Mollification infor extraction using Shearlets

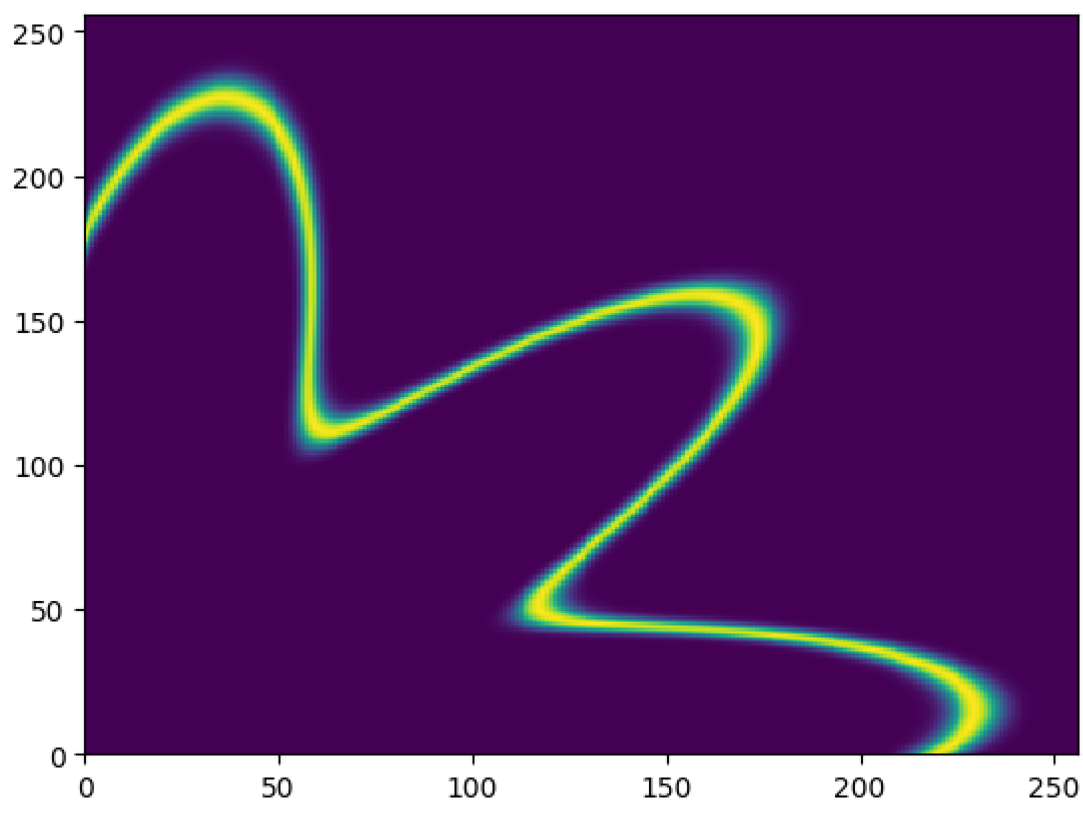
In [11]:
```
# Loard the Pkg
push!(LOAD_PATH,pwd()*"/../../src")
import Shearlab
#using Shearlab
using PyPlot
reload("Shearlab")
using Images
```

WARNING: replacing module Shearlab

In [12]:
```
using DataFrames
```

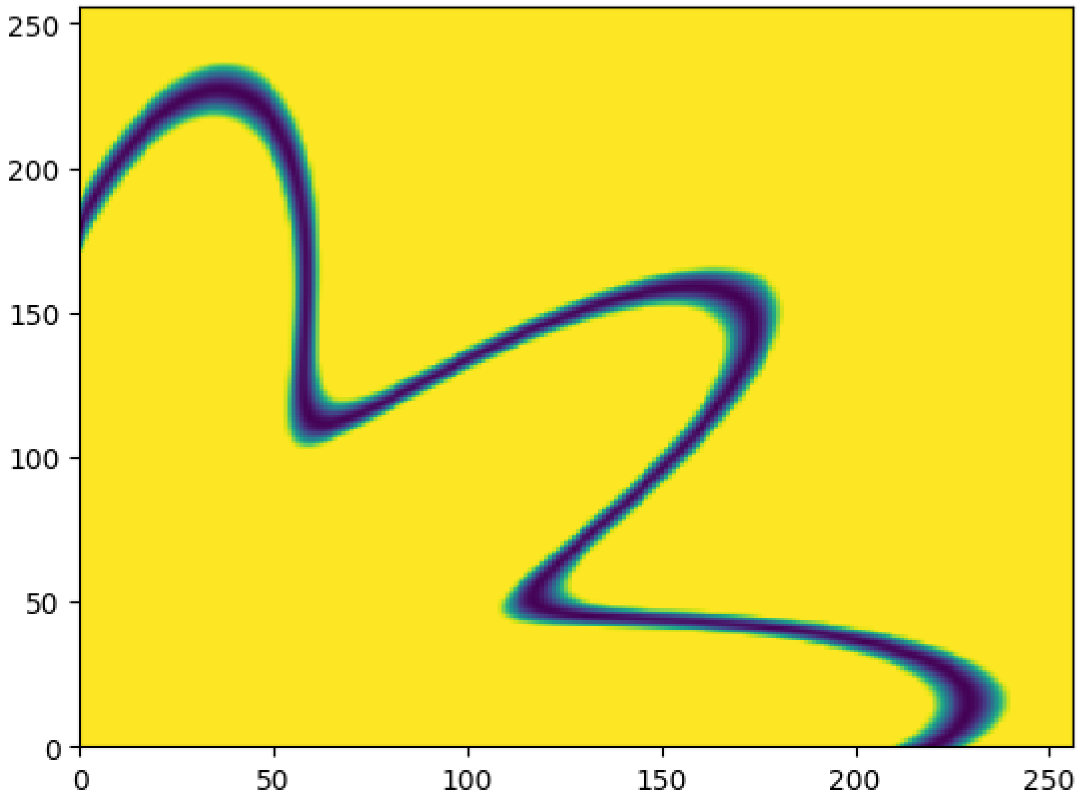
In [13]:
```
# Read Data
n = 256;
psi1 = Shearlab.load_image("./psi1.png",n)
psi1 = psi1[:,:,1];
psi2 = Shearlab.load_image("./psi2.png",n)
psi2 = psi2[:,:,1];
psi3 = Shearlab.load_image("./psi3.png",n)
psi3 = psi3[:,:,1];
```

