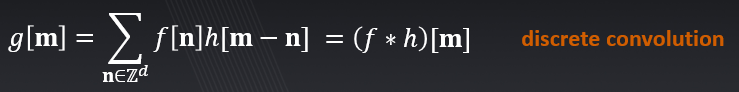
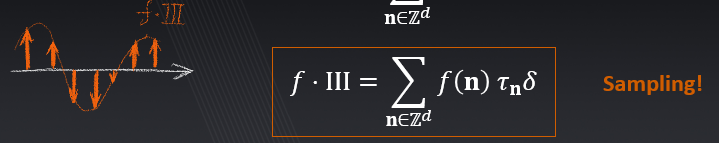
1. Let f(x) be the image that would be formed on the image plane if the camera were  
   absolutely still. The actually formed digital image of the k-th frame can be described as





where hk(x) is the effective PSF (point spread function) corresponding to the k-th frame. Express hk in terms of the camera motion.

Answer:

PSFs are obtained by sampling the continuous trajectory o(t) on a regular pixel grid using subpixel linear interpolation. We have o(t) and we want to get o[n], let T be 1/f, so:

**2.** Express the Fourier transform of *hk*.

**3.** Show that the action of the camera can be expressed fully in the digital domain as





Write an expression for the frequency response of the discrete kernel *Pk*.

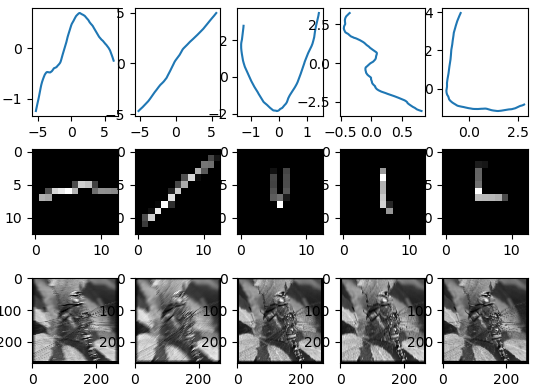
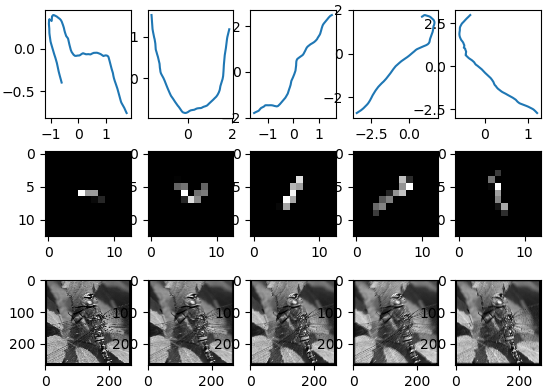
**4.** Write an upper bound on |Pk[]|.

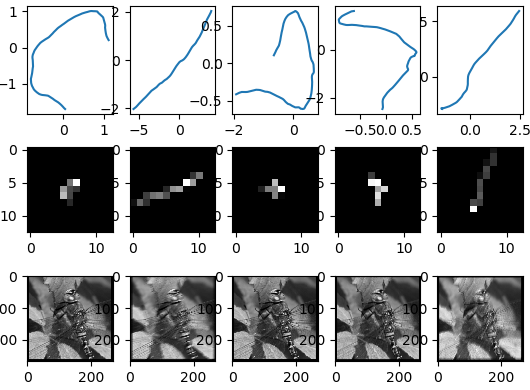
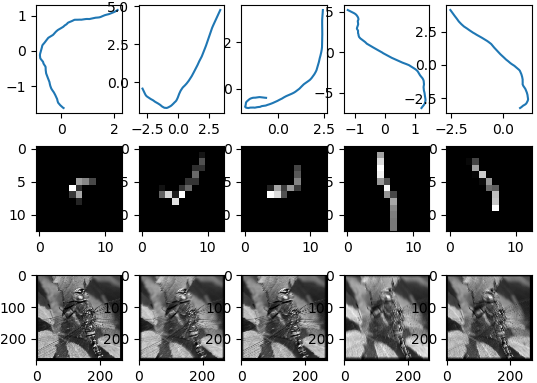
**5.** Assume that on the interval *τ 2* [*k; k* + 1), the camera moves with constant velocity *v*in the horizontal direction, **o**(*τ*) = (*τ - k -* 0*:*5) *v* **e**1, where **e**1 = (1*;* 0*;* 0)T. Express *Pk*[] in  
this case. How does |Pk[]|depend on *v*?

