the outline of a manufactured part can be described by a set of points,

$$P = \{p_1, p_2, ..., p_n\} \quad (1)$$

where

$$p_i = \begin{bmatrix} p_{i1} \\ p_{i2} \end{bmatrix}$$

is a vector in terms of the coordinates of the i-th sample point.We will have our desired result and consider P as the point patter of the object,in case when the number of points in P, n, is sufficiently large.In this case equation (1) will describe the object accurately.We can have different point pattern P' for the object.This can happen when we look at the object from a different distance and/or a different angle.It is an interesting problem to examine whether two given patterns are matched to within a scaled rotation and a translation.

The pattern-matching problem which we are considering can be formed like this: We have a database that contains N standard point patterns $\{P_1, P_2, ..., P_N\}$ where each P_i has the form of Eq.(1) and we need to find a pattern from the database that best matches a given point pattern $Q = \{q_1, q_2, ..., q_n\}$. For solving this problem we need to pay attention to two things. First, we need to understand what we mean by saying 'best matching'. Second, we need to develop a solution method to find an optimal pattern P^* from the database that best matches pattern Q based on the chosen measure.

2 Application

Point pattern matching is used in various fields as astronautics, document processing, computational biology and computational chemistry.

It is used in astronautics particularly in The Institute for Aerospace and Astronautics of the Technical University of Berlin for determining TUBSAT satellites attitude.They have cameras on them to take pictures of sky.This pictures are used to match the stars pattern with catalog information and help to control the satellite position and orientation.

In biology and chemistry, it is used for autoradiograph alignment, pharmacophore identification and protein structure alignment.

3 Similarity transformation

So, we said that we can have two point patterns P and P' for the same object.They can be called similar if we can obtain one pattern by applying a scaled rotation plus a translation to the other. If pattern P is given by Eq.(1) and

$$P' = \{p'_1, p'_2, ..., p'_n\}$$

where

$$p'_i = \left[p'_{i1}, p'_{i2}\right]^T$$

then P and P' are similar if and only if there exist a rotation angle θ, a scaling factor η, and a translation vector $r = \left[r_1, r_2\right]^T$ such that the relation

$$p'_i = \begin{bmatrix} cos\theta, -sin\theta \\ sin\theta,\ cos\theta \end{bmatrix} p_i + \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} \quad (2)$$

holds for $i = 1, 2, ..., n$. This kind of transformations that map pattern P to pattern Q are called *similarity transformations*. From Eq.(2), we see that a similarity transformation is characterized by the parameter column vector $\left[\eta, \theta\, r_1\, r_2\right]^T$

We can see that the similarity transformation is a nonlinear function of parameters η and θ. Because of this nonlinearity there can be an increase in the amount of computation required by the optimization process. We can fix this problem by applying the variable substitution.

$$a = \eta cos\theta,\ \ b = \eta sin\theta$$

to Eq. (2) to obtain

$$p'_i = \begin{bmatrix} a, -b \\ b,\ a \end{bmatrix} p_i + \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} \quad (3)$$

So, the parameter vector becomes $x = \left[a\, b\, r_1\, r_2\right]^T$ Now the similarity transformation depends linearly on the parameters.

4 Problem formulation

When solving a real-life problem, the probability that the point pattern Q chosen by the user and the point pattern in the database will be exactly the same is close to zero. The best we can do is identify the closest pattern to Q to within a similarity transformation. Let

$$q = \{q_1, q_2, ..., q_n\}$$

be a given pattern and

$$P'(x) = \{p'_1, p'_2, ..., p'_n\}$$

transformed version of pattern

$$P = \{p_1, p_2, ..., p_n\}$$

Let these patterns be represented by the matrices
$Q = [q_1\, q_2, ...q_n]$,
$P' = [p'_1\, p'_2, ...p'_n]$,
and
$P = [p_1\, p_2, ...p_n]$, respectively. A transformed pattern P' that matches Q can be obtained by solving the unconstrained optimization problem

$$\min_x \quad ||P'(x) - Q||_F^2 \quad (4)$$

where $||.||$ denotes the Frobenius norm . The solution of the above minimization problem corresponds to finding the best transformation that would minimize the difference between patterns P' and Q in the Frobenius sense. Since

$$||P'(x) - Q||_F^2 = \sum_{i=1}^{n} ||p'_i(x) - q_i||^2$$

the best transformation in the least-squares sense is obtained. Now if x^* is the minimizer of the problem in Eq.(4), then the error

$$e(P', Q) = ||P'(x^*) - Q||_F (5)$$

is a measure of the dissimilarity between patterns P' and Q. So for having a good result , $e(P', Q)$ should be as small as possible and the best case will be when it will be zero value .This case we will call perfect match.