In [14]:
```
pcolormesh(psi1)
```



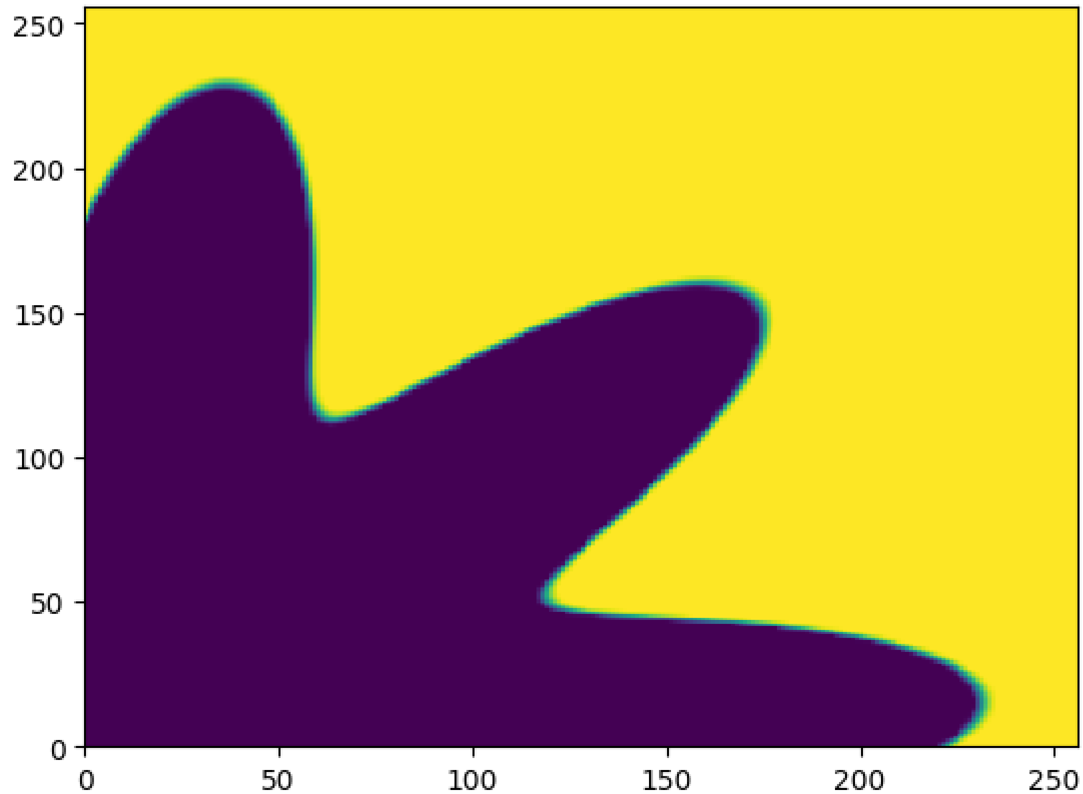
Out[14]: PyObject <matplotlib.collections.QuadMesh object at 0x1466a6710>

In [15]:
```
pcolormesh(psi2)
```



Out[15]: PyObject <matplotlib.collections.QuadMesh object at 0x12613c690>

In [16]:
```
pcolormesh(psi3)
```



Out[16]: PyObject <matplotlib.collections.QuadMesh object at 0x146446810>

In [17]:
```
# Parameters
rows = size(psi1,1)
cols = size(psi1,2);
nScales = 6;
shearLevels = ceil.((1:nScales)/2)
scalingFilter = Shearlab.filt_gen("scaling_shearlet");
directionalFilter = Shearlab.filt_gen("directional_shearlet");
waveletFilter = Shearlab.mirror(scalingFilter);
scalingFilter2 = scalingFilter;
full = 0;
```

In [18]:
```
# Compute the corresponding shearlet system without gpu
@time shearletSystem= Shearlab.getshearletsystem2D(rows,cols,nScales, shearI
                                directionalFilter,
                                scalingFilter,0);
```

