Technical tips: Step 3

1. Normalization Tips

Recall, normalization includes making the models similar in terms of various properties: position, size, alignment, orientation. Some useful tips about all these steps below:

(1) Position

This should really be very simple: If you want to make all shapes centered in the origin (that is, their barycenter **c** should be located at the origin of the coordinate frame), then the *update formula* for the coordinates $\mathbf{p}_i$ is

$$\mathbf{p}_i^{\text{updated}} = \mathbf{p}_i - \mathbf{c}$$

Proof: Simple. Put **c** instead of $\mathbf{p}_i$ in the above formula, and you'll see that $\mathbf{p}_i^{\text{updated}} = 0$.

Important note: All above quantities are *vectors*. So, the update is to be done coordinate-wise (for x, y, z separately).

(2) Alignment

As discussed in the practical session, a very simple way to align a shape's eigenvectors with the xyz coordinate frame is to use *projections*. Let again $\mathbf{p}_i$ be the unaligned shape coordinates. Let $\mathbf{e}_j$ be the eigenvectors computed from the shape (here, $j \in \{1,2,3\}$). Let $\mathbf{e}_1$, $\mathbf{e}_2$ be the major, respectively medium, eigenvectors. Assume next these are normalized (of length 1). Then the *update formula* for computing the aligned shape is

$$\mathbf{x}_i^{\text{updated}} = (\mathbf{p}_i - \mathbf{c}) \cdot \mathbf{e}_1$$
$$\mathbf{y}_i^{\text{updated}} = (\mathbf{p}_i - \mathbf{c}) \cdot \mathbf{e}_2$$
$$\mathbf{z}_i^{\text{updated}} = (\mathbf{p}_i - \mathbf{c}) \cdot (\mathbf{e}_1 \times \mathbf{e}_2)$$

Here, $\cdot$ indicates dot product and $\times$ indicates cross product. So, why don't we use $\mathbf{e}_3$ in the update formula for **z** above? Because, recall, PCA gives us only *directions*, not *orientations.* So, we cannot be sure that PCA outputs three vectors $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$ that form a right-handed coordinate system like the xyz one is. So, we 'force' this by using $\mathbf{e}_1 \times \mathbf{e}_2$ instead of $\mathbf{e}_3$.

Subtle point: So, since PCA doesn't give us orientations, but just directions, what about $\mathbf{e}_1$ and $\mathbf{e}_2$? That is, we 'fixed' $\mathbf{e}_3$ by replacing it with $\mathbf{e}_1 \times \mathbf{e}_2$. But what about $\mathbf{e}_1$ and $\mathbf{e}_2$? These can still be 'flipped', right? Yes, this is the case. But the flipping test (orientation normalization) discussed below will fix this! So, at this point, we don't care about the orientation of $\mathbf{e}_1$ and $\mathbf{e}_2$.

(3) Size

This is really very simple (see course slides). Compute the sizes D_x, D_y, D_z of the *oriented bounding box* that encloses the shape. Or, easier, compute the *axis-aligned bounding box* sizes D_x, D_y, D_z after alignment. Obviously, the above two operations yield the same result. Then, compute the largest dimension of the bounding box:

$D_{max} = \max(D_x, D_y, D_z)$

Then, compute the scaling factor that would *shrink* this size to one:

$\sigma = 1/D_{max}$

Then, the *update formula* is very simple:

$\mathbf{p}_i^{updated} = \sigma\, \mathbf{p}_i$

(4) Orientation ('flipping test')

The idea is very simple here too: For each axis (xyz) of the already axis-aligned shape, we compute in which part of the axis (positive half, negative half) most of the 'mass' (area, volume) of the shape resides. Then, we choose a conventional orientation – we want to make the most mass be in the positive half-axis. So, if this is already the case (indicated by the flipping test), then we don't do anything. If the mass is mostly in the negative half-axis, we mirror the shape along that axis. Note that mirroring is identical to scaling (see above) with a scaling factor equal to -1.
In detail: The flipping test computes a value f_i along each axis (that is, i=0 means x, i=1 means y, i=2 means z) as:

$$f_i = \sum_t sign(C_{t,i})(C_{t,i})^2$$

where the summation goes over all triangles *t* in the mesh, and $C_{t,i}$ is the i^{th} coordinate of the center of triangle *t*.
Then, the *update formula* is very simple: mirror the mesh using the following scaling factors:

$\mathbf{x}_i^{updated} = \mathbf{x}_i\, sign(f_0)$
$\mathbf{y}_i^{updated} = \mathbf{y}_i\, sign(f_1)$
$\mathbf{z}_i^{updated} = \mathbf{z}_i\, sign(f_2)$

(5) Order of normalization steps

Refining the ideas from the last technical tips document: What is a *good* order of all operations?

- Remeshing

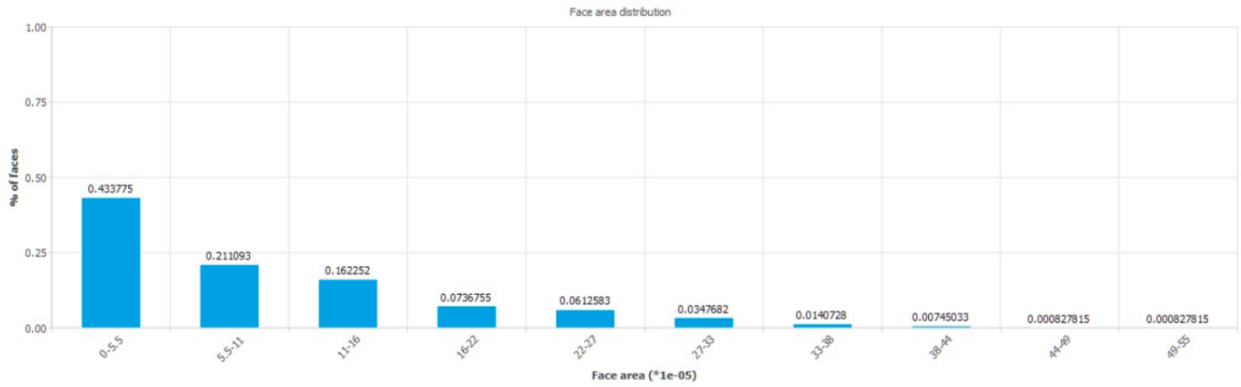
- Translation
- Pose (alignment)
- Flipping (mirroring)
- Size

Why this? Look at the actual formulas and think what would happen if you did these steps in other orders.

(6) Other normalization tips

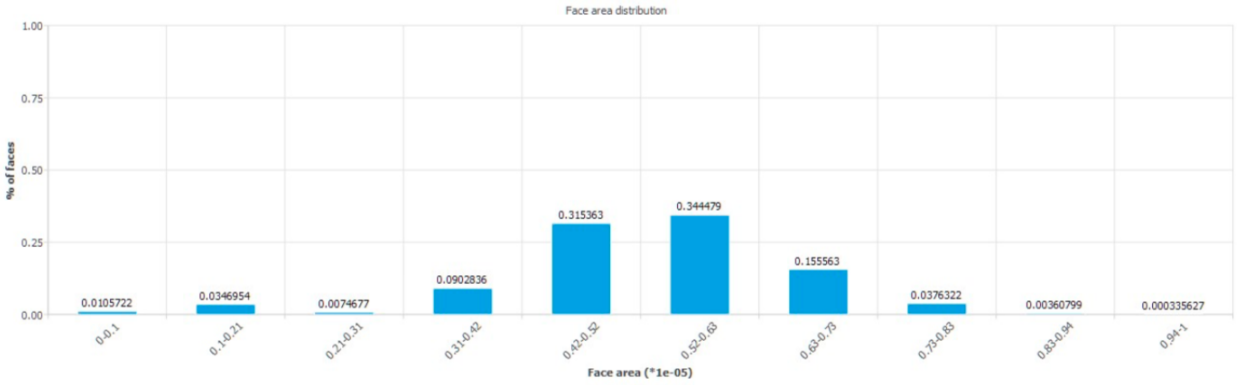
Should you use an oriented bounding box (OBB) or axis aligned bounding box (AABB) to do the size normalization? If you do this normalization *after* the pose alignment (as suggested above), the answer is that it does not matter, since after alignment, the OBB and AABB of a shape are identical.

What is a good example of using histograms to check normalization?
See the images below (taken from an actual student report). The first image shows the histogram of face areas *before resolution normalization* (i.e., before remeshing). We see that most faces are small, but there exist also larger faces. The 'bandwidth' of the distribution, i.e. difference between the minimal and maximal face areas, is quite large: from 1 to 50, roughly.



The next image shows the histogram of face areas *after resolution normalization* (i.e., after remeshing). We see two desirable things:

- The bandwidth of the distribution has considerably shrunk – from 1..50 to roughly 0..1. Hence, face sizes are far more uniform after this normalization
- The distribution is centered around an average face-area value of 0.5. *If this is our desired target face-area value, then this is a good distribution.*


Face area distribution
% of faces
1.00
0.75
0.50
0.25
0.00
0.0105722
0.0346954
0.0074677
0.0902836
0.315363
0.344479
0.155563
0.0376322
0.00360799
0.000335627
0-0.1
0.1-0.21
0.21-0.31
0.31-0.42
0.42-0.52
0.52-0.63
0.63-0.73
0.73-0.83
0.83-0.94
0.94-1
Face area (*1e-05)