3. Codigo Fonte em ep1_e3.py, rodar em python2 com o argumento sendo a imagem a filtrar. (Eu não consegui fazer o OpenCV funcionar com o python 3 aqui no meu computador, eu n faço ideia do que pode estar errado)

O processo usado no codigo foi relativamente simples, a maioria da dificuldade surgiu de falta de costume com python, ele consistiu de:

1. Aplicar a transformada de fourier do Numpy (np.fft.fft2(imagem))

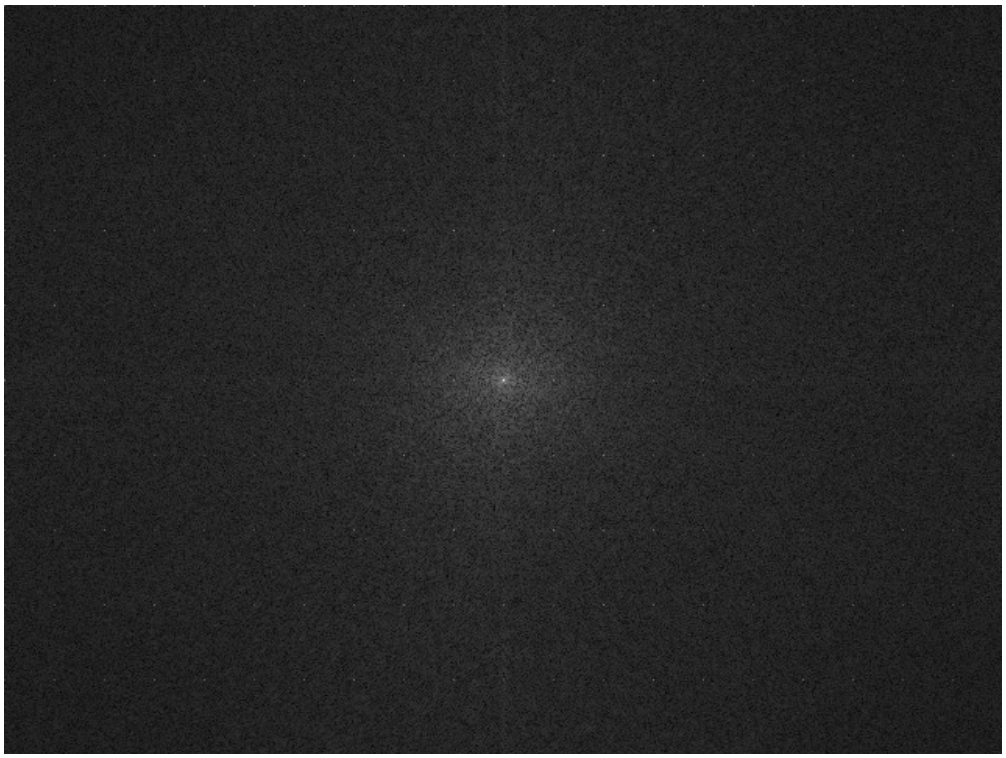

Imagem 9: Transformada aplicada, sem filtro nenhum, note os pontos de pico periodicos

