Polarization independence analysis of coherent laser speckle image

Correlation between s-polarization and p-polarization

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import os
import glob
import math
from PIL import Image
from PIL import ImageSequence
from matplotlib import rc, animation
rc('animation', html='html5')
from IPython.display import display, clear_output
```

In [2]:
```
folname=os.getcwd()+"\\190807\\pol\\"
flist=glob.glob(folname+"*tif*")
```

In [3]:
```
deglist=["{0:03d}".format(i*30) for i in list(range(12))]
print(deglist)
```

```
['000', '030', '060', '090', '120', '150', '180', '210', '240', '270', '300',
'330']
```

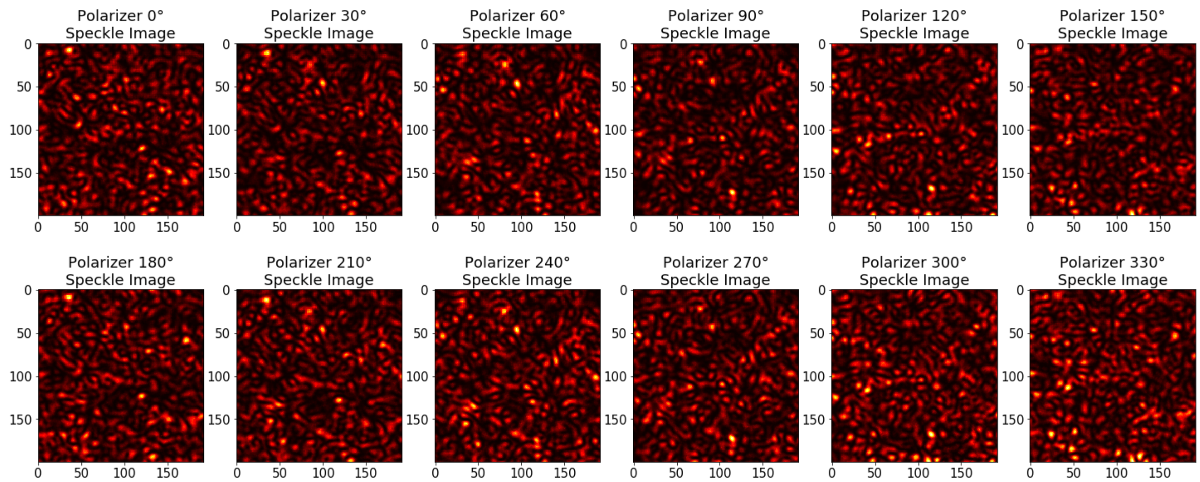
In [4]:
```
fdic={}
for deg in deglist:
    fdic[int(deg)]=[]