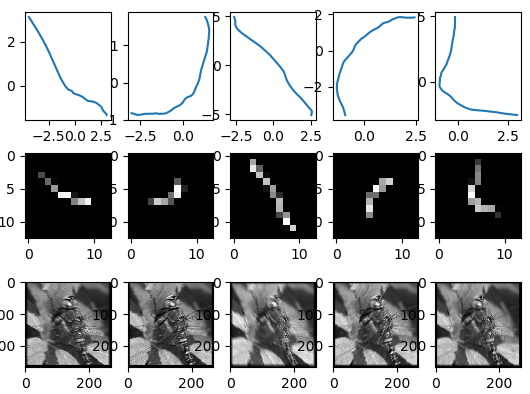
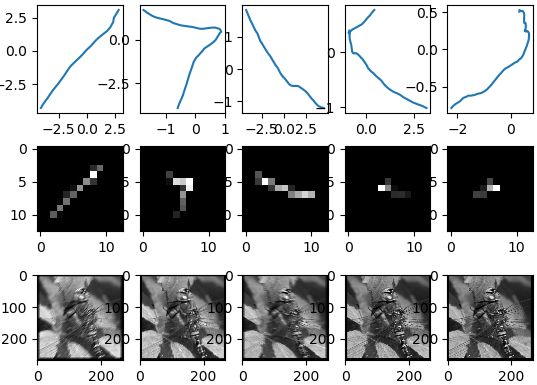
Wet part

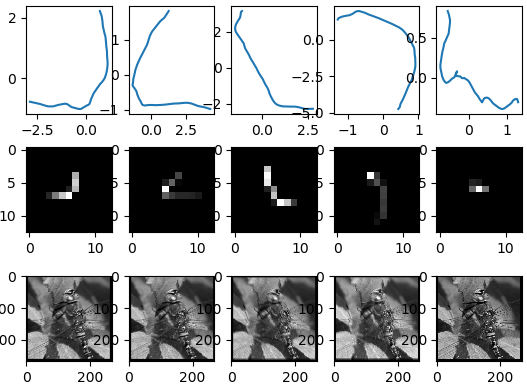
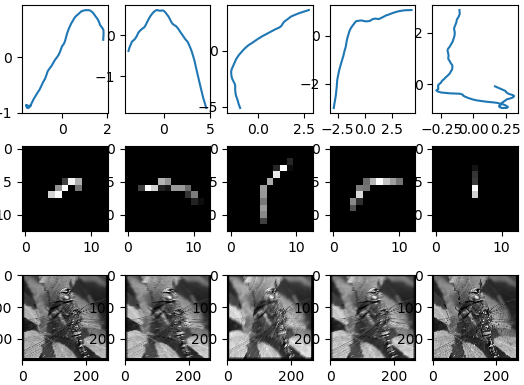
# Part 1 – Blurred images generation

