Assignment 5

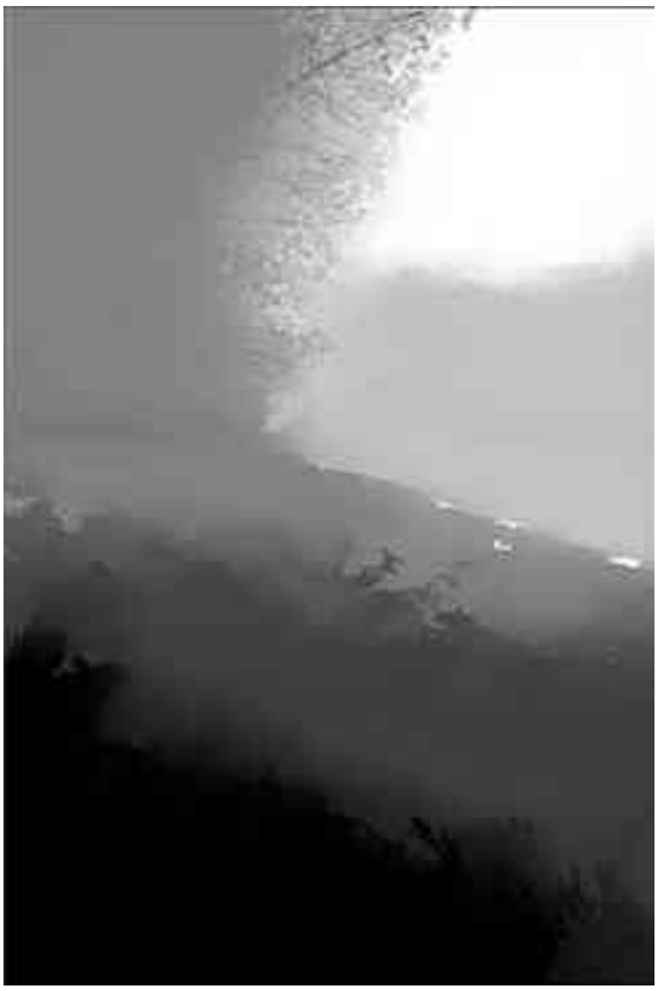
Digital Image Processing

Defocus blur from Depth

Objects at different depths, are blurred at different levels. Given the depth map, generating the blurred image is the task in this assignment.

So, if we can find the blur radius of every pixel from the depth map, we can reconstruct the defocused blurred image.

Depth map given is a grey image as shown below. Higher intensity pixels (bright pixels) have larger depth than the lower intensity pixels (darker pixels) having lower depth.



The depth map corresponds to the following image:

blur_radius = aperture size * image distance * abs((1/object distance)-1/(depth_img + 0.01))

sigma_image = rho * blur_radius

The +0.01 is to avoid division by zero error. At some pixels, depth map can give 0 value.

Rho is fixed for a given for a known f(focal length) and v(image distance). The units of Rho are pixels / cms because blur radius is measured in cms.

Analysis:

1. We are computing range filter here. So, for a given x, as variance increases, normal_distribution_value(x) decreases and as variance decreases, normal_distribution_value(x) increases.