for fname in flist:
    for deg in deglist:
        if deg+'deg' in fname:
            fdic[int(deg)].append(fname)
```

In [5]:
```
plt.rcParams['image.cmap'] = 'hot'
plt.rcParams['font.size'] = 15
```

In [6]:
```
imdic={}
#calculate the average of each photos with same respective angle
for deg in deglist:
    shape=plt.imread(fdic[int(deg)][0]).shape
    image=np.zeros(shape)
    counter=0
    for fname in fdic[int(deg)]:
        image+=plt.imread(fname)
        counter+=1
    imdic[int(deg)]=image/counter
```

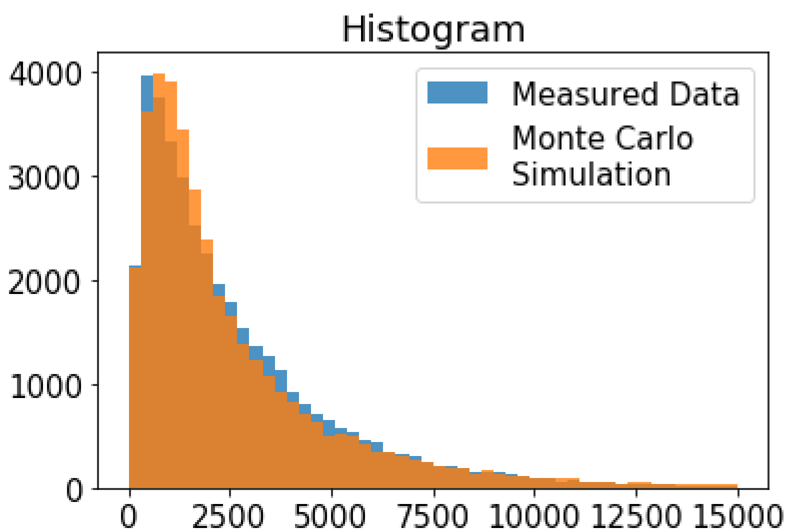
In [7]:
```
#Speckle images of each polarization angles
fig, ax= plt.subplots(ncols=6,nrows=2,figsize=(25,10),squeeze=True)
for y in range(ax.shape[0]):
    for x in range(ax.shape[1]):
        index=ax.shape[1]*y+x
        im=ax[y][x].imshow(imdic[index*30]);
        ax[y][x].set_title('Polarizer {0}'.format(index*30)+u'\xb0'+'\nSpeckle Image');
```



Polarized Speckle Pattern

A photon is a boson that follows Bose-Einstein statistics. Emission and detection of a photon is therefore a process that follows Poisson process. The electric field of each pixel for speckle image with enough resolution is a product of random walk process and should follow Gaussian distribution. The intensity which we detect with camera is square of the field, so the distribution of the intensity should follow χ^2 distribution with one degree of freedom.

In [8]:
```
Intensity=2000
MonteCarlo=np.random.chisquare(1, size=imdic[180].shape[0]*imdic[180].shape[1
]) #intensity matrix
MonteCarlo=np.random.poisson(MonteCarlo*Intensity,size=(1,MonteCarlo.size)) #d
etection with shot noise
Noise=np.abs(np.random.randn(1,38400)*800+300) #gaussian thermal noise & senso
r noise
MonteCarlo=MonteCarlo+Noise
plt.hist(imdic[180].ravel(),bins=50,histtype='stepfilled',label='Measured Dat
a',alpha=0.8,range=(0,15000));
plt.hist(MonteCarlo.ravel(),bins=50,histtype='stepfilled',label='Monte Carlo\n
Simulation',alpha=0.8,range=(0,15000))
plt.legend(loc='upper right')
plt.title('Histogram');
```



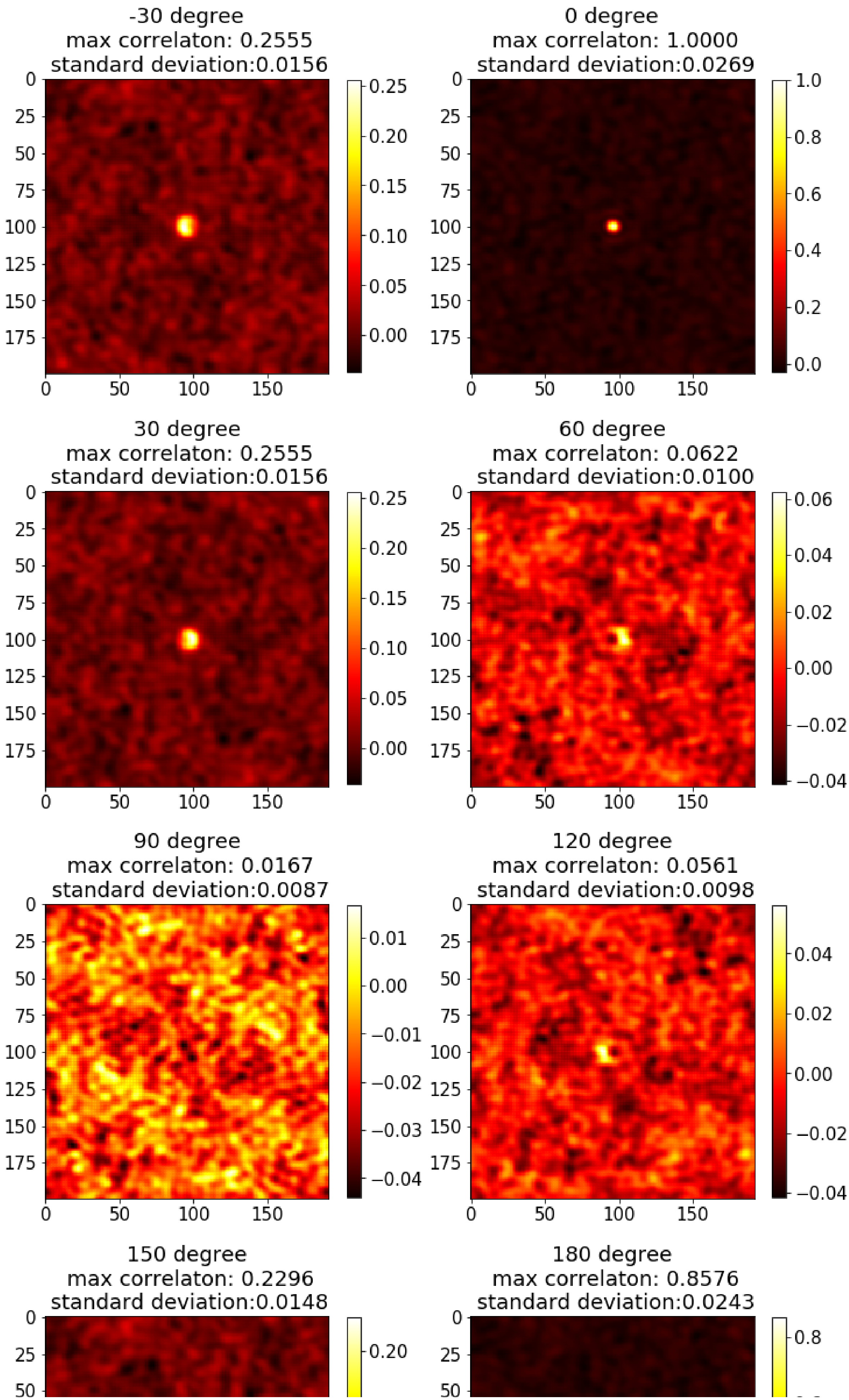
In [9]:
```
def correlator(X):
    jp_=np.mean(np.square(X));
    pj_=np.square(np.mean(X));
    xlen_=np.shape(X)[0];
    ylen_=np.shape(X)[1];
    tlen_=np.shape(X)[2];
    sj_=np.fft.fftn(X);
    sj_=np.abs(sj_);
    sj_=np.square(sj_);
    sj_=np.fft.ifftn(sj_);
    sj_=np.real(sj_);
    sj_=np.fft.fftshift(sj_);
    sj_=sj_/xlen_/ylen_/tlen_;
    if jp_==pj_:
        cor_=sj_/jp_
    else:
        cor_=(sj_-pj_)/(jp_-pj_);
    return cor_[:,:,:];
#    return cor_[math.floor(xlen_/2),math.floor(ylen_/2),:];
```

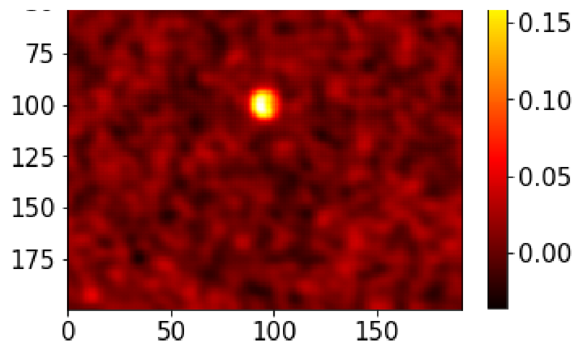
In [10]:
```
imagematrix=np.stack([imdic[int(deg)] for deg in deglist],axis=2)
```

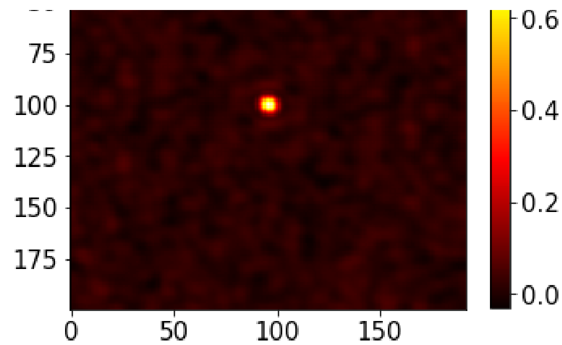
In [11]:
```
correlation=correlator(imagematrix)
```

In [12]:

```
fig, ax= plt.subplots(ncols=2,nrows=4,figsize=(12,25),squeeze=True)
for y in range(ax.shape[0]):
    for x in range(ax.shape[1]):
        index=ax.shape[1]*y+x
        im=ax[y][x].imshow(correlation[:,:,7-index]);
        fig.colorbar(im,ax=ax[y][x],shrink=0.85);
        #im.set_clim(1,0)
        ax[y][x].set_title('{1} degree\n max correlaton: {0:0.4f}\n standard d
eviation:{2:0.4f}'.format(np.max(correlation[:,:,7-index]),-30+30*index,np.std
(correlation[:,:,7-index])));
```


-30 degree
max correlaton: 0.2555
standard deviation:0.0156
0 degree
max correlaton: 1.0000
standard deviation:0.0269
30 degree
max correlaton: 0.2555
standard deviation:0.0156
60 degree
max correlaton: 0.0622
standard deviation:0.0100
90 degree
max correlaton: 0.0167
standard deviation:0.0087
120 degree
max correlaton: 0.0561
standard deviation:0.0098
150 degree
max correlaton: 0.2296
standard deviation:0.0148
180 degree
max correlaton: 0.8576
standard deviation:0.0243





180 degree correlation is not 1 because of displacement from the Glan-Thompson polarizer and misalignment