1.

a)

4	0.04	0.03	0.02	0.01	0.02
3	0.03	0.02	0.01	0.02	0.03
2	0.02	0.01	0.02	0.01	0.02
1	0.01	0	0.01	0.02	0.03
0	0	0.01	0.02	0.03	0.04
	0	**1**	**2**	**3**	**4**

4	1 or 2	2	1	1 or 3
3	2	1	2 or 3	2 or 3
2	1 or 2	2 or 3	1	1 or 3
1	1	3	3	1 or 3
	1	**2**	**3**	**4**

[0] -> [X 0] -> [-1 X 0] -> [0 -1 X 0]
-> [-1 -1 X 0]
-> [0] -> [1 0] -> [X 1 0] -> [0 X 1 0]
[X]-> [-1 X] -> [-1 -1 X] -> [0 -1 -1 X]
-> [-1 -1 -1 X]

b)

function [c, i] = **extend_match** (x, y, c1, c2, c3, oc)
[c, i] = min([c1 + oc, c2 + oc, sqrt((x-y)^2) + c3]) ;

Results)
extend_match(1,1,3.7,3.9,3.5,0.01)
ans = 3.5000

extend_match(1,0,3.7,3.9,3.5,0.01)
ans = 3.7100

c)

function d = stereo_1d(v1, v2, oc)
d1 = length(v1) ;
d2 = length(v2) ;

d = [] ;
CM = zeros(d1+1, d2+1) ; % For containing the matching cost
MM = zeros(d1, d2) ; % For reconstructing the optimum path

```
for i = 0 : d1
   for j = 0 : d2
      if i == 0 || j == 0
         CM(i+1,j+1) = oc * max([i j]) ;
      else
         [CM(i+1,j+1) MM(i, j)] = ...
            extend_match(v1(i), v2(j), CM(i, j+1), CM(i+1, j), CM(i, j), oc) ;
      end
   end
end

i = d1 ; j = d2 ;
disparity = 0 ;

while ( i ~= 0 && j ~= 0 )
   switch MM(i,j)
      case 1 ,
         disparity = disparity - 1 ;
         d = [' X ' d] ;
         i = i - 1 ;
      case 2 ,
         disparity = disparity + 1 ;
         j = j - 1 ;
      case 3 ,
         if disparity < 0
            d = [' -' char('0' + abs(disparity)) ' ' d] ;
         else
            d = [' ' char('0' + disparity) ' ' d] ;
         end
         i = i -1 ; j = j -1 ;
      otherwise,
         sprintf('\nError occured during computation.\n') ;
   end
end

for j = 1 : i
   d = [ d 'X'] ;
end
```

Results)

```
v1 = [1 0 1 1 0 1 1 0 0 1 1 0 1 1 1] ;
v2 = [1 0 1 0 1 0 1 1 0 0 1 1 1 1 1] ;
```

stereo_1d(v1, v2, 0.01)

```
ans =
 0  0  0  -1  -1  -1  -1  -1  -1  -1  -1  X  0  0  0
```

d)

```
function D = stereo_2D(V1, V2, oc)

if size(V1) ~= size(V2)
   error('DIM NEQ', 'The dimension of the images are not compatible', size(V1),
size(V2)) ;
end

D = [];
[n_rows n_cols] = size(V1)

for i = 1 : n_rows
   D = [D ; stereo_1dN(V1(i,:) , V2(i,:), oc, max(size(V1)) * oc * 2)] ;
end

D = (D - min(min(D))) ./ (max(size(V1)) * oc * 2) ;
```

Results)