5 Solution of the problem in Eq.(4)

From Eq.(3), Eq. (5) we get

$$||P'(x) - Q||_F^2 = \sum_{i=1}^{n} ||p'_i(x) - q_i||^2 = \sum_{i=1}^{n} \left|\left| \begin{bmatrix} ap_{i1} - bp_{i2} + r_1 \\ bp_{i1} + ap_{i2} + r_2 \end{bmatrix} - q_i \right|\right|^2 =$$

$$= \sum_{i=1}^{n} \left|\left| \begin{bmatrix} p_{i1} & -p_{i2} & 1 & 0 \\ p_{i2} & p_{i1} & 0 & 1 \end{bmatrix} x - q_i \right|\right|^2 = x^T H x - 2x^T b + k \quad (6a)$$

where

$$H = \begin{bmatrix} \sum_{i=1}^{n} R_i^T R_i & \sum_{i=1}^{n} R_i^T \\ \sum_{i=1}^{n} R_i & nI_2 \end{bmatrix}, R_i = \begin{bmatrix} p_{i1} & -p_{i2} \\ p_{i2} & p_{i1} \end{bmatrix} (6b)$$

$$b = \sum_{i=1}^{n} \begin{bmatrix} R_i & I_2 \end{bmatrix}^T q_i \quad (6c)$$

$$k = \sum_{i=1}^{n} ||q_i||^2 \quad (6d)$$

The Hessian H in Eq. (6b) is positive definite,so the objective function in Eq. (4) is globally strictly convex and, so,has a unique global minimizer. Using Eq. (6a), the gradient of the objective function is

$$g(x) = 2Hx - 2b$$

Now,if we want to find the unique global minimizer

$$g(x) = 2Hx - 2b = 0$$

$$x^* = H^{-1}b (7)$$

There are two conditions which should be satisfied in order to say that matrix H is invertible. First of all it should be a square matrix i.e it should have the same number of rows and columns. Second, the determinant of the matrix should not be equal to zero. As in our case H is positive definite 2 x 2 matrix, we can surely state that the inverse H^{-1} exists. The inverse of H can be calculated as follows. Let A be 2 x 2 matrix with non-zero determinant.

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

then

$$A^{-1} = \frac{1}{det(A)} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

where det(A) = ad-bc, as it is 2 x 2 matrix.

6 Alternative measure of dissimilarity

As can be seen in Eq.(6a), the Frobenius norm of a matrix can be related to the L_2 norm of its column vectors. If we define two new vectors $p'(x)$ and q as

$$p'(x) = \begin{bmatrix} p'_1(x) \\ p'_2(x) \\ \vdots \\ p'_n(x) \end{bmatrix} \; and \; q = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix}$$

then Eq. (6) implies that

$$||P'(x) - Q||_F^2 = ||p'(x) - q||^2$$

So we can express the dissimilarity measure defined in Eq. (5) as

$$e(P'Q) = ||p'(x) - q||$$

An alternative of the above dissimilarity measure can be defined in terms of the L_{2p} norm

$$e_{2p}(P', Q) = ||p'(x) - q||_{2p}$$

As p increases, $e_{2p}(P', Q)$ approaches the L_∞ norm of $p'(x) - q$ which is numerically equal to the maximum of the function. Therefore, solving the problem

$$\min_x e_{2p}(P', Q) = ||p'(x) - q||_{2p} \quad (8)$$

with a sufficiently large p amounts to minimizing the maximum error between symbols P' and Q. If we let

$$r_{i1} = \begin{bmatrix} p_{i1} & -p_{i2}, 1, 0 \end{bmatrix}^T$$

$$r_{i2} = \begin{bmatrix} p_{i2}, p_{i1}, 0, 1 \end{bmatrix}^T$$

$$q_i = \begin{bmatrix} q_{i1} \\ q_{i2} \end{bmatrix}$$

then the objective function in Eq. (8) can be expressed as

$$e_{2p}(x) = \left\{ \sum_{i=1}^{n} [(r_{i1}^T x - q_{i1})^{2p} + (r_{i2}^T x - q_{i2})^{2p}] \right\}^{1/2p} \quad (9a)$$

