This notebook provides an example to predict the reciporcal space coverage with known latiice constant and perferred sample orientation.

Here we provide a simple experimental plan tool for single crystal scattering exp
eriments. With known lattice constant, we can creat a virtue UB for experimental plan
ning, and predcit the reciprocal space coverage, and the peak position on the detecto
r space, saved in a peaktable for each rotation angle.

Please let me know if you have good suggestions/comments.
V.02 @ Yaohua Liu, liuyh@ornl.gov, Aug 25, 2017

In [1]:
```
from __future__ import print_function
import sys, math, getopt
import numpy as np
np.set_printoptions(precision=4)

import numpy.ma as ma

sys.path.append(r'/opt/mantid/bin')
#sys.path.append(r'/Applications/MantidPlot.app/Contents/MacOS')
import mantid
from mantid.simpleapi import *

from matplotlib import pyplot as plt
from matplotlib import rcParams

# Use this line for interactive plots. Note you will have to run each cell by h
and and hit the power button in the corner
# of the plot so that all plots are not over drawn on a single axis.
%matplotlib notebook
# Use this if you want to run the entire notebook with our user interaction.
#%matplotlib inline

import matplotlib
matplotlib.rc('xtick', labelsize=18)
matplotlib.rc('ytick', labelsize=18)

from matplotlib.colors import LogNorm
```

In [2]:
```
# a copy of this example code is saved here
%pwd
```

Out[2]: u'/SNS/lustre/CORELLI/shared/PythonScripts/ExpPlan'

generate a virtual UB

Requires lattice constants and smaple orientation as an input. Since we can easily rotate the sample around the vertical direction, therefore we only need to know the vertical axis.

See more details on CreateUB.py

In [3]:
```
# Provide necessary information
outputdir = '/SNS/CORELLI/shared/PythonScripts/ExpPlan/'
UBfile = outputdir + "ub/test_ub_hexagonal.mat"

# create a workspace (or you can load one)
ws=CreateSingleValuedWorkspace(5)

#set a UB matrix using the vector along k_i as 1,1,0, and the 0,0,1 vector in the horizontal plane, and the rotation axis is 1, -1, 0
# u is the vector along K_i when goniometer is at 0; and v is in-plane vector perpendicular to k_i, when goniometer is at 0
SetUB(ws,a = 5.0,b=5.0,c = 5.6,alpha=90, beta=90, gamma=120, u="1,1,0", v="0,0,1")
UB = ws.sample().getOrientedLattice().getUB()
SaveIsawUB(InputWorkspace=ws, Filename=UBfile)
```

load a few files:

```
required:
    V data:
Optional:
    IDF file
    Detector mask file
```

In [4]:
```
# One Vandanium Run is used to load the flux and detector information. No need to change.
filename='/SNS/CORELLI/shared/PythonScripts/ExpPlan/Vdata/COR_31663.nxs.h5'
Vdata = LoadEventNexus(Filename=filename, FilterByTimeStop=10)
```

In [5]:
```
#define IDF file. Ask instrument scientists for the updated information
IDFfile = ''    # using default
#IDFfile = '/SNS/CORELLI/shared/Calibration/CORELLI_Definition_910.xml'  # newly calibrated. April, 2017
```

In [6]:
```
#define the SE
SE = ""   # "CCR", "OC", "SlimSam"
if SE in ["", "CCR"] :
    mask = LoadMask(Instrument='CORELLI', InputFile='/SNS/CORELLI/shared/Python
Scripts/MaskFiles/CCR/maskfile_03262017.xml')
elif SE == "OC" :
    mask = LoadMask(Instrument='CORELLI', InputFile='/SNS/CORELLI/shared/Python
Scripts/MaskFile_20170221.xml')
elif SE == "SlimSam" :
    mask = LoadMask(Instrument='CORELLI', InputFile='/SNS/CORELLI/shared/Python
Scripts/MaskFiles/SlimSam/SlimSam.xml')
MaskDetectors(Workspace=Vdata,MaskedWorkspace=mask)
```

