CMSC 828D: Fundamentals of Computer Vision
Homework 4

Instructors: Larry Davis, Ramani Duraiswami,
Daniel DeMenthon, and Yiannis Aloimonos

Solution based on homework submitted by Haiying Liu

1. Show by direct calculation for the 2 by 2 matrices A and B that AB and BA have the same eigenvalues. Does this result hold in general (i.e. for A and B *N* by *N* matrices)?

Solution: Assume $\mathbf{A}=\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$, $\mathbf{B}=\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$. Thus

$\mathbf{AB}=\begin{bmatrix} a_{11}b_{11}+a_{12}b_{21} & a_{11}b_{12}+a_{12}b_{22} \\ a_{21}b_{11}+a_{22}b_{21} & a_{21}b_{12}+a_{22}b_{22} \end{bmatrix}$. Solve $|l\mathbf{I}-\mathbf{AB}|=0$, i.e.

$$\begin{vmatrix} l-(a_{11}b_{11}+a_{12}b_{21}) & -(a_{11}b_{12}+a_{12}b_{22}) \\ -(a_{21}b_{11}+a_{22}b_{21}) & l-(a_{21}b_{12}+a_{22}b_{22}) \end{vmatrix}$$

$$=(l-(a_{11}b_{11}+a_{12}b_{21}))(l-(a_{21}b_{12}+a_{22}b_{22}))-(a_{11}b_{12}+a_{12}b_{22})(a_{21}b_{11}+a_{22}b_{21})$$

$$=l^2-(a_{11}b_{11}+a_{12}b_{21}+a_{21}b_{12}+a_{22}b_{22})l+a_{11}b_{11}a_{21}b_{12}+a_{11}b_{11}a_{22}b_{22}+a_{12}b_{21}a_{21}b_{12}+a_{12}b_{21}a_{22}b_{22}-$$
$$a_{11}b_{12}a_{21}b_{11}-a_{11}b_{12}a_{22}b_{21}-a_{12}b_{22}a_{21}b_{11}-a_{12}b_{22}a_{22}b_{21}$$

$$=l^2-(a_{11}b_{11}+a_{12}b_{21}+a_{21}b_{12}+a_{22}b_{22})l+(a_{11}a_{22}b_{11}b_{22}+a_{12}a_{21}b_{12}b_{21}-a_{11}a_{22}b_{12}b_{21}-a_{12}b_{22}a_{21}b_{11})$$

$$=0$$

Similarly, $\mathbf{BA}=\begin{bmatrix} b_{11}a_{11}+b_{12}a_{21} & b_{11}a_{12}+b_{12}a_{22} \\ b_{21}a_{11}+b_{22}a_{21} & b_{21}a_{12}+b_{22}a_{22} \end{bmatrix}$. Solve $|l\mathbf{I}-\mathbf{AB}|=0$, i.e.

$$\begin{vmatrix} l-(b_{11}a_{11}+b_{12}a_{21}) & -(b_{11}a_{12}+b_{12}a_{22}) \\ -(b_{21}a_{11}+b_{22}a_{21}) & l-(b_{21}a_{12}+b_{22}a_{22}) \end{vmatrix}$$

$$=(l-(b_{11}a_{11}+b_{12}a_{21}))(l-(b_{21}a_{12}+b_{22}a_{22}))-(b_{11}a_{12}+b_{12}a_{22})(b_{21}a_{11}+b_{22}a_{21})$$

$$=l^2-(b_{11}a_{11}+b_{12}a_{21}+b_{21}a_{12}+b_{22}a_{22})l+b_{11}a_{11}b_{21}a_{12}+b_{11}a_{11}b_{22}a_{22}+b_{12}a_{21}b_{21}a_{12}+b_{12}a_{21}b_{22}a_{22}-$$
$$b_{11}a_{12}b_{21}a_{11}-b_{11}a_{12}b_{22}a_{21}-b_{12}a_{22}b_{21}a_{11}-b_{12}a_{22}b_{22}a_{21}$$

$$=l^2-(a_{11}b_{11}+a_{12}b_{21}+a_{21}b_{12}+a_{22}b_{22})l+(a_{11}a_{22}b_{11}b_{22}+a_{12}a_{21}b_{12}b_{21}-a_{11}a_{22}b_{12}b_{21}-a_{12}b_{22}a_{21}b_{11})$$

$$=0$$

The above two equations are the same. Therefore, they have the same eigenvalues.

The result holds in the general case. This can be proved as follows. Rewrite $[l\mathbf{I}-\mathbf{AB}]$ in the form $[l\mathbf{I}-\mathbf{AB}]=\begin{bmatrix} I & \mathbf{B} \\ \mathbf{A} & l\mathbf{I} \end{bmatrix}$. Notice that $|\mathbf{AB}|=|\mathbf{A}|\cdot|\mathbf{B}|$ for any square matrix. It can be verify that

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -l(l\mathbf{I}-\mathbf{AB})^{-1}\mathbf{A} & l(l\mathbf{I}-\mathbf{AB})^{-1} \end{bmatrix} \cdot \begin{bmatrix} I & \mathbf{B} \\ \mathbf{A} & l\mathbf{I} \end{bmatrix} = \begin{bmatrix} l\mathbf{I} & \mathbf{B} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

