Computational Photography
Assignment 2

Divya K Raman
EE15B085

17 March 2019

1 Introduction

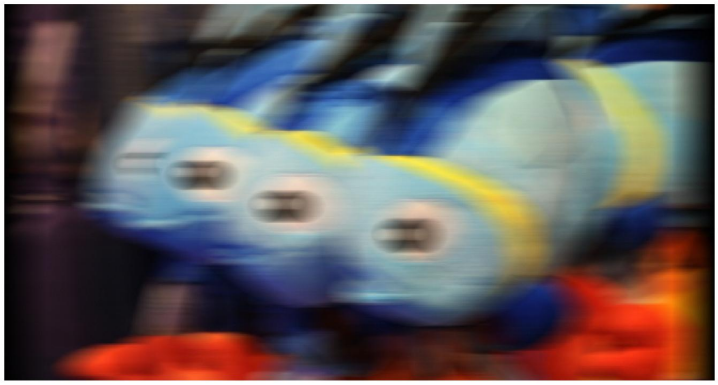
This assignment deals with motion deblurring. In the first part, we simulate experiments for motion deblurring of an image taken with a conventional camera and report results. The second part deals with motion deblurring with flutter shutter and the third part has experiments on deblurring with motion-invariant photography. The code is written in matlab.

2 Motion deblurring with conventional camera

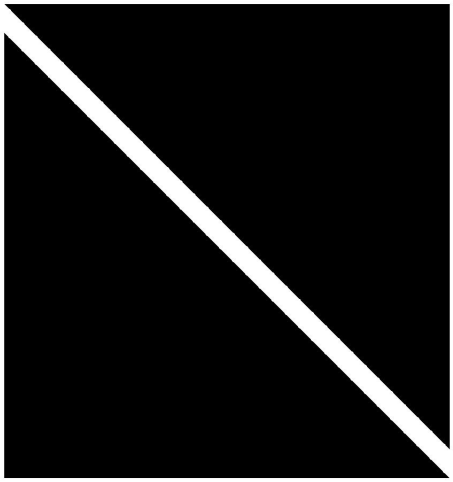
The code for this section is in Q1.m.

Given a clean image 'fish.png', we simulate the scene captured using a static conventional camera with the aperture kept open for the full exposure time which is 52 seconds. At time t = 0, the image has zero translation. Then,the whole image as seen by the camera moves at 1 pixel per second to the right(horizontal translation).

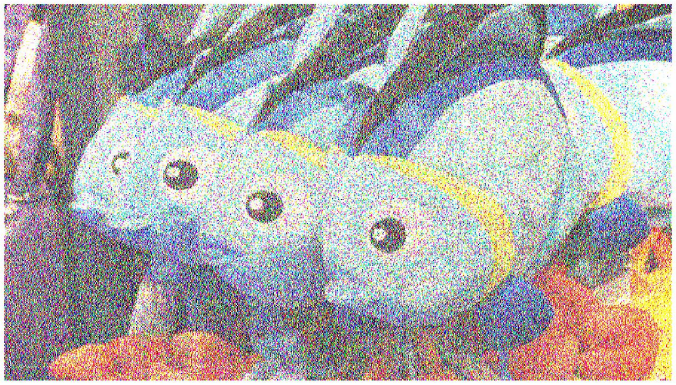
The blurred image is simulated and gaussian noise from gaussNoise.mat is added.The resulting image is as below.



The blur matrix corresponding to the horizontal translation described above is plotted below.



This blur matrix is used to deblur the image using least squares. The deblurred image is as below.



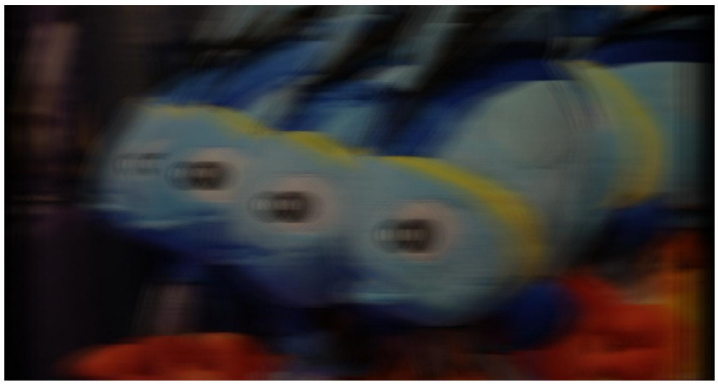
The deblurred image is noisy due to the gaussian noise that is added in the first step. The rmse between the clean image and this deblurred image is 0.4971(in the normalised scale) which corresponds to that of gaussian noise.

