Problem 1a)

\[
K = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{bmatrix}
\]

\[
R = \begin{bmatrix}
\cos(-45) & 0 & \sin(-45) \\
0 & 1 & 0 \\
-\sin(-45) & 0 & \cos(-45) \\
\end{bmatrix}
\]

\[
M = K \cdot R = \begin{bmatrix}
\cos(-45) & 0 & \sin(-45) \\
0 & 1 & 0 \\
-\sin(-45) & 0 & \cos(-45) \\
\end{bmatrix}
\]

\[
C = \begin{bmatrix}
0 \\
0 \\
-3 \\
\end{bmatrix}
\]

Projection Matrix \( P = [M \mid -MC] = \begin{bmatrix}
0.7071 & 0 & -0.7071 & -2.1213 \\
0 & 1 & 0 & 0 \\
0.7071 & 0 & 0.7071 & 2.1213 \\
\end{bmatrix}\)

Problem 1b)

\(P5 = [0; \frac{1}{2}; 0; 1]\)

\(P \cdot P5 = [-2.1213; .5; 2.1213]\)

Homogeneous image coordinates = [-2.1213; .5; 2.1213]
Non-Homogeneous image coordinates = [-1; .5 / 2.1213] = [-1, 0.2357]

\(P6 = [1; \frac{1}{2}; 0; 1]\)

\(P \cdot P6 = [-1.4142; .5; 2.8284]\)

Homogeneous image coordinates = [-1.4142; .5; 2.8284]
Non–Homogeneous image coordinates = [-1.4142/2.8284; .5/2.8284] = [-0.4035; 0.1768]

P7 = [1; \frac{1}{2}; 1; 1]

P * P7 = [-2.1213; .5; 3.5355]

Homogeneous image coordinates = [-2.1213; .5; 3.5355]
Non–Homogeneous image coordinates = [-2.1213/3.5355; .5/3.5355] = [-0.6000; 0.1414]

P8 = [0; \frac{1}{2}; 1; 1]

P * P8 = [-2.8284; 0; 2.8284]

Homogeneous image coordinates = [-2.8284; 0; 2.8284]
Non–Homogeneous image coordinates = [-2.8284/2.8284; .5/2.8284] = [-1; 0.1768]

**Problem 1c)**

The 3 vanishing points correspond to the columns of matrix M from above

Therefore the 3 image vanishing points in non-homogenous form are:

(0.7071/ 0.7071, 0/ 0.7071) = (1, 0)

(0/0, 1/0) = ideal

(-0.7071/0.7071, 0/0.7071) = (-1, 0)

**Problem 1d)**

Vanishing point of P5 and P7 would be:

Using Homogeneous image coordinates of P5, P7, P1, and P3:

P5 = [-2.1213; .5; 2.1213]
P7 = [-2.1213; .5; 3.5355]
P1 = [-2.1213; -.5; 2.1213]
P3 = [-2.1213; -.5; 3.5355]

(P5 x P7) x (P1 x P3) = (0, 0, 4.2425)

So the non-homogenous image vanishing point of line P5 P7 would be (0, 0).
**Problem 1e)**
The camera would need to be positioned so that it could only see 1 side of the cube; basically it would only see a square.

**Problem 2)**

\[ P = [M | M (-C)] \]

To show that \( PC = 0 \)

Start with \( M = \) and \( C = \)

\[
\begin{array}{ccc}
M_{11} & M_{12} & M_{13} \\
M_{21} & M_{22} & M_{23} \\
M_{31} & M_{32} & M_{33}
\end{array}
\]

Then \( P = \)

\[
\begin{array}{ccc}
M_{11} & M_{12} & M_{13} & -C_1*M_{11} - C_2*M_{12} - C_3 * M_{13} \\
M_{21} & M_{22} & M_{23} & -C_1*M_{21} - C_2*M_{22} - C_3 * M_{23} \\
M_{31} & M_{32} & M_{33} & -C_1*M_{31} - C_2*M_{32} - C_3 * M_{33}
\end{array}
\]

