ICBV HW #2

Dor Litvak Gal Elgavish

Question 1

Section A

In this section we were asked to free Willy the cat from behind the bars.
We can see that a sin wave in the X axis is creating the bars.
We need to find the wave frequency in order to diminish its effect on the image. One can notice that the sin wave complete approximately 20 periods over the horizontal axis (the rows). Moreover, the sin wave has only X axis component without Y.
Hence, if the sin wave complete 20 periods over N columns, than: $T = \frac{N}{20}$ which means that the frequency equals to: $f = \frac{1}{T} = \frac{20}{N}$, than: $f_0 = 20$.
We will use *Two Dimension Fourier Transform* to transform the wave location function to the wave frequency function.
The *Two Dimension Fourier Transform* furmula is as follows:

$F_{2D}(u,v) = F\{f(x,y)\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-(j2\pi(ux+vy))}\,dx\,dy$

And given important transform pairs known:

$f(x) = 1 \iff F(u) = \delta(u)$
$f(x) = \sin(2\pi f_0 x) \iff F(u) = \frac{1}{2j}(\delta(u-f_0) - \delta(u+f_0))$

We can say that the Fourier transformation correspond with the wave creating the bars which preventing willy to go out is:
$F_{2D}(u,v) = F\{f(x,y)\} = F\{\sin(2\pi f_0 x)\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \sin(2\pi f_0 x)e^{-(j2\pi(ux+vy))}\,dx\,dy =$

$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \sin(2\pi f_0 x)e^{-j2\pi ux}e^{-j2\pi vy}\,dx\,dy = \int_{-\infty}^{\infty} \sin(2\pi f_0 x)e^{-j2\pi ux}\,dx \int_{-\infty}^{\infty} e^{-j2\pi vy}\,dy$

Using the known transformations pairs:

$\int_{-\infty}^{\infty} \sin(2\pi f_0 x)e^{-j2\pi ux}\,dx = \frac{1}{2j}(\delta(u-f_0) - \delta(u+f_0))$

$\int_{-\infty}^{\infty} e^{-j2\pi vy}\,dy = \delta(v)$

And over all: $F_{2D}(u,v) = \frac{1}{2j}(\delta(u-f_0) - \delta(u+f_0)) \cdot \delta(v)$

Set $f_0 = 20$

$F_{2D}(u,v) = \frac{1}{2j}(\delta(u-20) - \delta(u+20)) \cdot \delta(v)$

The sin filter as a location function look as follows:

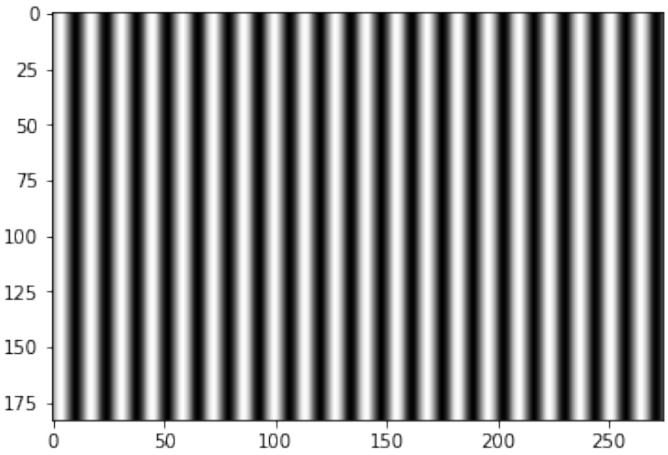


Figure 1: The sin wave illustration.

We Use the amplitude spectrum to locate and equate to zero the Fourier coefficients corresponding to the relevant frequency of the vertical bars. And Free Willy by applying the inverse DFT on the filtered Fourier spectrum using *numpy.fft.ifft2* function.

