In [2]:

```
from __future__ import print_function, unicode_literals, absolute_import, division
import numpy as np
import matplotlib
matplotlib.rcParams["image.interpolation"] = None
import matplotlib.pyplot as plt
%matplotlib inline
%config InlineBackend.figure_format = 'retina'

from glob import glob
from tqdm import tqdm
from tifffile import imread
from csbdeep.utils import Path, download_and_extract_zip_file

from stardist import relabel_image_stardist3D, Rays_GoldenSpiral, calculate_extents
from stardist import fill_label_holes, random_label_cmap
from stardist.matching import matching_dataset

np.random.seed(42)
lbl_cmap = random_label_cmap()
```

Data

This notebook demonstrates how the training data for *StarDist* should look like and whether the annotated objects can be appropriately described by star-convex polyhedra.

> The training data that needs to be provided for StarDist consists of corresponding pairs of raw images and pixelwise annotated ground truth images (masks), where every pixel has a unique integer value indicating the object id (or 0 for background).

For this demo we will download the file file `demo3D.zip` that contains synthetic train and test images with associated ground truth labels.

In [3]:

```
#download_and_extract_zip_file(
#    url       = 'https://github.com/mpicbg-csbd/stardist/releases/download/0.3.0/demo3
D.zip',
#    targetdir = 'data',
#    verbose   = 1,
#)
```

In [4]:

```
X = sorted(glob('data/train/images/*.tif'))
Y = sorted(glob('data/train/masks/*.tif'))
print(X)
#assert all(Path(x).name==Path(y).name for x,y in zip(X,Y))
```

```
['data/train/images\\C1-zone1Area1_images-sub1.tif', 'data/train/images\\C
1-zone1Area1_images-sub2.tif', 'data/train/images\\C1-zone1Area1_images-su
b3.tif', 'data/train/images\\C1-zone1Area2_images.tif']
```

Load only a small subset.

In [5]:

```
X, Y = X[:10], Y[:10]
print(X)
```

```
['data/train/images\\C1-zone1Area1_images-sub1.tif', 'data/train/images\\C
1-zone1Area1_images-sub2.tif', 'data/train/images\\C1-zone1Area1_images-su
b3.tif', 'data/train/images\\C1-zone1Area2_images.tif']
```

In [6]:

```
X = list(map(imread,X))
Y = list(map(imread,Y))
print(len(X))
print(len(Y))
```

```
4
4
```

In [7]:

```
extents = calculate_extents(Y)
print(str(extents))
anisotropy = tuple(np.max(extents) / extents)
print('empirical anisotropy of labeled objects = %s' % str(anisotropy))
anisotropy=(0.2069,0.2069,0.5)
print('given anisotropy of labeled objects = %s' % str(anisotropy))
```

```
[ 6.5 51.5 47.5]
empirical anisotropy of labeled objects = (7.923076923076923, 1.0, 1.08421
05263157895)
given anisotropy of labeled objects = (0.2069, 0.2069, 0.5)
```

