LabelMe3D database documentation*

Bryan C. Russell
Computer Science and Engineering
University of Washington
bcr@cs.washington.edu

Antonio Torralba
Computer Science and Artificial Intelligence Laboratory,
Massachusetts Institute of Technology
torralba@csail.mit.edu

1 Introduction

The goal of this document is to provide information on the conventions used in the LabelMe3D database. For more detailed information on how the system works, please see our CVPR 2009 paper.

The LabelMe3D database contains images, 2D object annotations (each annotation is specified as a string tag and 2D polygon), and 3D world coordinates for each of the objects. For each image in the database, there is an XML file containing the object annotations and 3D information. There is Matlab code for visualizing and manipulating the database.

In this document, we will refer to functions in the LabelMe3D Matlab toolbox. You may find

*Based on the paper *"Building a database of 3D scenes from user annotations"*, which appeared in CVPR 2009.

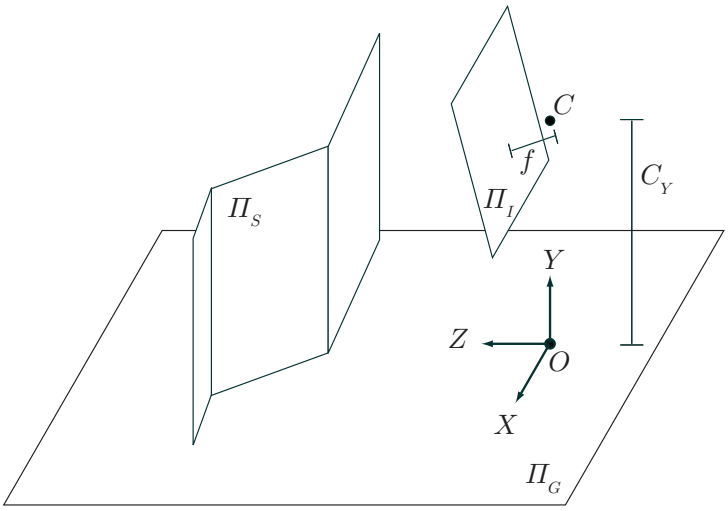

Figure 1. Camera and 3D scene layout assumptions. We assume that a scene is composed of objects, with each object represented as piecewise-connected planes Π_S standing on a ground plane Π_G. The 3D scene is projected onto the image plane Π_I via perspective projection. We assume that the world origin O lives on the ground plane at the foot of the camera center C. The origin of the image is at the center of the image plane.

it helpful to download the LabelMe3D Matlab toolbox and follow along. It will also be helpful to have a background in linear algebra.

This document is organized as follows: we start with an explanation of the 3D scene layout and camera model in Section 2. We provide a brief overview of projective geometry in Section 3. We give details on the structure of the LabelMe3D XML files in Section 5.

2 3D scene layout and camera model

This section provides details on the 3D scene layout and the camera model.

Figure 1 shows the 3D scene layout assumptions for LabelMe3D. We assume that a scene is composed of a set of objects standing on a ground plane Π_G. Each object is represented as a set of piecewise-connected planes Π_S. The 3D scene is projected onto the image plane Π_I through the camera center C via perspective projection.

We assume that the world origin O lives on the ground plane at the foot of the camera center. Therefore, the world coordinates of the camera center is $(0, C_Y, 0)^T$, where C_Y is the perpendicular distance (measured in world units) of the camera center to the ground plane. Also, all

points on the ground plane have Y coordinates equal to zero. The world axes are oriented so that the Z axis points forward into the scene, the X axis points left, and the Y axis points up. Note that the world axes are a right-handed coordinate system.

The image plane coordinate system has the x axis pointing to the right and the y axis pointing down, with the center of the top-left pixel having coordinates (1,1) and the center of the bottom-right pixel having coordinates (w, h). Here, w and h are the image width and height, respectively. Notice that this is a left-handed coordinate system. All annotations stored in the LabelMe3D XML files use the left-handed coordinate system.

3 Projective geometry overview

This section gives a brief overview of projective geometry. Projective geometry allows us to describe the relationship between 3D scene points and corresponding 2D image points. For more depth on projective geometry, please see the Hartley and Zisserman book.

