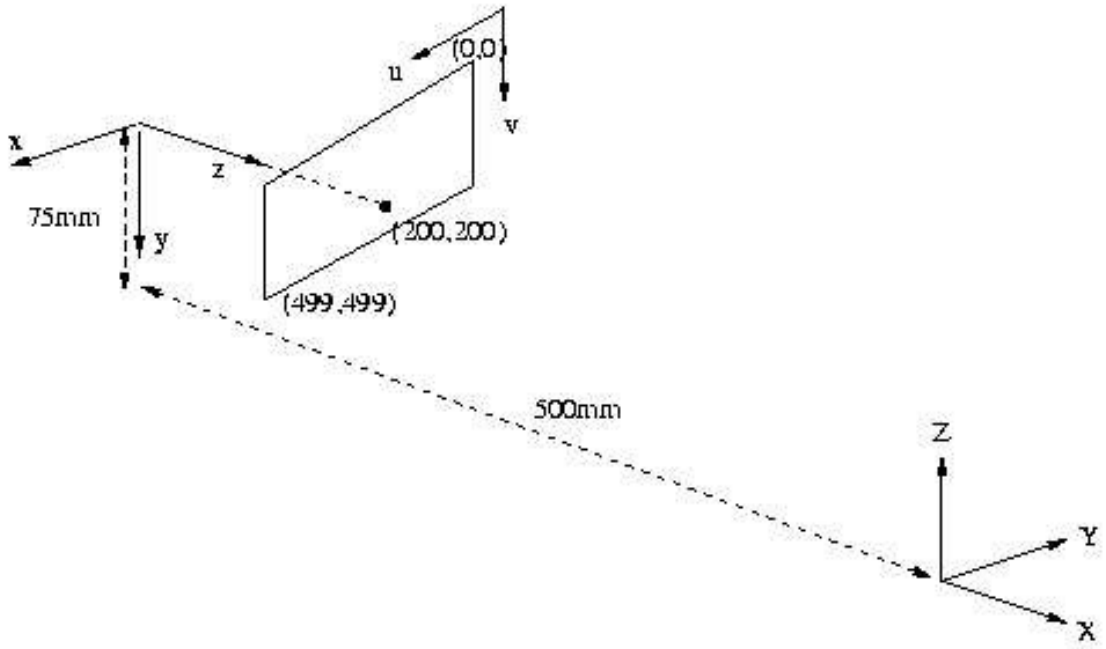
[15] **Projection**

A camera contains a lens with focal length 20mm, a 10mm × 10mm square CCD array, divided into 500 × 500 square pixels. The pixel at the top-left corner of the CCD array has image coordinates (0, 0) and the principal point of the CCD array is at pixel coordinates (200, 200). Consider a point in the scene at camera-centered coordinates (0, 75, 500), given in mm, as shown in the following figure.



(a) [8] Give general equations (of the form $u = ..., v = ...$) and also a specific numeric answer that specify the pixel coordinates (u, v) of the scene point at camera coordinates (0, 75, 500) under perspective projection for this camera configuration.

$$u = u_0 + \frac{k_u f x}{z} = 200 + \frac{(500/10)20x}{z} = 200 + \frac{1000x}{z}$$

$$v = v_0 + \frac{k_v f y}{z} = 200 + \frac{(500/10)20y}{z} = 200 + \frac{1000y}{z}$$

Here, $(x, y, z) = (0,75,500)$, so $(u, v) = (200,350)$.

(b) [4] What is the 3 × 3 *camera calibration matrix*, **K**, for this camera configuration?

$$\mathbf{K} = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_u & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

where $a_u = fk_u = 20(500/10) = 1000$ and $a_v = fk_v = 1000$, so

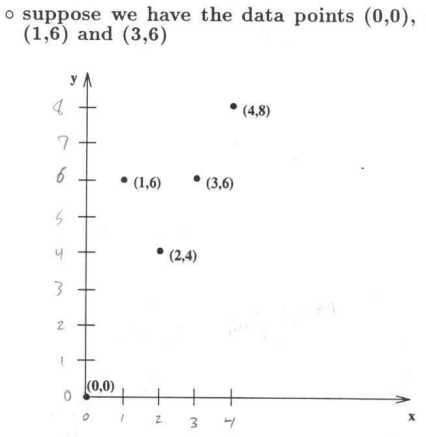
$$\mathbf{K} = \begin{pmatrix} 1000 & 0 & 200 \\ 0 & 1000 & 200 \\ 0 & 0 & 1 \end{pmatrix}$$

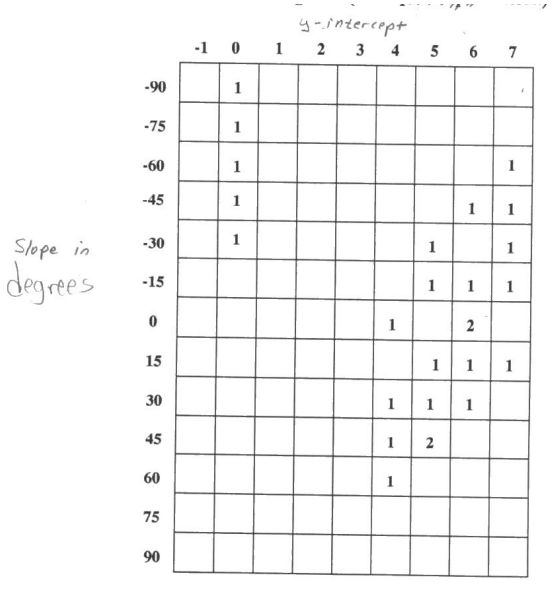
(c) [3] What image pixel would this scene point project to under orthographic projection using this set of camera parameters? Give the coordinates even if they are outside the 500 × 500 CCD array.

$$u = u_0 + k_u x = 200 + (500/10)x = 200 + 0 = 200$$

$$v = v_0 + k_v x = 200 + (500/10)y = 200 + 3750 = 3950$$

Hough Transform





[21] suppose we have the image points (0,0), (2,4), (4,8), (1,6) and (3,6). as seen in fig 2 (a). We use the slope-intercept parameter space (m–c space). The parameter space has been quantized as shown in fig 2 (b).