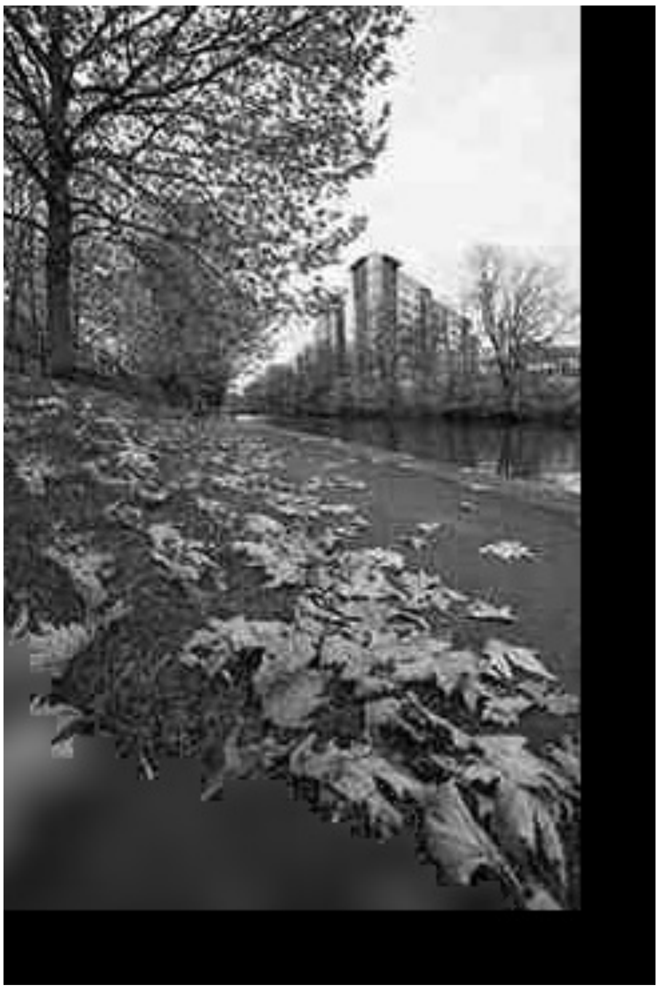
2. From the above formula, we can see that, at a particular pixel, as the depth increases, sigma decreases, normal_distribution_value(x) increases. When depth increases, blur increases. For objects at higher depth, pixels farther from the current pixel gets higher weight in the kernel. Similarly, for objects at lower depth, pixels farther from the current pixel gets lower weight.

3. While constructing the image, we are considering that, for pixels at higher depth, farther neighbours also carry important information. While, for pixels at lower depth, farther neighbours carry less information.

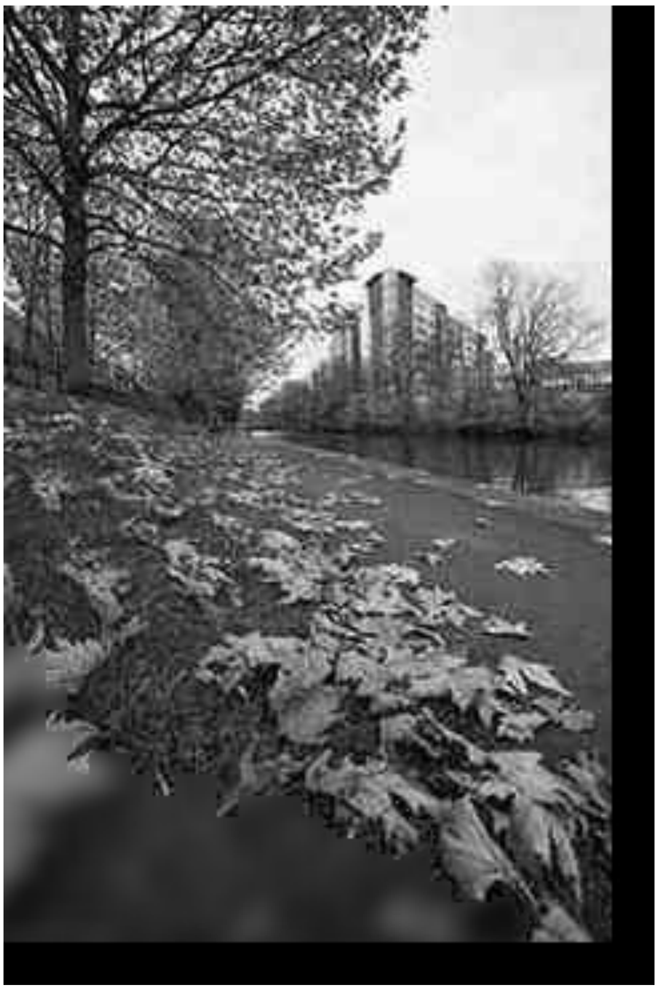
4. Its called **Space Variant blur.** Same filter is not used every where. Filter changes depending on the depth of the pixel. If depth is same for every pixel, then it is just a convolution operation with gaussian filter.