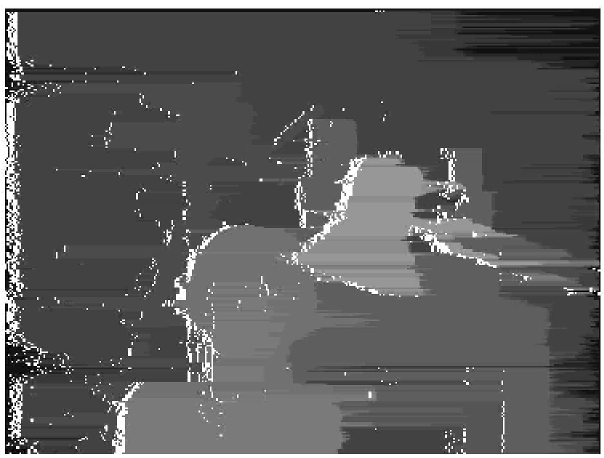
e)

```
function d = stereo_1DF(v1, v2, oc, DL)
d1 = length(v1) ;
d2 = length(v2) ;

d = [] ;
CM = ones(d1+1, d2+1) * max(d1,d2) * oc    ; % For containing the matching cost
MM = ones(d1, d2) + 2 ;        % For reconstructing the optimum path

for i = 0 : d1
   for j = max(0, i - DL) : min(d2,  i + DL)
      if i == 0 || j == 0
         CM(i+1,j+1) = oc * max([i j]) ;
      else
```

```
            [CM(i+1,j+1) MM(i, j)] = ...
                extend_match(v1(i), v2(j), CM(i, j+1), CM(i+1, j), CM(i, j), oc) ;
        end
    end
end

i = d1 ; j = d2 ;
disparity = 0 ;
sprintf('Done part 1') ;

while ( i ~= 0 && j ~= 0 )
    switch MM(i,j)
        case 1 ,
            disparity = disparity - 1  ;
            d = [DL+10 d] ;
            i = i - 1 ;
        case 2 ,
            disparity = disparity + 1 ;
            j = j - 1 ;
        case 3 ,
            if disparity > DL
                d = [DL+10 d] ;
            elseif disparity < -DL
                d = [-DL-10 d] ;
            else
                d = [disparity d] ;
            end
            i = i -1 ; j = j -1 ;
        otherwise,
            sprintf('\nError occured during computation.\n') ;
    end
end

for j = 1 : i
    d = [DL+10 d] ;
end

function D = stereo_2DF(V1, V2, oc)

if size(V1) ~= size(V2)
    error('DIM NEQ', 'The dimension of the images are not compatible', size(V1),
size(V2)) ;
end

D = [];
[n_rows n_cols] = size(V1) ;
```

```
for i = 1 : n_rows
   DM = stereo_1DF(V1(i,:) , V2(i,:), oc, 15) ;
   D = [D ; DM ] ;
end

D = D - min(min(D)) ;
D = D / max(max(D)) * 255 ;
D = uint8(D) ;
```

Results)



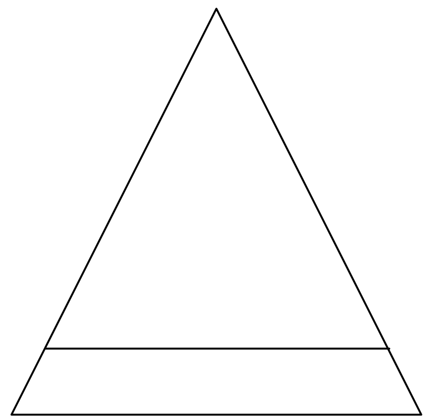
2.

a)

18 : 20 = z – 1 : z
20z – 20 = 18z
z = 10

Obviously the triangle is an isosceles triangle. . So

Pt = **(0 , 0 , 10)**



b)

16 : 20 = z-1 : z
20z -20 = 16z
z = 5

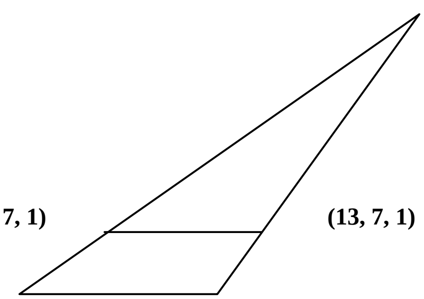
By the law of perspective projection
X = fx / z , Y = fy / z where f = 1, z = 5