Thus, we have

$$\left| \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -l(l\mathbf{I}-\mathbf{AB})^{-1}\mathbf{A} & l(l\mathbf{I}-\mathbf{AB})^{-1} \end{bmatrix} \cdot \begin{bmatrix} I & \mathbf{B} \\ \mathbf{A} & l\mathbf{I} \end{bmatrix} \right| = \begin{vmatrix} l\mathbf{I} & \mathbf{B} \\ \mathbf{0} & \mathbf{I} \end{vmatrix}$$

$$\Rightarrow \begin{vmatrix} \mathbf{I} & \mathbf{0} \\ -l(l\mathbf{I}-\mathbf{AB})^{-1}\mathbf{A} & l(l\mathbf{I}-\mathbf{AB})^{-1} \end{vmatrix} \cdot \begin{vmatrix} I & \mathbf{B} \\ \mathbf{A} & l\mathbf{I} \end{vmatrix} = \begin{vmatrix} l\mathbf{I} & \mathbf{B} \\ \mathbf{0} & \mathbf{I} \end{vmatrix}$$

$$\Rightarrow \left| l(l\mathbf{I}-\mathbf{AB})^{-1} \right| \cdot \begin{vmatrix} I & \mathbf{B} \\ \mathbf{A} & l\mathbf{I} \end{vmatrix} = \left| l\mathbf{I} \right|$$

$$\Rightarrow \frac{l}{\left| l\mathbf{I}-\mathbf{AB} \right|} \cdot \begin{vmatrix} I & \mathbf{B} \\ \mathbf{A} & l\mathbf{I} \end{vmatrix} = \left| l\mathbf{I} \right|$$

Equation 1

Similarly, it can also be verified that

$$\begin{bmatrix} l(l\mathbf{I}-\mathbf{BA})^{-1} & \mathbf{I}-l(l\mathbf{I}-\mathbf{BA})^{-1}\mathbf{A}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} I & \mathbf{B} \\ \mathbf{A} & l\mathbf{I} \end{bmatrix} = \begin{bmatrix} l\mathbf{I} & \mathbf{0} \\ \mathbf{A} & \mathbf{I} \end{bmatrix}$$

Thus, we have

$$\left| \begin{bmatrix} l(l\mathbf{I}-\mathbf{BA})^{-1} & \mathbf{I}-l(l\mathbf{I}-\mathbf{BA})^{-1}\mathbf{A}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} I & \mathbf{B} \\ \mathbf{A} & l\mathbf{I} \end{bmatrix} \right| = \begin{vmatrix} l\mathbf{I} & \mathbf{0} \\ \mathbf{A} & \mathbf{I} \end{vmatrix}$$

$$\Rightarrow \begin{vmatrix} l(l\mathbf{I}-\mathbf{BA})^{-1} & \mathbf{I}-l(l\mathbf{I}-\mathbf{BA})^{-1}\mathbf{A}^{-1} \\ \mathbf{0} & \mathbf{I} \end{vmatrix} \cdot \begin{vmatrix} I & \mathbf{B} \\ \mathbf{A} & l\mathbf{I} \end{vmatrix} = \begin{vmatrix} l\mathbf{I} & \mathbf{0} \\ \mathbf{A} & \mathbf{I} \end{vmatrix}$$

$$\Rightarrow \left| l(l\mathbf{I}-\mathbf{BA})^{-1} \right| \cdot \begin{vmatrix} I & \mathbf{B} \\ \mathbf{A} & l\mathbf{I} \end{vmatrix} = \left| l\mathbf{I} \right|$$

$$\Rightarrow \frac{l}{\left| l\mathbf{I}-\mathbf{BA} \right|} \cdot \begin{vmatrix} I & \mathbf{B} \\ \mathbf{A} & l\mathbf{I} \end{vmatrix} = \left| l\mathbf{I} \right|$$

Equation 2

Compare [Equation 2] with [Equation 1], we have $\left| l\mathbf{I}-\mathbf{AB} \right| = \left| l\mathbf{I}-\mathbf{BA} \right|$, which implies that if $\left| l\mathbf{I}-\mathbf{AB} \right| = 0$ then $\left| l\mathbf{I}-\mathbf{BA} \right| = 0$, i.e. $\mathbf{AB}$ and $\mathbf{BA}$ have the same eigenvalues.

2. For the 2 by 2 case obtain the relationship between the eigenvalues of A and of A^2 does the result you obtained hold for A and A^n in general (i.e. for general n and for A a *N* by *N* matrix)?

Solution: Assume that l is an eigenvalue of matrix $\mathbf{A}$, and l' is an eigenvalue of matrix $\mathbf{A}^2$. According to the definition of eigenvalue, we have $\mathbf{AX} = l\mathbf{X}$ and $\mathbf{A}^2\mathbf{X} = l'\mathbf{X}$. Since l is scalar, we have

$$\mathbf{A}^2\mathbf{X} = \mathbf{A}\cdot\mathbf{A}\mathbf{X} = \mathbf{A}\cdot l\mathbf{X} = l\cdot\mathbf{A}\mathbf{X} = l\cdot l\mathbf{X} = l^2\mathbf{X} = l'\mathbf{X}$$

Thus, the relationship between l and l' is $l' = l^2$.

This result holds for $\mathbf{A}$ and $\mathbf{A}^n$ in general. This can be proved by induction. The base part is proved as above. Now assume that l^n is an eigenvalue of matrix $\mathbf{A}^n$, then:

$$\mathbf{A}^{n+1}\mathbf{X} = \mathbf{A}\cdot\mathbf{A}^n\mathbf{X} = \mathbf{A}\cdot l^n\mathbf{X} = l^n\cdot\mathbf{A}\mathbf{X} = l^n\cdot l\mathbf{X} = l^{n+1}\mathbf{X}$$

Thus, l^{n+1} is an eigenvalue of $\mathbf{A}^{n+1}$.

Therefore, if l is an eigenvalue of matrix $\mathbf{A}$, l^n is an eigenvalue of matrix $\mathbf{A}^n$.

3. **Read chapter 2.6 of Numerical Recipes. To those who do not have the book, you can find a copy on the web at http://lib-www.lanl.gov/numerical/bookcpdf/c2-6.pdf. After you read this, solve the following problem. The solution of a linear equation Ax =b is often represented as x =A^{-1}b. As discussed in class and in numerical recipes, the equation can also be solved using the singular value decomposition using the expression x =A^+ b ,where A^+ =(V S^{-1} U^t),where S is the diagonal matrix of singular values. In case of zero or near zero, singular values the corresponding elements of S are set to zero. Write a Matlab function that computes the pseudoinverse of a given matrix after computing its SVD. (Do not use native Matlab functions to compute the pseudo inverse. You may use the Matlab svd function.) Find the singular value decomposition and the pseudoinverse of the following matrices:** $A\ (1\quad 1\quad 1\quad 1)$, $B\begin{pmatrix}0 & 1 & 0\\ 1 & 0 & 0\end{pmatrix}$, *and* $C\begin{pmatrix}1 & 1\\ 0 & 0\end{pmatrix}$

Solution: See appendix for the Matlab script and results.

