How the whole data collecting works:

1. Use Azure.py to collect rgb, depthmap and intrinsic matrix.
2. With the data we get from step 1, use Quan Miao's program to get every frame's corresponding rgb, depthmap, mask, bounding box, intrinsic and extrinsic matrix.
3. Generate corresponding 2d - 3d points through 3d_2d.py with the data from step 2.

I will set all the file directory for you, and will tell you where to put what data.

Azure.py part:

The rgb image will be put under ./rgb.
The depth map will be put under ./depth_map
The intrinsic matrix for depth camera will be put under ./intrinsic_depth
The intrinsic matrix for color camera will be put under ./intrinsic_rgb

Azure.py also has some commented lines to get imu_data/point_cloud/mask/bounding_box.

```
config = k4a.Config(
    color_resolution=k4a.ColorResolution.RES_720P,
    depth_mode=k4a.DepthMode.WFOV_2X2BINNED,
    synchronized_images_only=True,
    camera_fps=k4a.FPS.FPS_30,
)
```

Here to change the settings of cameras. First line to changes resolution color camera, second line changes depth camera, last line changes fps.

3d_2d.py part:
Here, all the data is from Quan Miao's program.

The intrinsics.json should be put under ./dataset
The depth.png should be put under ./dataset/depth
The mask.png should be put under ./dataset/mask
The pose.npy should be put under ./dataset/pose
The rgb.png should be put under ./dataset/rgb

The results from 3d_2d.py will be put under:

The point_cloud.ply will be put under ./dataset/point_cloud
The 2d_array.pkl and 2d_array.txt will be put under ./dataset/2d_array
The 3d_array.pkl and 3d_array.txt will be put under ./dataset/3d_array

Note: All the data should be named from 0, for example: 0.png, 1.png, 2.png...
You must have the same number of files of depthmap, mask, pose and rgb.

How to test if the result is right:
Use the first module of test.ipynb:

Change the path in :

```
points_2d = np.loadtxt("./dataset/2d_array/0.txt")
points_3d = np.loadtxt("./dataset/3d_array/0.txt")
```

Under our test, the error is 1/(640x480), this error may float a litte, but won't much. It is caused by the float to int during 3d to 2d.

You can change the error threshold here, now it's set to 1:

```
if error > 1:
        print('2D and 3D points do not correspond.')
    else:
        print('2D and 3D points correspond.')
```

The pickle format test is commented under loadtxt, just comment txt loading and make pickle reading valid, you can test pickle files.