From the camera 1 coordinate system,
7 = x / 5, x = 35 (in camera 2 coordinates x = 15)
7 = y / 5, y = 35 (in camera 2 coordinates y = 35)

So in the world coordinates,

x = 35 – 10 = 25 (in case of camera 2, x = 15 + 10)
y = 35 (in case of camera 2, y = 35)

(25, 35, 5)



c)

By the equation of $x_w + z_w = 0$ and $y = 0$,

The x coordinate of camera 1 coordinates is $x = fX/Z$, $x = (x_w+10) / -x_w$

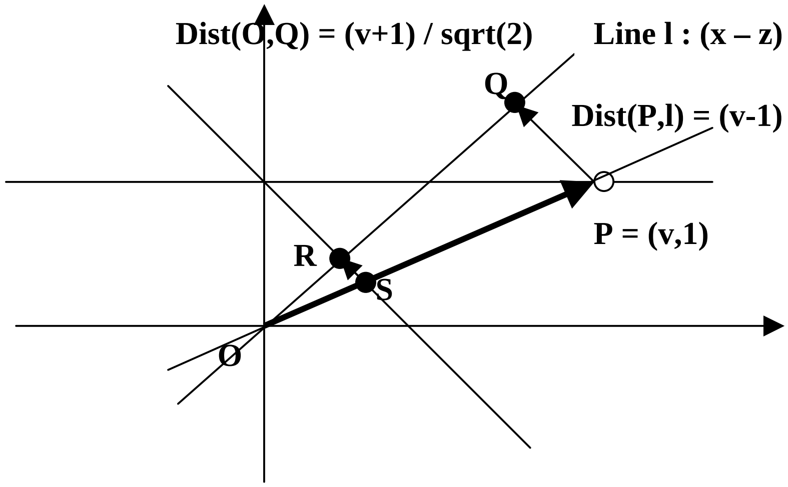
The x coordinate of camera 2 coordinates is $u = fU/Z$, $u = (x_w-10) / -x_w = -1 + 10 / x_w$

Then disparity between them is $d = x – u$

Finally apply the equation from camera 2 coordinate, $1 / x_w = (u+1)/10$,

$d = -20/x_w = -20 * (u+1) / 10 = -2 (u + 1)$

3.



Distance from P to line l : (v-1) / sqrt(2)
Distance from O to Q : (v+1) / sqrt(2)
Distance from O to R : 1 / sqrt(2)

By the rule of similar triangle,

Distance from R to S = u = ((v-1) / sqrt(2)) / (v+1) = (v-1) / (v + 1) * 1/sqrt(2)

[Another solution]

First, think about the coordinate system of the case that the image plane along the line z=1 (the second image). We can have this coordinate system as a reference coordinate system. Then any point (x,z) in that plane would be mapped to **v = x/z**.

If we observe that point (x,z) at the coordinate system of second image, it looks like the point is rotated around the y-axis –origin is the same- and the degree of the rotation is cos? = 1/sqrt(2) , sin? = 1/sqrt(2) , ? = p/4. (Think about the rotation matrix you've learned from the class)

Therefore if we transform the (x,z) to the second image coordinate system, we will have x' = (x-z) / sqrt(2) and z' = (x+z) / sqrt(2). Then the projection of that point on x + z = 1 is u = x '/z ' * (1/sqrt(2)) = (x-z) / (x+z) * (1/ sqrt(2)).

v = x / z, x = zv
u = (x-z) / (x+z) * (1/ sqrt(2))
= (v-1) / (v+1) * (1/ sqrt(2))

a) when v = 0, the corresponding u is (– 1 / sqrt(2))
b) (v-1) / (v+1) * (1/ sqrt(2))