# test stitching images with rotated detector using pyFAI own dummy data

## standard multigeometry

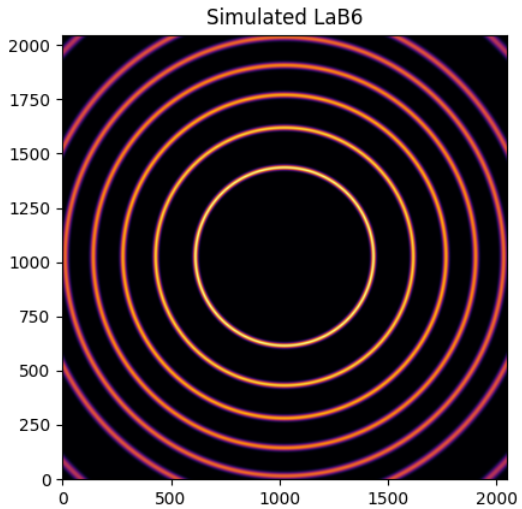
In [1]:
```
%matplotlib ipympl
# use `widget` for better user experience; `inline` is for documentation generation

import time
import copy
start_time = time.perf_counter()

import numpy as np
import matplotlib.pyplot as plt

import pyFAI
from pyFAI.multi_geometry import MultiGeometry
from pyFAI.integrator.azimuthal import AzimuthalIntegrator
from pyFAI.method_registry import IntegrationMethod
from pyFAI.gui import jupyter
print("Using pyFAI verison: ", pyFAI.version)
```

```
Using pyFAI verison:  2025.1.0
```

In [2]:
```
import pyFAI.calibrant
print("Number of known calibrants: %s"%len(pyFAI.calibrant.ALL_CALIBRANTS))
print(", ".join(pyFAI.calibrant.ALL_CALIBRANTS.keys()))

wavelength = 1e-10
LaB6 = pyFAI.calibrant.get_calibrant("LaB6")
LaB6.set_wavelength(wavelength)
print(LaB6)
print("Number of reflections for calibrant at given wavelength: %i"%len(LaB6.get_dSpacing()))
import pyFAI.detectors
det = pyFAI.detectors.Titan()
print(det)
p1, p2, p3 = det.calc_cartesian_positions()
print("Detector is flat, P3= %s"%p3)
poni1 = p1.mean()
poni2 = p2.mean()
print("Center of the detector: poni1=%s poni2=%s"%(poni1, poni2))

ai = AzimuthalIntegrator(dist=0.1, poni1=poni1, poni2=poni2, detector=det, wavelength=wavelength)
img = LaB6.fake_calibration_image(ai)
jupyter.display(img, label="Simulated LaB6")
pass
```

```
Number of known calibrants: 32
LaB6_SRM660b, Al, Si_SRM640d, Si_SRM640e, Ni, LaB6_SRM660a, C14H30O, CrOx, CeO2, vanadinite, Si_SRM640, Au, hydrocerussite, PBBA, alp
ha_Al2O3, AgBh, mock, Si_SRM640c, LaB6_SRM660c, Cr2O3, NaCl, cristobaltite, Si, TiO2, quartz, lysozyme, Si_SRM640b, Si_SRM640a, CuO,
ZnO, Pt, LaB6
LaB6 Calibrant with 59 reflections at wavelength 1e-10
Number of reflections for calibrant at given wavelength: 59
Detector Titan 2k x 2k%s         PixelSize= 60µm, 60µm
Detector is flat, P3= None
Center of the detector: poni1=0.06144 poni2=0.06143988
```