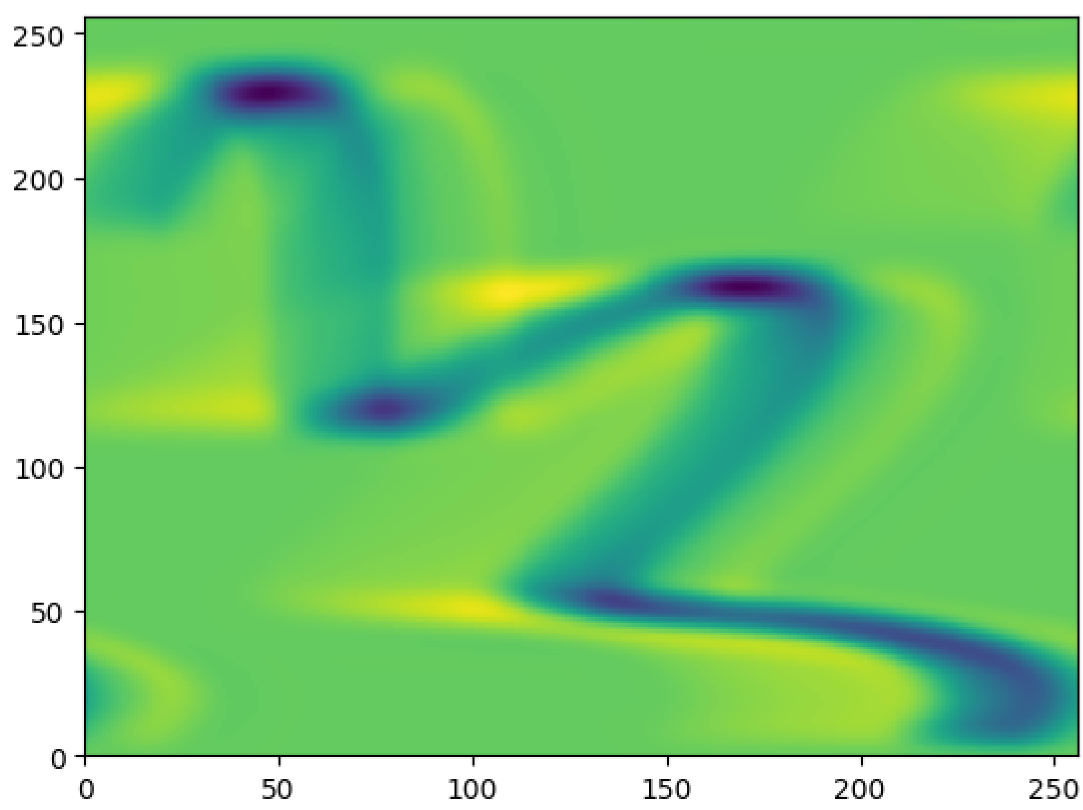
```
WARNING: The specified Shearlet system was not available for data of size
 256x256. Filters were automatically set to configuration 6(see operation
s.jl).

size(directionalFilterUpsampled'')=(36, 9)
size(filterLow2[size(filterLow2,2)-shearLevel]'')=(11,)
size(directionalFilterUpsampled'')=(72, 9)
size(filterLow2[size(filterLow2,2)-shearLevel]'')=(25,)
size(directionalFilterUpsampled'')=(144, 9)
size(filterLow2[size(filterLow2,2)-shearLevel]'')=(53,)
 10.074560 seconds (2.00 M allocations: 3.907 GiB, 26.33% gc time)
```

In [19]:
```
# Computing the coefficients
psi1coeffs = Shearlab.SLsheardec2D(psi1,shearletSystem);
psi2coeffs = Shearlab.SLsheardec2D(psi2,shearletSystem);
psi3coeffs = Shearlab.SLsheardec2D(psi3,shearletSystem);
```

In [20]:
```
size(psi1coeffs)
```

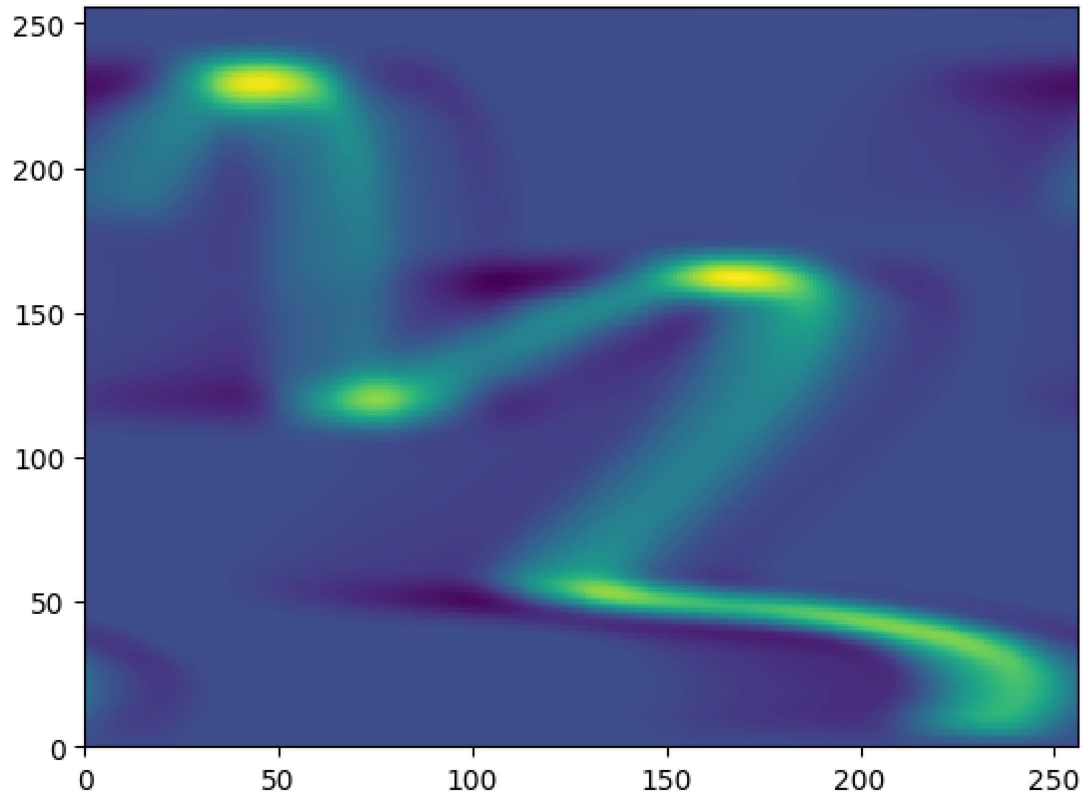
Out[20]: (256, 256, 113)