For convenience, we will represent points in homogeneous coordinates and work in projective space. For example, an image point (x, y) in $\mathbb{R}^2$ can be represented in homogeneous coordinates as:

$$\mathbf{x} = (x, y, 1)^T \tag{1}$$

To convert[1] from homogeneous coordinates (x, y, w) back to $\mathbb{R}^2$, we simply divide by the last component to get $(x/w, y/w)$. A 3D point is written in homogeneous coordinates as:

$$\mathbf{X} = (X, Y, Z, 1)^T \tag{2}$$

LabelMe3D recovers a 3×4 camera matrix $\mathbf{P}$ that projects 3D scene points to 2D image points on the image plane:

$$\mathbf{x} = \mathbf{PX} \tag{3}$$

The Matlab function `getCameraMatrix.m` accesses the recovered camera matrix.

The camera matrix factorizes into the following form:

$$\mathbf{P} = \mathbf{KR}\,[\mathbf{I} \mid -\mathbf{C}] \tag{4}$$

[1] Think of homogeneous coordinates as a *promise* to divide at the very end.

where $\mathbf{K}$ is the 3×3 internal camera parameter matrix, $\mathbf{R}$ is a 3×3 rotation matrix, $\mathbf{I}$ is the 3×3 identity matrix, and $\mathbf{C}$ is the 3D location of the camera center (three-dimensional vector). This decomposition can be achieved using the Matlab function `decomposeP.m`.

In general, the internal camera matrix has the following form:

$$\mathbf{K} = \begin{pmatrix} \alpha_x f & s & p_x \\ 0 & \alpha_y f & p_y \\ 0 & 0 & 1 \end{pmatrix} \tag{5}$$

where f is the focal length, (α_x, α_y) are the pixel width and height, s is skew, and (p_x, p_y) is the principal point (the 2D image location of the intersection of the image plane with the perpendicular line that passes through the camera center). In LabelMe3D, we assume square pixels ($\alpha_x = \alpha_y = 1$) and no skew ($s = 0$).

The rotation matrix has the following form:

$$\mathbf{R}(\theta, \mathbf{n}) = \mathbf{I} + \sin\theta[\mathbf{n}]_\times + (1 - \cos\theta)[\mathbf{n}]_\times^2 \tag{6}$$

where $\mathbf{n} = (n_x, n_y, n_z)^T$ is a unit length vector that specifies the 3D axis of rotation, θ is the angle of rotation through the axis, and $[\mathbf{n}]_\times$ is a skew-symmetric matrix:

$$[\mathbf{n}]_\times = \begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix} \tag{7}$$

In LabelMe3D, we assume that there is no yaw (i.e. no panning from left to right). Therefore, the Z axis faces forward into the scene and $n_y = 0$. We represent the rotation parameters by the vector $\omega = \theta \cdot \mathbf{n} = (\omega_x, 0, \omega_z)^T$.

In summary, the camera matrix $\mathbf{P}$ specifies the following parameters recovered by the system: f, p_x, p_y, ω_x, ω_z, C_Y.

4 Object representation

This section provides details on how objects are represented in the 3D scene in LabelMe3D. An object can be represented as one of the following geometric primitives: standing planes,

Figure 2. Different polygon and edge types. Polygons are classified as either *ground plane* **(green),** *standing planes* **(red), or** *part* **(yellow). Edges are classified as** *contact* **(white),** *occluded* **(black), or** *attached* **(gray). Only objects whose 3D coordinates can be found are shown. Best viewed in color.**

ground planes, or parts. Figure 2 illustrates objects that are represented by standing planes, ground plane, or part.

4.1 Standing planes and ground plane

A plane is parameterized as $\Pi = (\Pi_X, \Pi_Y, \Pi_Z, \Pi_W)$. 3D points (X, Y, Z) on a plane satisfy the following:

$$\Pi_X X + \Pi_Y Y + \Pi_Z Z + \Pi_W = 0 \tag{8}$$

Objects represented by standing planes are formed by a set of piecewise-connected 3D planes that are orthogonal to the ground plane. This implies that for standing planes, $\Pi_Y = 0$. An object represented as a ground plane has the plane passing through the world origin and X-Z axes. This implies that for a ground plane, $\Pi_X = \Pi_Z = \Pi_W = 0$.

