Homework III

Instructor : Gabriel Taubin*

1 Matching and merging of range images

Part I. The goal of this part is to mark corresponding points among different images and compute the 3D coordinates of such points. The images are pictures of the same object but from different points of view. So portions visible in picture 1 may no be visible in other pictures and viceversa.

The matlab file *markpoints.m* allows you to mark corresponding points among 3 pictures (*pieta356-1.jpg*, *pieta356-2.jpg* and *pieta356-3.jpg*) of the Michelangelo's Pieta'. Run the function and follow instructions to select mark points. Once you have selected a certain number of mark points (e.g. you may want to use as mark points the red dots within the statue's surface), *markpoints.m* will dump the mark point pixel coordinates into 3 files: *markpoints1.txt*, *markpoints2.txt* and *markpoints3.txt* attached to images 1,2 and 3 respectively. If a mark point has not been selected within a certain image, the corresponding pixel coordinates will be saved as $(-1, -1)$.

For each image, we provide the corresponding .wrl file (*pieta356-1.wrl*,*pieta356-2.wrl* and *pieta356-3.wrl*). Each .wrl file contains 3 parts: a list of the vertex indices of the triangles (called *coordIndex*) which accounts for the mesh connectivity; a list of the 3D coordinates of each vertex (called *coord*); a list of the pixel coordinates of each vertex (called *texCoord*). For any mark point, find the triangle that a given mark point p_m belongs to: if p_1, p_2 and p_3 are the vertices (in pixel coordinates) of such a triangle, find the affine coordinates (λ_1, λ_2 and λ_3) of p_m with respect to p_1, p_2 and p_3 (since the mark point belongs to the triangle defined by p_1, p_2 and p_3, the affine coordinates must be positive). If P_1, P_2 and P_3 are the corresponding 3D coordinates of p_1, p_2 and p_3, the 3D location P_m of the mark point can be expressed as a linear combination of P_1, P_2 and P_3 with the same affine coefficients λ_1, λ_2 and λ_3. Repeat this procedure for each mark point. As a result, you will get a list of 3D mark point locations for each figure.

Part II. The goal of this part is to use the mark points within each figure to compute the relative position of a given mesh with respect to the others. Assume that the absolute coordinate reference system is that attached to the mesh2 (i.e. the mesh corresponding to image2) . You are asked to estimate the transformations T_{12} and T_{32} which allow you to transform any point in the reference system attached to mesh1 and mesh3 (respectively) into a point in the reference system attached to mesh2. The transformation T_{ij} is a rotation combined with a translation. You can compute T_{12} (e.g.) by following the least square procedure given in class (see class notes). Call $p_1, p_2, \ldots p_N$ the 3D mark point coordinates of mesh1 and call $q_1, q_2, \ldots q_N$ the corresponding 3D mark point coordinates of mesh2. Call R and t the rotation matrix and translation vector of T_{12}. You can estimate R and t by minimizing the following quantity:

$$\sum_{i=1}^{N} |Rp_i + t - q_i|^2 \tag{1}$$

You can minimize (1) by performing the following steps.

1. Compute:

$$\bar{p} = \frac{1}{n}\sum p_i \qquad \bar{q} = \frac{1}{n}\sum q_i \tag{2}$$

$$P = [p_1 - \bar{p}\,, \ldots\ p_N - \bar{p}\,]; \quad Q = [q_1 - \bar{q}\,, \ldots\ q_N - \bar{q}\,] \tag{3}$$

$$M = PQ^T \tag{4}$$

2. Perform *svd* on M, getting:

$$M = U\Lambda V^T \tag{5}$$

3. Finally, compute R and t:

$$R = VU^T \qquad t = R\bar{p} - \bar{q} \tag{6}$$

Part III. The goal of this part is to transform all vertices of mesh1 and all vertices of mesh2 into vertices within the reference system attached to mesh2. In order to do that, we provide the program *ee193s08ifs.exe*. You can ran the following command:

```
ee193s08ifs -transform param inputfile.wrl outputfile.wrl
```

where *param* is the following list of parameters: r_x, r_y, r_z, t_x, t_y, t_z (note: they must be separated by a space). The parameters r_x, r_y, r_z are the components of the vector obtained by applying the rodriguez formula to R_{ij}. (Note: the modulus of such vector is $\sin(\alpha)$ instead of just α). The parameters t_x, t_y, t_z are the components of the translation vector t_{ij}.

As a result, you will get a file1.wrl and a file3.wrl, describing mesh1 and mesh3 in the mesh2 reference system. You may want to check the results in your java applet. You need now to assign a uniform color to the 3 meshes. You can do that by using the following command:

```
ee193s08ifs -color r g b inputfile.wrl outputfile.wrl
```

where r,g,b are 3 floats $\in [0, 1]$ encoding the color you may want to assign to the mesh. Finally you need to merge the 3 meshes. You can do that by running:

```
ee193s08ifs -merge file1.wrl file3.wrl outfile.wrl
```

The output file contains the 3 meshes depicted with different colors. Print the results by using the java applet.

Part IV. We provide 3 files *pieta356-1-032.rng*, *pieta356-2-032.rng* and *pieta356-3-032.rng* – other files at greater resolution are available. Such files contain the range data corresponding to image1, image2 and image3 respectively. In other words, for each pixel x_{ij} the distance Z_{ij} to the corresponding 3D point is available (that is, the distance from the image plane to Z_{ij}) . The format is the following:

- image width W (number of pixels along the row)
- image height H (number of pixels along the column)
- width size dx (in meters)
- height size dy (in meters)
- list of Z_{ij}

Thus, the (x,y) coordinates of the pixel x_{ij} (i varies faster that j) are:

*taubin@brown.edu
†

$y = (dy/H)(j + 0.5) - 0.5dy$
$x = (dx/W)(i + 0.5) - 0.5dx$

The Z_{ij} coordinate is the value of the corresponding pixel x_{ij}. $Z_{ij} = -1$ means no data available for that pixel.

Consider a bounding cube surrounding the whole mesh. Divide the bounding box in voxels (little cube) of a certain size. For each voxel, compute the centroid c and compute the distances D_1, D_2 and D_3 of c from image plane1, image plane2 and image plane3 respectively. Compute the pixels x_1, x_2 and x_3 attached to c in image plane1, image plane2 and image plane3 (assume orthographic projection); From the range data provided in *pieta356-1-032.rng*, *pieta356-2-032.rng* and *pieta356-3-032.rng* consider the range value Z_1, Z_2 and Z_3 associated to x_1, x_2 and x_3 respectively. Compare Z_1, Z_2 and Z_3 with D_1, D_2 and D_3. Label the voxel according to following algorithm:

for $i = 1 : 3$,
  $d_i = 1$ if $Z_i \leq D_i$ (inside)
  $d_i = -1$ if $Z_i > D_i$ (outside)
  $w_i = 1$ if the c is does correspond to a data point in range image i.
  $w_i = 0$ otherwise
end
voxel label $= \frac{\sum_i^3 w_i d_i}{\sum_i^3 w_i}$

Which value would you assign to voxel label if $\sum_i^3 w_i = 0$? Explain the reason. Repeat the whole thing for each voxel of your bounding box.

In the last step you need to generate isosurfaces. You don't need to implement the algorithm. Visit

http://mi.eng.cam.ac.uk/~gmt11/software/isosurf/isosurf.html

and download isosurf.exe. Read through instructions and examples in the site in order to figure out the input data file format. Isosurf.exe generate a .wrl file which you can read with your java applet. Print results and hand in all the codes you have generated.

For more information about the topics covered in this part you may want to refer at *A Volumetric Method for Building Complex Models from Range Images*, by Brian Curless and Marc Levoy – Siggraph '96 (see class web site).