In [21]:
```
pcolormesh(real(psi1coeffs[:,:,2]))
```



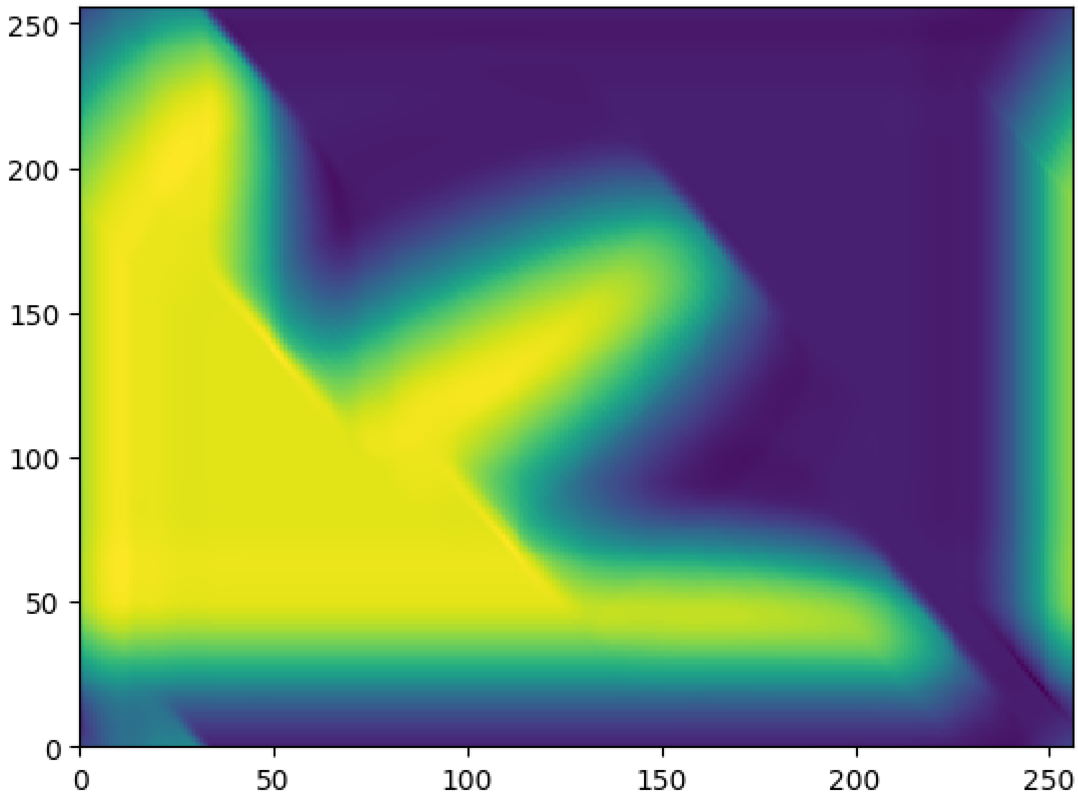
Out[21]: PyObject <matplotlib.collections.QuadMesh object at 0x1262ba650>

In [22]:
```
pcolormesh(real(psi2coeffs[:,:,1]))
```



Out[22]: PyObject <matplotlib.collections.QuadMesh object at 0x1389347d0>

In [23]:
```
pcolormesh(real(psi3coeffs[:,:,49]))
```

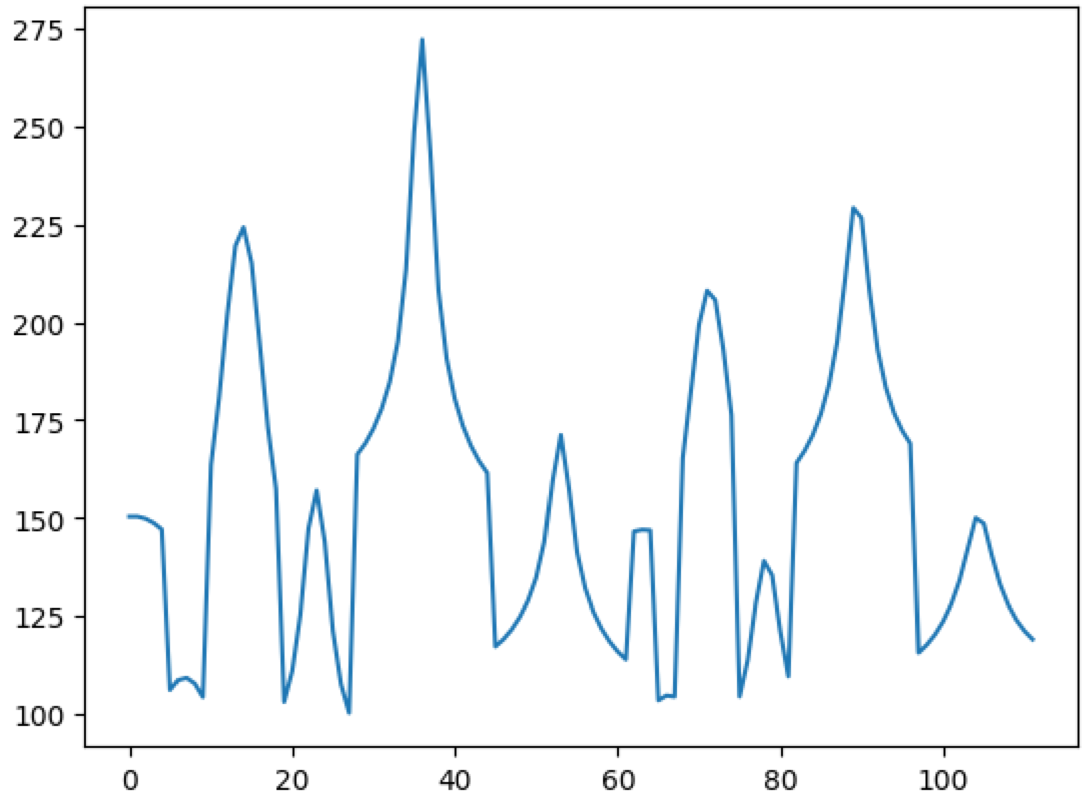


Out[23]: PyObject <matplotlib.collections.QuadMesh object at 0x1467b9950>

In [24]:
```
# Lets see how fast the coefficients decay
decay1 = [norm(psi1coeffs[:,:,i],2) for i in 1:(size(psi1coeffs,3)-1)];
decay2 = [norm(psi2coeffs[:,:,i],2) for i in 1:(size(psi2coeffs,3)-1)];
decay3 = [norm(psi3coeffs[:,:,i],2) for i in 1:(size(psi3coeffs,3)-1)];
```