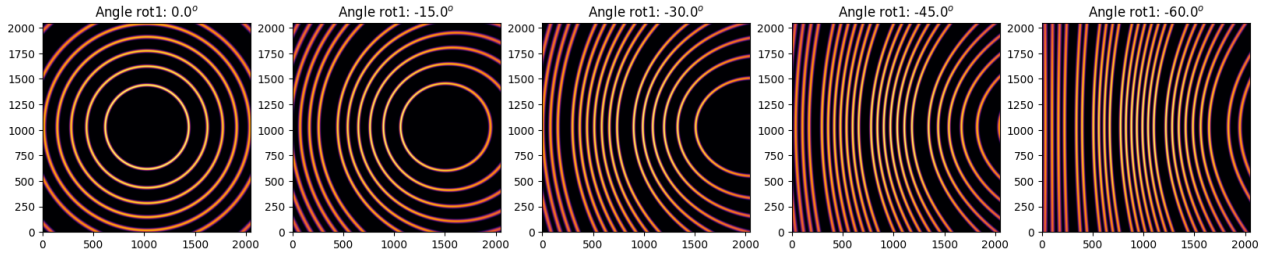
Figure




In [7]:
```
step = 15*np.pi/180
ais = []
imgs = []
fig, ax = plt.subplots(1, 5, figsize=(20,4))
for i in range(5):
    my_ai = copy.deepcopy(ai)
    my_ai.rot1 -= i*step
    #add in background to image
    my_img = (LaB6.fake_calibration_image(my_ai)*10)+2
    jupyter.display(my_img, label="Angle rot1: %.1f$^o$"%np.degrees(my_ai.rot1), ax=ax[i])
```

```
    ais.append(my_ai)
    imgs.append(my_img)
    print(my_ai)
```

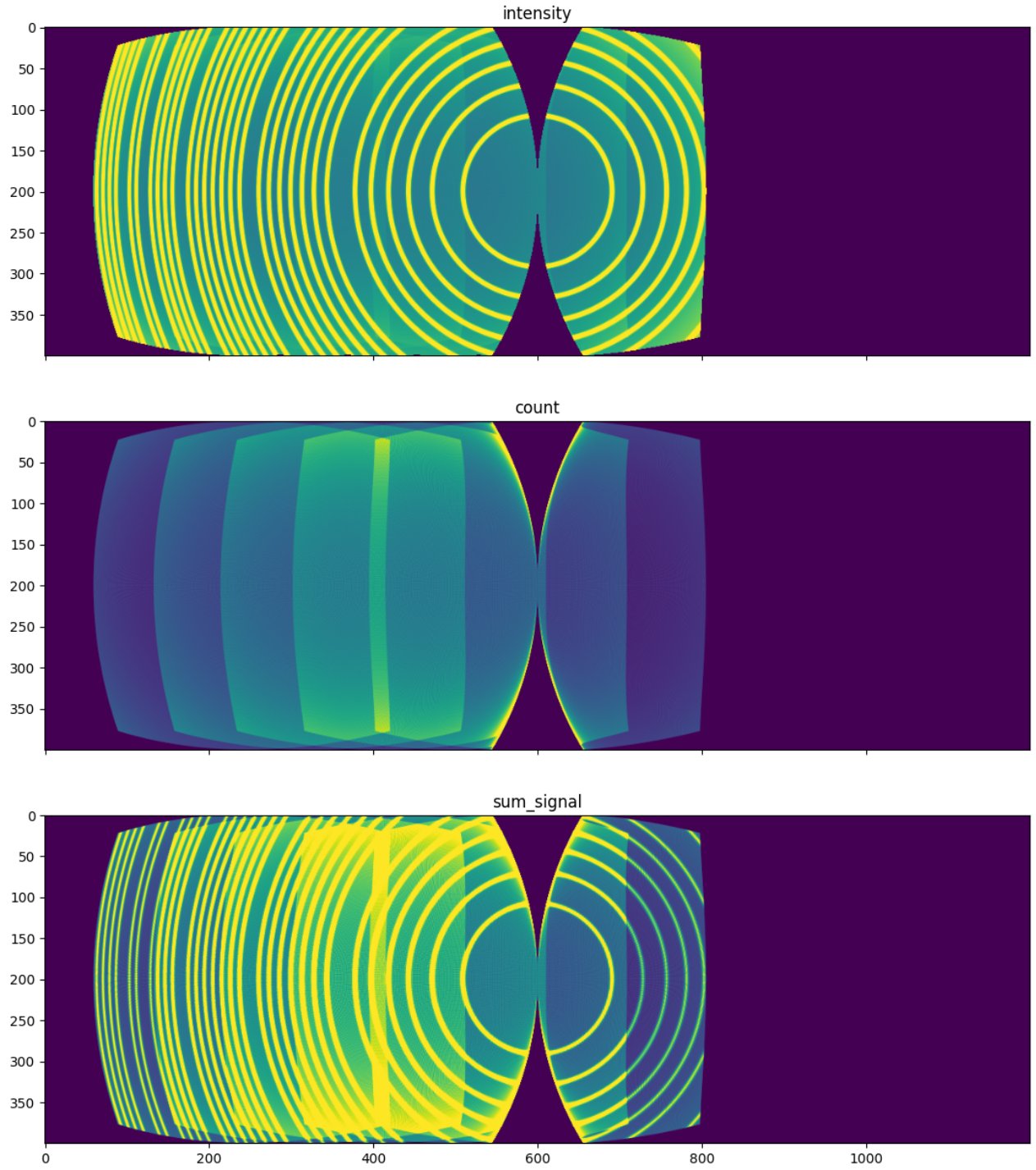
```
Detector Titan 2k x 2k%s         PixelSize= 60µm, 60µm
Wavelength= 1.000000e-10 m
SampleDetDist= 1.000000e-01 m   PONI= 6.144000e-02, 6.143988e-02 m      rot1=0.000000  rot2=0.000000  rot3=0.000000 rad
DirectBeamDist= 100.000 mm      Center: x=1023.998, y=1024.000 pix      Tilt= 0.000° tiltPlanRotation= 0.000° λ= 1.000Å
Detector Titan 2k x 2k%s         PixelSize= 60µm, 60µm
Wavelength= 1.000000e-10 m
SampleDetDist= 1.000000e-01 m   PONI= 6.144000e-02, 6.143988e-02 m      rot1=-0.261799  rot2=0.000000  rot3=0.000000 rad
