1. Projection matrix to intrinsic and extrinsic parameters.

Please provide citations to any sources you consult, along with your answers to the following:

   a. What is the $RQ$ factorization (also known as the $RQ$ decomposition) of a matrix $A$, and how can it help us recover the intrinsic and extrinsic parameters of a camera calibration matrix $M = \left[ A \ | \ b \right]$?

   b. Why is the standard notation for the $RQ$ factorization especially horrible in this context?

2. Algebraic vs. geometric error

Download, read, and run the `box3d` code posted to the course website. Then answer the following questions. You may also wish to consult external sources when answering them; if so, please cite.

   a. What is the difference between algebraic error and geometric error (a.k.a. reprojection error) when computing homographies from point correspondences or calibrating cameras?

   b. Which one is minimized by solving a homogeneous least squares problem?

   c. Which one is minimized by `cv2.calibrateCamera` or `cv2.findHomography`?

   d. Which one do we typically care more about minimizing?
3. Representing 3D rotations.

Although we can represent every rotation in 3D as a $3 \times 3$ matrix, 3D rotations only have three degrees of freedom. There are therefore several alternative parameterizations of 3D rotations, aside from the matrix representation.

- **Euler angles** describe rotations by successive rotations around coordinate axes. For instance, the roll-pitch-yaw convention rotates first around the $x$ axis (roll), next about the $y$ axis (pitch), and finally about the $z$ axis (yaw).

- **Unit quaternions** are an extension of the concept of complex numbers into higher dimensions. Just as a single complex number $(a + bi)$ with unit magnitude can represent a rotation in the plane, a unit quaternion $(a + bi + cj + dk)$ can represent rotations in space, given the appropriate multiplicative identities defined on $i$, $j$, and $k$.

- **Rotation vectors** are a compact way to encode rotations based on the fact that any rotation matrix can be encoded as a finite rotation of some angle $\alpha$ about a particular axis $a$, with $\|a\| = 1$. The rotation vector corresponding to this rotation is then simply $\alpha a$.

Read through section 2.1.4 of the book, and flip through up to about page 22 of the Diebel paper on the course website. Then, answer the following questions:

a. For each of the four parameterizations mentioned above, explain what the constraints on the representation are. A constraint is any property that prevents some sets of numbers from representing valid rotations for a particular parameterization.

b. In which parameterizations can we easily compute compositions of rotations – the rotation resulting from applying two arbitrary rotations in sequence?

c. Imagine performing gradient descent on a function whose input includes a 3D rotation, for example pose estimation of a 3D object in an image. Why is a minimal, three parameter rotation representation better to use in this case than a representation with more parameters (such as a full rotation matrix)? Think about how well the constraints are obeyed after taking a gradient step.

d. What is gimbal lock, and why is it undesirable?