(a) Write the equation describing the transformation from image space to the m – c parameter space. What is the disadvantage of using this space?

$$y = x \tan m + c$$

Disadvantages:

- The storage space required increases exponentially with the dimension of the parameter space
- Choosing the size of the buckets in the accumulator space can be an issue

(b) Apply the Hough transform algorithm for these points and fill out the accumulators in fig 2 (b). Part of the array has already been filled, so do not modify any of the accumulators with m < 15 or c > 3. Empty spaces mean 0. For your calculations tan 15 = 0.27, tan 30 = 0.58, tan 45 = 1, tan 60 = 1.73 and tan 75 = 3.73.

Please find the answer in notebook file.

(c) Detect and draw the line(s) in the image in fig 2 (a) using the results in the accumulator. Is this the result you were expecting by visually examining the image?

Please find the answer in notebook file.

(d) Could you have calculated that line using least squares? Why? If not, approximately sketch the line(s) you would have expected from least squares.

No, It will be tilted a little by the outlier (1, 6).

[14] **Laplacian**

A discrete approximation of the second derivative $\frac{\partial^2 I}{\partial x^2}$ can be obtained by convolving image $I(x, y)$ with the kernel

1	-2	1

(a) [3] Use the given kernel to derive a 3 × 3 kernel that can be used to compute a discrete approximation to the 2D Laplacian.

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} = I * \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

(b) [2] Apply your 3 × 3 kernel from (a) to compute the Laplacian at the center pixel of the image *I*:

3	10	11
1	4	12
2	3	5

```
(1)(10) + (1)(1) + (1)(12) + (1)(3) + (-4)(4) = 10
```

(c) [3] Is your kernel computed in (a) Separable? Linear? Implementable by a convolution?

This kernel is not separable but it is linear and can be implemented by a convolution.

(d) [2] What is the maximum number of non-zero elements in the convolution of two discrete 1D (one- dimensional) filters of size 1 × M and 1 × N?

M+N-1

(e) [4] Consider the properties of a general second-derivative kernel that is applied to a one-dimensional image. Answer each of the following True or False.

(i) The elements of the mask sum to 0.

True

(ii) Convolution with a constant image is 0 everywhere (ignoring any border effects).

True

(iii) Convolution with an image containing a step edge is 0 everywhere except +1 at the pixel where the step occurs.

False

(iv) Convolution with an image containing an intensity ramp (i.e., the intensity is linearly increasing from the left edge of the image to the right edge of the image) is 0.

True

[10] **Pyramids**

(a) [3] If there are *n* pixels in an input image, give an expression and an upper bound (in the limit) for the total number of pixels in *all other levels* of a Laplacian pyramid.

n/4 + n/16 + ... = n/3. In other words, at most 1/3 more pixels in all levels above the base of the pyramid.

(b) [7] Given that there are more pixels in the Laplacian pyramid than in the original image, why is the Laplacian pyramid considered a compact representation of the original image? Give your answer in two parts:

(i) [5] Give an expression and an explanation of how the original image can be approximately reconstructed from the Laplacian pyramid.

$L_0 + L_1 + \cdots + L_N = (G_0 - Expand(G_1)) + (G_1 - Expand(G_2)) + \cdots + (G_{N-1} - Expand(G_N)) + G_N$,So, first compute $G_N = L_N$. Second, compute $G_{N-1} = L_{N-1} + Expand\,(G_N)$. Continue $L_0 + Expand(G_1)$ until G_0 is computed from

(ii) [2] Explain qualitatively why the storage requirements for the Laplacian pyramid are generally much lower than the storage requirement for the original image, assuming each pixel in the original image requires 8 bits.

Because the values in each image in the Laplacian Pyramid are almost all very close to 0, we can re-quantize the Laplacian images so that each pixel is encoded by its sign plus, say, 2 bits. So, the total storage for the pyramid is approximately $(4n/3) * 3 = 4n$ bits, where *n* is the number of pixels in the original image. On the other hand, the original image requires $8n$ bits.

[15] **Gradient-based Operators for Edge Detection**

(a) [2] Define discrete, integer-valued kernels that can be used for computing $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$. (If you know them, give your answer as the Prewitt operator kernels.)

One of many possible answers are the Prewitt kernels:

$$\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$$

(b) [3] Using your kernels from (a), compute the gradient of the center pixel in the following image, *I*:

3	10	11
1	4	12
2	3	5

The gradient at the center pixel is (22, 14).

(c) [4] Based on your result in (b), overlay the gradient on the above image so that its tail is at the center pixel. Then put X's over those pixels whose gradients should be compared in order to perform non- maximum suppression at the center pixel, and justify why you selected those pixels. (Note: this is a qualitative question, so there is no single correct answer.)

With the tail of the gradient vector at the center pixel, the head of the vector is 22 pixels to the right and 14 pixels up; in other words the vector is pointing in roughly the northeast direction. Non-maximum suppression is done with pixels in the gradient direction, so in this case would include something like the following pixels:

		X
X	X	X
X		

(d) [6] Give two major reasons why the Canny edge operator is generally preferred over the Laplacian-of- Gaussian (LoG) edge operator for most applications.

The Canny operator doesn't round/distort corners and produces better long, thin edge contours because of thinning and hysteresis thresholding.