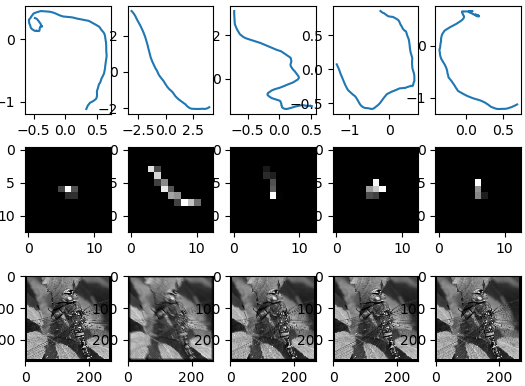
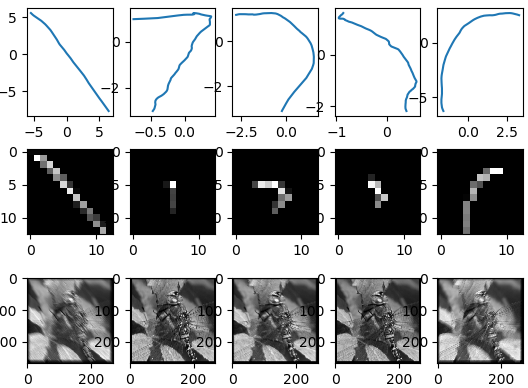
In this part we generated the PSFs by simply adding a positive value to corresponding pixel in a square zero kernel, according to round{(x,y)} values of a given trajectory. We used a 13x13 kernel as all trajectories bounded by those values. Here is a list of all trajectories, corresponding PSFs and blurred images:

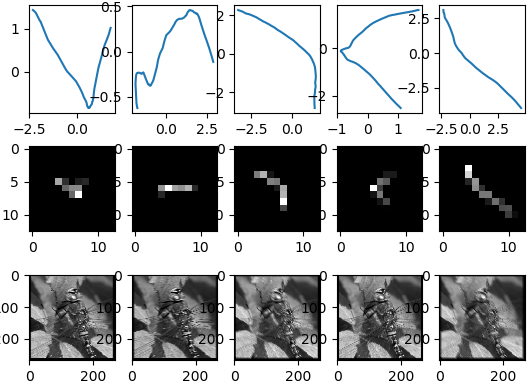
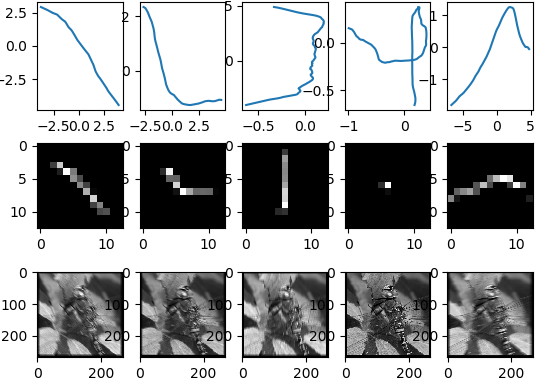


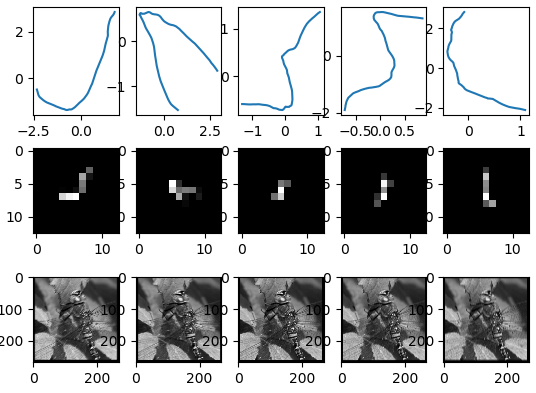
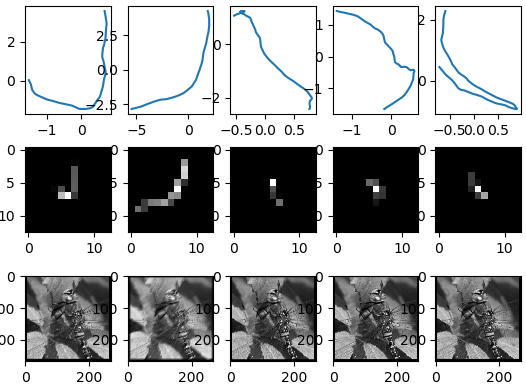