3 Motion deblurring with flutter shutter

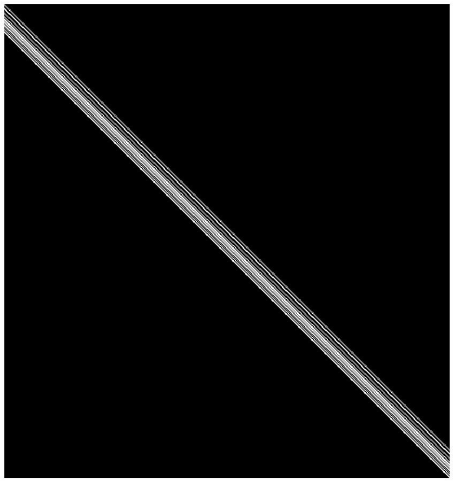
The code for this section is in Q2.m.

We simulate a few experiments just like the previous section, but with a flutter shutter camera whose code is provided.

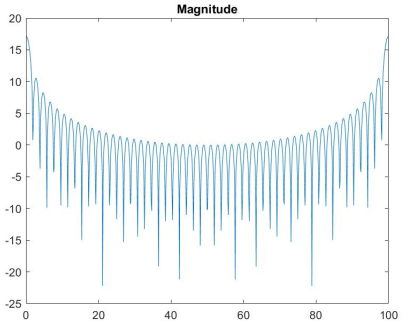
Blurred image with noise for the given exposure code:



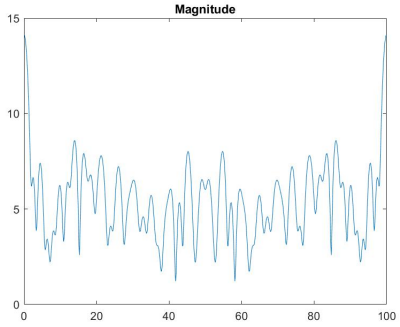
Blur matrix



DFT of the conventional code:



DFT of the flutter shutter code:



(We use the first column of the respective A matrices)

The DFT of the conventional code has a lot of zeros but the DFT of the flutter shutter code doesn't have any zeros. This makes it easy to invert while deblurring. Also, the conventional code's DFT has a lot of sudden fluctuations. The changes are more slow and gradual in flutter shutter code's DFT. This makes it more stable.

Deblurred output of image taken using conventional camera(no noise image):

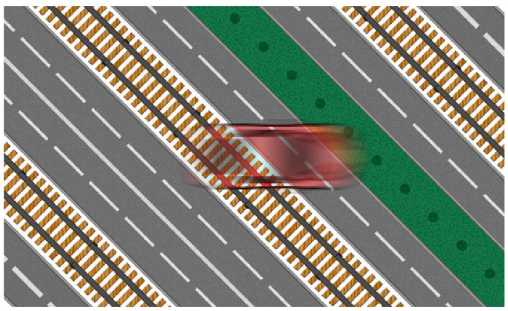
Deblurred output of image taken using flutter shutter camera(no noise image):

The first image is better in quality than the second image. This is probably because of large shifts in between consecutive frames which are added up in the flutter shutter case to give the final output - 0s in the code signify a blank frame at that instant. Also, the code needs to be optimal to generate good results. This code is not an optimal code. This leads to ringing effect caused by distortion or loss of high frequency information in image and dark vertical lines in the image.

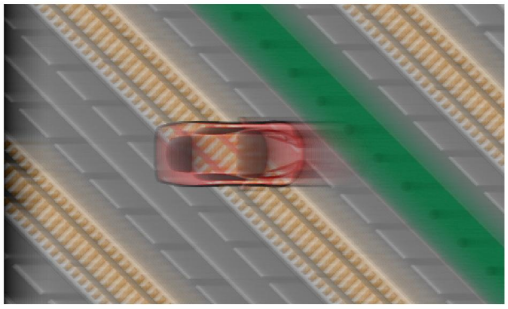
4 Deblurring with motion-invariant photography

The code for this section is in Q3.m.