Imagem 10: Pontos periodicos do ruido da imagem original

2. Aplicar um filtro removendo pixels a cada 32 pixels em x e 48 pixels em y

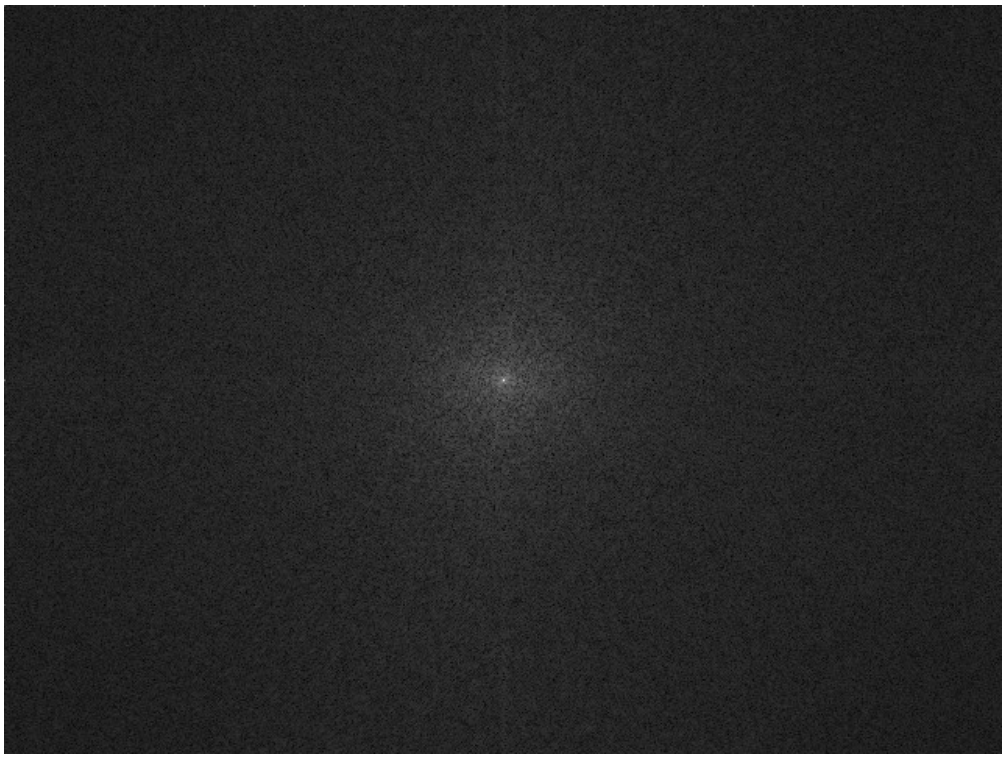

Imagem 11: Transformada aplicada, com um filtro sobre os picos periodicos, mantendo o centro intacto

2b. Não aplicar esse filtro no centro da imagem filtrada, para n perder informação demais da imagem

3. Aplicar a transformada inversa do Numpy (np.fft.ifft2(imagem))

Imagem 12: Imagem recuperada após aplicar o filtro

Fora os passos principais descritos acima, medidas foram tomadas como normalizar a imagem, para sempre lidar com numeros de ponto flutuante, manter em mente o shift da imagem (np.fft.fftshift()), e ignorar a parte complexa da imagem transformada quando for usar o cv2.imshow()

4. Código Fonte em ep1_e4.py rodar em python2 com o argumento sendo a imagem ao analisar.

Os mesmos cuidados tomados no ex3 foram tomados nesse exercicio também, de fato, a maior parte do código foi reaproveitada, então, normalização, fftshift e parte complexa foram tratadas como no ex3.

O processo usado esta descrito a seguir:

1. Aplicar transformada de fourier na imagem (O que leva algum tempo ja que as dimensões da imagem são bem grandes.)

2. Escolher uma parte do centro da imagem como area para analisar (Analisar a imagem inteira demoraria muito tempo, portanto somente um pequeno retangulo em volta do centro da transformada foi usado)

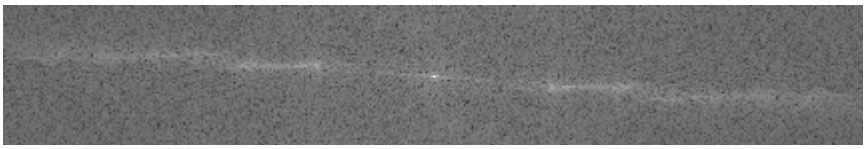

Imagem 13: Centro da transformada

3. Descobrir que pontos dentro dessa nova imagem tem valor acima de um valor prédeterminado (Treshold no codigo).

Imagem 14: Pontos escolhidos para ajuste

4. Ajustar uma reta passando por esses pontos (usando Numpy)

5. Aplicar arco-tangente ao valor do parametro para obter o angulo da reta, que está diretamente relacionado ao angulo da plantação

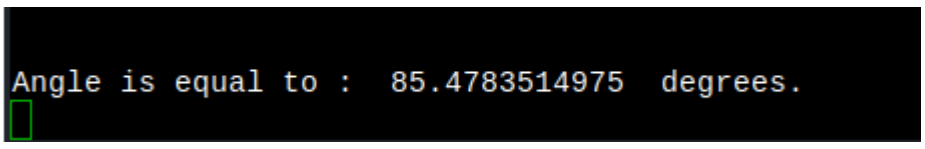

Imagem 15: Saída do programa