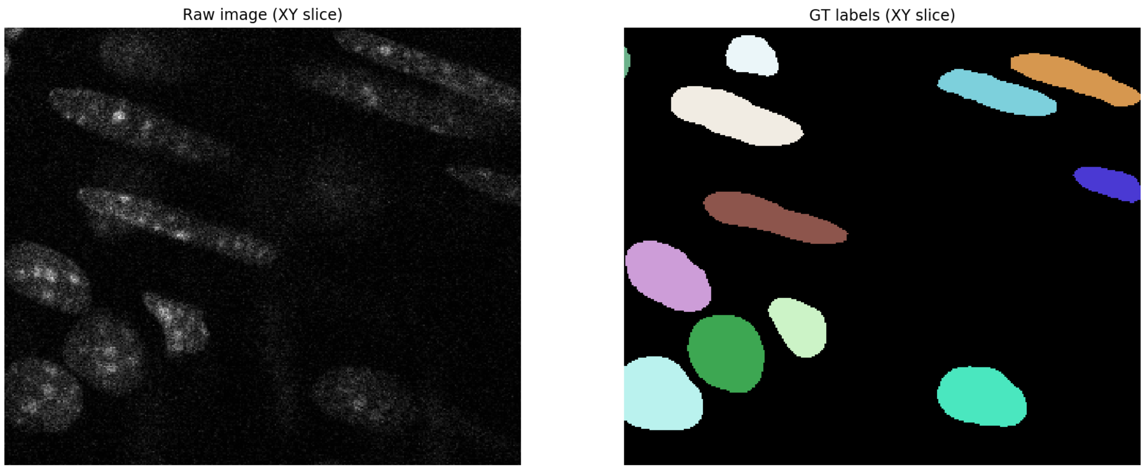
Example image

In [8]:

```
i = 0
img, lbl = X[i], fill_label_holes(Y[i])
assert img.ndim in (3,4)
# assumed axes ordering of img and lbl is: ZYX(C)
```

In [9]:

```
plt.figure(figsize=(16,10))
z = img.shape[0] // 2
y = img.shape[1] // 2
plt.subplot(121); plt.imshow(img[z],cmap='gray');   plt.axis('off'); plt.title('Raw image (XY slice)')
plt.subplot(122); plt.imshow(lbl[z],cmap=lbl_cmap); plt.axis('off'); plt.title('GT labels (XY slice)')
plt.figure(figsize=(16,10))
plt.subplot(121); plt.imshow(img[:,y],cmap='gray');   plt.axis('off'); plt.title('Raw image (XZ slice)')
plt.subplot(122); plt.imshow(lbl[:,y],cmap=lbl_cmap); plt.axis('off'); plt.title('GT labels (XZ slice)')
None;
```



Fitting ground-truth labels with star-convex polyhedra

In [10]:

```
def reconstruction_scores(n_rays, anisotropy):
    scores = []
    for r in tqdm(n_rays):
        rays = Rays_GoldenSpiral(r, anisotropy=anisotropy)
        Y_reconstructed = [relabel_image_stardist3D(lbl, rays) for lbl in Y]
        mean_iou = matching_dataset(Y, Y_reconstructed, thresh=0, show_progress=False).mean_true_score
        scores.append(mean_iou)
    return scores
```

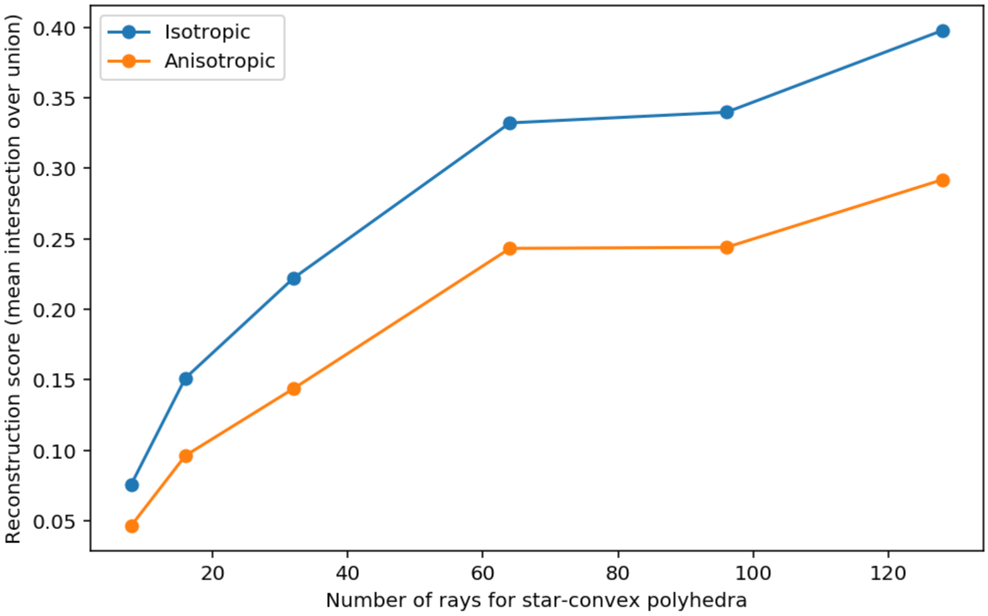
In [11]:

```
n_rays = [8, 16, 32, 64, 96, 128]
scores_iso   = reconstruction_scores(n_rays, anisotropy=None)
scores_aniso = reconstruction_scores(n_rays, anisotropy=anisotropy)
```

```
100%|██████████████████████████████████████████████████| 6/6 [51:47<00:00, 517.95s/it]
100%|██████████████████████████████████████████████████| 6/6 [46:24<00:00, 464.09s/it]
```