The following figure present our results:

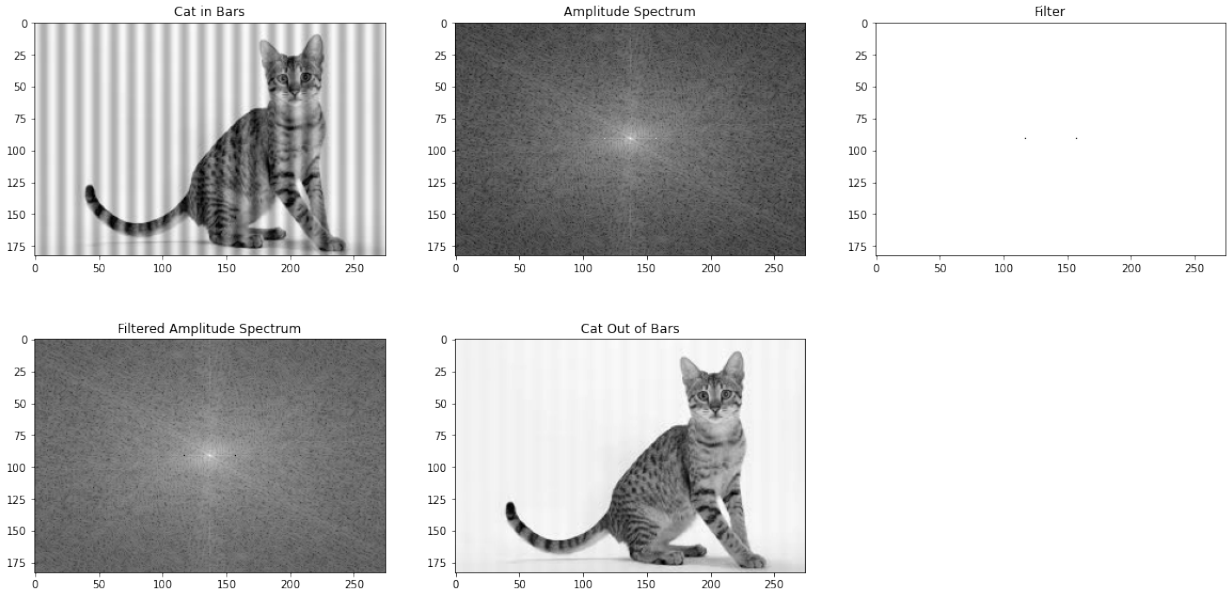


Figure 2: The sin wave illustration.

1. Cat in Bars - the original Willy's behind bars image.
2. Amplitude Spectrum - The full Amplitude Spectrum of Willy's behind bars image. We can wee two white spikes in the middle of the spectrum that suspicious as the bars frequency.

3. Filter - The Filter we found using the Fourier transformation of the sin wave we found fit to the bars. The filter has two spikes of energy located on the X-axis as expected. The two black spikes meaning we want to shut down this frequency.

4. Filtered Amplitude Spectrum - The Amplitude Spectrum of Willy's image after shutting down the bars frequency. As you can see the white spikes that were in the first Amplitude Spectrum image are now back, because we applied the bars removal filter.

5. Cat out of the Bar - The result of the Filter in the location spectrum. As you can see the Bars were removed completely. There are still marks of the bars on the image.
   Why is that?
   We completely shut down the frequency correspond with the bars. But it doesn't mean this frequency wasn't also a part of Willy's image itself and not only the bars. The marks shows that this frequency plays two rules in the image, and removing it does damaged a little but to the original image.

Section B

1.

The **G** kernel is what we called the Gaussian Filter and use especially for blurring. Its impulse response is a Gaussian function. It is considered the ideal time domain filter, just as the sinc is the ideal frequency domain filter. Gaussian blur is a low-pass filter, attenuating high frequency signals.

The **F** kernel, as it appears in the question, performs some kind of inverse (as in making the image negative - black to white and vice versa). We say that after examining the kernel with various images (an example of Willy is given below).
At first we thought that this kernel is performing edge-detection since it has numbers that match second order operators we saw in class and recitation, but after testing it on various images and trying to find the first order operator that this filter is build upon, we came to a conclusion that this is not the case.
But if the kernel was defined: $\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$ it would be suited for edge detection.

2.

The name of the signal image I is the Dirac signal function. It is zero everywhere except of a point mass. Which is exactly what shown in image I. The discrete case of Dirac function is the Kronecker function.

(a) Filter F applied

(b) Filter F and G applied

Figure 3: Willy with filters

3.

$$I * G = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 2 & 4 & 2 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

4.

$$F * (I * G) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -4 & -6 & -4 & -1 & 0 \\ 0 & -4 & -8 & -8 & -8 & -4 & 0 \\ 0 & -6 & -8 & -4 & -8 & -6 & 0 \\ 0 & -4 & -8 & -8 & -8 & -4 & 0 \\ 0 & -1 & -4 & -6 & -4 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

5.