4. **Verify the statement that the cross-ratio of four points on a line remains invariant when the points are mapped to a different configuration by a projective transformation (i.e. a projectivity,or homography).**

Solution: We use homogeneous coordinates to express 1D point. Assume the four points are $A=\begin{bmatrix}x_{A1}\\ x_{A2}\end{bmatrix}$, $B=\begin{bmatrix}x_{B1}\\ x_{B2}\end{bmatrix}$, $C=\begin{bmatrix}x_{C1}\\ x_{C2}\end{bmatrix}$, $D=\begin{bmatrix}x_{D1}\\ x_{D2}\end{bmatrix}$. According to the definition of cross-ratio,

$$\text{cross}(A,B,C,D) = \frac{|AB|}{|AD|}\div\frac{|CB|}{|CD|}.$$

Notice that $|AB| = \det[A\quad B] = \det\begin{bmatrix}x_{A1} & x_{B1}\\ x_{A2} & x_{B2}\end{bmatrix}$, we have

$$\text{cross}(A,B,C,D) = \frac{\det(A\quad B)}{\det(A\quad D)}\div\frac{\det(C\quad B)}{\det(C\quad D)},$$

Assume the projective transformation is H. Then the four points after the transformation are $A' = H\begin{bmatrix} x_{A1} \\ x_{A2} \end{bmatrix}$, $B' = H\begin{bmatrix} x_{B1} \\ x_{B2} \end{bmatrix}$, $C' = H\begin{bmatrix} x_{C1} \\ x_{C2} \end{bmatrix}$, $D' = H\begin{bmatrix} x_{D1} \\ x_{D2} \end{bmatrix}$. The cross-ratio of the new points is:

$$\begin{aligned}
\text{cross}(A', B', C', D') &= \frac{|A'B'|}{|A'D'|} \div \frac{|C'B'|}{|C'D'|} = \frac{\det(H[A \quad B])}{\det(H[A \quad D])} \div \frac{\det(H[C \quad B])}{\det(H[C \quad D])} \\
&= \frac{\det(H) \cdot \det([A \quad B])}{\det(H) \cdot \det([A \quad D])} \div \frac{\det(H) \cdot \det([C \quad B])}{\det(H) \cdot \det([C \quad D])} \\
&= \frac{\det(A \quad B)}{\det(A \quad D)} \div \frac{\det(C \quad B)}{\det(C \quad D)} \\
&= \text{cross}(A, B, C, D)
\end{aligned}$$

i.e. the cross-ratio remains invariant. The above prove can be easily extended to 3D.

1. **A robot vehicle is equipped with a camera. The camera image is a rectangle of height 500 pixels and width 600 pixels. The focal length is 690 pixels. The center of projection *C* of the camera is 3 m above the ground. The optical axis of the camera makes a 20 degree angle with the ground, so that the camera looks slightly downward toward the road. Large squares of size 4 m have been painted everywhere along the median lines of roads to facilitate automatic vehicle navigation. Two of the sides of the squares are parallel to the road edges. The vehicle is on a flat road where it sees only two squares. The vehicle faces the first square such that the optical axis of the camera passes through the center of the square, and is perpendicular to two sides of the square. Because the road turns, the next square is at an angle with the first square. The centers of the squares are on a circle of radius 100 meters (which is also the radius of the road turn), and the arc between the two centers is 10 degrees.**

The 4 coordinate systems are selected as follow [Figure 1]: coordinate 0 originated at the center of projection, coordinate 1 originated at the center of the first square, coordinate 2 originated at the center of the second square, and the image coordinate originated at the center of the image plane. All coordinates use right-hand-rule. The x- and z- axis of coordinate 1 and 2 are in the road plane and parallel to the two sides of the squares respectively. The relationships between these coordinates can be easily established by simple rotation and translation.

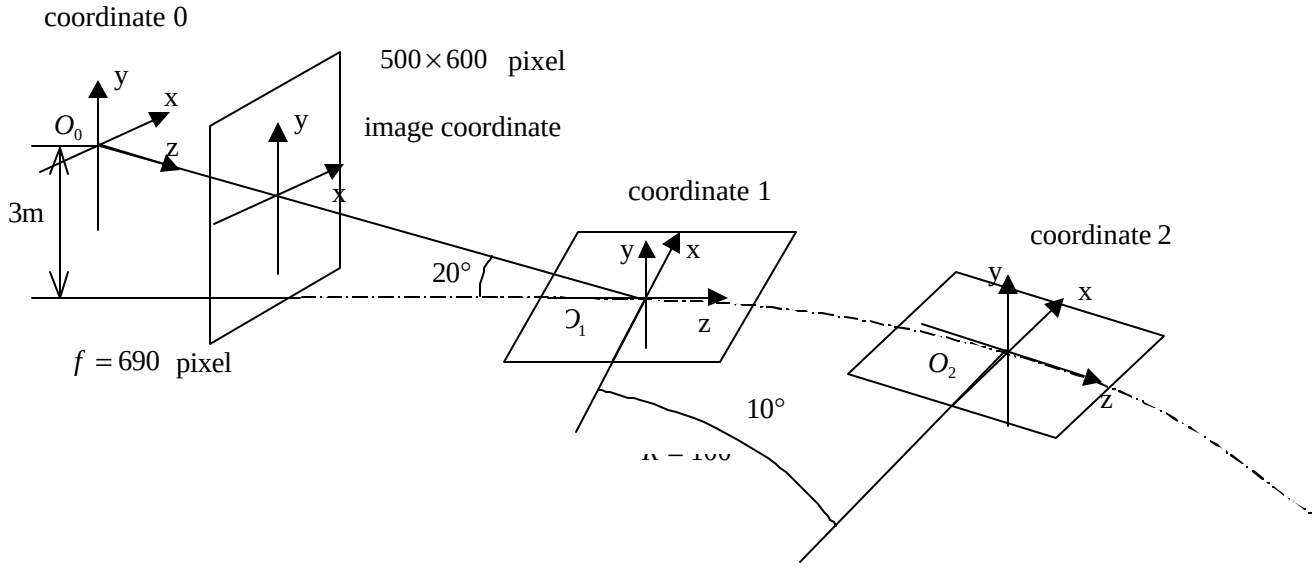


Figure 1: Robot navigation problem.

a. Compute the pixel positions of the images of the corners of the two squares.