The gradient and Hessian of $e_{2p}(x)$ can be evaluated as

$$\nabla e_{2p}(x) = \frac{1}{e_{2p}^{2p-1}(x)} \sum_{i=1}^{n} [(r_{i1}^T x - q_{i1})^{2p-1} + (r_{i2}^T x - q_{i2})^{2p-1}] (9b)$$

$$\nabla^2 e_{2p}(x) = \frac{(2p-1)}{e_{2p}^{2p-1}(x)} \sum_{i=1}^{n} [(r_{i1}^T x - q_{i1})^{2p-2} r_{i1} r_{i1}^T + (r_{i2}^T x - q_{i2})^{2p-2} r_{i2} r_{i2}^T] -$$

$$- \frac{(2p-1)}{e_{2p}(x)} \nabla e_{2p}(x) \nabla^T e_{2p}(x) \quad (9c)$$

It can be shown that the Hessian $\nabla^2 e_{2p}(x)$ in Eq. (9c) is positive semidefinite for any $x \in R^4$ and, therefore, the objective function $e_{2p}(x)$ is globally convex. Since the Hessian of $e_{2p}(x)$ is a 4x4 positive semidefinite matrix and is available in closed form, the Newton algorithm with the Hessian matrix H_k is an appropriate algorithm for the solution of the problem in Eq. (8). If the power

$2p$ involved in the optimization problem is a power of 2, i.e., $2p = 2^K$, then the problem at hand can be solved by first solving the problem for the case $p = 1$ using Eq. (7). The minimizer so obtained can then be used as the initial point to minimize the objective function for $p = 2$. This procedure is then repeated for $p = 4, 8, 16, ...$ until two successive optimizations give the same maximum error to within a prescribed tolerance.

7 Handwritten character recognition

The code which you can find in the last section of the paper is used for recognition of handwritten characters. The recognition is done by storing the coordinates of "standard" characters in the database. Each character in the database can be represented by point pattern from Eq.(1) with $n = 20$. The number here is just for our convenience. You can choose the number to be anything you want. However, note that the higher is the number of points, the larger is precision of final output. You can find in Figure 1 the point pattern for the number 5. Lets denote it by P_5.

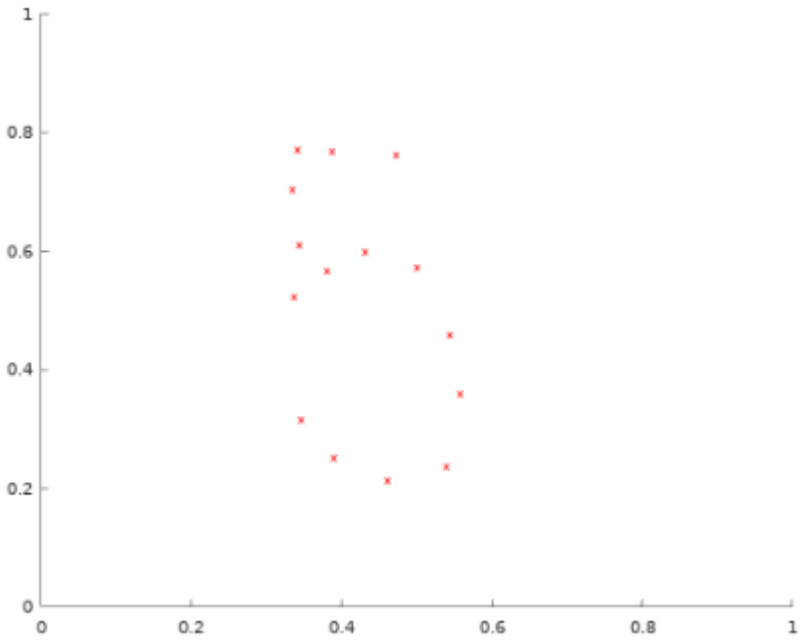


Figure 1: Sample points in pattern P_5

In Figure 2 there is the pattern which should be recognized.