Since convolutions are commutative and associative, we can say that:

$$F * (I * G) = I * (G * F) \Longrightarrow H = G * F$$

$$H = \begin{bmatrix} -1 & -4 & -6 & -4 & -1 \\ -4 & -8 & -8 & -8 & -4 \\ -6 & -8 & -4 & -8 & -6 \\ -4 & -8 & -8 & -8 & -4 \\ -1 & -4 & -6 & -4 & -1 \end{bmatrix}$$

And check if we get the same result as in part 4.

$$F * (I * G) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -4 & -6 & -4 & -1 & 0 \\ 0 & -4 & -8 & -8 & -8 & -4 & 0 \\ 0 & -6 & -8 & -4 & -8 & -6 & 0 \\ 0 & -4 & -8 & -8 & -8 & -4 & 0 \\ 0 & -1 & -4 & -6 & -4 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = F * H$$

Question 2

Section A

Original image:

$$I(x, y) = x + \sin y$$

$\underline{\mathbf{E}_1}$

The gradient's magnitude:

$$I_x = 1, \quad I_y = \cos(y) \Longrightarrow |\nabla I(x,y)| = \sqrt{1+\cos^2 y}$$

$$Set: f(y) = \sqrt{1+\cos^2 y}$$

First derivative of $f(y)$:

$$f'(y) = \frac{-\sin 2y}{2\sqrt{1+\cos^2 y}} \Longrightarrow$$

Extremum at:

$$y = \pi n \quad \forall n = 0, 1, 2 \ldots$$

$$y = \pi n - \frac{\pi}{2} \quad \forall n = 0, 1, 2 \ldots$$

Second derivative of $f(y)$:

$$f''(y) = -\frac{4\cos 2y \cdot (1+\cos^2 y) + \sin^2 2y}{4(1+\cos^2 y)^{\frac{3}{2}}}$$

The second derivative is negative at: $\quad y = \pi n \quad \forall n = 0, 1, 2 \ldots$

The second derivative is positive at: $\quad y = \pi n - \frac{\pi}{2} \quad \forall n = 0, 1, 2 \ldots$

Over all, the edge detector method E_1 finds all the edges of the given function at:
$(x, \pi \cdot n) \quad \forall n = 0, 1, 2 \ldots \quad and \quad \forall x \in \mathbb{R}$

$\underline{\mathbf{E}_2}$

The second derivatives:

$$\frac{\partial^2 I}{\partial x^2} = 0, \quad \frac{\partial^2 I}{\partial y^2} = -\sin y$$

$$\Delta I(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} = -\sin y \Longrightarrow \text{laplacian is zero at:} \quad y = \pi n \quad \forall n = 0, 1, 2 \ldots$$

Over all, the edge detector method E_2 finds all the edges of the given function at:
$(x, \pi \cdot n) \quad \forall n = 0, 1, 2 \ldots \quad and \quad \forall x \in \mathbb{R}$

Section B

In this section we were asked to detect the edges of the coins present in the attached *coins.jpg* image.
We first used gaussin blur to smooth the image, then calculated the LoG (the laplacian of gaussian) for the image.
Then we used double threshold values **a** and **b** such that $\mathbf{a} \geq 0 \geq \mathbf{b}$. And did zero-crossing only for pixels that passed the thresh-holding.
The following figure present our results:

1. Original - The original greyscale coins image.

2. Smoothed - The result after applying the Gaussian filter over the original image. You can see that the image is more smooth. We used kernel size (7,7) and set sigma to be 2.

3. LoG - The result of the smoothed image after applying the laplacian kernel. You can see the contrast between color changes.