DirectBeamDist= 103.528 mm      Center: x=1470.580, y=1024.000 pix      Tilt= 15.000° tiltPlanRotation= 0.000° λ= 1.000Å
Detector Titan 2k x 2k%s         PixelSize= 60µm, 60µm
Wavelength= 1.000000e-10 m
SampleDetDist= 1.000000e-01 m   PONI= 6.144000e-02, 6.143988e-02 m      rot1=-0.523599  rot2=0.000000  rot3=0.000000 rad
DirectBeamDist= 115.470 mm      Center: x=1986.248, y=1024.000 pix      Tilt= 30.000° tiltPlanRotation= 0.000° λ= 1.000Å
Detector Titan 2k x 2k%s         PixelSize= 60µm, 60µm
Wavelength= 1.000000e-10 m
SampleDetDist= 1.000000e-01 m   PONI= 6.144000e-02, 6.143988e-02 m      rot1=-0.785398  rot2=0.000000  rot3=0.000000 rad
DirectBeamDist= 141.421 mm      Center: x=2690.665, y=1024.000 pix      Tilt= 45.000° tiltPlanRotation= 0.000° λ= 1.000Å
Detector Titan 2k x 2k%s         PixelSize= 60µm, 60µm
Wavelength= 1.000000e-10 m
SampleDetDist= 1.000000e-01 m   PONI= 6.144000e-02, 6.143988e-02 m      rot1=-1.047198  rot2=0.000000  rot3=0.000000 rad
DirectBeamDist= 200.000 mm      Center: x=3910.749, y=1024.000 pix      Tilt= 60.000° tiltPlanRotation= 0.000° λ= 1.000Å
```