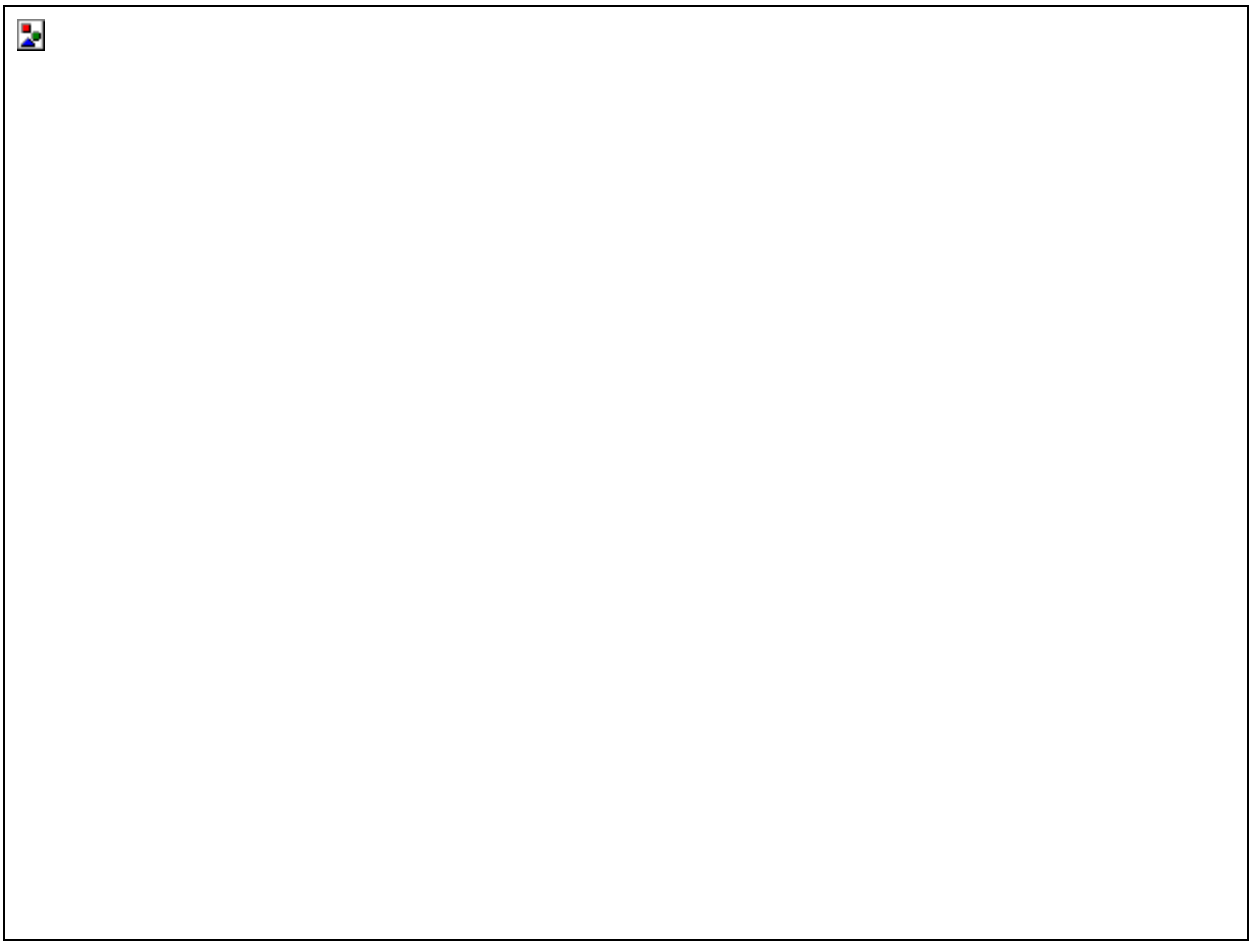
Solution: The basic idea of this solution is establishing the rotation-translation relationship between

- coordinate 2 and coordinate 1: rotate around axis y by $-10°$ and translate from O_1 to O_2;
- coordinate 1 and coordinate 0: rotate around axis x by $20°$ and translate from O_0 to O_1;
- coordinate 0 and image coordinate: translation and scaling.

See the Matlab script for detailed definition of the transformation matrix for the above rotation and translation. The Matlab script and pixel positions of the images of the corners of the two squares are listed in appendix.

b. Make a drawing of the images of the two squares and of the rectangular frame of the image using Matlab.

Solution: The Matlab script is listed in appendix. The image is as follows:



c. To navigate, the robot computes a projectivity (i.e. homography) that transforms the images of the squares back to their actual geometry on the road,in order to compute how the road turns. To do that, it does not use the camera focal length and the camera position with respect to the road. Instead, it uses its knowledge that the squares it sees are always of size 4 m, and computes a projectivity matrix using the four corners of the closest square it sees. Using the images of the square corners found in (a),compute this matrix using Matlab.