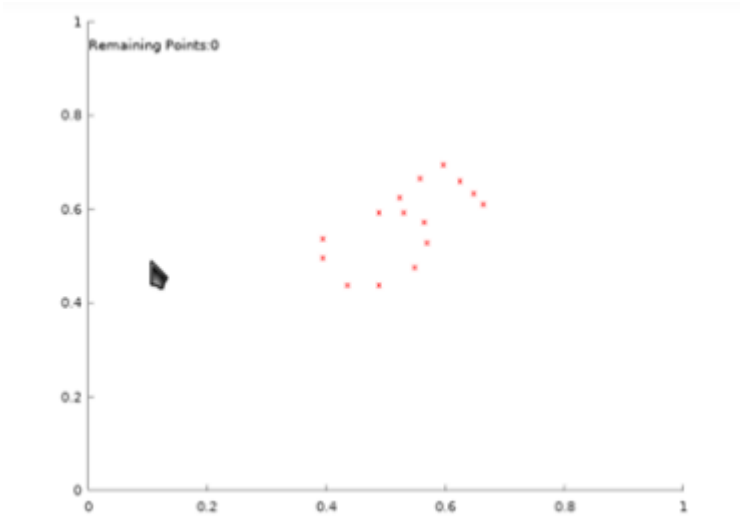


Figure 2: A number to be recognized

You can see that it is a little bit rotated and its size is larger compared to the corresponding one in the database. In order to recognize the point pattern, first of all we should represent it by a point pattern Q with n = 20 (the same structure of patterns in the database). The dissimilarity between pattern P_5 and pattern Q is measured in terms of $e(P_5, Q)$ in Eq. (5). Next steps of the algorithm are exactly the same as described during previous sections.

So the result is the right recognition of the handwritten character

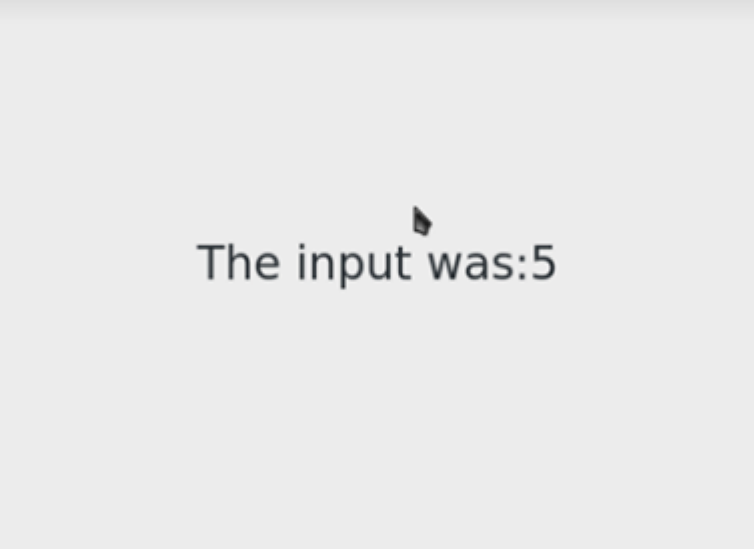
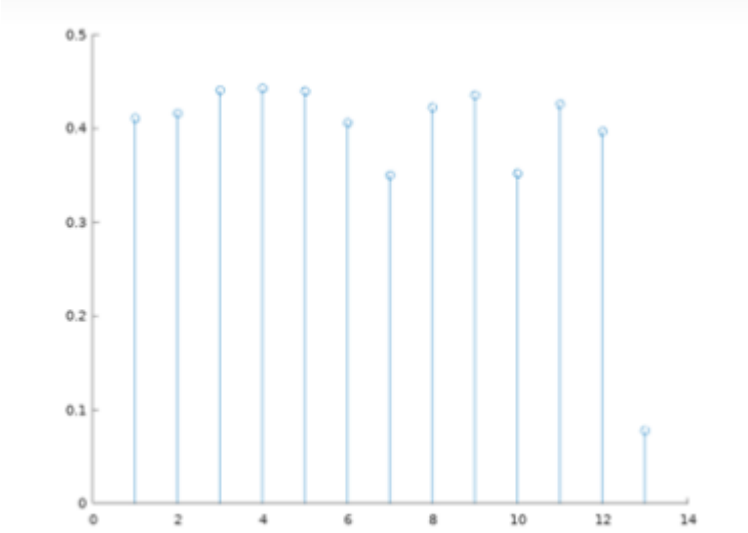