4. Zero Crossing - The LoG image after applying zero crossing double threshold. We scaled $a = 0.0035$ and $b = -0.006$.

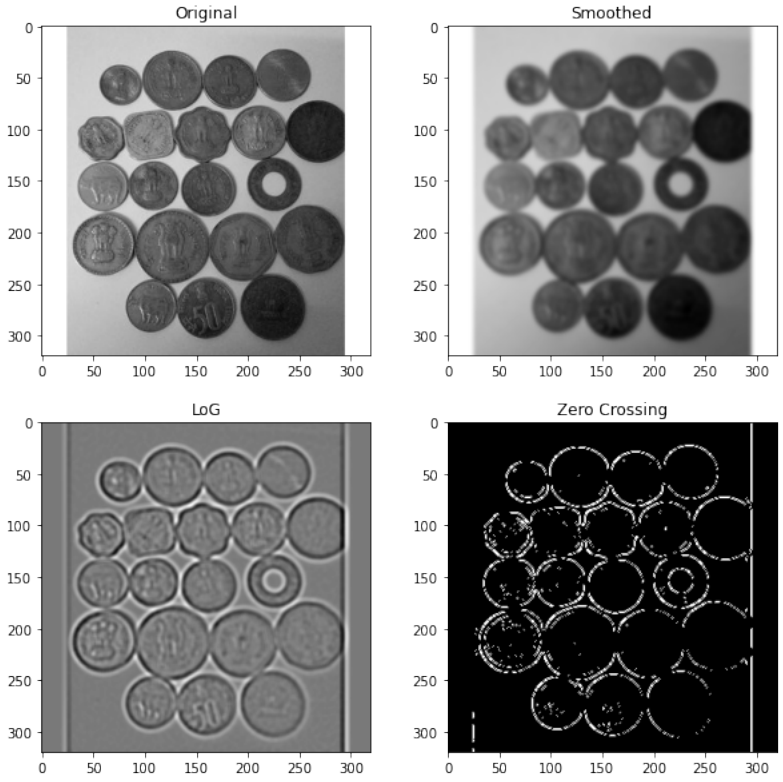


Figure 4: Edge detector for *coins.jpg*.

Section C

In this section we were asked to preform two edge detection methods over a noisy version of the coins image.

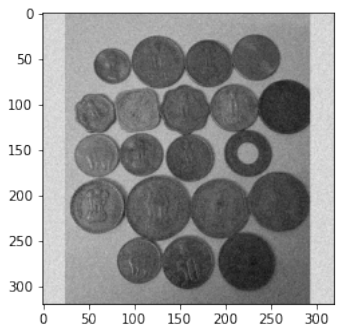


Figure 5: Coins image plus some gaussian noise.

We used two methods for edge detection the first one is the zero-crossing method we created in section B, and the second one using OpenCV's implementation of Canny's algorithm.
About Canny's algorithm: Canny Edge Detection is a popular edge detection algorithm. It was developed by John F. Canny in 1986. It is a multi-stage algorithm, the stages are as follows:

1. Noise Reduction - Remove the noise in the image with a Gaussian filter.
2. Finding Intensity Gradient of the Image - Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (G_x) and vertical direction (G_y). From these two images, we can find edge gradient and direction for each pixel.
3. Non-maximum Suppression - After getting gradient magnitude and direction, a full scan of image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, pixel is checked if it is a local maximum in its neighborhood in the direction of gradient. The result you get is a binary image with "thin edges".
4. Hysteresis Thresholding - This stage decides which are all edges are really edges and which are not. For this, we need two threshold values, minVal and maxVal. Any edges with intensity gradient more than maxVal are sure to be edges and those below minVal are sure to be non-edges, so discarded. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. If they are connected to "sure-edge" pixels, they are considered to be part of edges. Otherwise, they are also discarded. This stage also removes small pixels noises on the assumption that edges are long lines.

The following parameters scaling got the best edge detection results:

Our zero-crossing method - We set $a = 0.000035$ and $b = -0.009$

Canny's algorithm - We set $canny_{t1} = 200$ and $canny_{t2} = 400$

Why is Canny's Algorithm did better them ours?
Canny's Algorithm contain another step that our algorithm doesn't have. It is the Non-maximum Suppression stage. After getting gradient magnitude and direction, a full scan of image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, pixel is checked if it is a local maximum in its neighborhood in the direction of gradient. The result you get is a binary image with "thin edges". As you can see Canny's algorithm produce thinner lines in the output then ours. And a better representation of the thin butteries lines due to the removal of pixels that does not consist with gradient direction. Because