Solution: Define the transformation matrix as $\mathbf{H} = \left[h_{ij}\right]_{3\times 3}$. Assume image coordinate is $[u, v, 1]^T$, and correspondent space coordinate is $[x', z', s]$. Note that the square is on a flat plane. We then have the equation:

$$\begin{bmatrix} x' \\ z' \\ s \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Notice that $x = \frac{x'}{s}, z = \frac{z'}{s}$, we have:

$$x = \frac{h_{11}u + h_{12}v + h_{13}}{h_{31}u + h_{32}v + 1} \Rightarrow h_{11}u + h_{12}v + h_{13} - h_{31}ux - h_{32}vx = x$$

$$y = \frac{h_{21}u + h_{22}v + h_{23}}{h_{31}u + h_{32}v + 1} \Rightarrow h_{21}u + h_{22}v + h_{23} - h_{31}uz - h_{32}vz = z$$

Since we have 4 correspondent points, we have 8 equations, which can be expressed in the form of $\mathbf{Ah} = \mathbf{b}$:

$$\begin{bmatrix} & & & & \cdots & & & \\ u_i & v_i & 1 & 0 & 0 & 0 & u_i x_i & v_i x_i \\ 0 & 0 & 0 & u_i & v_i & 1 & u_i z_i & v_i z_i \\ & & & & \cdots & & & \end{bmatrix} \cdot \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ \vdots \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} x_1 \\ z_1 \\ \vdots \\ \vdots \\ \vdots \\ x_4 \\ z_4 \end{bmatrix}.$$

Therefore, the $\mathbf{H}$ can be easily solved by $\mathbf{A}^{-1}\mathbf{b}$. By using the image corners from (a) part, we can also calculate the new road coordinate of the 4 corners of the square 2. The Matlab script and result ($\mathbf{H}$) for this sub-problem are listed in appendix.

d. Draw the two reconstructed squares. Compute the radius of the road turn.

Solution: The reconstructed squares are show as follow [Figure 3]. Once we get the geometry relationship between square 1 and square 2, we have the their position information on the road plane. Then the arc q between the two centers of the squares can be computed by two side of the two squares respectively. Combining with the distance D between the two centers, the radius is given by $\frac{D}{2\sin(q/2)}$. This is illustrated in [Figure 2]. See details in the Matlab script in appendix.

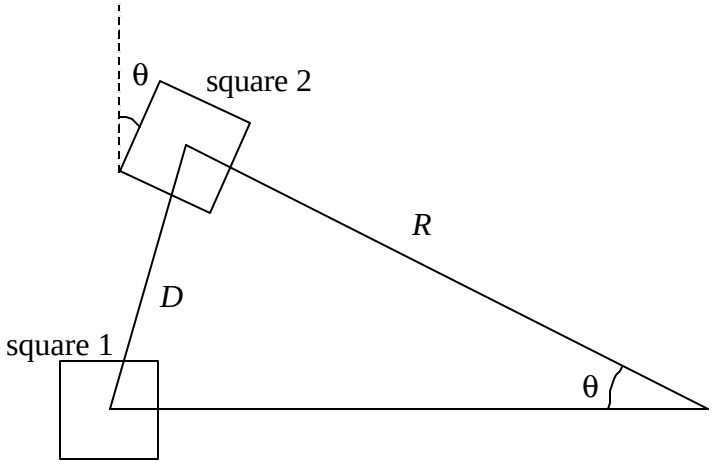


Figure 2: Geometry relationship of the two squares.

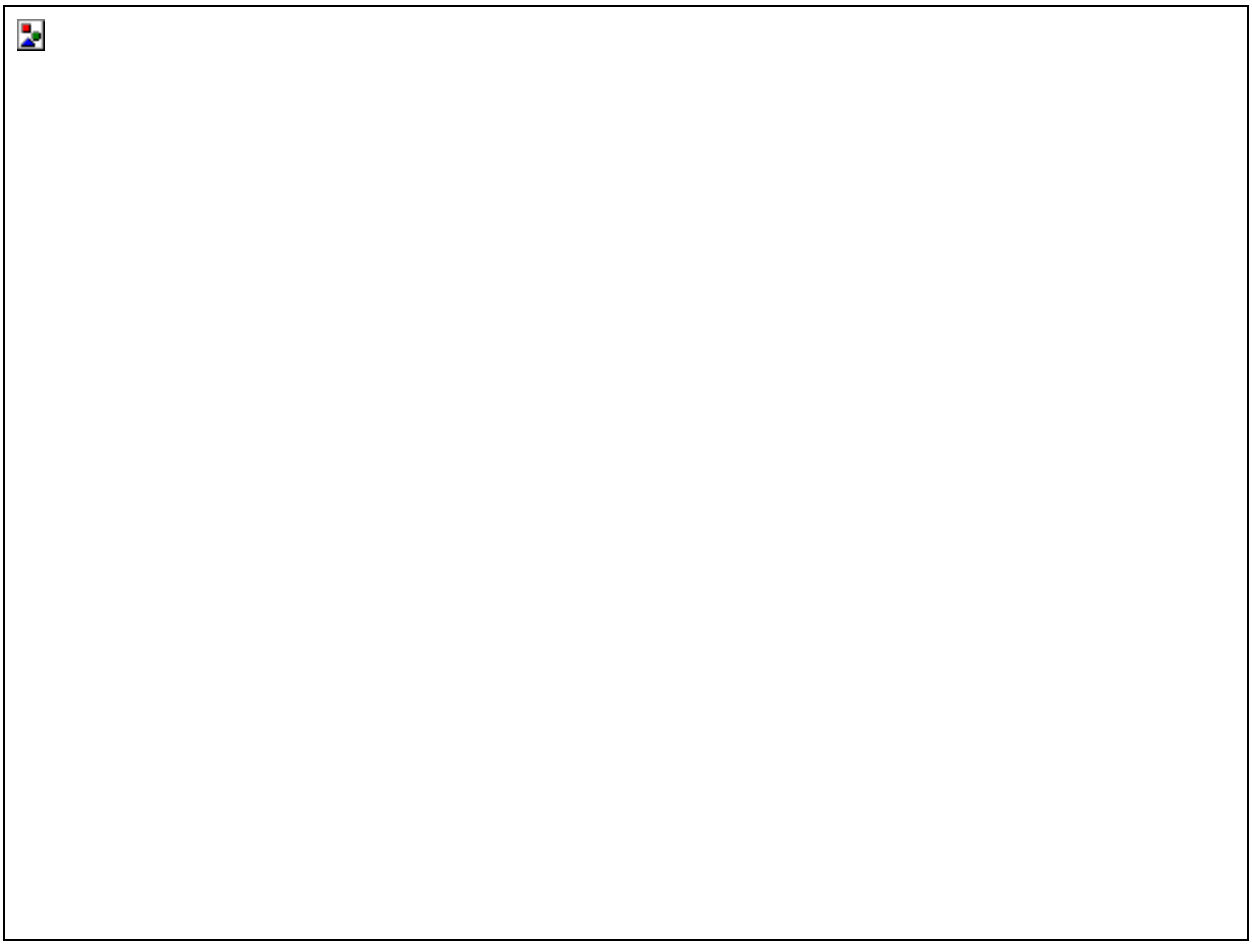


Figure 3: Geometry of the two squares.

