EUROGRAPHICS Workshop on Graphics and Cultural Heritage (2016)
C. E. Catalano and L. De Luca (Editors)

3D Semantic Modelling of Scale Models from 2D Historical Plans

C. Chevrier[1]

[1]MAP-CRAI, ENSA Nancy, UMR MAP n^o 3495 CNRS, Nancy, France - chevrier@crai.archi.fr

Abstract

The French collection of Plans-Reliefs, scale models of fortified towns, are an exceptional architectural heritage. Many cities, represented on these plans-reliefs, would like to expose, develop and exploit this historical knowledge. However, the fragility, the dimension of the supports and the exposure conditions make this acquisition difficult. Thus, the creation and the exploitation of a virtual model is an interesting alternative. This paper presents a new method exploiting historical documentary for the 3D semantic modelling of Plans-Reliefs as more than the half physical Plans-Reliefs are currently enclosed in containers in Paris. From 2D plans, ground outlines and facades of buildings are partially-automatically extracted to create automatically the 3D textured model of each building and ground elements (streets, rivers and courtyards). Another specificity of the method is the use of graphical schemes for the description of parametric objects.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: 3D modelling—cultural heritage

1. Introduction

The French collection of *Plans-Reliefs* are an exceptional architectural heritage (*Figure 1*). These scale models of fortified towns were built from the 17^{th} to the 19^{th} centuries. From an initial collection of 260 *plans-reliefs*, only one hundred models remain. Only 41 are exposed in two museums. The others are enclosed in containers in Paris. Many cities, represented on these *plans-reliefs*, would like to expose, develop and exploit this historical knowledge for tourism or urban issues. However, the fragility, the dimension of the supports and the conditions of their exposure make this acquisition very difficult and expensive. Thus, the creation and the exploitation of a virtual model is an interesting alternative for these cities. This creation of a digital model requires the availability of the town scale model as experiments led with Toul [CJP10], Aire sur la Lys [Ing13] and Saint-Omer [OnS14]. In this paper we present a new approach based on historical documentary resources. These historical documents bring together all the topographic surveys made on the ground by the engineers of the time and were used as specification for the construction of the physical *plans-reliefs*. Some of these documents have been digitised. Our method utilizes these documents as a data source for the reconstruction of a virtual historical city model similar to the original *plan-relief*. In a previous paper [Che15], the method was described for 3D textured meshes as input data. The same library of parametric objects is used but all the process of reconstruction is new and now specific to 2D plans. Another specificity of the new method is the use of graphical schemes for the description of parametric objects.

After an overview of the related works (section 2), a presentation of the historical documents is exposed (section 3), then our overall methodology is explained (section 4). Each step of the methodology is further exposed starting with the drawing of the polygons in the 2D plans (section 5), followed by the definition of relations between the ground polygons and their elevations (section 6) and the generation of realistic textures (section 7). The graphical schemes used for the description of the parametric models is then exposed (section 8), followed by the automatic generation of the 3D objects of the virtual version of the *plan-relief* (section 9.1). Finally results are detailed (section 10) and further enhancement discussed (section 11).

Figure 1: *Picture of the plan-relief of Verdun (currently stored in containers)*

2. Related works

For a couple of years, there are lots of ongoing works for the modelling of city scale models. In most projects, the 3D modelling is done without any automatic processes (Virtual Leodium [PCDB09], Pragues [Pra13]) or is not documented (Nantes [LKB08], Geneva [PRA11], Sarajevo [RPO15]) since the work is carried out by private companies. The focus is set to the outcome of the modelling rather than on the method used for the 3D modelling itself.

A first modelling project is the 3D modelling of Prague based on the scale model of Antonin Langweil realised between 1826 and 1837. More than a hundred people worked for the 3D model reconstruction step. The digitising was easy because the model could be divided into 52 parts (1.6 x 1m for the biggest with a scale of 1/480). The model was carried out thanks to the collaboration of Autodesk and the use of a photogrammetric software adapted and developed especially for this project [SZ09] [Pra13].