define the UB file, sample rotation range, HKL projection directions and do the simulations

you can load a new UB file here. Otherwise, we will use the one generated above

In [7]:
```
#UBfile = outputdir + "ub/test_ub_hexagonal.mat"
# define the angles for virtual runs
Omegas = range(0, 61, 10)
totalrun = len(Omegas)
print("Total number of runs %d" %totalrun)

#define the projected orientation in the HKL space
proj=['1,1,0', '0,0,1', '1,-1,0']
```

```
Total number of runs 7
```

In [8]:
```
# Load the data and convert to Q space for mesh plot and peak finding.
toMerge1=[]

for index, Omega in enumerate(Omegas):
    print("Run %d of %d, Processing converting MD for run : %s" %(index+1, tota
lrun, Omega))
    ows='COR_'+str(Omega)+'deg'
    toMerge1.append(ows)
    print('Omega = %5.2f deg.' %(Omega))

    CloneWorkspace(InputWorkspace= Vdata, OutputWorkspace=ows)
    if len(IDFfile) > 0:
        LoadInstrument(Workspace= ows, Filename=IDFfile,RewriteSpectraMap=False
)
    SetGoniometer(ows,Axis0=str(Omega)+',0,1,0,1')


data = GroupWorkspaces(toMerge1)
LoadIsawUB(InputWorkspace=data,Filename=UBfile)
ConvertToMD(InputWorkspace=data,OutputWorkspace='md',QDimensions='Q3D',dEAnalys
isMode='Elastic', Q3DFrames='HKL',Uproj= proj[0],Vproj=proj[1],Wproj=proj[2],
    QConversionScales='HKL',LorentzCorrection='0', MinValues='-10.1,-10.1,-10.1
',MaxValues='10.1,10.1,10.1')
mdmesh=MergeMD('md')
```

```
Run 1 of 7, Processing converting MD for run : 0
Omega =  0.00 deg.
Run 2 of 7, Processing converting MD for run : 10
Omega = 10.00 deg.
Run 3 of 7, Processing converting MD for run : 20
Omega = 20.00 deg.
Run 4 of 7, Processing converting MD for run : 30
Omega = 30.00 deg.
Run 5 of 7, Processing converting MD for run : 40
Omega = 40.00 deg.
Run 6 of 7, Processing converting MD for run : 50
Omega = 50.00 deg.
Run 7 of 7, Processing converting MD for run : 60
Omega = 60.00 deg.
```

save the results

to view the results, you can use mantidplot, nexypy or the following notebook example
,
/SNS/CORELLI/shared/PythonScripts/notebook_examples/ImageSlice/ImageSlices.ipynb

In [9]:
```
outputfile = outputdir + "Predicted_QSpace.nxs"
SaveMD('mdmesh',Filename=outputfile)
```

to have a quick view of a slice of the coverage.

