20 Objects Light Dataset

October 30, 2014

All data regarding our ECCV 14 paper can be downloaded from our project page: `http://www.inf.tu-dresden.de/index.php?node_id=3629&ln=en`. If you run into problems contact: eric <*dot*> brachmann <*at*> tu-dresden.de.

1 Overview

We recorded 20 textured and texture-less objects, each under three different lighting conditions resulting in three sequences per object. We used a standard RGB-D (Kinect) camera. Kinect Fusion tracked the 6DoF camera pose of the camera, and, additionally, provided a 3D mesh of the scene for each sequence. We segmented the 3D meshes with a tight 3D bounding box around the object. The 2D projection of this 3D segmentation yields the object mask for each image. All three sequences of each object are aligned using the marker board the object stands on, so all given poses of each object live in the same global coordinate system.

Each object was recorded under the following three lighting conditions:

light bright artificial light (diffuse light)

dark darker natural light (diffuse light)

spot artificial spot light (directional light)

Additional information about the recording procedure can be found in our ECCV14 paper[1] and its supplementary material. If you use this dataset, please cite the aforementioned paper.

2 Structure

The dataset is structured as follows: At the top level there are 60 folders, one for each sequence recorded (20 objects x 3 lighting conditions). These sequence folders are named according to the following scheme:

`Kinfu_<object name>_<lighting condition>`

Each sequence folder contains 4 sub-folders with the actual sequence data. Each sequence data item is named after the following scheme:

`<data prefix>_<image number>_<data extension>`

Note that not all image numbers are taken. The sequence data is split into `rgb_noseg`, `depth_noseg`, `mask` and `info`. Each sub-folder will be explained below. The sequence folders ending with "light" contain two additional data items: `object.obj` and `object.xyz`. They will also be explained below.

2.1 rgb_noseg

These folders contain rgb images. Each image is a 3 channel 8 bit (unsigned char) PNG file.

2.2 depth_noseg

These folders contain depth images. Each image is a 1 channel 16 bit (unsigned short) PNG file. The depth values are stored in millimeters. A depth value of 0 means missing depth.

2.3 mask

These folders contain object segmentation masks. Each image is a 1 channel 8 bit (unsigned char) PNG file. A pixel is either black which means background or white which means object.

2.4 info

These folders contain meta data files (pose + object size). Each file is a text file of the following format:

```
image size
<iw> <ih>
<sequence name>
rotation:
<r1> <r2> <r3>
<r4> <r5> <r6>
<r7> <r8> <r9>
center:
<t1> <t2> <t3>
extent:
<ow> <oh> <od>
```

Image width `<iw>` and image height `<ih>` are measured in pixels, and are always 640 resp. 480. The rotation and center entries are combined to transformation $T_{o \to c}$ in the following way:

$$T_{o \to c} = \begin{bmatrix} r1 & r2 & r3 & t1 \\ r4 & r5 & r6 & t2 \\ r7 & r8 & r9 & t3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$T_{o\to c}$ maps 3D coordinates in the object coordinate system (i.e. *object coordinates*) to 3D coordinates in the camera coordinate system. Note that the camera viewing direction is the negative Z-axis. All coordinates are assumed to be measured in meters. The inverse transform $T_{c\to o} = T_{o\to c}^{-1}$. The last three entries `<ow>`, `<oh>` and `<od>` represent object width, object height and object depth, respectively. They are measured in meters.

2.5 object.obj

This Wavefront OBJ file contains an untextured mesh of the object. This file is only given for "light" sequences, and is to be used for the "dark" and "spot" sequences of the associated object as well. The coordinates of the mesh file are measured in meters, and live in the coordinate system of the first camera of the "light" sequence. Note that in order to render the mesh according to the given poses you have to transform its vertices with the inverse transformation of the first "light" camera, and then with the given transformation of your current camera.

2.6 object.xyz

This file contains a point cloud of the object. Each line contains `<x> <y> <z>` of one 3D point. This file is only given for "light" sequences, and is to be used for the "dark" and "spot" sequence of the associated object as well. The coordinates of the points are measured in meters and live in the coordinate system of the first camera of the "light" sequence. Note that in order to render the point cloud according to the given poses you have to transform its vertices with the inverse transformation of the first "light" camera, and then with the given transformation of your current camera.

3 Calculation of Additional Information

This section explains how to derive additional information from depth data.

3.1 Camera Coordinates

For each pixel of an RGB-D image you can calculate the X_c, Y_c, Z_c-coordinates of the pixel in the camera coordinate system in meters:

$$X_c = (x - \frac{w_i}{2})\frac{d}{1000f} \tag{2}$$

$$Y_c = -(y - \frac{h_i}{2})\frac{d}{1000f} \tag{3}$$

$$Z_c = \frac{-d}{1000} \tag{4}$$

Where x and y are pixel coordinates measured from the top left image corner, and d is the associated depth in millimeters. Image width w_i and image height h_i are 640 and 480, respectively. We use a focal length f of 575.816px. The factor 1000 in all three equations converts millimeters in meters.

3.2 Object Coordinates

Each camera coordinate can be converted to an object coordinate by applying the transformation $T_{c\to o}$:

$$\begin{pmatrix} X_o \\ Y_o \\ Z_o \\ 1 \end{pmatrix} = T_{c\to o} \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} \tag{5}$$

Note that this only makes sense if the pixel lies within the object segmentation mask.

References

[1] Brachmann, E., Krull, A., Michel, F., Gumhold, S., Shotton, J., Rother, C.: Learning 6d object pose estimation using 3d object coordinates. In Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., eds.: Computer Vision – ECCV 2014. Volume 8690 of Lecture Notes in Computer Science. Springer International Publishing (2014) 536–551