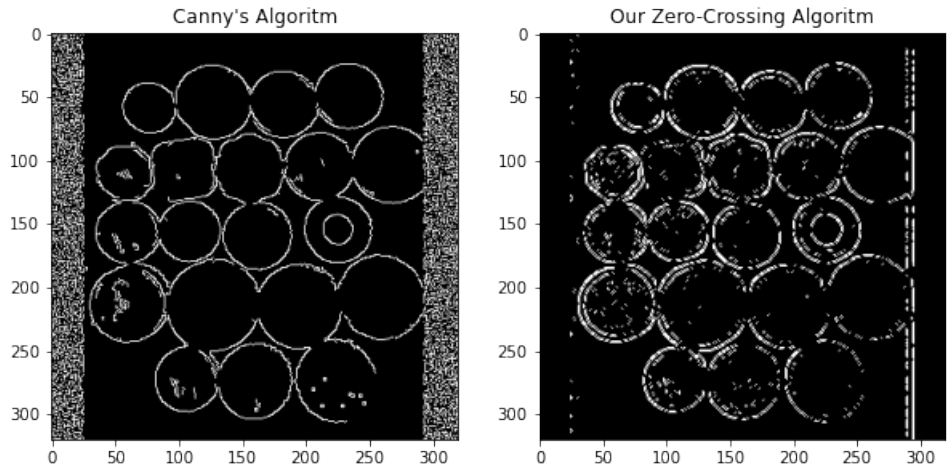


Figure 6: On the left the Canny's algorithm edge detection result, on the right our zero-crossing algorithm result.

of the 3rd stage at Canny's algorithm the 4th stage also is does better job then in our method.The result is a better edge detection tool.

Question 3

Section A

A line in 3D is described by the following parametric description:

$$\vec{\alpha(t)} = \begin{bmatrix} x_0 + t \cdot a_1 \\ y_0 + t \cdot a_2 \\ z_0 + t \cdot a_3 \end{bmatrix}$$

In order to get the parametric description of those points on the x-y image plane we perform:

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = K \cdot F \cdot \begin{bmatrix} \vec{\alpha(t)} \\ 1 \end{bmatrix} = \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{bmatrix} = \begin{pmatrix} \frac{\tilde{x}}{\tilde{z}} \\ \frac{\tilde{y}}{\tilde{z}} \end{pmatrix}$$

And because we are performing linear transformations on $\vec{\alpha(t)}$, the result coordinates can be described as:

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \begin{bmatrix} x'_0 + t \cdot b_1 \\ y'_0 + t \cdot b_2 \\ z'_0 + t \cdot b_3 \end{bmatrix} = \begin{pmatrix} \frac{x'_0 + t \cdot b_1}{z'_0 + t \cdot b_3} \\ \frac{y'_0 + t \cdot b_2}{z'_0 + t \cdot b_3} \end{pmatrix}$$

Now, in order to prove that the curve on the x-y plane is also a straight line, we will show that the slope for some t_1, t_2 is constant:

$$\frac{\Delta x}{\Delta y} = \frac{\hat{x}(t_2) - \hat{x}(t_1)}{\hat{y}(t_2) - \hat{y}(t_1)} = \frac{\frac{x'_0 + t_2 \cdot b_1}{z'_0 + t_2 \cdot b_3} - \frac{x'_0 + t_1 \cdot b_1}{z'_0 + t_1 \cdot b_3}}{\frac{y'_0 + t_2 \cdot b_2}{z'_0 + t_2 \cdot b_3} - \frac{y'_0 + t_1 \cdot b_2}{z'_0 + t_1 \cdot b_3}}$$

After (many ☺) algebraic simplifications, we are ended up with:

$$\frac{\Delta x}{\Delta y} = \frac{b_1 \cdot z'_0 - b_3 \cdot x'_0}{b_2 \cdot z'_0 - b_3 \cdot y'_0} = \text{constant}$$

So, we proved that a perspective image of a straight line in the 3D world is a straight line in the image plane, by showing that its slope is constant between some two points at t_1, t_2.

