The MATLAB code below demonstrates how we can use QR decomposition in order to extract the components $\mathbf{K}$, $\mathbf{R}$ and $\mathbf{c}$ from a given 3×4 camera matrix $\mathbf{P}$, such that:

- $\mathbf{K}$ is a 3×3 homogeneous calibration matrix containing the intrinsic parameters of the camera;
- $\mathbf{R}$ is a 3×3 orthogonal (rotation) matrix and $\mathbf{c}$ a 3×1 Euclidean vector that together relate the camera coordinate system to the world coordinate system;
- and $\mathbf{KR}\big[\mathbf{I} \mid -\mathbf{c}\big] = \mathbf{P}$.

The code is a straightforward implementation of the procedure outlined in section 5.2.2 of the class notes.

```
function [K,R,c] = decomposeP(P)

% The input P is assumed to be a 3-by-4 homogeneous camera matrix.
% The function returns a homogeneous 3-by-3 calibration matrix K,
% a 3-by-3 rotation matrix R and a 3-by-1 vector c such that
%   K*R*[eye(3), -c] = P.

W = [0 0 1; 0 1 0; 1 0 0];

% calculate K and R (up to sign)
[Qt,Rt] = qr((W*P(:,1:3))');
K = W*Rt'*W;
R = W*Qt';

% correct for negative focal length(s) if necessary
D = [1 0 0; 0 1 0; 0 0 1];
if K(1,1) < 0, D(1,1) = -1; end
if K(2,2) < 0, D(2,2) = -1; end
if K(3,3) < 0, D(3,3) = -1; end
K = K*D;
R = D*R;

% calculate c
c = -R'*inv(K)*P(:,4);

end
```

Willie Brink
August 2017