4. As the v(image distance) decreases, blur radius decreases. Keep the **rho, u and r** fixed. Vary the **v**.

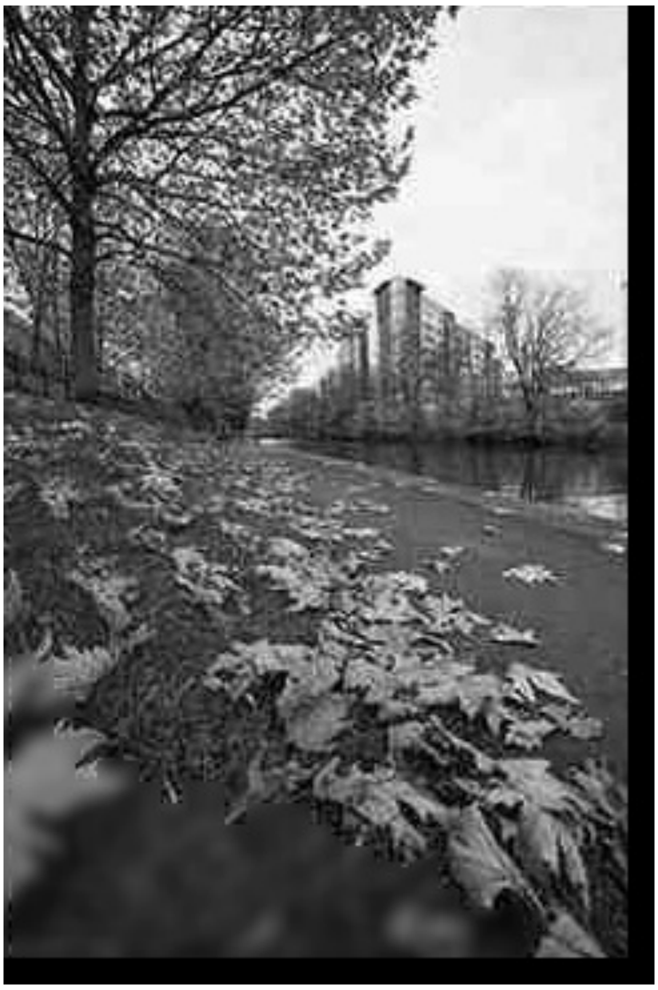
rho = 1, u = 15, r = 0.2, **v = 0.5:**



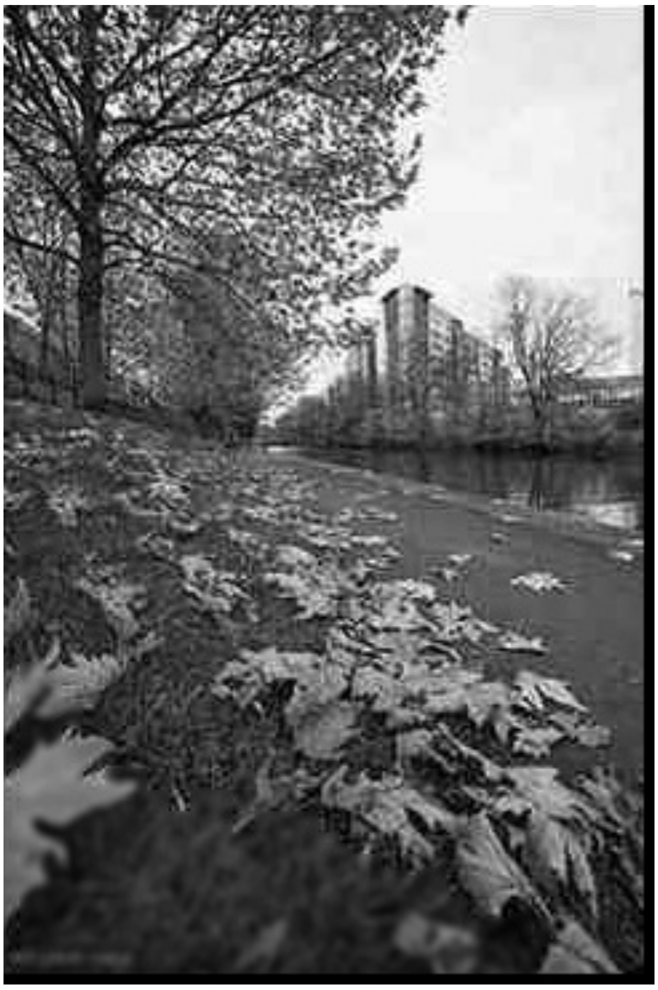
rho = 1, u = 15, r = 0.2, **v = 0.3:**



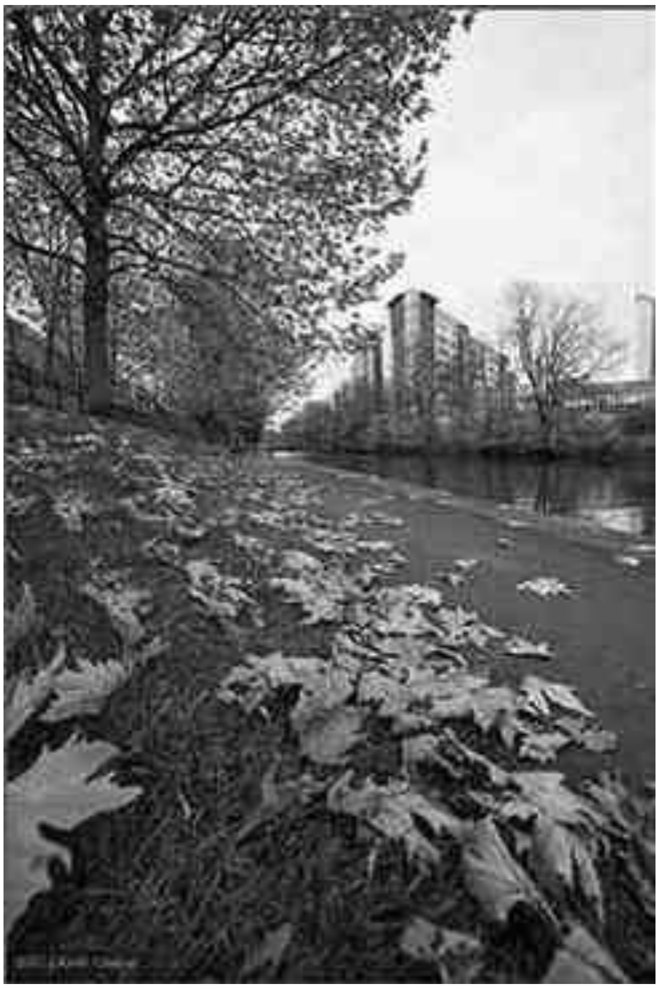
rho = 1, u = 15, r = 0.2, **v = 0.2:**



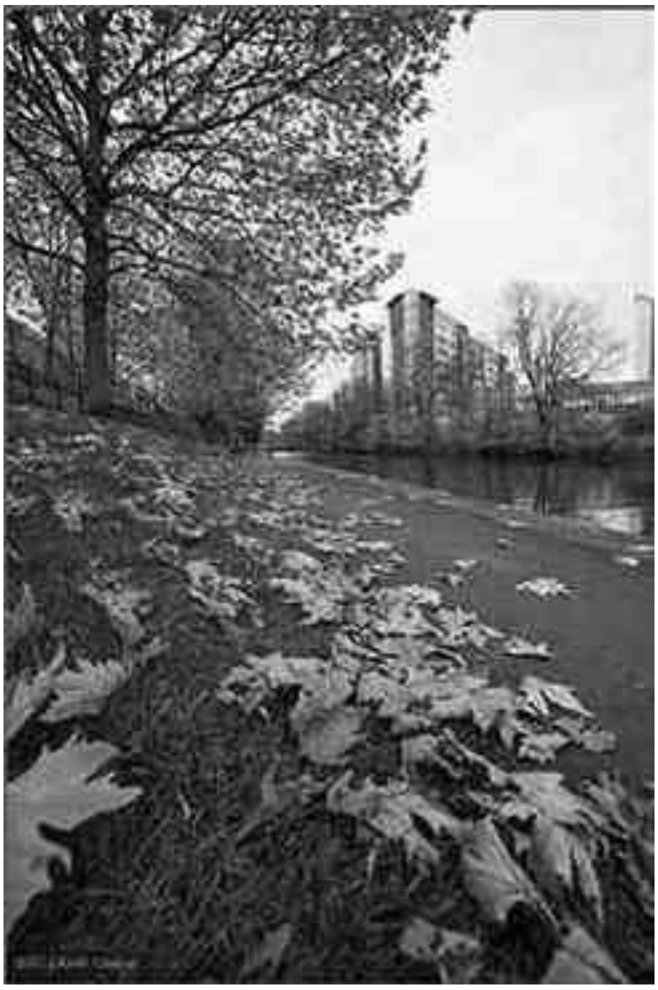
rho = 1, u = 15, r = 0.2, **v = 0.1:**



rho = 1, u = 15, r = 0.2, **v = 0.01:**



rho = 1, u = 15, r = 0.2, **v = 10^-4:**



The clarity in this image is very close to the original image.

Space Invariant Deblurring using Gradient Descent

In this assignment, original image must restored, given a blurred image using Gradient Descent method.

For this, first Emma.jpg was resized to 128*128 pixels. Image is blurred using a smoothening filter (H) like below.

[0, 0.5,0]

[0.5,1,0.5]

[0,0.5,0]

Y = Hx : x is the original image, H is the filter used and Blurred image is obtained.

