READ ME - Dataset

Capturing Hands in Action using Discriminative Salient Points and Physics Simulation

D. Tzionas, L.Ballan, A. Srikantha, P. Aponte, A. Taneja, M. Pollefeys and J. Gall

http://files.is.tue.mpg.de/dtzionas/hand-object-capture.html

In case you use this dataset, please cite the above paper.
Bibtex file will be uploaded in the future, the project is under review.

For questions please refer to: tzionas@iai.uni-bonn.de

INDEX_BOUNDS.txt

Contains information about the total number of aligned frames, as well as information about the alignment between the motion and the video frames. This information is the same for all the sequences and its practical meaning is that the first video frame corresponds to the second motion frame. The reason for this is that the first motion frame defines the rigging pose.

The file structure:

```
TotalAlligned   <int numbFrames>
MotionnOffset   1                       always same value
VideoooOffset   0                       always same value
```

MODELS_INFO.txt

Contains information about the models contained in the sequence.

The file structure:

```
<int numberOfModels>
<text modelName1>
<text modelName2>     (if applicable)
etc..                 (if applicable)
```

MODELS_BULLET_INFO.txt (if applicable)

Contains information about the shape primitives used in Bullet in order to model the static scene of each sequence, namely the resting bases of the objects. For the present sequences only cylindrical primitives are used.

The file structure:

```
<int numberOfModels>
<text modelName1> <double height1> <double radius1>                (if applicable)
                  <double T1(0)> <double T1(1)> <double T1(2)>
                  <double R1(0)> <double R1(1)> <double R1(2)> <double R1(3)>
etc..
```

where, T and R define the translation and rotation (quaternion) of each model.

.mp4 files

Use just for quick visualization, in order to give an insight about the sequence.

Folder "depth"

Contains the depth frame in yml format, that can be easily read with OpenCV. Invalid depth is stored with value 0.

Folder "depth_viz"

Contains the depth frame in png format. However this is not the raw depth information, but has been processed to enable easy visualization. Thus, it should be used only for visualization and not for actual processing.

Folder "detections"

Contains the detections found by our fingertip detector.

The file structure:

```
  <int totalDetections> <int ignoreThis> <int ignoreThis> <int ignoreThis>

  <double confidence>   (this group defines 1 detection and takes up 1 line)
  <int topLeft_X>
  <int topLeft_Y>
  <int height>
  <int width>
  <double ignoreThis>
  <int    ignoreThis>

  etc.. (rest of detections)
```

where:
confidence – the confidence of the detection
topLeft_X // the top-left corner coordinates
topLeft_Y // of the bounding box of the detection
height   - the height of the bounding box of the detection
width   - the width of the bounding box of the detection

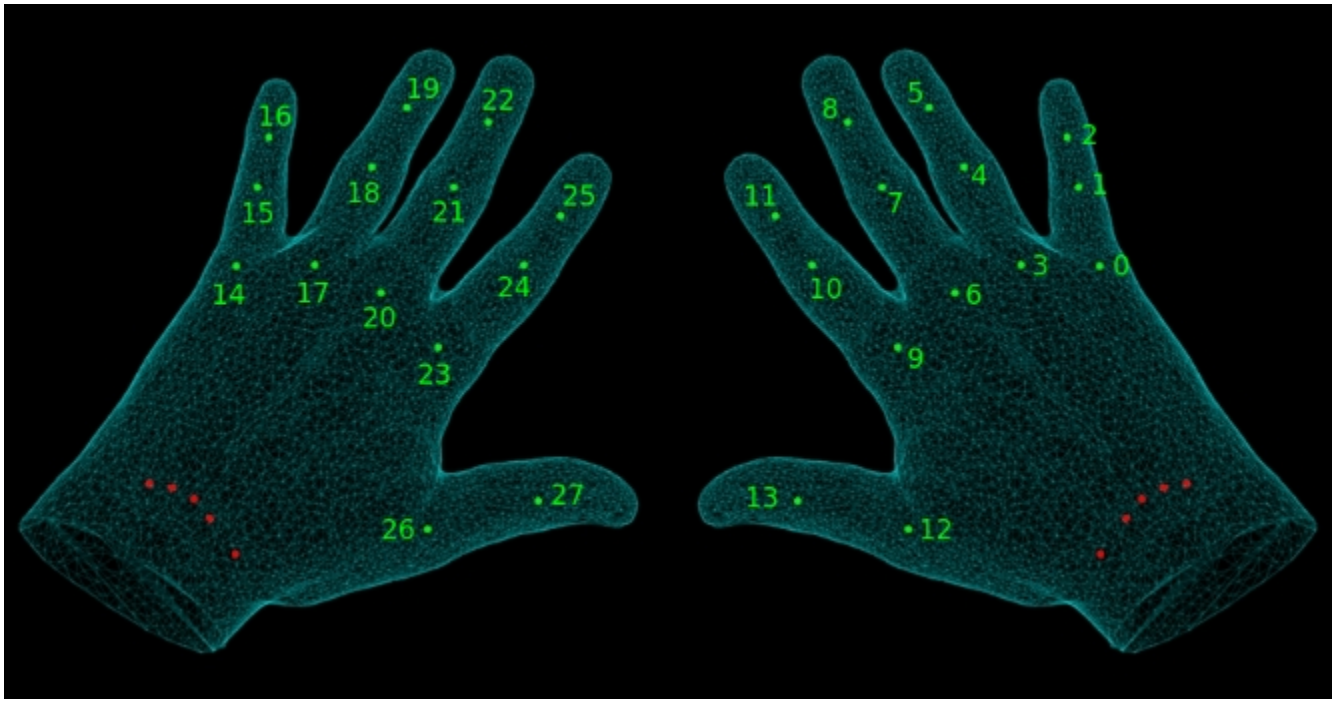
Folder "joints_2D_GT"

Contains the ground-truth joints. The annotation takes place every 5th frame.
For each frame there are either 14 or 28 joints, depending on the number of hands (1 and 2 hands respectively).
The joint IDs follow a specific order, depicted in the following picture.
Please note that numbering starts from the right hand, with a direction from the little finger to the thumb and from the inside to the outside.

In case of an occluded joint, the stored coordinates (x,y) have zero value (0,0) and should not be taken into account in the error metric.

The file structure:

```
  <int jointID>  <int x>  <int y>
  etc...
```



Folder “oni” (if applicable)

Contains the .oni (OpenNI) file that packs all the frames of a “Set B” sequence (GCPR_2014_Dataset_Sequences_INFO.pdf) in a single file.

Folder “pcl” (if applicable)

Contains the incoming point cloud frames in the PCL format. A description of the format can be found at the following link. http://pointclouds.org/documentation/tutorials/pcd_file_format.php#pcd-file-format

Folder “rgb”

Contains the incoming RGB frames.

Folder “rgbd”

Contains the incoming RGBD frames. These are created using the aligned RGB and depth images. Pixels with invalid depth information are assigned black color.

Folder “models”

Contains the model files that are used in each sequence. A description of the files that are found in this folder follows in the next:

Camera file format

```
-----------------------
Filename: Cameras.txt
-----------------------
```

The fixed camera export output (from the 3DSMax PlugIn) is:

Let R(i) and T(i) be respectively the rotation matrix 3x3 and the translation matrix 3x1 of camera i and K(i) is the intrinsic matrix of camera i. Let v be the number of cameras.

The file structure:

```
 <v = number of cameras>
 <K(0)(0,0)> <K(0)(0,1)> <K(0)(0,2)>
 <K(0)(1,0)> <K(0)(1,1)> <K(0)(1,2)>
 <K(0)(2,0)> <K(0)(2,1)> <K(0)(2,2)>
 <R(0)(0,0)> <R(0)(0,1)> <R(0)(0,2)> <T(0)(0,0)>
 <R(0)(1,0)> <R(0)(1,1)> <R(0)(1,2)> <T(0)(1,0)>
 <R(0)(2,0)> <R(0)(2,1)> <R(0)(2,2)> <T(0)(2,0)>
   .           .           .           .
   .           .           .           .
   .           .           .           .
   .           .           .           .
 <K(v-1)(0,0)> <K(v-1)(0,1)> <K(v-1)(0,2)>
 <K(v-1)(1,0)> <K(v-1)(1,1)> <K(v-1)(1,2)>
 <K(v-1)(2,0)> <K(v-1)(2,1)> <K(v-1)(2,2)>
 <R(v-1)(0,0)> <R(v-1)(0,1)> <R(v-1)(0,2)> <T(v-1)(0,0)>
 <R(v-1)(1,0)> <R(v-1)(1,1)> <R(v-1)(1,2)> <T(v-1)(1,0)>
 <R(v-1)(2,0)> <R(v-1)(2,1)> <R(v-1)(2,2)> <T(v-1)(2,0)>
```

Motion file format

```
-------------------------
File Extension: .MOTION
-------------------------
```

Given a kinematic tree composed of a set of s bones. Each bone has a name n[i] and a transformation matrix M[i,t] defined at each time t (starting from 0 to z-1, where z is the number of frames). M[i,0] is part of the rigging pose.

M[i,t] is a 3x4 matrix which can be decomposed in

$$M[i,t] = (3x3, 3x1) = (R[i,t] , T[i,t])$$

where T[i,t] is the origin of the bone.

The file structure:

```
  <text        n[0]>                                  (occupies 1 line)
  <matrix(3,3) R[0,0]>   <matrix(3,1) T[0,0]>         (occupies 1 line)
  <matrix(3,3) R[0,1]>   <matrix(3,1) T[0,1]>
  <matrix(3,3) R[0,2]>   <matrix(3,1) T[0,2]>
    .          .             .
    .          .             .
    .          .             .
  <matrix(3,3) R[0,z-1]> <matrix(3,1) T[0,z-1]>
```

```
  <text        n[1]>
  <matrix(3,3) R[1,0]>   <matrix(3,1) T[1,0]>
  <matrix(3,3) R[1,1]>   <matrix(3,1) T[1,1]>
  <matrix(3,3) R[1,2]>   <matrix(3,1) T[1,2]>
    .          .           .
    .          .           .
    .          .           .
  <matrix(3,3) R[1,z-1]> <matrix(3,1) T[1,z-1]>


    .
    .
    .


  <text        n[s-1]>
  <matrix(3,3) R[s-1,0]>   <matrix(3,1) T[s-1,0]>
  <matrix(3,3) R[s-1,1]>   <matrix(3,1) T[s-1,1]>
  <matrix(3,3) R[s-1,2]>   <matrix(3,1) T[s-1,2]>
    .          .           .
    .          .           .
    .          .           .
  <matrix(3,3) R[s-1,z-1]> <matrix(3,1) T[s-1,z-1]>
```

Object File Format

File Extension: **.OFF**

Object File Format (.off) files are used to represent the geometry of a model by specifying the polygons of the model's surface. The polygons can have any number of vertices.
The .off files in the Princeton Shape Benchmark conform to the following standard. OFF files are all ASCII files beginning with the keyword OFF. The next line states the number of vertices, the number of faces, and the number of edges. The number of edges can be safely ignored.
The vertices are listed with x, y, z coordinates, written one per line. After the list of vertices, the faces are listed, with one face per line. For each face, the number of vertices is specified, followed by indices into the list of vertices. See the examples below.
Note that earlier versions of the model files had faces with -1 indices into the vertex list. That was due to an error in the conversion program and should be corrected now.

```
OFF numVertices numFaces numEdges
x y z
x y z
... numVertices like above
NVertices v1 v2 v3 ... vN
MVertices v1 v2 v3 ... vM
... numFaces like above
```

Note that vertices are numbered starting at 0 (not starting at 1), and that numEdges will always be zero.

A simple example for a cube:

```
OFF
```

```
8 6 0
-0.500000 -0.500000 0.500000
0.500000 -0.500000 0.500000
-0.500000 0.500000 0.500000
0.500000 0.500000 0.500000
-0.500000 0.500000 -0.500000
0.500000 0.500000 -0.500000
-0.500000 -0.500000 -0.500000
0.500000 -0.500000 -0.500000
4 0 1 3 2
4 2 3 5 4
4 4 5 7 6
4 6 7 1 0
4 1 7 5 3
4 6 0 2 4
```

Skeleton file format

```
----------------------
File Extension: .SKEL
----------------------

The kinematic tree structure is stored as a text file as follows:
 <z = number of frames>

 <bone 0 father name>
 <n[0] = bone 0 name>
 <bone 0 length>

 <bone 1 father name>
 <n[1] = bone 1 name>
 <bone 1 length>

   .
   .
   .

 <bone s-1 father name>
 <n[s-1] = bone s-1 name>
 <bone s-1 length>
```

Skin file format

```
----------------------
File Extension: .SKIN
----------------------

The skinning parameters are stored as a pxs' matrix matrix, where s'<=s is the
number of bones of the kinematic tree actually used by the skin operator and p is
the number of vertices of the mesh.
Each column of the matrix is associated to a bone which name is n'[i] (which in
general it differs from the vector n[i]).


The file structure:
```

```
 <int s'>
 <text    n'[0]>   <text    n'[1]>    ...  <text    n'[s'-1]>
 <double A(0,0)>   <double A(0,1)>    ...  <double A(0,s'-1)>
 <double A(1,0)>   <double A(1,1)>    ...  <double A(1,s'-1)>
 <double A(2,0)>   <double A(2,1)>    ...  <double A(2,s'-1)>
   .                 .                       .
   .                 .                       .
   .                 .                       .
 <double A(p-1,0)> <double A(p-1,1)>  ...  <double A(p-1,s'-1)>
```

Vertices-Of-Interest file format

```
---------------------
File Extension: .VOI
---------------------
```

This file contains the vertex IDs that define one extreme point per finger. More vertices can be found for each fingertip through proximity search.

The file structure:

```
 <int vertexID_littleFinger>
 <int vertexID_ringFinger>
 <int vertexID_middleFinger>
 <int vertexID_pointer>
 <int vertexID_thumb>
```

Limits file format

```
------------------------
File Extension: .LIMITS
------------------------
```

The file structure:

```
 <text Joint_name>
 <int t> <int rx> <int ry> <int rz>
<double rot_axis_x(0)> <double rot_axis_x(1)> <double rot_axis_x(2)>
<double rot_axis_y(0)> <double rot_axis_y(1)> <double rot_axis_y(2)>
<double rot_axis_z(0)> <double rot_axis_z(1)> <double rot_axis_z(2)>
 <int min_rx> <int max_rx>
 <int min_ry> <int max_ry>
 <int min_rz> <int max_rz>
```

where
t = 0 or 1 indicates if the joint can translate
rx = 0 or 1 indicates if the joint can rotate along the x-axis
ry = 0 or 1 indicates if the joint can rotate along the y-axis
rz = 0 or 1 indicates if the joint can rotate along the z-axis
rot_axis_x(0) // A 3d unit vector that forms the rotation axis around
rot_axis_x(1) // which the bone “Joint_name” rotates.
rot_axis_x(2) // Each rotation axis forms one DoF.
etc...
min_rx = min rotation angle (around rot_axis_x)
max_rx = max rotation angle (around rot_axis_x)
etc...

The name of the joint <text Joint_name> is given by the bone that starts at this 3d location.

Angles are expressed in degrees w.r.t. the pose at frame 0 (rigging pose). They constraint the twist of each bone (i.e., the 3D vector in so(3) which can be transformed to a rotation matrix using the exponential map).

Some of the material in this document are borrowed from Luca Ballan's work:
http://cvg.ethz.ch/research/ih-mocap/
http://cvg.ethz.ch/research/ih-mocap/FileFormat.txt
The chapter about the Object File Format is borrowed from:
http://segeval.cs.princeton.edu/public/off_format.html
