```
%==========================
%  Name:                hw4_1.m
%
%  Author:              Kairi Kozuma
%
%==========================

% Points in the world
pwpts = [13.1800000000000,24.7500000000000,14.4700000000000,15.8300000000000,22.5100000000000,12.5500000000000;-3.36200000000000,-11.7900000000000,-9.
sz = size(pwpts);
qwpts = [pwpts ; ones(1,sz(2))];

% Points in the image
rpts = [157,92,416,3,91,167;376,139,97,26,387,238;1,1,1,1,1,1];

% Get M matrix
M = -calibrateM(qwpts, rpts); % Why negative?
fprintf('M matrix:\n');
disp(M);

% Test M matrix to see if original results obtained
test = (M * qwpts);
test = test./test(3,:);
fprintf('M * qwpts:\n');
disp(round(test(1:2,:)));

%============================== calibrateM ==============================
%
%  function [M] = calibrateM(qPts, rPts)
%
%
%  INPUT:
%    qPts   - The points in world coordinates.
%    rPts   - The image points (in ray form).
%
%============================== calibrateM ==============================
function [M] = calibrateM(qPts, rPts)


%--(1) For each world point and image point pair, create the 2-row matrix,
%      and use them to create a master matrix.

sz = size(qPts);
masterMatrix = zeros(2*sz(2),2*sz(2));

for index = 1:sz(2)
    rmat = makeRMat(rPts(:,index));
    Qmat = makeQMat(qPts(:,index));
    mat = rmat * Qmat;
    index2 = 2*index;
    masterMatrix(index2:index2 + 1,:) = mat(1:2,:);
end


%--(2) Perform SVD using the master matrix and extract the projection
%      matrix.  Be careful about rows versus columns ...

[UU SS VV] = svd(masterMatrix);
szvv = size(VV);

M = VV(:,szvv(2));

M = (reshape(M, 4, 3))';

end


%============================== makeRMat ==============================
%
%  function mat = makeRMat(vector)
%
%
%  INPUT:
%    vector - 3 x 1 vector
%
%============================== makeRMat ==============================
function mat = makeRMat(vector)
mat = [0, vector(3), -vector(2); -vector(3), 0, vector(1); vector(2), -vector(1), 0];
end


%============================== makeQMat ==============================
%
%  function mat = makeQMat(vector)
%
%
%  INPUT:
%    vector - 4 x 1 vector
%
%============================== makeQMat ==============================
```

```
function mat = makeQMat(qPts)
mat = [qPts',zeros(1,8); zeros(1,4), qPts', zeros(1,4);zeros(1,8), qPts'];
end
```

```
M matrix:
    0.0331   -0.0003   -0.0937    0.9567
    0.0519    0.0730   -0.0127    0.2581
    0.0002   -0.0000   -0.0001   -0.0002

M * qwpts:
   157    92   416     3    91   167
   376   139    97    26   387   238
```

Published with MATLAB® R2016b