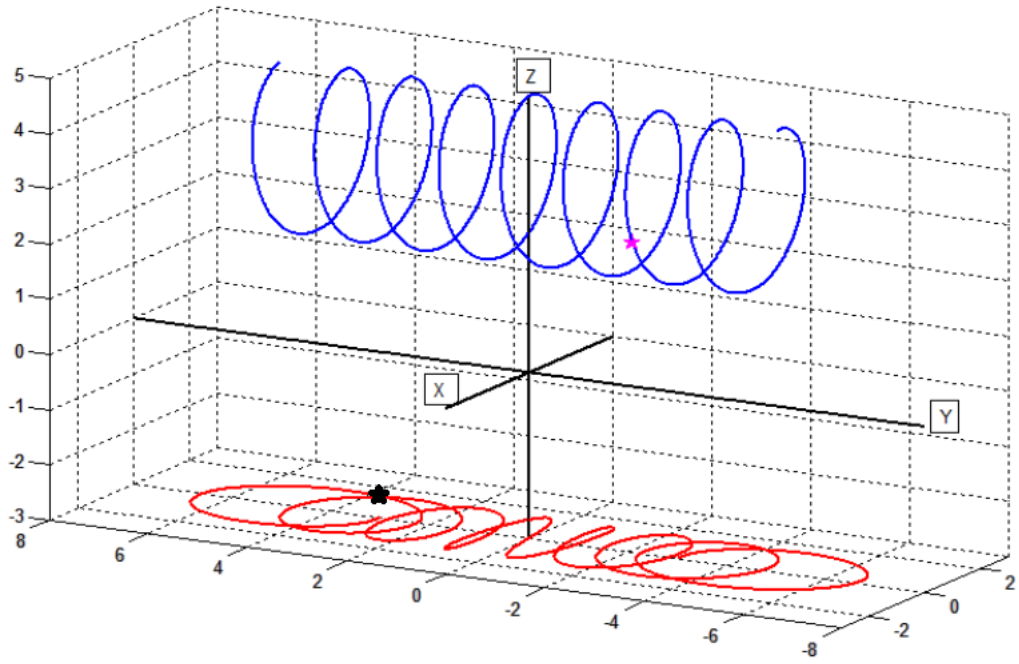
Section B

(1)

The blue object is a parametric curve since its a graph (a helix) that can be described by the following continuous functions of s on an interval:

$$\alpha(s) = (k + a \cdot \sin s, l + b \cdot s, h + a \cdot \cos s)$$

(2)



(3)

The parametric equation of the helix is:

$$\alpha(s) = (k + a \cdot \sin s, l + b \cdot s, h + a \cdot \cos s)$$

z-coordinates:

$$z(s) = h + a \cdot \cos s$$

We are given that:

$$z(s) \in [2, 5], \quad z(0) = 5, \quad z(-8\pi) = 5$$

So we can conclude:

$$a + h = 5, \quad a = \frac{5-2}{2} = \frac{3}{2} \longrightarrow h = \frac{7}{2}$$

And finally:

$$z(s) = \frac{7}{2} + \frac{3}{2}\cos s$$

x-coordinates:

$$x(s) = k + a \cdot \sin s$$

We are given that:

$$x(0) = 0, \quad x(-8\pi) = 0$$

So we can conclude:

$$k + \frac{3}{2}\sin 0 = k + \frac{3}{2}\sin -8\pi = 0 \longrightarrow k = 0$$

And finally:

$$x(s) = \frac{3}{2}\sin s$$

y-coordinates:

$$y(s) = l + b \cdot s$$

We are given that:

$$y(0) = 0, \quad y(-8\pi) = \frac{-8\pi}{5}$$

So we can conclude:

$$l + b \cdot s = l + b \cdot 0 = 0 \longrightarrow l = 0$$

$$l + b \cdot s = 0 + b \cdot (-8\pi) = \frac{-8\pi}{5} \longrightarrow b = \frac{1}{5}$$

And finally:

$$y(s) = \frac{1}{5}s$$

To conclude the curve is defined:

$$\alpha(s) = \begin{pmatrix} x^{(\alpha)}(s) \\ y^{(\alpha)}(s) \\ z^{(\alpha)}(s) \end{pmatrix} = \begin{pmatrix} \frac{3}{2}\sin s \\ \frac{1}{5}s \\ \frac{7}{2} + \frac{3}{2}\cos s \end{pmatrix}$$

(4)