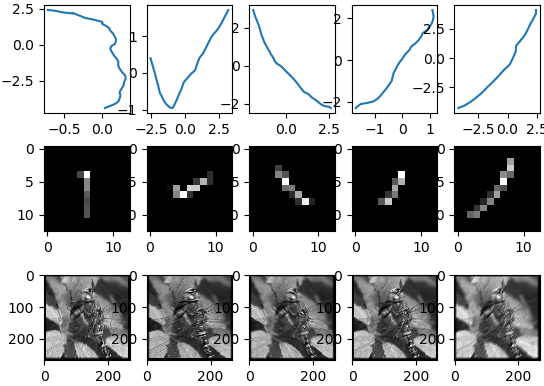
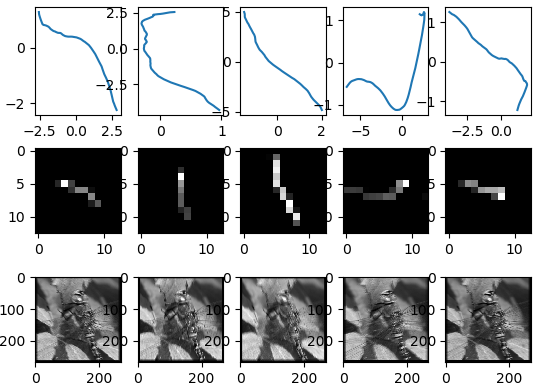


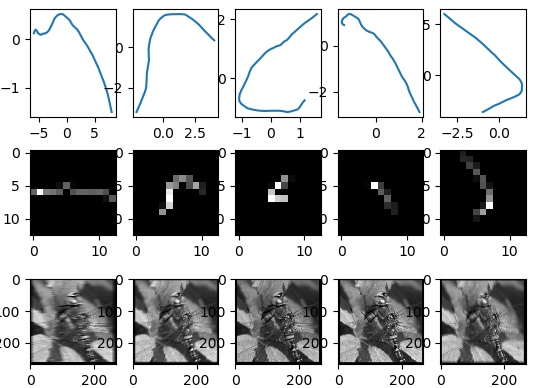
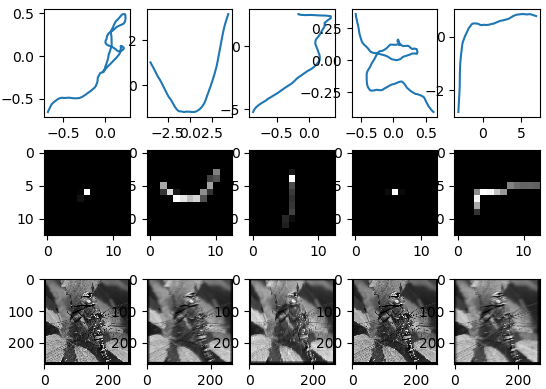


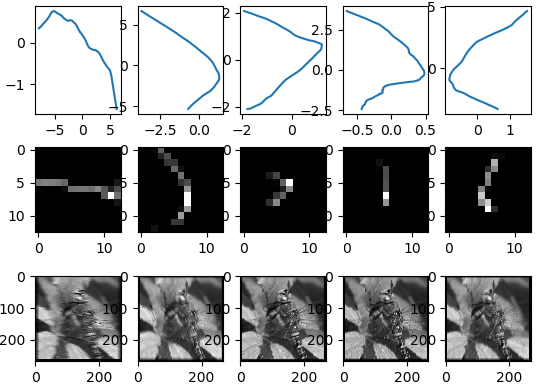
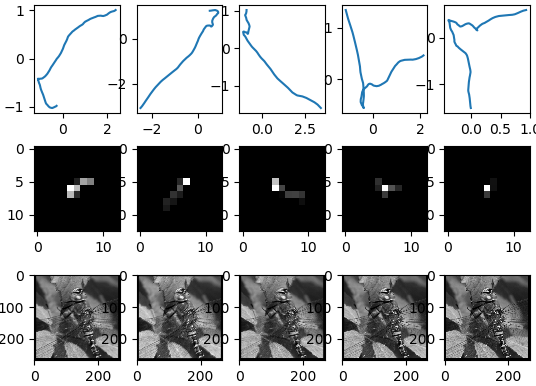