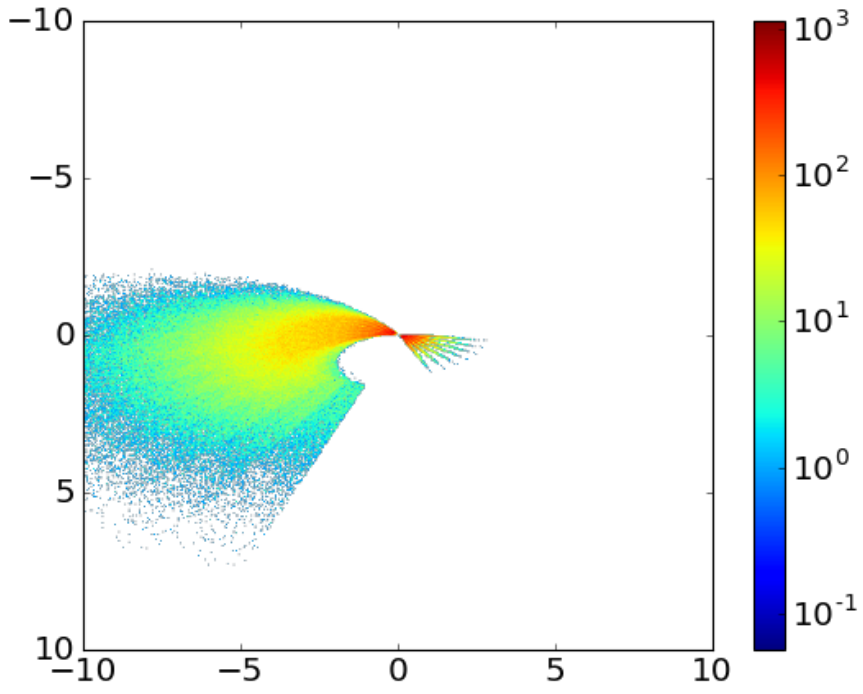
In [10]:
```
dim0 = mdmesh.getDimension(0)
d0name, d0min, d0max = dim0.name, dim0.getMinimum(), dim0.getMaximum()
d0name, d0min, d0max
```

Out[10]: ('[H,H,0]', -10.100000381469727, 10.100000381469727)

In [11]:
```
d1name = mdmesh.getDimension(1).name
d2name = mdmesh.getDimension(2).name
```

In [12]:
```
_slicedata = BinMD(InputWorkspace='mdmesh',
                   AlignedDim0= d0name +',-10,10,500',
                   AlignedDim1= d1name + ',-10,10,500',
                   AlignedDim2= d2name + ',-0.2,0.2,1',
                   )
_slice = (mtd['_slicedata'].getSignalArray() + 0)[:, :, 0]
```

In [13]:
```
imscale = [_slice.max()/1e4, _slice.max()*2]
plt.imshow(_slice, norm=LogNorm(vmin=imscale[0], vmax=imscale[1]), extent=[-10,
10, 10, -10])
#plt.imshow(_slice, norm=LogNorm(vmin=imscale[0], vmax=imscale[1]))
plt.colorbar()
```



Out[13]: <matplotlib.colorbar.Colorbar at 0x12d86650>

to predict the peaks on the detector space

```
prrovide a d range.
it is usually slow if d < 1 A.
```

In [14]:
```
#toggle the function to PredictPeaks. If turns on, need to update the Dmin and
Dmax.
predictpeaks = 1 # 1 to turn on Predictpeaks.
dRange = [1.5, 10.0]
if predictpeaks == 1:
    ppeaks =  PredictPeaks(InputWorkspace=data, WavelengthMin=0.8,
                  WavelengthMax=2.9, MinDSpacing=dRange[0], MaxDSpacing=dRange[1
])
```

In [15]:
```
print('total number of peak table %d' %(len(ppeaks)))
ppeaks[0]
```

```
total number of peak table 7
```

Out[15]:
```
PeaksWorkspace
Columns: 17
Rows: 3
1 kB
Instrument: CORELLI (2014-Feb-25 to 2100-Jan-31)Instrument from: /SNS/CORELLI/s
hared/PythonScripts/ExpPlan/Vdata/COR_31663.nxs.h5

Parameters from: /opt/Mantid/instrument/CORELLI_Parameters.xml
Run start: 2016-Sep-02 09:38:13
Run end:  2016-Sep-02 11:46:31
Sample: a 5.0, b 5.0, c 5.6; alpha 90, beta 90, gamma 120
Inelastic: ki-kf
```

In [16]:
```
peaktable0 = ppeaks[0]
peak0 = peaktable0.getPeak(0)
peak0.getHKL(), peak0.getWavelength()
```

Out[16]:
```
([-1,2,-2], 1.391057516296631)
```