Now, there are 2 objectives in the process of restoring x from known Y and H:

I. Compute H` from the filter using the Toeplitz matrix technique.

II. Recursively apply gradient descent method to restore x from known Y and H`.

Implementation trick:

I. Compute H` from the filter using the Toeplitz matrix technique.

1. x is a 128 * 128 matrix. First it must be vectorised into 128^2 * 1 vector.

2. After smoothening, Y will also be a 128 * 128 matrix which must be vectorised into 128^2 * 1 vector.

3. Using the Toeplitz matrix technique, find the H` which will be a 128 * 128 matrix, so that it is compatible for multiplication. See the H` matrix below.

```
[[  0.   67.   66.  ... 115.   98.5   0. ]
 [ 67.  267.  331.  ... 571.5 416.5  98.5]
 [ 67.  333.5 399.5 ... 702.  538.  104.5]
 ...
 [ 93.  464.5 555.5 ... 720.  600.  120. ]
 [ 93.  371.5 461.5 ... 600.  480.  120. ]
 [  0.   93.   92.5 ... 120.  120.    0. ]]
```

To cross check if the obtained H` matrix is correct, compute Y` using the vectorised image and the above H`. Compare it with original Y obtained by convolving x with 3 * 3 filter H.

If both of them turn out to be same, then obtained H using the above technique is correct.

Following python snippet can be used to do it:

```
from scipy import signal
lib_output = signal.convolve2d(I, F, “full")
```

II. Recursively apply gradient descent method to restore x from known Y and H`.

1. E = ||Y - Hx||^2 + f(x)
2. X(t+1) = X(t) - eta * (Delta E).
3. Compute E at X(t+1). if X(t+1) < delta, then stop.

Find (Delta E) from 2, substitute X(t+1) in 1 and find energy for the next iteration.

Delta E is the prior term and should be differentiable in case of Gradient Descent method.

If eta is too large, oscillations will be created in the local minima / maxima.

If eta is too small, it will take very long to converge.

So, initially reduce eta in larger steps and then reduce it in smaller steps.