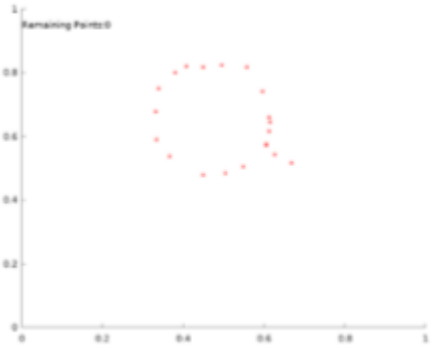
Figure 3: Result

Figure 4 represents the error between given input and each dataset



8 Implementation

Attached to the paper You can find the implementation code.The practical part of the project consists of 4 files: 2 example and 2 main files. ppm.m - file implements ppm function that does point pattern matching algorithm, mentioned above, which calculates error of any two datasets that have same dimensions with x, y coordinates, representing rows. The function returns error of given datasets. dataManager.m - file implements dataManager function that gets input from user and passes it and given dataset to ppm.m file. The dataset is given as argument by cell object that has any number of dataset. These datasets are given to ppm one by one together with input. letters.m - has initial cell with letter datasets, which are given to dataManager for further processing. After getting results, it displays the predicted letter numbers.m - has initial cell with number datasets, which are given to dataManager for further processing. After getting results, it displays the predicted number. letter.m and numbers.m are given as example how dataManager.m and ppm.m can be used. They are not mandatory!



8.1 dataManage.m

```
## This program is free software; you can
## redistribute it and/or modify it
## under the terms of the GNU General Public
## License as published by
## the Free Software Foundation; either
##  version 3 of the License, or
## (at your option) any later version.
##
## This program is distributed in the hope
##  that it will be useful,
## but WITHOUT ANY WARRANTY; without even
##  the implied warranty of
## MERCHANTABILITY or FITNESS FOR A
## PARTICULAR PURPOSE.  See the
## GNU General Public License for
## more details.
##
## You should have received a copy of
##  the GNU General Public License
## along with this program.  If not,
##  see <http://www.gnu.org/licenses/>.

## -*- About Function -*-
## This function gets P cell of matrix of
##  datapoints and takes input from user
## in order to compare it with items in P
##  with ppm( Point-Pattern Matching)
## algorithm.
```

```
## @deftypefn {} {@var{res} = array} dataManager
## (@var{P}, @var{dim}, @var{prec}, @var{vbs})
##    Input Arguments:
##      P -> cell of matrix of datapoints
## to be compared with.
##          The input data will be compared
##  with each element of P.
##      dim -> Specifies the number of
## datapoints for each input and for P's
##          datapoints
##          default: 15
##      prec(optional) -> rounds the datapoints'
## coordinates. E.g. if prec==10, the
##          algorithm rounds till nearset tenths
##          default: 100
##      vbs(optional) -> displays errors of each
## dataset in the same order as it
##          was given with stem plot.
##          default: false
##
```

```
##    Returns:
##      res -> array containing errors
## @seealso{}
## @end deftypefn

## Created: 2018-05-21

function [res] = dataManager (P, dim=15, prec=100, vbs=false)

  %Get user input
  Q = [];
  h2 = text(0, 0.95, strcat('Remaining Points: ', int2str(dim)));
  for pts = dim:-1:1
    [X,Y] = ginput(1);
    h1 = text(X,Y,'*', ...
                    'HorizontalAlignment','center', ...
                    'Color', [1 0 0], ...
                    'FontSize',8);
    delete(h2);
    h2 = text(0, 0.95, strcat('Remaining Points: ', int2str(pts-1)));
    hold on;
    Q = horzcat(Q, vertcat(X, Y));
  endfor

  %Calling point pattern matchin algorithm for each dataset and input
  res = [];
  for k = 1:1:length(P)
    res(end+1) = ppm(round(Q*prec)/prec, round(P{k}*prec)/prec, verbose
  endfor



  %If verbose is true, creates stem plot, displaying all errors
  if vbs
    figure
    stem(res)
    hold on;
  endif

endfunction
```