Assuming static camera and the velocity of the car(redcar.png) as 1 pixel per second to the right relative to a static camera, we generate the blurred image captured for an exposure time of 52 seconds. The background(background.png) is static.

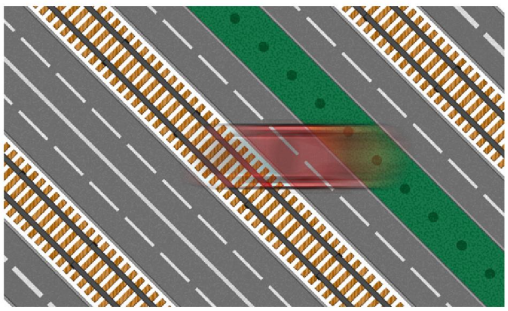


We now consider the same scenario captured using motion invariant photography (i.e. the camera also moves during the exposure). The translational values of the static background due to camera motion at each second are taken as in CameraT.mat. The blurred image captured in this case with both camera and object motions for an exposure time of 52 seconds is captured.

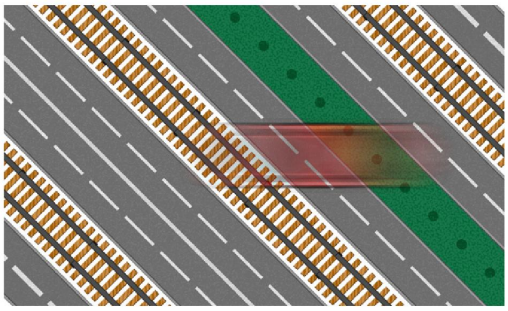


In the second image, the entire scene is blurry. However, only the car is blurry in the first image. Thus, in the second case, the entire image can be deblurred with a single kernel. We don't need to segment and apply different blur kernels to different parts of the image as we have captured a motion invariant blurred image. Thus, the second image is easier to work with.

Object velocity: 2 pixels/second:

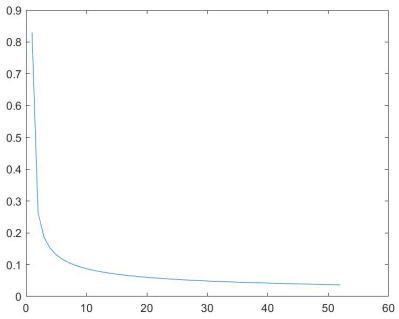


Object velocity: 3 pixels/second:

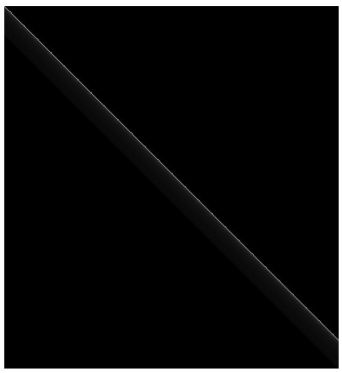


The image with object velocity 3 pixels per second is more blurred than the image with object velocity 2 pixels per second. This is because the object moves faster and hence there is more motion within the same timeframe resulting in more blur.

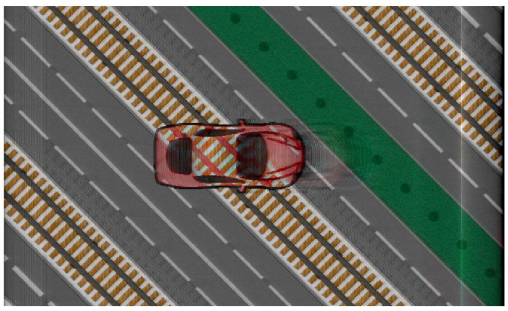
We now plot the PSF of the motion invariant blur image. The PSF is taken to be a 1D function. If x is the camera translation then PSF weight at x is 1 ax. x = [0.1,1,2,...,51] and a = 1/13 . The psf is normalised after assigning these values.



Blur matrix A corresponding to the above PSF:



The image is now deblurred using least squares. The output is given below.



The image is slightly dark and has vertical lines. This is because we add a column of zeroes to the original image to match dimensions for easy pseudo

inverse calculations while deblurring using least squares. The quality of the output is very good otherwise.