Appendix:

- hw4_3.m:

```
function hw4_3
% Syntax: hw4_3
%
% Description: Test my_pinv
%
% CMSC 828D: Fundamentals of Computer Vision
%                       Homework4
%
% Instructors : Larry Davis, Ramani Duraiswami,
%               Daniel DeMenthon, and Yiannis Aloimonos
% Student     : Haiying Liu
%
% Date        : Sept. 23, 2000
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dbstop if error

msg = nargchk(0, 0, nargin);
if (~isempty(msg))
  error(strcat('ERROR:', msg));
end

clear msg;

% Prepare record result in file
filename = 'hw4_3.txt';
if exist(filename, 'file')
  delete(filename);
end

diary off;
diary(filename);

%=========================================================================

% Test A
disp('-----------------------------------');
A = [1 1 1 1]
disp(':::::::::::::::::::::::::::::::');
disp(': [U, S, V] = svd(A) returns: :');
disp(':::::::::::::::::::::::::::::::');
[U, S, V] = svd(A)
disp(':::::::::::::::::::::::');
disp(': my_pinv(A) returns: :');
disp(':::::::::::::::::::::::');
my_pinv(A)

% Test B
disp('---------');
B = [0 1 0; 1 0 0]
disp(':::::::::::::::::::::::::::::::');
```

```
disp(': [U, S, V] = svd(B) returns: :');
disp('::::::::::::::::::::::::::::::');
[U, S, V] = svd(B)
disp('::::::::::::::::::::::');
disp(': my_pinv(B) returns: :');
disp('::::::::::::::::::::::');
my_pinv(B)

% Test C
disp('---------');
C = [1 1; 0 0]
disp('::::::::::::::::::::::::::::::');
disp(': [U, S, V] = svd(C) returns: :');
disp('::::::::::::::::::::::::::::::');
[U, S, V] = svd(C)
disp('::::::::::::::::::::::');
disp(': my_pinv(C) returns: :');
disp('::::::::::::::::::::::');
my_pinv(C)

% Stop recording
diary off;
```

- my_pinv.m:

```
function result = my_pinv(M)
% Syntax: result = my_pinv(M)
%
% Description: Compute the pseudoinverse of a given matrix by SVD
%
% CMSC 828D: Fundamentals of Computer Vision
%                       Homework4
%
% Instructors : Larry Davis, Ramani Duraiswami,
%               Daniel DeMenthon, and Yiannis Aloimonos
% Student     : Haiying Liu
%
% Date        : Sept. 23, 2000
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dbstop if error

msg = nargchk(1, 1, nargin);
if (~isempty(msg))
  error(strcat('ERROR:', msg));
end

clear msg;

%=========================================================================

% Do the SVD
[U, S, V] = svd(M);
```

```
% Compute pseudoinverse matrix of M
nRow     = size(S, 1);
nCol     = size(S, 2);
S_inv    = zeros(nCol, nRow);
nElement = min(nRow, nCol);

for index = 1:nElement
  entry = S(index, index);
  if entry ~= 0
    S_inv(index, index) = 1 ./ entry;
  end
end

result = V * S_inv * U';
```

- Result from my_pinv.m:

```
-----------------------------------

A =

     1     1     1     1

::::::::::::::::::::::::::::::
: [U, S, V] = svd(A) returns: :
::::::::::::::::::::::::::::::

U =

     1


S =

     2     0     0     0


V =

    0.5000   -0.8660         0         0
    0.5000    0.2887   -0.5774   -0.5774
    0.5000    0.2887    0.7887   -0.2113
    0.5000    0.2887   -0.2113    0.7887

:::::::::::::::::::::::
: my_pinv(A) returns: :
:::::::::::::::::::::::

ans =

    0.2500
    0.2500
    0.2500
    0.2500

---------
```

```
B =

     0     1     0
     1     0     0

::::::::::::::::::::::::::::::
: [U, S, V] = svd(B) returns: :
::::::::::::::::::::::::::::::

U =

     0     1
     1     0


S =

     1     0     0
     0     1     0


V =

     1     0     0
     0     1     0
     0     0     1

:::::::::::::::::::::::
: my_pinv(B) returns: :
:::::::::::::::::::::::

ans =

     0     1
     1     0
     0     0

---------

C =

     1     1
     0     0

::::::::::::::::::::::::::::::
: [U, S, V] = svd(C) returns: :
::::::::::::::::::::::::::::::

U =

     1     0
     0     1


S =
```

```
    1.4142         0
         0         0


V =

    0.7071   -0.7071
    0.7071    0.7071

::::::::::::::::::::::
: my_pinv(C) returns: :
::::::::::::::::::::::

ans =

    0.5000         0
    0.5000         0
```

- hw4_5.m:

```
function hw4_5
% Syntax: hw4_5
%
% Description: Robot navigation problem.
%
% CMSC 828D: Fundamentals of Computer Vision
%                       Homework4
%
% Instructors : Larry Davis, Ramani Duraiswami,
%               Daniel DeMenthon, and Yiannis Aloimonos
% Student     : Haiying Liu
%
% Date        : Sep. 23, 2000
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dbstop if error

msg = nargchk(0, 0, nargin);
if (~isempty(msg))
  error(strcat('ERROR:', msg));
end

clear msg;

% Turn on the diary to save the result.
filename = 'hw4_5.txt';
if (exist(filename, 'file'))
  delete(filename);
end

diary off;
eval(['diary ', filename]);
```

```
disp('» hw4_5');

%=========================================================================
% Part a.

% Define corners of square 1 in coordinate 2
squre1_corners = [ ...
    +2 0  2 1; ...
    +2 0 -2 1; ...
    -2 0 -2 1; ...
    -2 0  2 1; ...
  ];

% Define corners of square 2 in coordinate 1
squre2_corners = [ ...
    +2 0  2 1; ...
    +2 0 -2 1; ...
    -2 0 -2 1; ...
    -2 0  2 1; ...
  ];

% Convert square 1 corners to image plane
nCorners1      = size(squre1_corners, 1);
image_corners1 = zeros(nCorners1, 3);

M0toImage      = coord0toImage;
M1to0          = coord1to0;
M1toImage      = M0toImage * M1to0;

for index = 1:nCorners1
  point = squre1_corners(index, :)';
  image_corners1(index, :) = (M1toImage * point)';
  scale = image_corners1(index, 3);
  image_corners1(index, :) = image_corners1(index, :) ./ scale;
end

% Convert square 2 corners to image plane
nCorners2      = size(squre2_corners, 1);
image_corners2 = zeros(nCorners2, 3);

M2to1          = coord2to1;
M2toImage      = M0toImage * M1to0 * M2to1;

for index = 1:nCorners2
  point = squre2_corners(index, :)';
  image_corners2(index, :) = (M2toImage * point)';
  scale = image_corners2(index, 3);
  image_corners2(index, :) = image_corners2(index, :) ./ scale;
end

% Display the image coordinates of corners.
disp(' ');
disp('::::::::::::::::::::::::::::::::::::::::::::');
disp(':: Image coordinate of corners of square 1 ::');
disp('::::::::::::::::::::::::::::::::::::::::::::');
disp(' ');
disp('id   |      x        y');
```

```
disp('-----+----------------------');
for index = 1:nCorners1
  disp([num2str(index), '    |   ', ...
    num2str(image_corners1(index, 1), '%3.4f'), '   ', ...
    num2str(image_corners1(index, 2), '%3.4f')]);
end

disp(' ');
disp('::::::::::::::::::::::::::::::::::::::::::::');
disp(':: Image coordinate of corners of square 2 ::');
disp('::::::::::::::::::::::::::::::::::::::::::::');
disp(' ');
disp('id   |      x        y');
disp('-----+----------------------');
for index = 1:nCorners2
  disp([num2str(index), '    |   ', ...
    num2str(image_corners2(index, 1), '%3.4f'), '   ', ...
    num2str(image_corners2(index, 2), '%3.4f')]);
end

disp(' ');

%=========================================================================
% Part b.

% Draw the two squares in the image plane
image_corners1 = [image_corners1; image_corners1(1, :)];
image_corners2 = [image_corners2; image_corners2(1, :)];

left   = 0;
right  = 600;
up     = 0;
bottom = 500;

figure;
axis ij;
axis equal;
axis([left, right, up, bottom]);
hold on;
xlabel('x');
ylabel('y');
title('Robot vision of the road');

% .. Draw frame
plot([left, right, right, left, left], [up, up, bottom, bottom, up], 'k');

% .. Draw corners for square 1
plot(image_corners1(:, 1), image_corners1(:, 2));
fill(image_corners1(:, 1), image_corners1(:, 2), 'k');

% .. Draw corners for square 2
plot(image_corners2(:, 1), image_corners2(:, 2));
fill(image_corners2(:, 1), image_corners2(:, 2), 'k');

print -djpeg hw4_5_1

%=========================================================================
```

```
% Part c.

% Construct A and b for Ax = b
squre1_corners_xz = [ ...
    +2  2 1; ...
    +2 -2 1; ...
    -2 -2 1; ...
    -2  2 1; ...
  ];

A = zeros(8);
for index = 1:nCorners1
  road_corner  = squre1_corners_xz(index, :);
  x            = road_corner(1);
  z            = road_corner(2);
  image_corner = image_corners1(index, :);
  u            = image_corner(1);
  v            = image_corner(2);
  row1         = [u, v, 1, 0, 0, 0, -u * x, -v * x];
  row2         = [0, 0, 0, u, v, 1, -u * z, -v * z];
  A(2 * index - 1, :) = row1;
  A(2 * index    , :) = row2;
  b(2 * index - 1, :) = x;
  b(2 * index    , :) = z;
end

HH = inv(A) * b;
H  = zeros(3, 3);

H(1, 1) = HH(1);
H(1, 2) = HH(2);
H(1, 3) = HH(3);
H(2, 1) = HH(4);
H(2, 2) = HH(5);
H(2, 3) = HH(6);
H(3, 1) = HH(7);
H(3, 2) = HH(8);
H(3, 3) = 1;

disp(' ');
disp('::::::::::::::::::::::::::::::::::::::::::::::::::::::::');
disp(':: Transformation matrix (from image to geometry) ::');
disp('::::::::::::::::::::::::::::::::::::::::::::::::::::::::');

H

%=========================================================================
% Part d.

% Calculate geometry for square 2.
squre2_corners_xz = zeros(nCorners2, 3);
for index = 1:nCorners2
  image_corner = image_corners2(index, :);
  squre2_corners_xz(index, :)  = (H * image_corner')';
  squre2_corners_xz(index, :)  = ...
    squre2_corners_xz(index, :) ./ squre2_corners_xz(index, 3);
end
```

```
squre1_corners_xz = [squre1_corners_xz; squre1_corners_xz(1, :)];
squre2_corners_xz = [squre2_corners_xz; squre2_corners_xz(1, :)];

% Draw the two squares.
figure
hold on;
axis equal;
xlabel('-x');
ylabel('z');
title('Geometry of the two squares');

plot(-squre1_corners_xz(:, 1), squre1_corners_xz(:, 2), 'k');
fill(-squre1_corners_xz(:, 1), squre1_corners_xz(:, 2), 'k');

plot(-squre2_corners_xz(:, 1), squre2_corners_xz(:, 2), 'k');
fill(-squre2_corners_xz(:, 1), squre2_corners_xz(:, 2), 'k');

print -djpeg hw4_5_2;

% Calculate radius.
center1_x = (squre1_corners_xz(1, 1) + squre1_corners_xz(3, 1)) ./ 2;
center1_z = (squre1_corners_xz(1, 2) + squre1_corners_xz(3, 2)) ./ 2;
center1   = [center1_x; center1_z];

center2_x = (squre2_corners_xz(1, 1) + squre2_corners_xz(3, 1)) ./ 2;
center2_z = (squre2_corners_xz(1, 2) + squre2_corners_xz(3, 2)) ./ 2;
center2   = [center2_x; center2_z];

distance  = norm(center1 - center2);

% Calculate arc.
vector1         = squre1_corners_xz(1, :) - squre1_corners_xz(2, :);
vector1(1, 3)   = 0;
vector1         = vector1 ./ norm(vector1);

vector2         = squre2_corners_xz(1, :) - squre2_corners_xz(2, :);
vector2(1, 3)   = 0;
vector2         = vector2 ./ norm(vector2);

sin_theta       = cross(vector1, vector2) * [0, 0, 1]';
cos_theta       = vector1 * vector2';

sin_theta_by2   = sqrt((1 - cos_theta) / 2);

%Calculate radius.
disp('::::::::::::');
disp(':: Radius ::');
disp('::::::::::::');

radius = (distance ./ 2) / sin_theta_by2

% Turn off diary.
diary off;
```