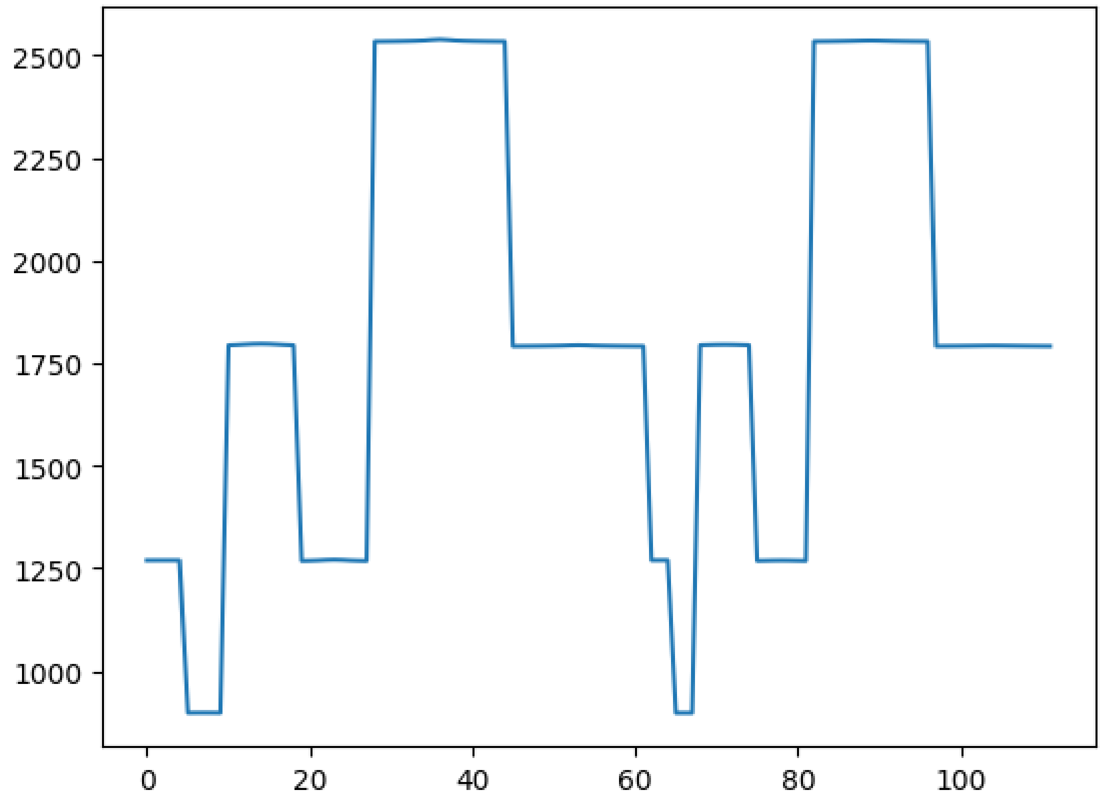
In [25]:
```
idxs = shearletSystem.shearletIdxs;
idxs = ["["*string(idxs[i,1])*","*string(idxs[i,2])*","*string(idxs[i,3])"]'
```

In [26]:
```
plot(decay1)
```



Out[26]:
```
1-element Array{PyCall.PyObject,1}:
 PyObject <matplotlib.lines.Line2D object at 0x147864750>
```