In [8]:
```
unit_qip_name = "qip_A^-1"
unit_qoop_name = "qoop_A^-1"
mg = MultiGeometry(ais, unit=(unit_qip_name,unit_qoop_name), radial_range=(-10, 10))
print(mg)
```

```
MultiGeometry integrator with 5 geometries on (-10, 10) radial range ((
qip_A^-1
Incident_angle=0.0°
Tilt_angle=0.0°
Sample orientation=1
,
qoop_A^-1
Incident_angle=0.0°
Tilt_angle=0.0°
Sample orientation=1
)) and None azimuthal range (deg)
```

In [9]:
```
res2d = mg.integrate2d(imgs, 1200,400,method=("no", "csr", "cython"))


fig,axs=plt.subplots(3,1,figsize=(15,15),sharey=True,sharex=True)

axs[0].imshow(res2d.intensity,vmax=res2d.intensity.mean()*2)
axs[0].set_title('intensity')
axs[1].imshow(res2d.count,vmax=res2d.count.mean()*5)
axs[1].set_title('count')
axs[2].imshow(res2d.sum_signal,vmax=res2d.sum_signal.mean()*2)
axs[2].set_title('sum_signal')

pass
```




## Fiber multigeometry

In [27]:
```
%matplotlib ipympl
# use `widget` for better user experience; `inline` is for documentation generation

import time
start_time = time.perf_counter()

import matplotlib.pyplot as plt
import copy

import pyFAI
from pyFAI.method_registry import IntegrationMethod
from pyFAI.gui import jupyter
print("Using pyFAI version: ", pyFAI.version)
import pyFAI.calibrant
print("Number of known calibrants: %s"%len(pyFAI.calibrant.ALL_CALIBRANTS))
print(", ".join(pyFAI.calibrant.ALL_CALIBRANTS.keys()))
```

```
Using pyFAI version:  2025.1.0
Number of known calibrants: 32
LaB6_SRM660b, Al, Si_SRM640d, Si_SRM640e, Ni, LaB6_SRM660a, C14H30O, CrOx, CeO2, vanadinite, Si_SRM640, Au, hydrocerussite, PBBA, alp
ha_Al2O3, AgBh, mock, Si_SRM640c, LaB6_SRM660c, Cr2O3, NaCl, cristobaltite, Si, TiO2, quartz, lysozyme, Si_SRM640b, Si_SRM640a, CuO,
ZnO, Pt, LaB6
```

In [28]:
```
wavelength = 1e-10
LaB6 = pyFAI.calibrant.get_calibrant("LaB6")
LaB6.set_wavelength(wavelength)
```

```
print(LaB6)
print("Number of reflections for calibrant at given wavelength: %i"%len(LaB6.get_dSpacing()))
import pyFAI.detectors
det = pyFAI.detectors.Titan()
print(det)
p1, p2, p3 = det.calc_cartesian_positions()
print("Detector is flat, P3= %s"%p3)
poni1 = p1.mean()
poni2 = p2.mean()
print("Center of the detector: poni1=%s poni2=%s"%(poni1, poni2))
```

```
LaB6 Calibrant with 59 reflections at wavelength 1e-10
Number of reflections for calibrant at given wavelength: 59
Detector Titan 2k x 2k%s         PixelSize= 60µm, 60µm
Detector is flat, P3= None
Center of the detector: poni1=0.06144 poni2=0.06143988
```

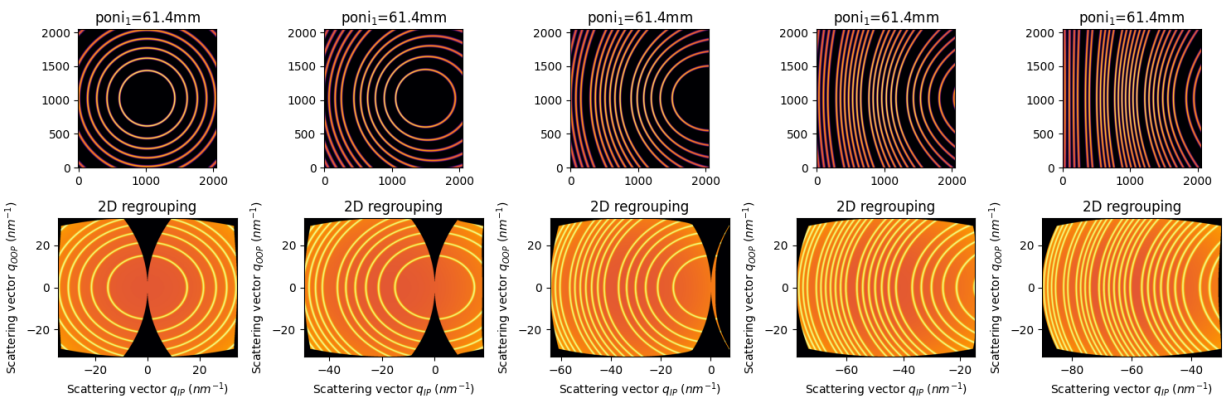
In [30]:
```
from pyFAI.integrator.fiber import FiberIntegrator
fi = FiberIntegrator(dist=0.1, poni1=poni1, poni2=poni2, detector=det, wavelength=wavelength)
print(fi)

#Selection of the methods for integrating
method1d = IntegrationMethod.parse("full", dim=1)
method2d = IntegrationMethod.select_one_available(("pseudo","histogram","cython"), dim=2)
print(f"Integration method in 1d: {method1d}\nIntegration method in 2d: {method2d}")
```

```
Detector Titan 2k x 2k%s         PixelSize= 60µm, 60µm
Wavelength= 1.000000e-10 m
SampleDetDist= 1.000000e-01 m   PONI= 6.144000e-02, 6.143988e-02 m      rot1=0.000000  rot2=0.000000  rot3=0.000000 rad
DirectBeamDist= 100.000 mm      Center: x=1023.998, y=1024.000 pix      Tilt= 0.000° tiltPlanRotation= 0.000° λ= 1.000Å
Incident angle: 0.00° Tilt angle 0.00° Sample orientation 1
Integration method in 1d: IntegrationMethod(1d int, full split, histogram, cython)
Integration method in 2d: IntegrationMethod(2d int, pseudo split, histogram, cython)
```