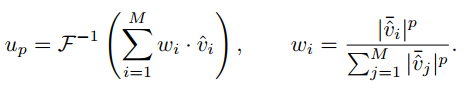




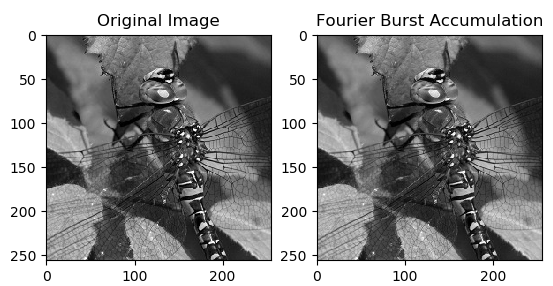


# Part 2 – Deblurring Through Fourier Burst Accumulation

Our solution for this part is based on “Burst Deblurring: Removing Camera Shake  
Through Fourier Burst Accumulation”. Given a list of blurred images, we performed a Fourier transform on each one, and calculated a weighted sum. Then we perform an inverse Fourier transform the get the deblurred image. i.e.:

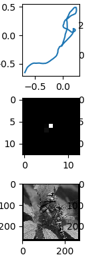
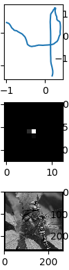


We choose value p=10 as it gave us good visual results. Iterating trough different values of p from 0 to infinity, and calculating PSNR could gave us the best p value. Here is an original image and deblurred image:



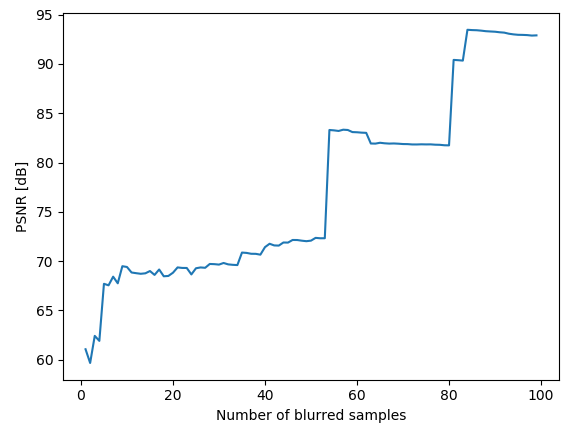
# Part 3 - PSNR calculation

For fixed p=10 value we calculated PSNR of deblurred image form different number of samples. We expect PSNR to increase as we add more and more samples. Also, we expect PSNR to rise drastically as we add blurred images which are has minimum amount of blur.



Let’s look at those blurred images: those are blurred images of index 53 and 81. As we can see, their PSFs are practically an Identity kernel (Both has minimum ammount of blur), thus we expect the PSNR to rise drastically at 53th an 81th iterations. This happens because those images get the majority of the weight in the weighted sum during the process of deblurring.

As we can see from the graph our expectations were correct. As we add more and more images the PSNR rises, while it rises drastically after 53th and 81th iterations.



Here are all deblurred images from all iterations. To make the result more visible, we cropped the images. Unfortunately, all of them visually undistinguishable, as the first PSF doesn’t blur the image drastically, so we get pretty good results starting from first iteration.



To see the differences between first and last iteration, let’s zoom the images. We can see that the first iteration is practically the first blurred image, which is already not strongly blurred. The 100th iteration produces much sharper image which looks like the original.