In [12]:

```
plt.figure(figsize=(8,5))
plt.plot(n_rays, scores_iso,   'o-', label='Isotropic')
plt.plot(n_rays, scores_aniso, 'o-', label='Anisotropic')
plt.xlabel('Number of rays for star-convex polyhedra')
plt.ylabel('Reconstruction score (mean intersection over union)')
plt.legend()
None;
```

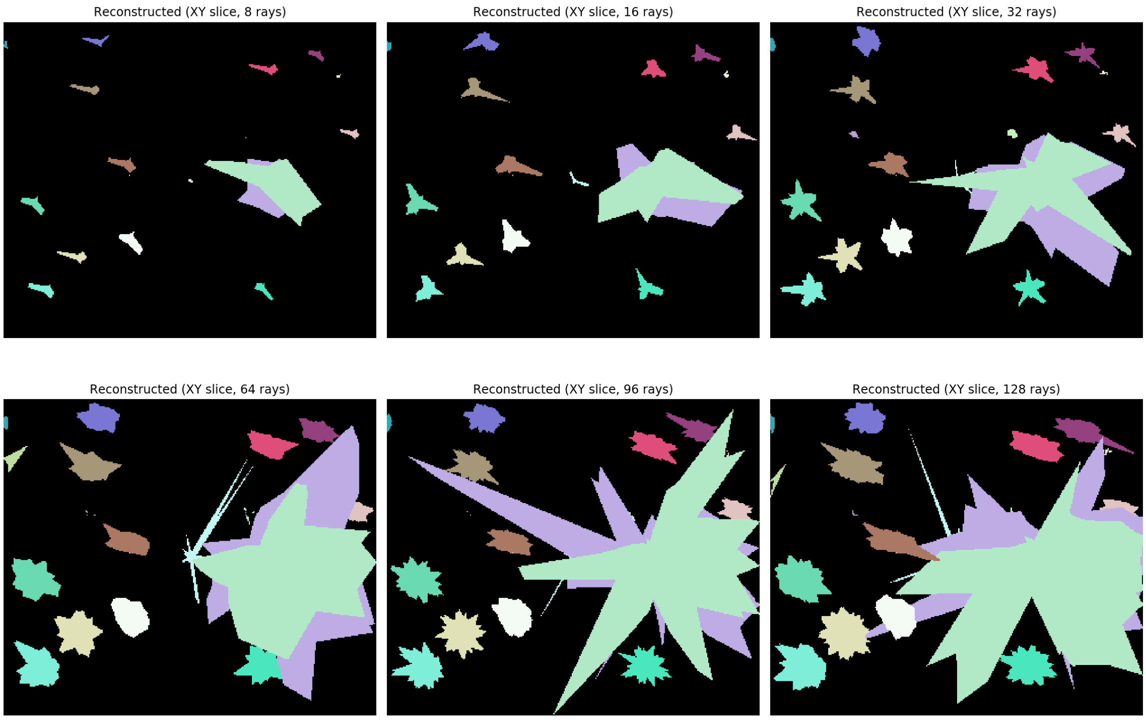


Example image reconstructed with various number of rays

Without taking anisotropy into account

In [13]:

```
fig, ax = plt.subplots(2,3, figsize=(16,11))
for a,r in zip(ax.flat,n_rays):
    z = lbl.shape[0] // 2
    rays = Rays_GoldenSpiral(r, anisotropy=None)
    a.imshow(relabel_image_stardist3D(lbl, rays)[z], cmap=lbl_cmap)
    a.set_title('Reconstructed (XY slice, %d rays)' % r)
    a.axis('off')
plt.tight_layout();
```



Taking anisotropy into account

In [ ]:

```
fig, ax = plt.subplots(2,3, figsize=(16,11))
for a,r in zip(ax.flat,n_rays):
    z = lbl.shape[0] // 2
    rays = Rays_GoldenSpiral(r, anisotropy=anisotropy)
    a.imshow(relabel_image_stardist3D(lbl, rays)[z], cmap=lbl_cmap)
    a.set_title('Reconstructed (XY slice, %d rays)' % r)
    a.axis('off')
plt.tight_layout();
```