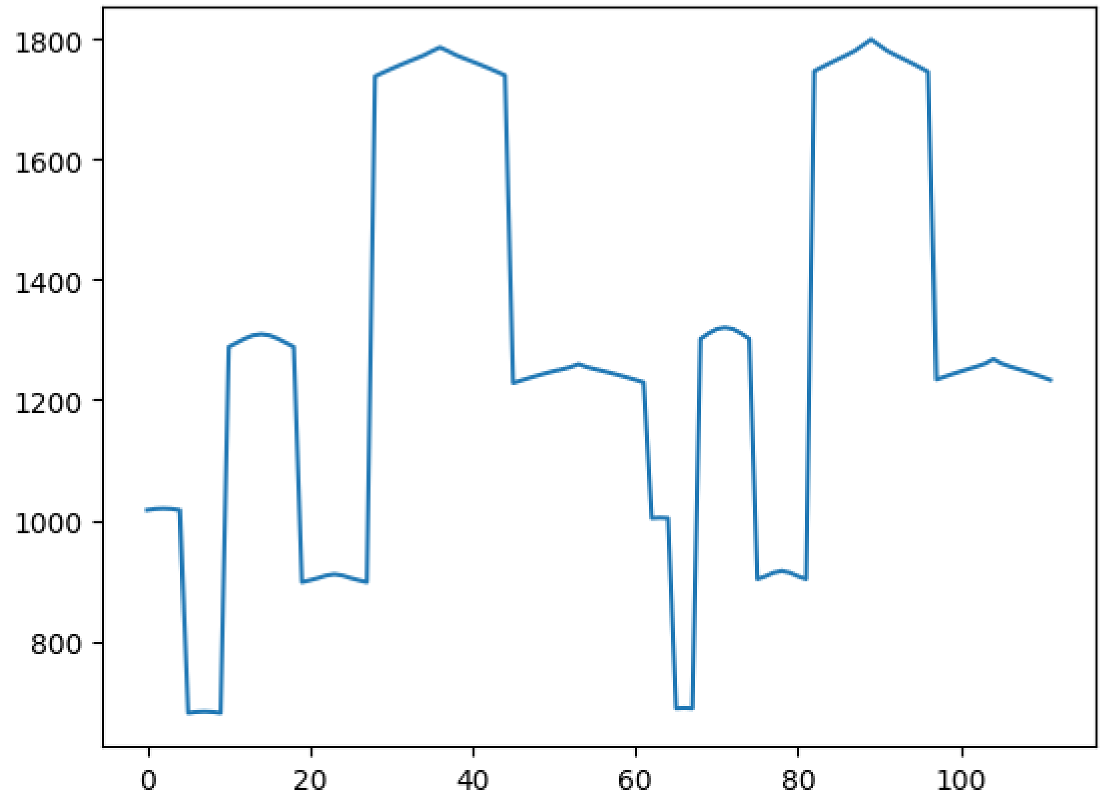
In [27]:
```
plot(decay2)
```



Out[27]:
```
1-element Array{PyCall.PyObject,1}:
 PyObject <matplotlib.lines.Line2D object at 0x14824ad10>
```

In [28]:
```
idxs = shearletSystem.shearletIdxs;
idxs = ["["*string(idxs[i,1])*","*string(idxs[i,2])*","*string(idxs[i,3])"]'
```

In [29]:
```
plot(decay3)
```



Out[29]:
```
1-element Array{PyCall.PyObject,1}:
 PyObject <matplotlib.lines.Line2D object at 0x14789cdd0>
```

Extracting the curve and the mollified part.

We wil use a thresholding approach so we can extract the curve (high frequency domain) and the smoth distribution (low frequency domain).

Lets visualize first the data in 3D plots using Plotly.

In [30]:
```
using Plotly
```

Plotly javascript loaded.

To load again call
`init_notebook(true)`

In [31]:
```
init_notebook(true)
```

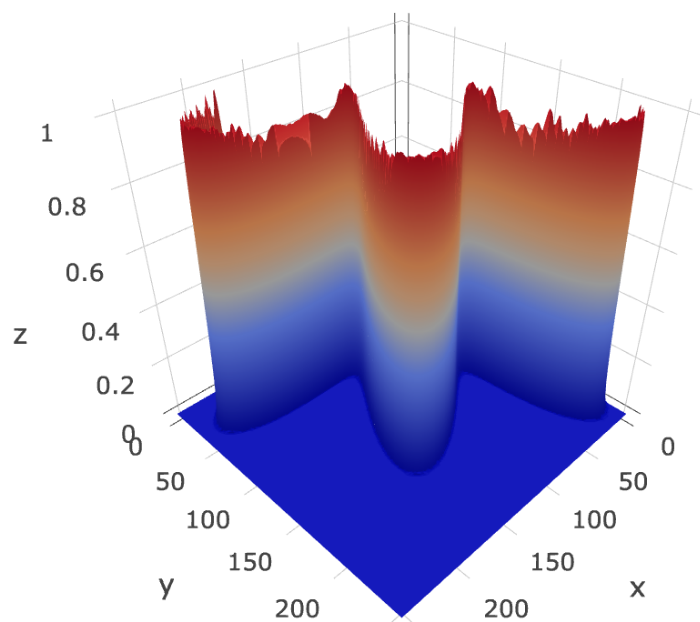
Plotly javascript loaded.

To load again call
`init_notebook(true)`

Out[31]: true

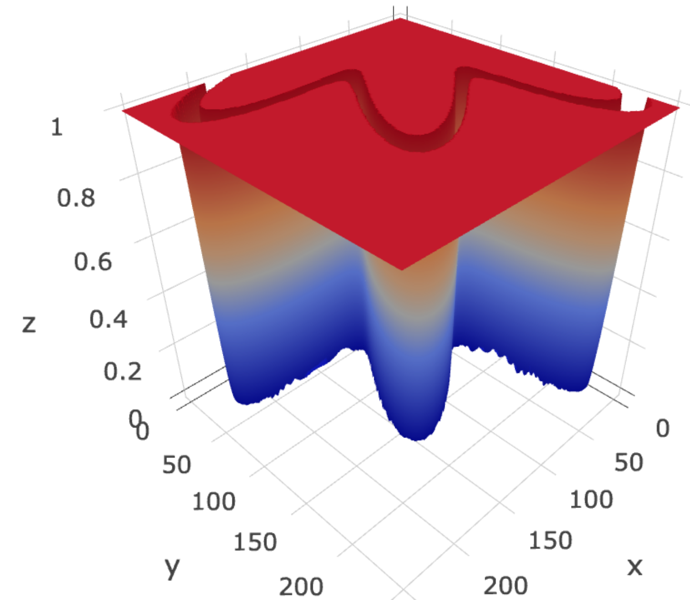
In [32]:
```
z = psi1
u = linspace(0, size(z,1), size(z,1))
v = linspace(0, size(z,2), size(z,2))
import PlotlyJS
PlotlyJS.plot([PlotlyJS.surface(x=u,y=v,z=z)])
```

Out[32]:



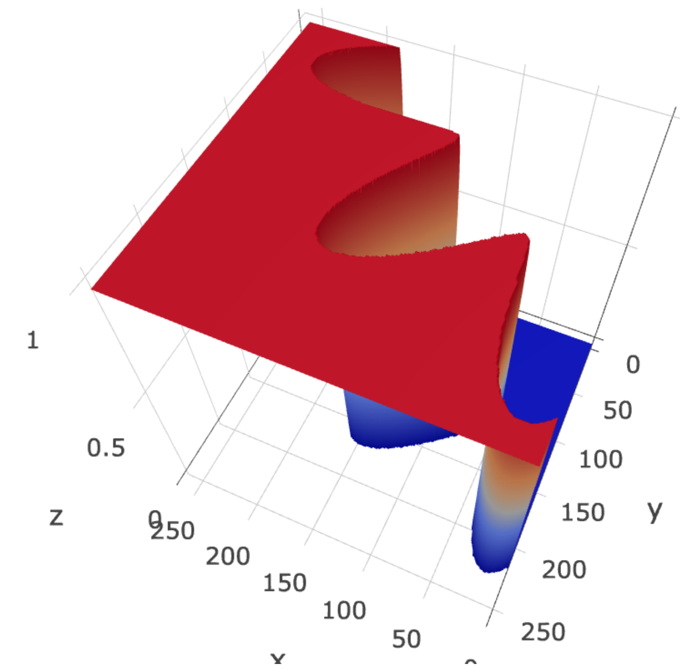
In [33]:
```
z = psi2
u = linspace(0, size(z,1), size(z,1))
v = linspace(0, size(z,2), size(z,2))
import PlotlyJS
PlotlyJS.plot([PlotlyJS.surface(x=u,y=v,z=z)])
```

Out[33]:



In [34]:
```
z = psi3
u = linspace(0, size(z,1), size(z,1))
v = linspace(0, size(z,2), size(z,2))
import PlotlyJS
PlotlyJS.plot([PlotlyJS.surface(x=u,y=v,z=z)])
```

Out[34]:



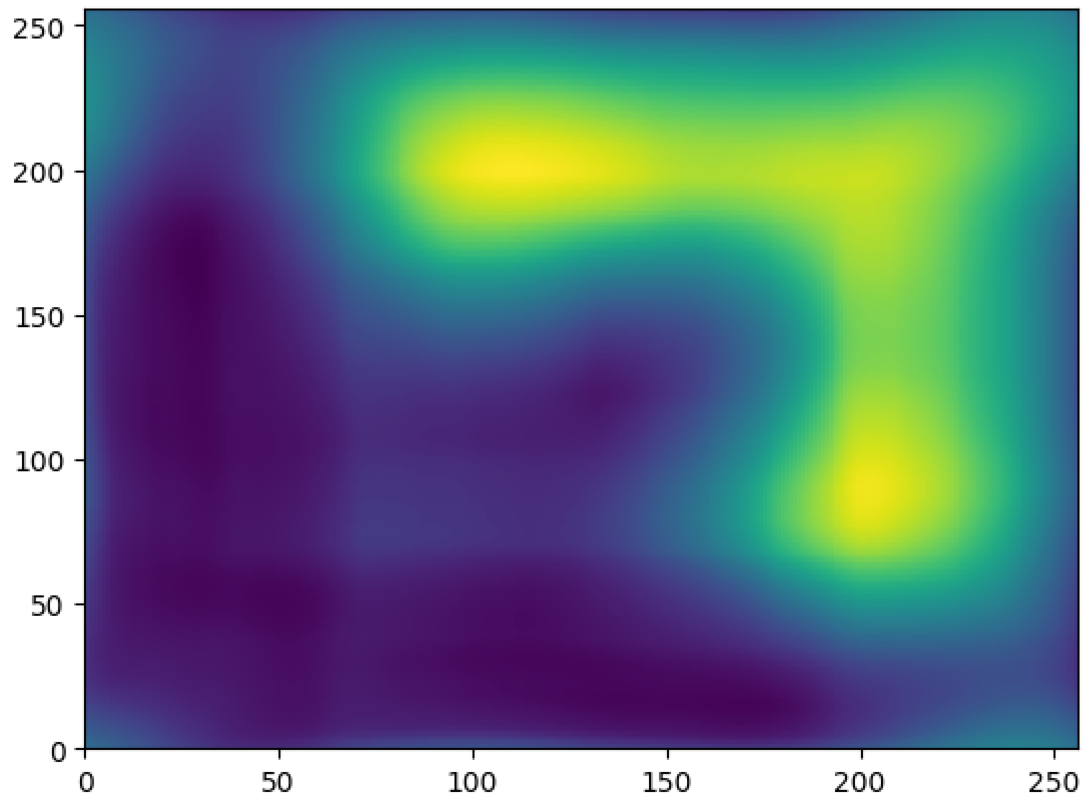
In [35]:
```
# Extract smooth or nonsmoth part function
# This function takes the coeffs from which you want to extract the smooth c
# the shearlet system
# the sigma factor of the thresholding, the thresholding factor, and the bi
# is one for the smooth part and zero for the nonsmooth part
function extract_smooth(coeffs, shearletSystem, sigma, thresholdingFactor, s
    # Thresholding indices for the coefficients
    threshold_idx=(abs.(coeffs).> thresholdingFactor*reshape(repmat(shearlet
            size(coeffs,1),size(coeffs,2),length(shearletSystem.RMS))*sigma)
    #Thresholded
    if smooth == 1
        thresholded = coeffs.*threshold_idx;
    else
        thresholded = coeffs.*(.~threshold_idx);
    end
    # Reconstruction
    Shearlab.shearrec2D(thresholded, shearletSystem);
end
```

Out[35]: extract_smooth (generic function with 1 method)

In [36]:
```
# Lets start with the thresholding setting.
sigma = 50;
thresholdingFactor = 0.9;
coeffs = psi3coeffs;
# Extracting the smooth and nonsmooth part
smooth = 1;
psi3_smooth = extract_smooth(coeffs, shearletSystem, sigma, thresholdingFact
smooth = 0;
psi3_nonsmooth = extract_smooth(coeffs, shearletSystem, sigma, thresholdingF
```

Visualize the smooth part (smooth scalar field)

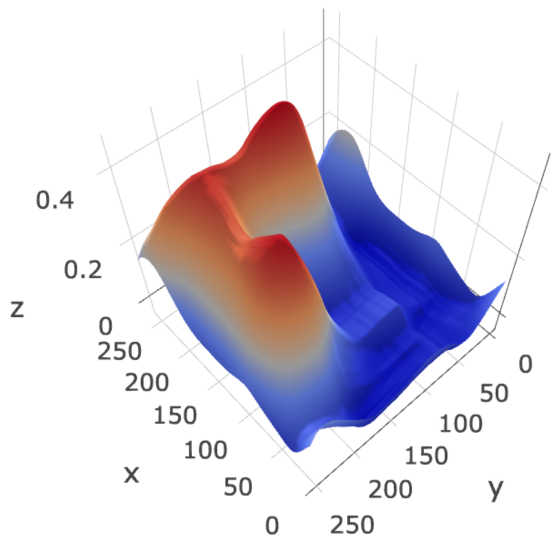
In [39]:
```
pcolormesh(psi3_smooth)
```



Out[39]: PyObject <matplotlib.collections.QuadMesh object at 0x116631bd0>

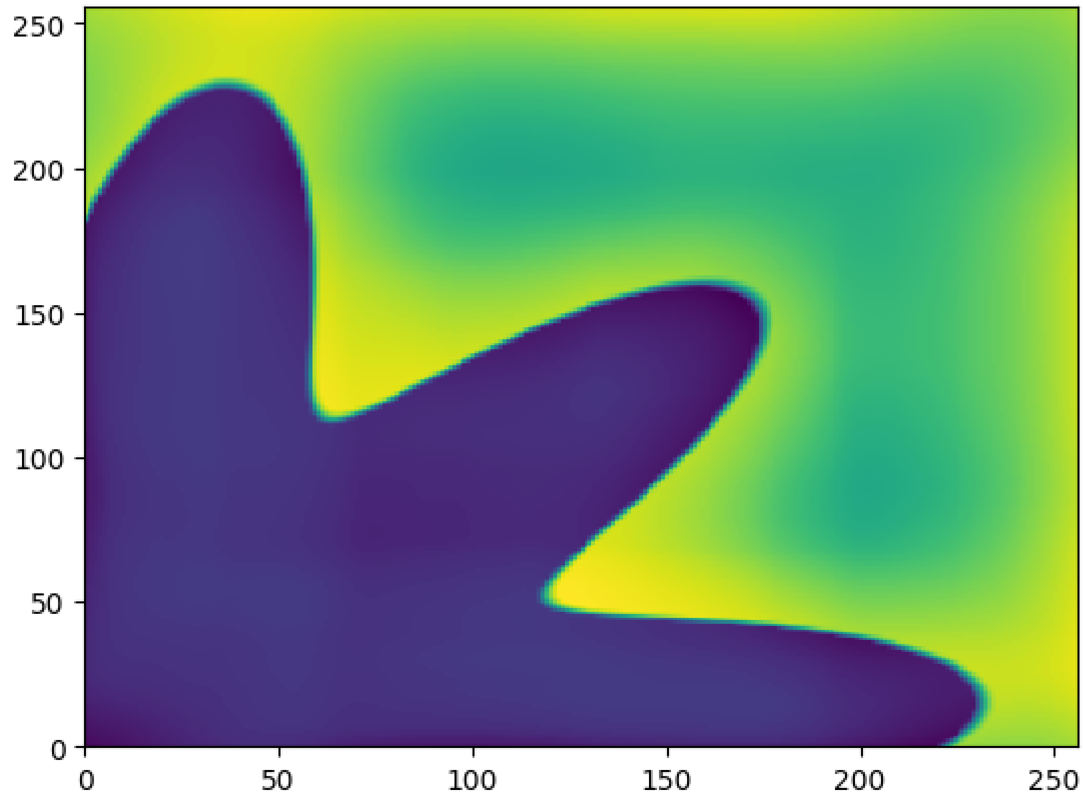
In [37]:
```
z = psi3_smooth
u = linspace(0, size(z,1), size(z,1))
v = linspace(0, size(z,2), size(z,2))
import PlotlyJS
PlotlyJS.plot([PlotlyJS.surface(x=u,y=v,z=z)])
```

Out[37]:



Visualize the nonsmooth part (the curve)

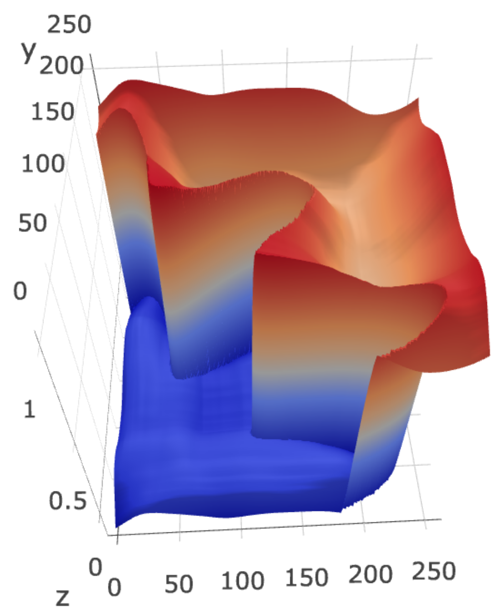
In [40]:
```
pcolormesh(psi3_nonsmooth)
```



Out[40]: PyObject <matplotlib.collections.QuadMesh object at 0x120794d50>

In [38]:
```
z = psi3_nonsmooth
u = linspace(0, size(z,1), size(z,1))
v = linspace(0, size(z,2), size(z,2))
import PlotlyJS
PlotlyJS.plot([PlotlyJS.surface(x=u,y=v,z=z)])
```

Out[38]:



In [ ]:
```
```