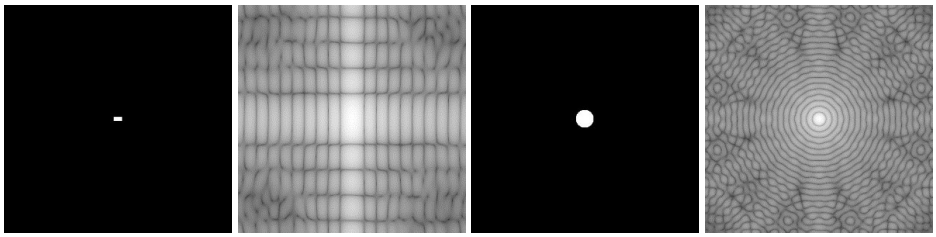
Labo 3: Filters in het frequentiedomein

Gianni Allebosch, Martin Dimitrievski

1 De 2-D discrete Fouriertransformatie

De Fouriertransformatie wordt gebruikt om beelden te kunnen filteren in het frequentiedomein. De vraag is nu: wat is de betekenis van dat frequentiedomein en zijn variabelen voor beelden? De frequentie is gerelateerd aan de spatiale veranderingen/variaties van de intensiteit in een beeld.

De traagste frequentie komt overeen met de gemiddelde grijswaarde in een beeld, deze traagste frequentie komt overeen met de oorsprong, (u,v)=0, van het Fourierdomein (ook wel de DC-component genoemd). Dicht bij het nulpunt zitten de traag veranderende variaties, bv. voor een vlak met een homogene kleur in het beeld. Hoe verder van dat nulpunt, hoe sneller de variaties in grijswaarden optreden in een beeld. De hoogste frequenties komen overeen met plotse veranderingen in een beeld, zoals bij randen, of door ruis.



(a) (b) (c) (d)

Figuur 1. (a) `block.jpg` en (b) het fft-beeld van `block.jpg`; (c) `disk.jpg` en (d) het fft-beeld van `disk.jpg`.

Laad het beeld 'block.jpg' en transformeer met de Fast Fourier Transform (FFT). De FFT is een algoritme dat het oplossen van de discrete Fouriertransformatie in O(NlogN) toelaat. Het resultaat na FFT is hierboven ook getoond als een fft-beeld, waarbij de intensiteitswaarden de grootte van elke component in het Fourierdomein voorstellen.

In Python gebruiken we de modules numpy.fft of scipy.fftpack voor implementaties in het Fourierdomein:

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

```
def magnitude_spectrum(fshift):
    magnitude_spectrum = np.log(np.abs(fshift)+1)
    magnitude_spectrum -= np.min(magnitude_spectrum)
    magnitude_spectrum *= 255./np.max(magnitude_spectrum)
    return magnitude_spectrum


def main():
    image = cv2.imread("disk.jpg", cv2.IMREAD_GRAYSCALE)
    # fourier transform
    fbeeld = np.fft.fft2(image)
    fshift = np.fft.fftshift(fbeeld)
    plt.imshow(magnitude_spectrum(fshift), cmap='gray')
    plt.show()

    # inverse fourier transform
    ifshift = np.fft.ifftshift(fshift)
    ifbeeld = np.fft.ifft2(ifshift)
    ifbeeld = ifbeeld.real
    plt.imshow(ifbeeld, cmap='gray')
    plt.show()


if __name__ == '__main__':
    main()
```

Het fft-beeld `fbeeld` is een matrix van complexe getallen, waarvan we nu alleen een maat voor de grootte van de vectoren willen visualiseren: de `abs` functie berekent de magnitude van de fft-waarden. De vectoren kunnen zeer uiteenlopende groottes aannemen, het logaritme van de magnitude, na schaling, geeft ons een duidelijker beeld van de beschikbare informatie.

De functie `fftshift` verwisselt kwadrant I met III en II met IV. Op deze manier komt de DC-component in de oorsprong van de Fourierruimte (`fbeeld`) te liggen. Dit is al gedaan voor het fft-beeld voor `block.jpg` in bovenstaande figuur. Merk op dat de hoogste waarden zich rond het centrum concentreren aangezien het oorspronkelijke beeld bestaat uit 2 homogene kleurvlakken. Kijk ook naar de plaatsen van de nuldoorgangen in het spectrum: in de verticale richting liggen deze 2x verder uit elkaar dan in de horizontale richting. Deze verhouding komt invers overeen met de verhoudingen van de rechthoek in het oorspronkelijke beeld. Waarom invers? Omdat variatie in de verticale richting (dus loodrecht op het smallere deel) sneller is (want het blok is daar smaller), dan in de horizontale richting. **Vraag:** ***Volgens welk patroon komen deze nulpunten voor in het spectrum? En waarom?***

Er bestaat ook een functie `ifftshift` om de verschuiving van de kwadranten ongedaan te maken. Ook al heeft deze functie hetzelfde effect, het gebruik ervan kan een programma leesbaarder maken. Doe de bovenstaande bewerkingen eens opnieuw met het beeld: `disk.jpg`, en probeer te begrijpen wat er getoond wordt in het Fourierdomein.

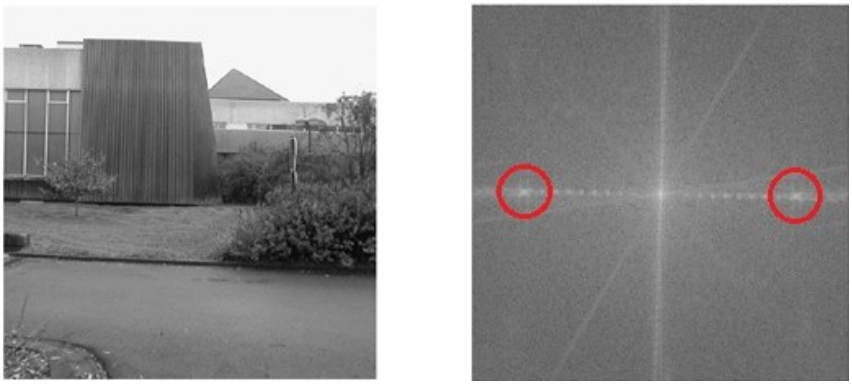
Om terug te keren naar het spatiale domein, kan je de functie `ifft2` gebruiken. Het invers getransformeerde beeld zou een reëel beeld moeten zijn, maar door afrondingsfouten blijven er nog kleine imaginaire componenten over, gebruik de functie `real` om deze te verwijderen uit het invers getransformeerd beeld. Denk eraan dat voor de inverse transformatie de DC component teruggeschoven moet worden met `ifftshift`.

2 Filteren in het frequentiedomein

Een convolutie in het spatiale domein komt overeen met een vermenigvuldiging in het frequentiedomein: $f(x,y) * h(x,y) \Leftrightarrow H(u,v)F(u,v)$, en vice versa. Wanneer we een beeld en een filter hebben in het frequentiedomein moeten we dus enkel de matrices element per element vermenigvuldigen. Na inverse Fouriertransformatie bekomen we het gefilterde beeld.

2.1 Opdrachten

1. **Ideaal laagdoorlaatfilter**
   Een ideaal laagdoorlaatfilter is een filter dat alles binnen de afsnijfrequentie onverzwakt doorlaat, en alles erbuiten wegfiltert (dus met een discontinue overgang op de afsnijfrequentie). Als we werken op een beeld waar de DC-component in het midden ligt (d.m.v. `fftshift`), dan wordt een ideaal laagdoorlaatfilter gevormd door een matrix van dezelfde grootte van het fft-beeld, met in het midden een schijf met filtercoëfficiënten = 1 (wit) met straal bepaald door de afsnijfrequentie, en de rest nullen (zwart). Na vermenigvuldiging blijven enkel de lage frequentiecomponenten (centraal in het fft-beeld) over.
   In de foto `campus.jpg` vertoont de muur van het gebouw een verticaal lijnenpatroon. `fftcampus.jpg` toont de FFT van het beeld na `fftshift` in het beeld erlangs. **Vraag:** ***Waarmee komt de frequentie-inhoud met hoogste magnitude overeen?*** In `fftcampus.jpg` kun je de componenten van de spatiale frequentie van het lijnenpatroon duidelijk zien in de hogere frequenties op de horizontale as aan de buitenrand van het fft-beeld (zie rode cirkels). Voer nu een laagdoorlaatfiltering uit op het beeld, zodanig dat het lijnenpatroon onderdrukt wordt. Toon hierbij ook het gefilterde fft-beeld die net de frequenties van het lijnenpatroon afsnijdt.
   Gebruik voor het maken van de matrix voor het ideaal laagdoorlaatfilter de routine **circle(R,M,N)** die een matrix maakt van M rijen, N kolommen waarin centraal een cirkel met straal R staat. Gebruik de functie



(a) (b)

Figuur 2. (a) `campus.jpg` en (b) het fft-beeld van `campus.jpg`.

show_results(image, fft_image, filtered_fft, filtered_image) om de resultaten te tonen. **Vraag:** ***Verklaar wat er gebeurd als je de straal van de cirkelvormig filter laat variëren in grootte.***

```
def circle(R, M, N):
    x, y = np.meshgrid(np.linspace(0,N,N), np.linspace(0,M,M),sparse=True)
    cx, cy = [N/2, M/2]
    return (((x-cx)**2 + (y-cy)**2) <= R**2)

def show_results(image, fft_image, filtered_fft, filtered_image):
    fig = plt.figure()
    ax1 = fig.add_subplot(221)
    ax1.set_title("Original")
    ax2 = fig.add_subplot(222)
    ax2.set_title("FFT")
    ax3 = fig.add_subplot(223)
    ax3.set_title("Filtered FFT")
    ax4 = fig.add_subplot(224)
    ax4.set_title("Filtered Image")
    ax1.imshow(image, cmap = "gray")
    ax2.imshow(magnitude_spectrum(fft_image), cmap = "gray")
    ax3.imshow(magnitude_spectrum(filtered_fft), cmap = "gray")
    ax4.imshow(filtered_image, cmap = "gray")
    ax1.axis('off')
    ax2.axis('off')
    ax3.axis('off')
    ax4.axis('off')
    plt.tight_layout()
    plt.show()
```

De discontinuïteit in het ideale laagdoorlaatfilter zorgt voor *ringing* in het teruggetransformeerde beeld: intensiteitsvariaties in regio's waar er constante intensiteit verwacht wordt, dikwijls in de buurt van de randen van regio's van constante intensiteit in het oorspronkelijke beeld. Deze effecten zijn duidelijk zichtbaar in het resultaat voor deze opdracht. Het minimaliseren van ringing-effecten is een belangrijk criterium in filter-design.
De oorzaak ligt in de eigenschappen van de fouriergetransformeerde van het ideaal filter in het spatiale domein. Denk aan de relatie tussen ideale filters en de sinc-functie, en ook aan het Gibbs-fenomeen bij discontinue overgangen. Je kan dit effect bekijken door het masker van de ideale filter te transformeren naar het spatiale domein. **Vraag:** ***Voer dit uit en bespreek de bovenstaande effecten.***

2. **Gaussiaans laagdoorlaatfilter**
   We bekijken nu een aantal filters met een vloeiender overgang, die geen impulsantwoord geven dat oscilleert of negatief kan worden, waardoor bovenstaande effecten verminderen. De eerste filter die we bekijken is een Gaussiaans laagdoorlaatfilter. De frequentierespons van een Gaussiaan is weer een Gaussiaan, wat vele voordelen geeft. **Vraag:** ***Bespreek deze voordelen in commentaar, maar ook de nadelen.*** Om dit soort filter in het frequentiedomein toe te passen, beschrijven we de functie waaraan een Gaussiaans filter H voldoet:

$$\begin{aligned} H(u,v) &= e^{\frac{-D^2(u,v)}{2D_0^2}} \\ D(u,v) &= \sqrt{u^2+v^2} \end{aligned} \tag{1}$$

   $D(u,v)$ stelt de afstand tot de oorsprong van het filter voor en D_0 is een maat voor de straal (en dus de afsnijfrequentie) van het filter: bij $D(u,v) = D_0$ is het filter gezakt tot 0,607 van zijn maximale waarde van 1. Implementeer en pas dit filter toe op `campus.jpg`. Bekijk zeker ook de vorm van dit filter (bv. als een beeld of een oppervlakteplot) en het fft-beeld van de gefilterde afbeelding.

3. **Butterworth laagdoorlaatfilter**
   Vraag: ***Enkele nadelen van het Gaussiaans filter worden opgevangen in een 2de orde Butterworth laagdoorlaatfilter. Bespreek welke.***
   De Butterworth laagdoorlaatfilter voldoet aan de volgende vergelijkingen:

$$\begin{aligned} H(u,v) &= \frac{1}{1+\left[\frac{-D(u,v)}{D_0}\right]^{2n}} \\ D(u,v) &= \sqrt{u^2+v^2} \end{aligned} \tag{2}$$

   n is de orde van de filter en D_0 is een maat voor de straal: bij $D(u,v) = D_0$ is het filter gezakt tot 50% van zijn maximale waarde. Bekijk en vergelijk ook dit filter met de voorgaande laagdoorlaatfilters, en test verschillende waardes voor de parameters.