Next let \( C \) be:

\[
\begin{array}{c}
C_1 \\
C_2 \\
C_3 \\
1
\end{array}
\]

So that \( P * C = \)

\[
\begin{array}{c}
C_1*M_{11} + C_2*M_{12} + C_3 * M_{13} -C_1*M_{11} - C_2*M_{12} - C_3 * M_{13} \\
C_1*M_{21} + C_2*M_{22} + C_3 * M_{23} -C_1*M_{21} - C_2*M_{22} - C_3 * M_{23} \\
C_1*M_{31} + C_2*M_{32} + C_3 * M_{33} -C_1*M_{31} - C_2*M_{32} - C_3 * M_{33}
\end{array}
\]

Which equals \([0; 0; 0] \)

**Problem 3)**

The first 3 columns of the 3x4 camera matrix correspond to the 3 vanishing points of the parallel lines. And the last column is the translation and rotation between the camera origin and the world origin.
Problem 4)

Camera center \(-C = (\text{inv}(M) \ast \text{last column of } P)\)

\[
\begin{bmatrix}
0.0009 & -0.0015 & 0.7510 & 1440000 & 1.5603\times10^3 \\
-0.0020 & 0.0001 & -0.9526 & -632000 & 3.6263\times10^3 \\
0.0001 & 0.0018 & 0.2169 & -918 & -1.2109\times10^3 \\
\end{bmatrix}
\]

So camera center \(C = [-1.5603\times10^3; -3.6263\times10^3; 1.2109\times10^3]\)

To find the calibration parameters of \(P\) compute the RQ decomposition of Matrix \(M\)

\(M = RQ\)

Using Matlab gives:

\[
M = \\
\begin{bmatrix}
350.0000 & 339.0000 & 277.0000 \\
-103.0000 & 23.3000 & 459.0000 \\
0.7070 & -0.3530 & 0.6120 \\
\end{bmatrix}
\]

\(R = \text{calibration Matrix:}\)

\[
\begin{bmatrix}
-465.9239 & -92.6561 & -297.4555 \\
0 & -426.4365 & -199.9619 \\
0 & 0 & -0.9995 \\
\end{bmatrix}
\]

\(Q = \text{rotation Matrix:}\)

\[
\begin{bmatrix}
-0.4136 & -0.9093 & -0.0467 \\
0.5732 & -0.2202 & -0.7892 \\
-0.7074 & 0.3532 & -0.6123 \\
\end{bmatrix}
\]

Problem 5)

The vanishing point \(v\) on line \(abc\) is associated with the line \(ABC\) at infinity so using the cross ratio we can say:

\[
\text{Cross } (A, B, C, \text{inf}) = \text{Cross } (a, b, c, v)
\]

\[
\frac{((A-B)/(C-B))}{((A-\text{inf})/(C-\text{inf})} = \frac{((a-b)/(c-b))}{((a-v)/(c-v))}
\]

The \(\text{inf}\)'s cancel out leaving

\[
AB/BC = (ab/bc) / ((a - v)/(c - v))
\]
From here we can say

\[
\frac{(a - v)}{(c - v)} = \frac{(ab/bc)}{(AB/BC)}
\]

Where the only unknown is \(v\) and we can use algebra to solve for \(v\).

**Problem 6a)**

If we know the image point \(m'\) then we can use the cross ratio to find the world point \(M\) (viewing ray \(OM\))

Cross \((a, b, c, m') = \text{Cross} \ (A, B, C, M)\)

\[
\left(\frac{|ab|}{|am'|}\right) / \left(\frac{|cb|}{|cm'|}\right) = \left(\frac{|AB|}{|AM|}\right) / \left(\frac{|CB|}{|CM|}\right)
\]

From here we use algebra to solve for \(M\).

