CSE 344: Computer Vision
Assignment 1

Nitika Saran
#2014068

Q1. Calibration toolbox from OpenCV used : http://docs.opencv.org/3.0-beta/doc/tutorials/calib3d/camera_calibration/camera_calibration.html

a)

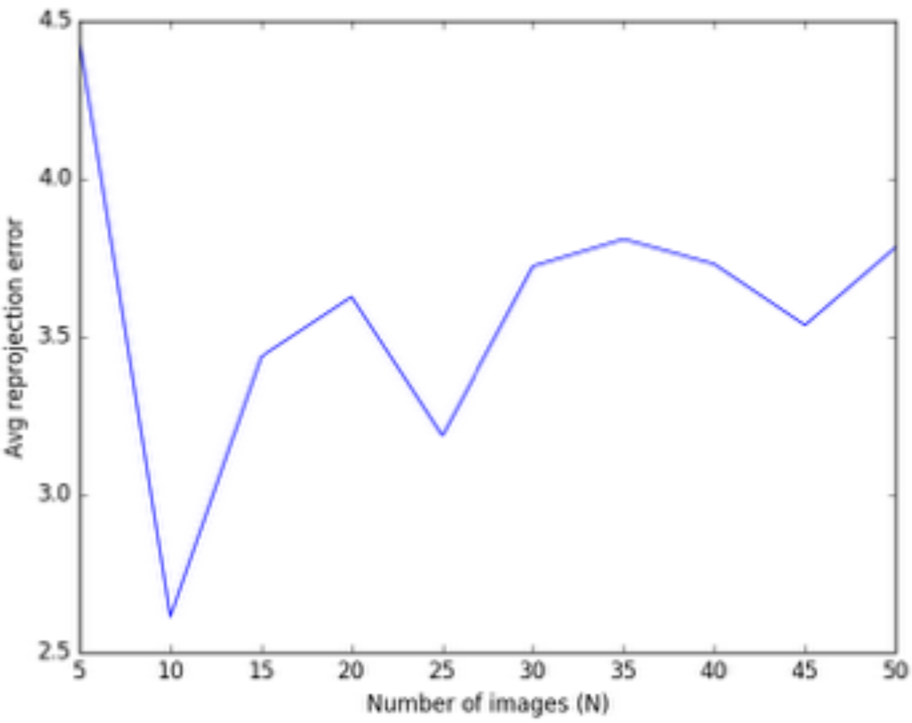
S1 : for 20 images with all DOFs covered

- Average re-projection error = 3.16271482714
- Estimated intrinsics:
  - fx = 3.8584530251284455e+03
  - fy = 3.8584530251284455e+03
  - cx = 1.1835000000000000e+03
  - cy = 2.1035000000000000e+03
- Estimated distortion coefficients: [-1.4186e-02, 1.8714e+00, 9.7445e-04, -4.6332e-03, -1.0061+01]

S2: for first 20 images

- Average re-projection error for first 20 images: 4.58034775838
- Estimated intrinsics:
  - fx = 3.8311637596940427e+03
  - fy = 3.8311637596940427e+03
  - cx = 1.1835000000000000e+03
  - cy = 2.1035000000000000e+03
- Estimated distortion coefficients: [-8.5676e-03, 9.2993e-01, -1.8579e-02, 3.5245e-03, -1.6453e+00]

b) I used random sampling implemented in python's libraries to calibrate for different, randomly chosen sets of images with length N = 5,10…50. The plot for Mean re-projection error vs N is shown below:



Q2.

Assumption: The rotation from car frame (V) to mount frame (M) is -30 degrees about the X axis. This is what I interpreted from the question and figure provided.

We have three transformations:
vTw: transformation from world frame (W) to car frame (V)
mTv: transformation from car frame (V) to mount frame (M)
cTm: transformation from mount frame (M) to camera frame (C)

These transformations are calculated one by one as explained in the code.
They are as follows:

vTw =

```
[[ 0.8660254   0.5         0.        -1.19615242]
 [-0.5         0.8660254   0.         9.92820323]
 [ 0.          0.          1.        -1.        ]
 [ 0.          0.          0.         1.        ]]
```

mTv =

```
[[ 1.         0.          0.          0.        ]
 [ 0.         0.8660254  -0.5         2.        ]
 [ 0.         0.5         0.8660254  -3.46410162]
 [ 0.         0.          0.          1.        ]]
```

cTm =

```
[[1   0   0   0]
 [0   1   0   2]
 [0   0   1   0]
 [0   0   0   1]]
```

We want to transform a point from world frame P_w to camera frame P_c.
This is done by combining the three transformations above as follows:

- P_w --> P_v --> P_m --> P_c
- P_c = cTm * (P_m)
- P_c = cTm * (mTv * P_v)
- P_c = cTm * (mTv * (vTw * P_w))
- P_c = cTm * mTv * vTw * P_w
- P_c = cTw * P_w
- cTw = cTm * mTv * vTw

Transformation from world frame (W) to camera frame (C):

cTw =

```
[[ 0.8660254   0.5         0.        -1.19615242]
 [-0.4330127   0.75       -0.5        13.09807621]
 [-0.25        0.4330127   0.8660254   0.6339746 ]
 [ 0.          0.          0.          1.        ]]
```