5 Structure of LabelMe3D XML files

The LabelMe3D code takes as input a LabelMe XML annotation file and inserts 3D information in two places: the `<camera>` and `<world3d>` tags. The `<camera>` tag contains the camera matrix $\mathbf{P}$. The `<world3d>` tag contains the 3D information for an object. The following is the basic structure of a LabelMe3D XML file (note that in general some of the LabelMe XML tags, such as `<source>`, may not be present):

`<annotation>`
- `<filename>` - Image filename
- `<folder>` - Image folder
- `<source>`
  - `<sourceImage>` - Source of image
  - `<sourceAnnotation>` - Source of annotation
  - `<submittedBy>` - Indicates who uploaded the image
- `</source>`
- `<scenedescription>` - Keywords for scene
- `<object>`
  - `<name>` - Object name
  - `<deleted>` - `1` indicates that the object is deleted
  - `<verified>` - Object polygon has been verified
  - `<date>` - Timestamp when polygon was created
  - `<polygon>`
    - `<username>` - LabelMe username of person who created polygon
    - `<pt>`
      - `<x>` - X coordinate of polygon point in image
      - `<y>` - Y coordinate of polygon point in image
    - `</pt>`
  - `</polygon>`
  - `<viewpoint>`
    - `<azimuth>` - Azimuth angle of object
  - `</viewpoint>`
  - `<id>` - Unique ID of object in this scene
  - `<world3d>`
    - `<type>` - 3D type: {`standingplanes` | `part` | `groundplane`}

<!-- Object 3D information goes here (see below) -->
</world3d>
</object>
<imagesize>
<nrows> - Number of rows in image
<ncols> - Number of columns in image
</imagesize>
<camera>
<units> - World units type (e.g. 'meters')
<pmatrix>
<p11><p12><p13><p14> - First row of camera matrix
<p21><p22><p23><p24> - Second row of camera matrix
<p31><p32><p33><p34> - Third row of camera matrix
</pmatrix>
</camera>
</annotation>

The following subsections describe the XML tags for the different 3D object types.

5.1 Standing planes

The following is 3D information for objects that are represented by standing planes (see Section 4.1 for more information on the standing planes representation):

<world3d>
<type>standingplanes</type>
<stale> - 1 indicates that the 3D information is not up-to-date
<polygon3d>
<pt>
<x> - 3D X coordinate
<y> - 3D Y coordinate
<z> - 3D Z coordinate
<added> - 1 indicates that LabelMe3D added this point to the polygon
<planeindex>
<index> - Index of which plane the point belongs to (zero-based)

```
      </planeindex>
    </pt>
  </polygon3d>
  <contact>
    <x> - X coordinate of contact point in image
    <y> - Y coordinate of contact point in image
  </contact>
  <plane>
    <pix><piy><piz><piw> - 3D plane parameters (see Section 4.1)
  </plane>
</world3d>
```

Note that each 3D point in `<polygon3d>` corresponds to a 2D point in `<polygon>` (i.e. the number of 2D and 3D polygon points are equal, with the ith point in each polygon being in correspondence). Furthermore, LabelMe3D automatically inserts additional 2D/3D points where a polygon edge crosses the 3D line that intersects two adjacent planes (these added points are indicated by the `<added>` tag). Finally, planes are stored as they appear in the 3D scene from left to right.

5.2 Ground plane

The following is 3D information for objects that are represented as a ground plane:

```
<world3d>
  <type>groundplane</type>
  <stale> - 1 indicates that the 3D information is not up-to-date
  <polygon3d>
    <pt>
      <x> - 3D X coordinate
      <y> - 3D Y coordinate
      <z> - 3D Z coordinate
    </pt>
  </polygon3d>
  <plane>
```

<pix><piy><piz><piw> - 3D plane parameters (see Section 4.1)
</plane>
</world3d>

5.3 Parts

The following is 3D information for objects that are represented as a part:

<world3d>
<type>part</type>
<stale> - 1 indicates that the 3D information is not up-to-date
<parentid> - ID of part's parent
<rootid> - ID of part's root parent
<polygon3d>
<pt>
<x> - 3D X coordinate
<y> - 3D Y coordinate
<z> - 3D Z coordinate
<added> - 1 indicates that LabelMe3D added this point to the polygon
<planeindex>
<index> - Index of which root parent's plane the point belongs to (zero-based)
</planeindex>
</pt>
</polygon3d>
</world3d>