8.2 ppm.m

```
## This program is free software; you can redistribute
## it and/or modify it
## under the terms of the GNU General Public
##License as published by
## the Free Software Foundation; either version
## 3 of the License, or
## (at your option) any later version.
##
## This program is distributed in the hope that it
## will be useful,
## but WITHOUT ANY WARRANTY; without even the implied
##warranty of
## MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
##  See the
## GNU General Public License for more details.
##
## You should have received a copy of the
##GNU General Public License
## along with this program.  If not,
##see <http://www.gnu.org/licenses/>.

## --About Function--
## The function takes two mandatory arguments
## that specify two datasets with
## same number of points and compares their
##compatibility with point pattern
## matching algorithm. The resulting error
##is returned.
```

```
## @deftypefn {} {@var{error} = number} ppm (@var{Q},
##@var{P}, @var{verbose})
##Input Arguments:
##Q -> matrix with 2 rows that specify x and
##y coordinates of each point.
##Number of columns depends on number of datapoints
##that describe the
## object. Specifies the first matrix to be compared.
## P -> matrix with 2 rows that specify x and y
##coordinates of each point.
## Number of columns depends on number of
##datapoints that describe the
## object. Specifies the second matrix to
## be compared.
## erbose (optional) -> if true prints error
## default: false
##
```

```
function [error] = ppm (Q, P, verbose=false)

  R = [];
  R_T = [];
  Sum_R = zeros(2,2);
  Sum_R_T = zeros(2,2);
  Sum_RRT = zeros(2,2);
  I_2 = eye(2);
  b = zeros(4,1);

  for i = 1:length(P)
    R = horzcat(R, [P(1,i) -P(2,i); P(2,i) P(1,i)]);
    R_T = horzcat(R_T, transpose([P(1,i) -P(2,i); P(2,i) P(1,i)]));
  end

  for i= 1:length(P)
    R_i = [R(1, 2*i-1) R(1, 2*i); R(2, 2*i-1) R(2, 2*i)];
    R_T_i = [R_T(1, 2*i-1) R_T(1, 2*i); R_T(2, 2*i-1) R_T(2, 2*i)];

    Sum_RRT = Sum_RRT + R_T_i*R_i;

    Sum_R(1,1) = Sum_R(1,1) + R_i(1, 1);
    Sum_R(1,2) = Sum_R(1,2) + R_i(1, 2);
    Sum_R(2,1) = Sum_R(2,1) + R_i(2, 1);
    Sum_R(2,2) = Sum_R(2,2) + R_i(2, 2);

    Sum_R_T(1,1) = Sum_R_T(1,1) + R_T_i(1, 1);
    Sum_R_T(1,2) = Sum_R_T(1,2) + R_T_i(1, 2);
    Sum_R_T(2,1) = Sum_R_T(2,1) + R_T_i(2, 1);
    Sum_R_T(2,2) = Sum_R_T(2,2) + R_T_i(2, 2);
```

```
  %  temp = transpose(horzcat(R_i, I_2));
    b = b + transpose(horzcat(R_i, I_2))*Q(:, i);
  end


  I_2 = length(P)*I_2;

  H = vertcat(horzcat(Sum_RRT,Sum_R_T), horzcat(Sum_R, I_2));

  x_opt = inv(H)*b;

  P_est = [];
  for i = 1:length(P)
    rot = [x_opt(1) -1*x_opt(2); x_opt(2) x_opt(1)]*P(:, i);
    P_est = horzcat(P_est, rot+[x_opt(3);x_opt(4)]);
  end

  error = norm(P_est-Q,'fro');
  if verbose
    disp(error);
  endif
endfunction
```

9 Conclusion

The project represented the point pattern matching problem which is studied all over the world and has its applications in various fields.The project consists of theoretical and practical parts.First part of the project gave some introduction about this problem and then more details were represented.The last part included the code of implementation.

References

Practical Optimization Algorithms and Engineering applications.
[3, Ch 9.2], http://ieeexplore.ieee.org/document/400572/