I used this matrix to transform all cube corners to the camera frame, and then projected these 3D points according to the perspective projection formula (fx*X/Z + cx, fy*Y/Z + cy) to get 2D pixel coordinates:

```
[[0 0 0 1]]  -->  [[-6096.44143844]] [[-6096.44143844]]
[[1 0 0 1]]  -->  [[-2133.85382338]] [[-2133.85382338]]
[[1 1 0 1]]  -->  [[ 1985.77308215]] [[ 1985.77308215]]
[[0 1 0 1]]  -->  [[-1333.93523668]] [[-1333.93523668]]
[[1 0 1 1]]  -->  [[ 164.47632413]] [[ 164.47632413]]
[[0 0 1 1]]  -->  [[-1893.36528927]] [[-1893.36528927]]
[[0 1 1 1]]  -->  [[-206.07773982]] [[-206.07773982]]
[[1 1 1 1]]  -->  [[ 1572.94858645]] [[ 1572.94858645]]
```

The normalised image coordinates were calculated with f = 1 and standard procedure, to get these for cube corners:

```
[[0 0 0 1]]  -->  [[-1.88675135]] [[-1.88675135]]
[[1 0 0 1]]  -->  [[-0.85976266]] [[-0.85976266]]
[[1 1 0 1]]  -->  [[ 0.2079261]] [[ 0.2079261]]
[[0 1 0 1]]  -->  [[-0.65244678]] [[-0.65244678]]
[[1 0 1 1]]  -->  [[-0.26410162]] [[-0.26410162]]
[[0 0 1 1]]  -->  [[-0.79743495]] [[-0.79743495]]
[[0 1 1 1]]  -->  [[-0.36013857]] [[-0.36013857]]
[[1 1 1 1]]  -->  [[ 0.10093387]] [[ 0.10093387]]
```

Theory questions on further pages.

Q3 We wish to rotate a vector x about an ~~angle~~ axis u by angle θ.

x has 2 components, - parallel to u, and perpendicular to u.

$$\vec{x} = \underbrace{(\vec{u}\cdot\vec{x})\vec{u}}_{\substack{\vec{x}_{\parallel} \\ \text{(parallel)}}} + \underbrace{\vec{x} - (\vec{u}\cdot\vec{x})\vec{u}}_{\substack{\vec{x}_{\perp} \\ \text{(perpendicular)}}}$$

The component $\vec{x}_{\parallel}$ does not change on rotation as it is about the rotation axis.

$\vec{x}_{\perp}$ rotates as follows:

$$|\vec{x}_{\perp}| = |\vec{x}_{\perp\, rot}| \quad \text{(retain its magnitude)}$$

$$\rightarrow \quad \vec{x}_{\perp\, rot} = \underbrace{\cos\theta\, \vec{x}_{\perp}} + \underbrace{\sin\theta(\vec{u} \times \vec{x}_{\perp})}$$

These represent 2 perpendicular axes in the plane orthogonal to $\vec{u}$.

The above comes from simple planar rotation.

$$\vec{x}_{\perp\, rot} = \cos\theta\, \vec{x}_{\perp} + \sin\theta\,(\vec{u} \times (\vec{x} - \vec{x}_{\parallel}))$$

$$= \cos\theta\, \vec{x}_{\perp} + \sin\theta(\vec{u} \times \vec{x}) - \cancel{\sin\theta(\vec{u} \times \vec{x}_{\parallel})}$$

0 as $\vec{u} \parallel \vec{x}_{\parallel}$.

~~x_R~~ $\vec{x}_{\perp rot} = \cos\theta\, \vec{x}_{\perp} + \sin\theta(\vec{u} \times \vec{x})$

Now, $\vec{x}_{rot} = \vec{x}_{\perp rot} + \vec{x}_{\parallel rot}$

$= \vec{x}_{\parallel} + \cos\theta\, \vec{x}_{\perp} + \sin\theta(\vec{u} \times \vec{x})$

$= \vec{x}_{\parallel} + (\vec{x} - \vec{x}_{\parallel})\cos\theta + \sin\theta(\vec{u} \times \vec{x})$

$\cancel{= \vec{x}\cos\theta + (}$

$= \cos\theta\, \vec{x} + \sin\theta(\vec{u} \times \vec{x}) + (1-\cos\theta)\vec{x}_{\parallel}$

$R x = \cos\theta\, \vec{x} + \sin\theta(\vec{u} \times \vec{x}) + (1-\cos\theta)(\vec{u}\cdot\vec{x})\vec{u}$

which is the desired result.

Q4. O : optical center in world frame.
M : perspective projection matrix.

We know that M contains world to camera transformation.

~~The origin of the camera coordinates~~

$\Rightarrow$ $M = K \times T$
where T is 3D transform ($w \to c$)
and K is intrinsic matrix.

$$\rightarrow M = K \times \begin{bmatrix} {}^{c}R_{w} & {}^{c}O_{w} \\ 0 & 1 \end{bmatrix}$$

We have $O = \begin{pmatrix} {}^{w}O_{c} \\ 1 \end{pmatrix}$

Now, if we calculate MO,

$$MO = K \times \begin{bmatrix} {}^{c}R_{w} & {}^{c}O_{w} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{w}O_{c} \\ 1 \end{bmatrix}$$

$$= K \left[{}^{c}R_{w} \, {}^{w}O_{c} + {}^{c}O_{w} \right]$$

$\left[\because Tx = Rx + t \right]$

$\rightarrow$

Now, ${}^{c}R_{w}\,{}^{w}O_{c}$ is the rotated ~~vector~~ position vector of the camera origin, in the camera frame.

This will give us the inverse of world origin in camera frame ${}^{c}O_{w}$.

$${}^{c}R_{w} = -{}^{c}O_{w}$$

$$\rightarrow \quad MD = K(-{}^{c}O_{w} + {}^{c}O_{w})$$

$$= 0.$$

$\therefore$ Hence proved.

This is intuitively correct as the optical center is the projection of the origin ~~in the~~ of the camera onto the image plane, which is by definition 0 vector.