In [31]:
```
step = 15*np.pi/180
fis = []
imgs = []
fig, ax = plt.subplots(2,5, figsize=(15,5))
for i in range(5):
    my_fi = copy.deepcopy(fi)
    #my_fi.poni1 -= i*poni1
    my_fi.rot1 -= i*step
    #add in background to image
    my_img = (LaB6.fake_calibration_image(my_fi)*10)+2
    jupyter.display(my_img, label="poni$_1$=%3.1fmm"%(1e3*my_fi.poni1), ax=ax[0,i])
    res2d = my_fi.integrate2d_grazing_incidence(data=my_img)
    jupyter.plot2d(res2d, ax=ax[1,i])
    fis.append(my_fi)
    imgs.append(my_img)
    print(my_fi)
fig.tight_layout()
```

```
Detector Titan 2k x 2k%s         PixelSize= 60µm, 60µm
Wavelength= 1.000000e-10 m
SampleDetDist= 1.000000e-01 m   PONI= 6.144000e-02, 6.143988e-02 m      rot1=0.000000  rot2=0.000000  rot3=0.000000 rad
DirectBeamDist= 100.000 mm      Center: x=1023.998, y=1024.000 pix      Tilt= 0.000° tiltPlanRotation= 0.000° λ= 1.000Å
Incident angle: 0.00° Tilt angle 0.00° Sample orientation 1
Detector Titan 2k x 2k%s         PixelSize= 60µm, 60µm
Wavelength= 1.000000e-10 m
SampleDetDist= 1.000000e-01 m   PONI= 6.144000e-02, 6.143988e-02 m      rot1=-0.261799  rot2=0.000000  rot3=0.000000 rad
DirectBeamDist= 103.528 mm      Center: x=1470.580, y=1024.000 pix      Tilt= 15.000° tiltPlanRotation= 0.000° λ= 1.000Å
Incident angle: 0.00° Tilt angle 0.00° Sample orientation 1
Detector Titan 2k x 2k%s         PixelSize= 60µm, 60µm
Wavelength= 1.000000e-10 m
SampleDetDist= 1.000000e-01 m   PONI= 6.144000e-02, 6.143988e-02 m      rot1=-0.523599  rot2=0.000000  rot3=0.000000 rad
DirectBeamDist= 115.470 mm      Center: x=1986.248, y=1024.000 pix      Tilt= 30.000° tiltPlanRotation= 0.000° λ= 1.000Å
Incident angle: 0.00° Tilt angle 0.00° Sample orientation 1
Detector Titan 2k x 2k%s         PixelSize= 60µm, 60µm
Wavelength= 1.000000e-10 m
SampleDetDist= 1.000000e-01 m   PONI= 6.144000e-02, 6.143988e-02 m      rot1=-0.785398  rot2=0.000000  rot3=0.000000 rad
DirectBeamDist= 141.421 mm      Center: x=2690.665, y=1024.000 pix      Tilt= 45.000° tiltPlanRotation= 0.000° λ= 1.000Å
Incident angle: 0.00° Tilt angle 0.00° Sample orientation 1
Detector Titan 2k x 2k%s         PixelSize= 60µm, 60µm
Wavelength= 1.000000e-10 m
SampleDetDist= 1.000000e-01 m   PONI= 6.144000e-02, 6.143988e-02 m      rot1=-1.047198  rot2=0.000000  rot3=0.000000 rad
DirectBeamDist= 200.000 mm      Center: x=3910.749, y=1024.000 pix      Tilt= 60.000° tiltPlanRotation= 0.000° λ= 1.000Å
Incident angle: 0.00° Tilt angle 0.00° Sample orientation 1
```