Projection on the image plane (x-y plane):

$$\begin{pmatrix} x^{(\beta)}(s) \\ y^{(\beta)}(s) \end{pmatrix} = K \cdot F \cdot \begin{bmatrix} x^{(\alpha)}(s) \\ y^{(\alpha)}(s) \\ z^{(\alpha)}(s) \\ 1 \end{bmatrix} = \begin{bmatrix} -3 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot I \cdot \begin{bmatrix} \frac{3}{2}\sin s \\ \frac{1}{5}s \\ \frac{7}{2} + \frac{3}{2}\cos s \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{9}{2}\sin s \\ -\frac{3}{5}s \\ \frac{7}{2} + \frac{3}{2}\cos s \end{bmatrix} = \begin{pmatrix} \frac{-18\sin s}{14+6\cos s} \\ \frac{-6s}{35+15\cos s} \end{pmatrix}$$

So, the curve for $\beta(s)$ is:

$$\beta(s) = \begin{pmatrix} x^{(\beta)}(s) \\ y^{(\beta)}(s) \\ -3 \end{pmatrix} = \begin{pmatrix} \frac{-18\sin s}{14+6\cos s} \\ \frac{-6s}{35+15\cos s} \\ -3 \end{pmatrix}$$

Question 4

In this section we will discussed whether the frog's visual system, is a linear system. Why the frog? The frog visual system is relatively simple. His eyes do not move, as do ours, but also are actively stabilized. He has no fovea, or region of greatest acuity vision. He also has only a single visual system, retina to colliculus, not a double one such as ours.
The experiments finding shows there are four separate operations on the image in the frog's eye. Each has its result transmitted by a particular group of fibers, uniformly distributed across the retina, and they are all nearly independent of the general illumination.
The operations are:

- Sustained-contrast detection
- Net convexity detection
- Moving-edge detection
- Net dimming detection

We will prove non-linearity for each of the operations independently.

1. **Non-linearity of sustained-contrast detection -** [Page 241] In the paper they made several experiments on a single fiber, and compare these responses with full illumination 60 watt and with 1/300 light. they stated that both experiments with both 60 watt and 1/300 light gave almost the same response. A result of for sure not a linear system. A linear system get only one value (or none) for each of the y in the range.

2. **Non-linearity of net convexity detection -** [Page 244] In figure 3, they present 8 experiments, we will discuss the first two. In the first experiment A, shown in figure 3(a) a 1° block disk is imported into the receptive field and left there. In the second experiment B shown in figure 3(b) the same as is event occurs, but now the light is turned off then on again. The spikes graph shows that after the light turned of then on the spiked number and frequency decreased significantly. Although, in linear system we would expect to the same response as before since the values are back to "normal". Another example for this part is the figures c and d and although there is a difference in the multiplication we get the same response. Different from what we would expected from a linear system.

3. **Non-linearity of moving-edge detection -** [Page 245] In the paper they presented an experiment Figure 4g of moving edge detector, a 7° block disk passed thought the receptive field under dim light, with an ON OFF response of half a second. The second experiment is shown in Figure 4h the same disk is moving in dim light but this time an ON OFF response happens for two seconds. Mark the results as $f(first) = x$ and $f(second) = y$, we would expected in linear system that if $second = 4 \cdot first$. The response would be $f(second) = 4 \cdot f(first)$, but as you can see in figure 4, this not what happening, instead, there is approximately $f(second) = 2 \cdot f(first)$ which mean it is not a linear system.

4. **Non-linearity of net dimming detection -** [Page 247] In the paper they preform the following experiment, suppose they turn of the light, and set up a prolong response, mark it as $f(off) = x$. If they darken the general lightning by a factor of 100 they got a prolonged discharge too, mark it as $f(100) = x$. However, if they turned off the light completely a few seconds after the 100/1 dimming and then turn it back on to the same dim level, the discharge is increasing by the second dimming and is completely or almost abolish by relighting.

   Lets mark the darken the general lightning by a factor of 100 and then turned off the light response as $f(100 + off) = y$. We also know that the discharge is increasing by the second dimming, then $y > x$. Another thing we know is that after relighting the discharge increasing completely or almost abolish, let make it as $f(100 + off + 100) = x$

   The system is a non-linear system because it is not preserve the addition rule: $f(100 + off) = y$ and $f(100) = x$ and $y > x$ but $f(100 + off + 100) = x \neq x + y = f(100 + off) + f(100)$