**Problem 6b)**

A line in the image say between \(a\) and \(m'\) would contain all the rays that passed through the origin or camera center such that any ray \(P\) times vector \(l\) equals 0.

\[l*P = 0\]

Vector \(l\) will be at the intersection of the two rays \(a\) and \(m'\) and will be at the origin/camera center. We can simply compute \(l\) by taking the cross product of the two rays:

\[l = a \times m'\]

**Problem 7a)**

For this problem I chose the origin of the world coordinates to be the center of the road/circle so the center of each square will be 100 meters from the origin.

Using matlab I chose the world coordinates of the first square to be:

\[
\begin{align*}
slul &= [98; 2; 1; 1]; \quad \% \text{upper left} \\
slur &= [102; 2; 1; 1]; \quad \% \text{upper right} \\
sllr &= [102; -2; 1; 1]; \quad \% \text{lower right} \\
slll &= [98; -2; 1; 1]; \quad \% \text{lower left}
\end{align*}
\]

Then I used the following rotation matrix to rotate the first square by 10 degrees around the z axis to get the second square.

\[
R = [\cos(\pi/18), -\sin(\pi/18), 0, 0; \sin(\pi/18), \cos(\pi/18), 0, 0; 0, 0, 1, 0; 0, 0, 0, 1];
\]

The coordinates of the second square are then computed from the rotation of the coordinates from the first square:
Next I set up my camera calibration and rotation matrix. The robot camera is facing downwards at the road at an angle of 20 degrees so the camera rotation is $20 + 90 = 110$ degrees around the x-axis compared to the world orientation.

\[
K = \begin{bmatrix}
690 & 0 & 300 \\
0 & -690 & 250 \\
0 & 0 & 1
\end{bmatrix};
\]

\[
Rx = \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos(11\pi/18) & -\sin(11\pi/18) \\
0 & \sin(11\pi/18) & \cos(11\pi/18)
\end{bmatrix};
\]

\[
M = K \times Rx;
\]

Next I set the location of the robot to be slightly behind the first square with the camera 3 meters of the ground. (note the ground starts at 1 meter)

\[
C = \begin{bmatrix}
100 \\
-9 \\
4
\end{bmatrix};
\]

\[
Mc = M \times (-C);
\]

\[
P = \text{horzcat}(M, Mc);
\]

Multiplying the above projection matrix $P$ with each world coordinate and then dividing by the $z$ component gives the following set of image coordinates:

<table>
<thead>
<tr>
<th>World Coordinate</th>
<th>Image Coordinate</th>
</tr>
</thead>
<tbody>
<tr>
<td>118.5144, 211.4400</td>
<td>square 1 lower left</td>
</tr>
<tr>
<td>481.4856, 211.4400</td>
<td>square 1 lower right</td>
</tr>
<tr>
<td>178.5498, 307.2725</td>
<td>square 1 upper left</td>
</tr>
<tr>
<td>421.4502, 307.2725</td>
<td>square 1 upper right</td>
</tr>
<tr>
<td>208.2419, 407.8921</td>
<td>square 2 lower left</td>
</tr>
<tr>
<td>322.6724, 410.3992</td>
<td>square 2 lower right</td>
</tr>
<tr>
<td>203.1327, 420.5240</td>
<td>square 2 upper left</td>
</tr>
<tr>
<td>302.5425, 422.4047</td>
<td>square 2 upper right</td>
</tr>
</tbody>
</table>
Problem 7b)
The following is the image of the 8 points of the 2 squares in matlab:

![Figure 1](image)

Problem 7c)
Using the 4 world points and image points of the first square I create the A such that $Ah = 0$

$$A = \begin{bmatrix} 118.5144, & 211.4400, & 1, & 0, & 0, & 0, & (-98 \times 118.5144), & (-98 \times 211.4400), & -98; \\ 0, & 0, & 0, & 118.5144, & 211.4400, & 1, & (2 \times 118.5144), & (2 \times 211.4400), & 2; \\ 481.4856, & 211.4400, & 1, & 0, & 0, & 0, & (-102 \times 481.4856), & (-102 \times 211.4400), & -102; \\ 0, & 0, & 0, & 481.4856, & 211.4400, & 1, & (2 \times 481.4856), & (2 \times 211.4400), & 2; \\ 178.5498, & 307.2725, & 1, & 0, & 0, & 0, & (-98 \times 178.5498), & (-98 \times 307.2725), & -98; \\ 0, & 0, & 0, & 178.5498, & 307.2725, & 1, & (2 \times 178.5498), & (2 \times 307.2725), & -2 \\ 421.4502, & 307.2725, & 1, & 0, & 0, & 0, & (-102 \times 421.4502), & (-102 \times 307.2725), & -102; \\ 0, & 0, & 0, & 421.4502, & 307.2725, & 1, & (2 \times 421.4502), & (2 \times 307.2725), & -2 \\ \end{bmatrix}$$

Then using matlab to solve for $h$ I get the following homography matrix:
$H = \begin{bmatrix} 0.0064 & -0.1995 & 98.0888 \\ 0 & 0.0201 & -5.4141 \\ 0 & -0.0020 & 1.0000 \end{bmatrix}$

Now I can compute world coordinate = $H \ast$ image coordinate

**Problem 7d)**
Using matlab to convert the 8 image points back into world geometry I have plotted the points below:

As you can see from the above plot the radius is still 100 meters because the first square is still centered at (100, 0).

**Problem 8)**
The parachuter is lower than the person taking the picture because the parachuter is lower than the horizon.
Problem 9)

First plotting the 1-D image below described by the function

\[ Y = \cos(X/100) \]

Next I smooth the image using a gaussian with sigma = 1.5 which yields the following kernel:

\[ [0.3088, \ 0.3825, \ 0.3088] \]
The resulting smoothed image is shown below:

Next I compute the magnitude with the following code:

```matlab
function output = Magnitude1D( image )
    [rows, columns] = size(image);
    A = zeros(rows,columns);
    for i = 1:columns
        if (i < columns)
            A(1,i) = abs(image(1, i+1) - image(1, i));
        end
    end
    output = A;
end
```
The resulting magnitudes are plotted below:

As you can see from the above plot the local max values correspond to the straight lines in the original functions (aka the edges). These local max values occur at a threshold $T$ of around 0.1 however you could probably make the threshold range for $T$ to be anywhere from about 0.03 to 0.1.

Also because this function changes so slowly I think that the smoothing does very little to alter the final result for this function so I think that sigma could really be any value.

**Problem 10a)**

Using two sobel filters I calculate at point (7)

$$S_x = \frac{1}{8} (-4 - 12 - 8 + 6 + 16 + 10) = \frac{1}{8} \times 8 = 1$$

$$S_y = \frac{1}{8} (4 + 10 + 6 - 8 - 18 - 10) = \frac{1}{8} \times -16 = -2$$

So the magnitude $= (1^2 + (-2)^2)^{.5} = 5^{.05}$

And the direction $= \arctan(-2/1) = 63.4$ degrees
Problem 10b)

I would expect the intensity to slightly increase at (3.1, 7.3) to be about 17.1. This is because the gradient is pointing down and to the left and this new point is down and farther to the right. Therefore I would expect a slightly higher intensity because we are moving farther away from the direction that the gradient is pointing to.

Problem 10c)

Using two sobel filters I calculate at pixel (3,2)

\[
S_x = \frac{1}{8} (-9 -8 -1 + 13 +16 +5) = \frac{1}{8} \times 16 = 2 \\
S_y = \frac{1}{8} (9 + 20 + 13 - 1 - 4 - 5) = \frac{1}{8} \times 32 = 4
\]

Magnitude = \((2^2 + 4^2)^{.5}\) = \((20)^{.5}\)

Direction = \arctan \(4/2\)

Problem 11)

My filter is:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>-2</td>
<td>-1</td>
<td>0</td>
</tr>
</tbody>
</table>

After applying the filter to the image the following are the results for the middle row:

6 0 -6 -12 -18 -24 -30