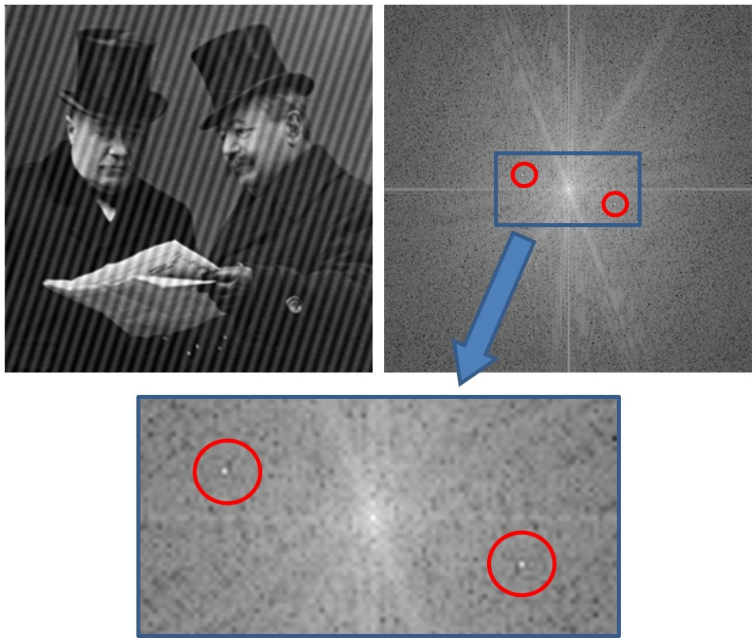
4. **Hoogdoorlaatfilters**
   Een hoogdoorlaatfilter kan makkelijk geconstrueerd worden uit de transferfunctie van een laagdoorlaat filter volgens de relatie: $H_{HP} = 1 - H_{LP}$. Hoogdoorlaatfilters filteren de DC-component weg, zodat de gemiddelde waarde gereduceerd wordt naar 0. Dit effect kan gecompenseerd worden door een offset bij een hoogdoorlaatfilter op te tellen. Als deze offset gecombineerd wordt met een vermenigvuldiging van het filter met een constante groter dan 1, spreken we van een high-frequency emphasis filter: $H_{HFE} = a + bH_{HP}$. Zolang a kleiner is dan b zullen de hoge frequenties het meest benadrukt worden. Probeer dit uit op het beeld `Xraywrist.jpg` met behulp van een Butterworthfilter. Toon hierbij ook de filter en het gefilterde fft-beeld.
5. **Butterworth notch filter**
   De figuur `oldtv.jpg` toont een beeld van een oude tv. Het beeld is vervormd door sinusoïdale ruis. Het storend patroon vertoont een constante variatie over het beeld, die eenvoudig weg te filteren is met een bandsperfilter.
   Een Butterworth notch filter kan berekend worden met:

$$\begin{aligned} H(u,v) &= \frac{1}{1 + \left[\frac{D_0^2}{D_1(u,v)D_2(u,v)}\right]^n} \\ D_1(u,v) &= \sqrt{(u-u_0)^2 + (v-v_0)^2} \\ D_2(u,v) &= \sqrt{(u+u_0)^2 + (v+v_0)^2} \end{aligned} \tag{3}$$

   D_0 is een maat voor de straal van de notch, (u_0, v_0) en $(-u_0, -v_0)$ zijn de locaties van de 2 notches. De plaats van de notches kan gevonden worden op de het fft-beeld van `oldtv.jpg`. De plaats is zichtbaar als een kleine vlek van verhoogde intensiteit van dit beeld (zie rode cirkels).
   Probeer met een bandsperfilter het patroon zoveel mogelijk te onderdrukken, zonder de kwaliteit van de oorspronkelijke figuur te verminderen. Bereken de matrix voor het filter op een efficiënte manier (gebruik **meshgrid**) en **plot** een grafiek van het filter ter controle. Plaats de originele en bewerkte figuur naast elkaar; de ruis moet duidelijk verdwenen zijn zonder het beeld zichtbaar te *blurren*.

3 Indienen

Indienen van het labo gebeurt via Opdrachten op Ufora. De code dien je in als 1 pythonscript genaamd **labo3.py** waarin alle opdrachten sequentieel worden uitgevoerd. Voeg een **README.txt** bestand toe waarin je programma beschreven is en waarin de vragen beantwoord worden. Alle gevraagde outputafbeeldingen geef je mee in een .zip-file genaamd **output.zip**.



Figuur 3. `oldtv.jpg` en zijn fft-beeld. De rode cirkels duiden de notches van de ruis aan.