Figure



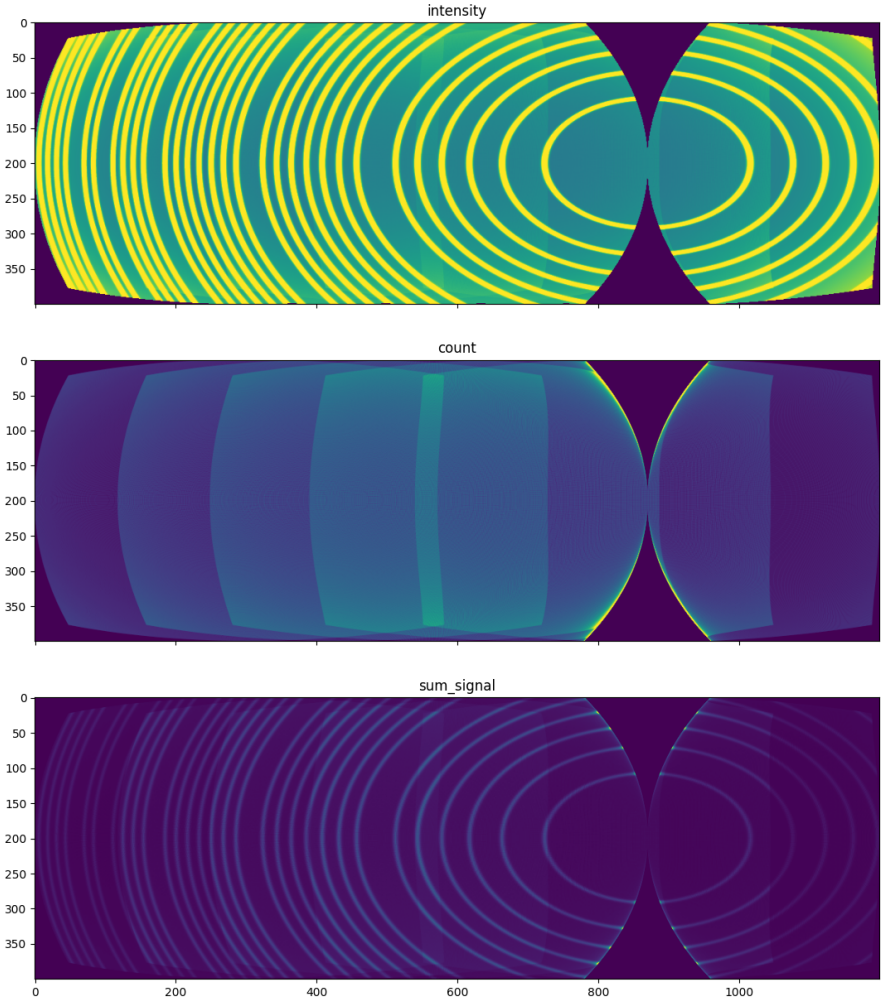

In [32]:
```
from pyFAI.multi_geometry import MultiGeometryFiber
```

```
mg = MultiGeometryFiber(fis, unit=("qip_nm^-1", "qoop_nm^-1"))
print(mg)
```

```
MultiGeometry integrator with 5 geometries on None radial range ((
qip_nm^-1
Incident_angle=0.0°
Tilt_angle=0.0°
Sample orientation=1
,
qoop_nm^-1
Incident_angle=0.0°
Tilt_angle=0.0°
Sample orientation=1
)) and None azimuthal range (deg)
```

In [33]:
```
res2d = mg.integrate2d(imgs, 1200,400)


fig,axs=plt.subplots(3,1,figsize=(15,15),sharey=True,sharex=True)
axs[0].imshow(res2d.intensity,vmax=res2d.intensity.mean()*1.2)
axs[0].set_title('intensity')
axs[1].imshow(res2d.count,vmax=res2d.count.mean()*5)
axs[1].set_title('count')
axs[2].imshow(res2d.sum_signal,vmax=res2d.sum_signal.mean()*20)
axs[2].set_title('sum_signal')

pass
```

Figure




In [ ]:

In [ ]: