A Computer Modeling of the China Central Television Headquarters in Beijing

Valentina Forcella

Politecnico di Milano, Milano D.I.C.A. 20131, Italy

Abstract: This paper deals with the computer modeling of structures starting from a point cloud. The CCTV (China Central Television) tower headquarters is the case for study because the shape of this building is non-stellar, concave and multi-connected. It is composed of sowns and chains. The sown is the representation of a horizontal plane formed by dense points. The chain is a planar path modeled by rare points. The CCTV structure is defined only by the three orthogonal Cartesian coordinates of the points. The proposed computer modeling uses a sequence of procedures and the desired outputted 3D model is consistent. The first procedure is devoted to attributing points to their voxel and to estimating three values needed afterwards. The second procedure is devoted to analyzing clusters vertically and horizontally, to preliminarily distinguishing chains from sowns and to generating relational matching. The third procedure is devoted to building closed paths between all chains and all their projections on sowns. The fourth procedure is devoted to connecting points with triangles. The fifth procedure, still being implemented, is devoted to interpolating triangles with triangular splines. The results show it is possible to achieve the 3D model using the above mentioned procedures. These procedures are written, implemented and tested and they form a library of people’s own software. The code is written using Matlab. It is not possible to obtain the required 3D model if the procedures are applied in the wrong order or one step is skipped. To conclude, it is possible to obtain the computer model of the CCTV using the provided sequence of procedures.

Key words: CCTV tower, cluster analysis, Delaunay triangulation, Bézier spline.

1. Introduction

This paper deals with the computer modeling of a building, starting from a point cloud. The test example is the China Central Television Tower Headquarters, located in Beijing.

The construction of the building began in September 2004 on the 20 hectare site of an abandoned motorcycle factory in Beijing’s new Central Business District and was completed by the OMA (Office for Metropolitan Architecture) in August 2008 [1].

There are two modeling problems. The first one relates to the fact that the structure is modeled by points, acquired from pictures available on the web and in architectural literature, and defined only by their three orthogonal Cartesian coordinates, without any other distinction, as shown in Table 1, extracted by the input data set.

The second one relates to the fact that the shape of this building is:

1. non-stellar: starting from one point, to reach all the other points, it may be necessary to go outside the structure;
2. concave;
3. multi-connected: this means that there is a hole.

The structure is composed of sowns and vertical or almost vertical walls (treated as chains).

To solve these problems, the solution strategy is composed of procedures. The first procedure checks the input data set [2], identifying the minimums and the maximums of X, Y and Z. The “voxel” procedure, a method based on octree encoding, is used also to estimate three values needed afterwards. The second procedure divides the point cloud into clusters, first vertically and then horizontally [3]. This second procedure also classifies the data into sets of chains
and sowns and relates points belonging to different clusters. The third procedure generates closed paths between points classified as belonging to the same horizontal cluster. These closed paths are built differently depending on the fact the cluster is classified as a chain or as a sown. The fourth procedure connects points with a Delaunay triangulation. The last procedure, still being implemented, will create a triangular interpolation using triangular splines.

The results show it is possible to obtain the 3D model using the above mentioned solution strategy. The strategy is composed of coded using Matlab [4]. It is not possible to obtain the required 3D model if the procedures are applied in the wrong order or one step is skipped.

To conclude, it is possible to obtain the computer model of a structure if its shape is non-stellar, concave and multi-connected. The case study was the CCTV and the provided sequence of procedures outputted the desire consistent 3D model.

The imposition of an arbitrary reference system is needed to remove the uncertainty of the origin, orientation and scales. The origin is placed on the left hand side, the axes as right-hand coordinate system, and 1 m in reality is 1 unit in the data set.

The results will be given in the same reference system in which the coordinates are provided as shown in Fig. 1.

The first 3D plot shows the structure represented by the group of points in Fig. 2.

![Fig. 1 Reference system adopted.](image)

<table>
<thead>
<tr>
<th>ID</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>253</td>
<td>0.000</td>
<td>120.000</td>
<td>-20.000</td>
</tr>
<tr>
<td>273</td>
<td>200.000</td>
<td>120.000</td>
<td>-20.000</td>
</tr>
<tr>
<td>274</td>
<td>0.000</td>
<td>130.000</td>
<td>-20.000</td>
</tr>
<tr>
<td>294</td>
<td>200.000</td>
<td>130.000</td>
<td>-20.000</td>
</tr>
<tr>
<td>295</td>
<td>0.000</td>
<td>140.000</td>
<td>-20.000</td>
</tr>
</tbody>
</table>

![Fig. 2 3D plot of point cloud.](image)
As mentioned, a sequence of procedures have to be adopted to obtain three dimensional model. The procedures used by the author are given below.

2. Voxel

A voxel is an element of the total volume, representing a value on a regular grid in 3 dimensional space. The voxel information is organized in matrix and sparse data structure must be handled [5]. The position of a voxel is inferred based upon its position relative to the other voxels.

The first procedure is devoted to checking the data set, to detecting the minimums and the maximums of X, Y and Z, to attributing points to their voxel and to estimating the three values needed afterwards.

The input file is CCTV.txt containing a list with the:
(1) point ID;
(2) X-coordinate;
(3) Y-coordinate;
(4) Z-coordinate.

The data set was checked to ensure that both the ID point and its coordinates are correct, consistent and without any duplication. If one of these two cases occurs, a list with the number ID, coordinate X, Y and Z is provided. If there are no double points, it is shown. The service file CCTV_RID.txt contains the checked data which will be used subsequently.

In this specific case, there are neither double points nor points out of height so CCTV.txt is equal to CCTV_RID.txt.

The “voxel” procedure is also used due to the estimation of the three values needed afterwards:
(1) The first parameter defines vertical scanning and divides horizontal surfaces (which model sowns) from chains;
(2) The second parameter defines horizontal clustering and generates finite point sets at the same level, if necessary;
(3) The third parameter is used to match vertically between points belonging to consecutive levels; this parameter is also used to distinguish sowns from chains.

In CCTV modeling, the produced parameters are:
(1) height step: 7.969 m;
(2) planimetric distance: 17.678 m;
(3) planimetric tolerance between points at different levels: 4.000 m.

It is also necessary to find minimums and maximums for each coordinate, in order to set the voxel parameters.

In this specific case, the values given are:
(1) X minimum: 0.000 m;
(2) X maximum: 20.000 m;
(3) Y minimum: 0.000 m;
(4) Y maximum: 20.000 m;
(5) Z minimum: -20.000 m;
(6) Z maximum: 235.000 m.

For each order of voxel, a list provides:
(1) the total number of points;
(2) the voxel ID;
(3) the number of points that belong to that voxel.

Another list contains only the full voxels, their full-voxel ID, their voxel ID, and the number of points that belong to that full voxel.

The “voxel” procedure continues up to the order in which the voxels are all empty.

In this specific case, the results are:
(1) voxel 1° has no point;
(2) voxel 2° has no point;
(3) voxel 3° has 1 point (in yellow in Fig. 3);
(4) voxel 4° has 167 points (in blue in Fig. 3);
(5) voxel 5° has 1077 points (in red in Fig. 3);
(6) voxel 6° has 1802 points (in green in Fig. 3).

At the end of this procedure, the optimal order voxel is found with its amplitude in X, Y and Z.

In this specific case, the parameters produced are:
(1) optimal voxel: 6°;
(2) X amplitude: 3.125 m;
(3) Y amplitude: 3.125 m;
(4) Z amplitude: 3.984 m.
3. Contours

The second procedure is devoted to checking the data set, to analyzing clusters vertically and horizontally, to preliminarily distinguishing chains from sowns, and to generating relational matching.

The input file could be CCTV.txt or CCTV_rid.txt. In this case, CCTV.txt is used because in the initial data set there are neither double points nor points out of height. If there were outliers, the used file would be CCTV_RID.txt and not the initial data set.

The input data set is listed with the point ID and X, Y and Z coordinates for each points.

The point ID can be the name of the point or the position in the list: both are used and listed.

The output files are:
(1) CCTV_ord.txt;
(2) CCTV_intermediate.txt;
(3) CCTV_assemby.txt.

The service file is CCTV_connections.txt, the data set can be processed in two different ways:
(1) by checking the input;
(2) by checking the data set and analyzing it in terms of clusters and relational matching.

The data set is checked in terms of planimetric distance. If points are too close together, only one of them is kept. After this checking procedure, the data set is analyzed.

Input coordinates could be supplied in all possible combinations (X, Y and Z; X, Z and Y; …), the order chosen is 123 (X, Y and Z).

Input data set can also be scaled and rotated, the adopted scales are:
(1) +1.000 m in X axis;
(2) +1.000 m in Y axis;
(3) +1.000 m in Z axis.

and the adopted rotations are:
(1) 0.000 m in X axis;
(2) 0.000 m in Y axis;
(3) 0.000 m in Z axis.

The estimation of the three values needed (the height step between points at consecutive levels, the planimetric distance between points at the same level and the planimetric tolerance between points at consecutive levels) is done using “voxel” procedures.

The parameters adopted in “contours” procedure are:
(1) height step: 8.000 m;
(2) planimetric distance: 15.000 m;
(3) planimetric tolerance between points at different levels: 4.000 m.
The height step parameter, estimated using the “voxel” procedure, is necessary to do vertical clustering.

For each level, the information is listed in this way:

1. vertical cluster ID;
2. number of points belonging to that vertical cluster;
3. height;
4. point IDs at that level.

A typical extract of the vertical clusters is shown in Table 2.

In this case, fifty two 2-dimensional plots were obtained. The 3-dimensional vertical clusters is shown in Fig. 4. Two 2-dimensional plots of the fifty-two plots are shown in Fig. 5.

The planimetric distance parameter, estimated using the “voxel” procedure, is necessary to do horizontal clustering. In order to generate contours, finite point sets at each level must be constructed.

These sets could involve all points belonging to a vertical cluster, or there could be different horizontal clusters at the same level, if necessary. This second case is displayed in Fig. 6.

As shown in Table 3, each horizontal cluster is described by:

1. its horizontal cluster ID;
2. the number of points that belong to the clustered level;
3. its level;
4. the point ID that belongs to the cluster.

There are two horizontal clusters at level 55.000 m: this is the case in which two different sets are done at the same level.

There are 83 horizontal clusters.

The preliminary distinction between sows and chains is made considering firstly the number of points and secondly, and in the case of ambiguity, the local density.

Table 2 Vertical clusters.

<table>
<thead>
<tr>
<th>Vertical ID</th>
<th># points</th>
<th>Height (m)</th>
<th>Point IDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>80</td>
<td>-20.000</td>
<td>1 441</td>
</tr>
<tr>
<td>16</td>
<td>38</td>
<td>55.000</td>
<td>2793 2830</td>
</tr>
<tr>
<td>52</td>
<td>153</td>
<td>235.000</td>
<td>4338 4496</td>
</tr>
</tbody>
</table>

Fig. 4 Vertical clusters.
A Computer Modeling of the China Central Television Headquarters in Beijing

Fig. 5  Two of the 52 vertical clusters.

Fig. 6  Two horizontal clusters are needed.
Generally, points that belong to sowns are denser than the ones that belong to chains. If the number of points is the same, it is necessary to check the local density.

The 3D plot is shown in Fig. 7. Although it is a preliminary distinction, all clustered levels are correctly classified, except height 45.000 m which is classified as a chain instead of a sown. The information is listed in this way:

1. horizontal cluster ID;
2. type of horizontal cluster;
3. vertical cluster ID.

At this point, the data set is analyzed in terms of relational matching and the planimetric tolerance parameter, estimated with the "voxel" procedure, is used.

The top picture in Fig. 8 shows an incorrect model due to the incorrect matching, while the figure at the
bottom shows a consistent relational matching. The connection must satisfy the following properties:

1. The points in question must be as close as possible while still belonging to consecutive levels;
2. If points belong to the same vertical, they are easily matched;
3. If not belonging to the same vertical, the strategy is to create connections along the same direction which resulted in the right picture in Fig. 8 above.

The relational matching is done separately for chains and for sowns. The former is classified by:

1. Relational matching ID (between one chain and another);
2. Horizontal cluster ID matched to the other horizontal cluster ID;
3. Level matching the other level;
4. The number of points belonging to that matching;
5. The point ID matched to the other or others point(s) ID.

Table 4 shows the output provided. The latter is classified by:

1. Relational matching ID (between the sown and a chain);
2. Horizontal cluster ID matched to the other horizontal cluster ID;
3. Level matching the other level;
4. The number of points belonging to that matching;
5. The point ID matched to the other or others point(s) ID;
6. The number of sides matched.

In this specific case, there is a perfect correspondence between each point so the matching type is a one to one and the ID point is related to a single point ID.

The total number of sides is also available. In the modeling of CCTV, the matched chain sides are 148, the matched sown sides are 18 and the total number is 166. The specific results are omitted for brevity.

4. Closed Paths

The third procedure is devoted to building closed paths between all chains and all their projections on sowns.

The input files needed are the ones generated in “contour” procedures and are:

1. CCTV_ord.txt;
2. CCTV_connections.txt;
3. CCTV_assembly.txt.

The output files are:

1. CCTV_closedpath.txt;
2. CCTV_connections_cla.txt;
3. CCTV_contours.txt.

The four values needed are the three generated with the “voxel” procedure and the fourth is a switch parameter in order to choose what type of interpolation should be used.

The switch parameter can assume two values:

1. 0, for classic paths;
2. 1, for Catmull-Rom paths.

The approach is to use the classic paths to model the closed paths of points representing vertices of the object, and to use Catmull-Rom lines for the smooth structure.

The values of the first three parameters adopted in the “closed-paths” procedure are:

1. Height step: 8.000 m;
2. Planimetric distance: 15.000 m;
3. Planimetric tolerance between points at different levels: 4.000 m.

For CCTV modeling, this parameter is set at 0 because of the shape of the building.

In the first step, the data set is horizontally clustered,

<table>
<thead>
<tr>
<th>Hor_clu ID</th>
<th>Height (m)</th>
<th># of points</th>
<th>Point ID (first Hor_clu and second Hor_clu)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-20</td>
<td>80</td>
<td>2 3 22 43</td>
</tr>
<tr>
<td>2</td>
<td>-15</td>
<td>80</td>
<td>443 444 463 484 445</td>
</tr>
</tbody>
</table>
in the second, the data are connected one other, and each horizontal cluster is finally classified as a chain or as a sown. At this point, it is possible to generate the closed paths, one for each horizontal clusters as previously done. All the points must be taken only once and all sides, if taken, only once too.

The generation of closed paths, simple in elementary cases, becomes complex as the shape of the path becomes more complicated. The shapes of four letters F, G, R and W were created starting from singular points in order to test this procedure. The results are shown in Fig. 9.

To find a closed path, first, the author connect the first point to its closest point. Then this second point is connected to its closest point, excluding the first in order to keep a unique sense of advancement, and so on. To obtain the desired closed path, the sides do not have to cross each other. If there are points not connected to other points, they must be connected. To do this, different hierarchic ways are used and topological information is analyzed. In fact, topology and geometry, in dimensions up through 3, are very intricately related [6]

Typically, these isolated points belong to levels, separated by a unique level, quite often consisting in one or a few points. Moreover, further specific analyses are developed, using to the same mechanism.

Only at this step, it is possible to follow the path of the path. Indeed following a level structure, the path between its root and the farthest leaf identifies the first half of the path. The second part is done attempting to understand if starting from the farthest leaf the root is reached.

In some cases, it is really impossible to reach the root. Therefore, the whole path is formed by adding single steps, starting from the point where the previous step finishes. After all points have been taken into account, all the sides inside the closed path will be deleted.
4.1 Piecewise Catmull-Rom’s Lines

It is possible to continuously model every path with piecewise Catmull-Rom’s lines. Starting from three points, every straight line passes through the second point and it is parallel to the line joining the first and the third points.

Fig. 10 shows the Catmull-Rom closed path between five generic points.

The advantage of a continuous interpolation is the possibility to extrapolate other points, e.g., the intersections between consecutive straight lines.

The output given is divided into two groups: the first one for chains and the second for sows’ contours.

The former are organized as follows:
1. chain perimeter ID;
2. horizontal cluster ID;
3. side ID;
4. point ID with the ID of the next point on the perimeter;
5. the total number of points in the perimeter that belong to chains.

The latter (closed paths for the sows’ contour) is organized as follows:
1. sows perimeter ID;
2. horizontal cluster (sow) ID and horizontal cluster (chain) ID;
3. side ID;
4. point ID with other point ID consecutively in the perimeter;
5. the total number of perimetred points that belong to chains;
6. the total number of the points on the perimeter.

Interpolation using Catmull-Rom resulted in the contours of chains and contours of sows. The reports are organized like so:
1. interpolation ID;
2. horizontal cluster ID;
3. side ID;
4. ID of the three consecutive points on the perimeter;
5. type of straight line:
   \[ y = ax + by \]
   \[ x = cy + dx \]
6. in the first case, a \( y \) is displayed while in the second, an \( x \);
7. the slope of the straight line;
8. \( y \)-intercept or \( x \)-intercept.

Table 5 shows an extract of the interpolation results for the chains.

If Catmull-Rom interpolation is set, the straight lines intersect each other so the “closed-path” procedure lists some additional information, divided for chains and sows:
1. ID of the three consecutive points (belonging to chains or to sows) on the perimeter;
2. X, Y and Z coordinates of the intersection point;
3. number of intersected points for chains;
4. number of intersected points for sows;
5. total number of intersected points.

4.2 Classic Way

It is also possible to model every path in the classic way, in order to make a comparison between the two modes (Catmull-Rom and the classic way).

Table 5  Chains interpolation.

<table>
<thead>
<tr>
<th>Side ID</th>
<th>Point IDs</th>
<th>Type</th>
<th>Slope</th>
<th>Intercept</th>
</tr>
</thead>
<tbody>
<tr>
<td>81</td>
<td>442 443 444</td>
<td>y</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>82</td>
<td>443 444 445</td>
<td>y</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>
Fig. 11 shows the classic contour between the same five points displayed in Fig. 10. Fig. 12 shows the image of the contour of the chains, and Fig. 13 that of the sowns. Fig. 14 shows the perimeter done in the classic way:

The final distinction between sowns and chains is done considering separate points that belong to connections classified as sowns-chains or sowns-sowns and points that belong to connections classified as chains-chains.

For the former, the numbers of points is compared: if the first group is composed of a greater number than the other one, the first is classified as a sown while the second one as a chain. On the contrary, if the first group is composed of a lower number than the other one, the first is classified as a chain while the second one as a sown.

For the latter, first the existence of all sowns or all chains is checked. If it is not so, a loop begins in order to fix all chains making a comparison between the number of points that belong to subsequently horizontal clusters.

At the end, if there are some non-classified connections, the type is set as a sown.

As shown in Fig. 15, all the clusters are correctly classified.

The levels 0.000, 45.000, 205.000 and 235.000 are correctly classified as sowns, all the others as chains.

![Fig. 11 Classic perimeter.](image)

![Fig. 12 Classic perimeter for a chain (10.000 m).](image)
Fig. 13  Classic perimeter for the contour of a sown (235.000 m).

Fig. 14  Classic perimeter.
5. Triangles

The fourth procedure is devoted to connecting points in triangles. These points belong to subsequent chains or subsequent sowns or chains and their projections upon the nearest sowns. There are many possible triangulations, the adopted one is Delaunay triangulation. Triangles are as equilateral as possible and each triangle must not contain any other points. This last fact implies that a circumcircle passing through three points, should not contain any other point. Delaunay triangulations maximize the smallest of all the angles of the triangles in the triangulation. This triangulation was invented by Boris Delaunay in 1934 [7].

Delaunay triangulation of a discrete point set $P$ in a general position corresponds to the dual graph of the Voronoi tessellation for $P$.

Fig. 16 shows a random triangulation but it is wrong because:

1. The triangle should be as equilateral as possible;
2. A circumcircle should not contain any of the other points (it is shown in Fig. 17);
3. The smallest inner angle must be at its maximum breadth.

If the structure is concave, Delaunay triangulation does not work so the following procedure is adopted:

If the middle point of the side of the triangle is inside the closed path or on the perimeter or outside the closed path. If it is inside or on the perimeter, the side is kept. If the line is outside the closed path, it is deleted.

For example, the middle points $P_{11}$ and $P_{12}$ in Fig. 18, are outside the perimeter so sides $P_7$-$P_6$ and $P_7$-$P_2$ are deleted.

If the body has a hole, the 3D model must have the same hole. So if a side is found within the hole, it will be deleted.

For example, the middle points $P_9$ in Fig. 19, is inside the hole so the side $P_1$-$P_5$ is deleted.

The input files needed in the “triangles” procedure are:

1. CCTV_ord.txt;
2. CCTV_connections_cla.txt;
3. CCTV_assembly.txt;
4. CCTV_closedpaths.txt;
5. CCTV_contours.txt.
**Fig. 16** Random triangulation.

**Fig. 17** Delaunay triangulation (in blue), Voronoi tessellation (in orange) and circumcircle (in yellow) [8].

**Fig. 18** Wrong Delaunay triangulation for a concave body.

**Fig. 19** Wrong Delaunay triangulation for a multi-connected body.

**Fig. 20** Delaunay triangulation (level -5.000).
The Delaunay triangulation is also applied to points that belong to different but consecutive levels. This is done in order to create a rough 3D surface. Fig. 21 provides the 3D plot.

The output files given are:
(1) CCTV_triangles.txt;
(2) CCTV_newpoints.txt.

The three values needed are estimated using “voxel” procedures and are:
(1) height step: 8.000 m;
(2) planimetric distance: 15.000 m;
(3) planimetric tolerance between points at different levels: 4.000 m.

The triangulation can connect points belonging to the horizontal cluster, at the same level, as shown in Fig. 20.

Each triangulation is associated with a Voronoi tessellation.

6. Bézier Spline

The fifth procedure, still being implemented, is carried out using one of people’s own software called “tri_spline”. It is devoted to interpolating triangles with triangular splines. For this task, the Bézier spline is used, because it forms continuous surfaces with one continuous derivative surfaces.

Fig. 22 shows 7 points (in red) with their point IDs (in black) and two connected Bézier splines (in blue). The endpoint of the first Bézier spline is the start point of the second Bézier spline.

The computer modeling becomes smoother, even if surface breaklines are accepted. It is noticed that discontinuities of second derivatives are geometrically a bit less evident and therefore neglected.

It is necessary to point out that it is difficult to identify breaklines not highlighted in the input. Another theoretical problem is the following: a pair of coordinates should be used instead of the one dimensional curvilinear coordinate. This leads to analytical complexity of the solution because this pair of coordinates is not yet globally defined.

7. Future Works

At the moment, some theoretical aspects should be studied in detail, recognizing the elegance of the Bézier spline, even in one-dimensional space.

Fig. 21  Delaunay triangulation.

Fig. 22  Two splines connected in the middle point.

It is evident that future work has to be focus on solving existing problems. First of all, the evaluation of the first three parameters done using the “voxel” procedure are an underestimation of the real values.

The second problem to be analyzed is the possibility to model almost-horizontal sowns correctly. In this specific case, the top of CCTV is not horizontal, while in the 3D model it is analyzed and treated as a horizontal plan.

The third problem is the fact that the Bézier spline is well known, but there is no definite coordinate pair so some extra studies are needed in order to find a solution.
8. Conclusions

The author’s first comment regards the results as a whole: they try to take a step forward compared to a wide range of previously collected samples. In any case, all programs (except for computer graphics applications) are coded, implemented, tested and used forming a library of own software.

The second comment relates to the approach used: the user is provided a graphic result and *.txt files where all the results are analyzed.

Not only convex structures, but also concave, non-stellar and multi-connected can be processed.

The geometric characteristics of the model are put first than the graphic outputted model. It implies the attention is focused on how to obtain the computer 3D model automatically and not on the graphical restitution. For this reason, the final model do not contain shadows, light effects and so on.

References