- coord2to1.m:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function M = coord2to1
% Syntax: M = coord2to1
%
% Description: Compute matrix that converts the point from
%              coordinate 2 to 1
%
% Author: Haiying Liu
% Date  : Sep. 24, 2000

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dbstop if error

msg = nargchk(0, 0, nargin);
if (~isempty(msg))
  error(strcat('ERROR:', msg));
end

clear msg;

%=========================================================================

radius        = 100;
theta         = 10 * pi / 180;
theta_y       = -theta;
sin_theta_by2 = sin(theta / 2);
cos_theta_by2 = cos(theta / 2);

delta_x       = - 2 * radius * sin_theta_by2 * sin_theta_by2;
delta_y       = 0;
delta_z       = + 2 * radius * sin_theta_by2 * cos_theta_by2;

Mt            = translate3d(delta_x, delta_y, delta_z);
Mr            = rotate3dy(theta_y);

M             = Mt * Mr;
```

- coord1to0.m:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function M = coord1to0
% Syntax: M = coord1to0
%
% Description: Compute matrix that converts the point
%              from coordinate 1 to 0
%
% Author: Haiying Liu
% Date  : Sep. 24, 2000

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dbstop if error
```

```
msg = nargchk(0, 0, nargin);
if (~isempty(msg))
  error(strcat('ERROR:', msg));
end

clear msg;

%=========================================================================

theta_x = 20 * pi / 180;

delta_x  = 0;
delta_y  = 0;
delta_z  = 3 ./ sin(theta_x);

Mt       = translate3d(delta_x, delta_y, delta_z);
Mr       = rotate3dx(theta_x);

M        = Mt * Mr;
```

- coord0toImage.m:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function M = coord0toImage
% Syntax: M = coord0toImage
%
% Description: Convert the point from coordinate 0 to image plane
%
% Author: Haiying Liu
% Date  : Sep. 24, 2000

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dbstop if error

msg = nargchk(0, 0, nargin);
if (~isempty(msg))
  error(strcat('ERROR:', msg));
end

clear msg;

%=========================================================================

% Initialization.

f        = 690;
ku       = -1;
kv       = -1;
u0       = 300;
v0       = 250;

M = [ ...
    f * ku, 0,      u0, 0; ...
    0,      f * kv, v0, 0; ...
```

```
    0,      0,      1,  0; ...
  ];
```

- translate3d.m:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function M = translate3d(x, y, z)
%
%   Returns a matrix that performs a 3-D translation.
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M      = eye(4);
M(1,4) = x;
M(2,4) = y;
M(3,4) = z;
```

- rotate3dx.m:

```
function M = rotate3dx(theta)
%
%   Returns a matrix that performs a rotation about the X axis
%   3-D rotation matrix

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M      =  eye(4);
M(2,2) =  cos(theta);
M(3,3) =  M(2,2);
M(2,3) =  sin(theta);
M(3,2) = -M(2,3);
```

- rotate3dy.m:

```
function M = rotate3dy(theta)
%
%   Returns a matrix that performs a rotation about the Y axis
%   3-D rotation matrix

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M      =  eye(4);
M(1,1) =  cos(theta);
M(3,3) =  M(1,1);
M(1,3) =  sin(theta);
M(3,1) = -M(1,3);
```

- Result from hw4_5.m:

```
» hw4_5

::::::::::::::::::::::::::::::::::::::::::
:: Image coordinate of corners of square 1 ::
::::::::::::::::::::::::::::::::::::::::::
```

```
id   |      x        y
-----+----------------------
1    |   170.4322   205.6852
2    |   99.7687   318.4832
3    |   500.2313   318.4832
4    |   429.5678   205.6852

::::::::::::::::::::::::::::::::::::::::::::
:: Image coordinate of corners of square 2 ::
::::::::::::::::::::::::::::::::::::::::::::

id   |      x        y
-----+----------------------
1    |   297.3911   79.6510
2    |   276.6427   92.3421
3    |   394.6090   95.0052
4    |   399.4584   81.6324


::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:: Transformation matrix (from image to geometry) ::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

H =

  1.0e+003 *

   -0.0028    0.0000    0.8405
    0.0000   -0.0082    2.0480
    0.0000    0.0009    0.0010

::::::::::::
:: Radius ::
::::::::::::

radius =

  100.0000
```