A second modelling project is the Rome reborn project [GFDs*05] [Rom13]. It aimed at illustrating the urban development of the Ancient Rome from 1000 BC to 500 AD thanks to the Plastico di Roma antica scale model. Because of the size of the model (280m^2), the accent was put on the 5% of well documented buildings (Circus Maximus, Colosseum, etc.), with a manual reconstruction. The remaining buildings (roughly 10,000) were modelled with CityEngine [Cit13], a procedural tool that creates credible but not necessarily true buildings [DMU*09].

A third project, the Hamburg project [KKSS12] was the only one in which automatic steps allow the modelling of the buildings. However lots of assumptions were made: saddle and symmetrical roofs, buildings with four sides. The scale of their physical model is 1:1000, so the geometry is simplified, most of the buildings respect the constraints (92%) and are correctly modelled.

Since our first experiments in 2010 [CJP10], other modelling projects of the *plans-reliefs* have been undertaken. The *plans-reliefs* of Marsal, Aire sur la Lys and Saint-Omer were digitised by private companies (3D textured mesh). In 2012, Google and the French Defense Ministry conducted the digitising of parts of 8 *plans-reliefs* for a visualisation in Google Earth [Alo11]. However the modelling was very disappointing: manual and inaccurate, as much for the geometry as for the photometry.

In computer vision, for full scale towns, extraction of segments or regions from images are also another research field of many teams [WZ02] [SV02] [JC08] [AF00] [TD04]. Other [HVF*97] [KD07] [MFVVG98] use geometrical properties to label each segment of a roof. [KD07] divide data into three layers: outline segments, every segment inside the outline and the roof slopes. They use TIN (Triangulated Irregular Network) algorithms to triangulate roofs but problems involving complex roofs still appears.

A visual programming language (VPL) is a programming language that lets users create programs by graphically manipulating program elements rather than by specifying them textually. Lots of VPL exist in various domains (education, games, multimedia (Quartz composer [Qua16]), simulation, automation... [vlp16] As far as 3D modelling is concerned we can cite Grasshopper [Gra17] a generative modelling interface for Rhinoceros 3D, Maya [AM13] or Blender [Ble16] software that include node editors to create shading programs as graphs. The aim of these VPL is to simplify the task of programming, making available the creation of software by non-specialists. Grasshopper is a visual programming language and runs within the Rhinoceros 3D CAD application. Programs are created by dragging components onto a canvas. The outputs to these components are then connected to the inputs of subsequent components. Many of Grasshopper's components create 3D geometry. Non computer scientists can learn to develop with Grasshopper and create their own parametric objects as in [JCH13] but it requires nevertheless a long training period.

Revit software [Rev16] is specifically built for Building Information modelling (BIM). Revit allows a user to describe graphically new parametric elements from a 3D model by specifying parameters and constraints. In our method, we describe graphically the 3D parametric models of the buildings. The simplicity of the drawings allows a better reliability and fewer mistakes compared to the former method. These descriptions can be made by non-computer scientists with no training period which is important for us as we often work with non-scientist students.

3. Historical documents

In [CJH*15] is presented an overview of the project with possible applications. You can also find a presentation and characteristics of historical documents in [CJH*15]. In this paper we focus on the modelling process.

Problems of access to the physical scale models have led to the decision of using these documents to develop a new approach of 3D semantic modelling for the *plans-reliefs*.

4. Principles of the method

Our modelling process is explained in *Figure 2* and was implemented in our own software. It is a knowledge-based approach that consists of two steps: the first one is common to all projects, whereas the second one is specific to each scale model.

- Step 1: a knowledge model of the studied architectural elements (buildings and fortification works) is carried out (a) thanks to reference literature (e.g. classical and military treatises). Each kind of element is described in a library of parametric entities (b). This library can be enriched at anytime if required. For more details about the library, you can read [Che15].
- Step 2: the modelling process consists of:
  - Semi-automatic step (d): 2D drawing of polygons in the plans: ground outline of the buildings and elevations (Section 5).
  - Relations between the ground outline of buildings and their elevations are then manually specified. Each segment of the ground outline can be connected to a corresponding elevation (d) (Section 6).
  - More realistic textures are automatically computed from the 2D plans (Section 7).
  - Finally, from a ground outline and its corresponding elevations, we automatically compute the 3D textured model of the building (e) according to a given parametric object of the library (Section 9).

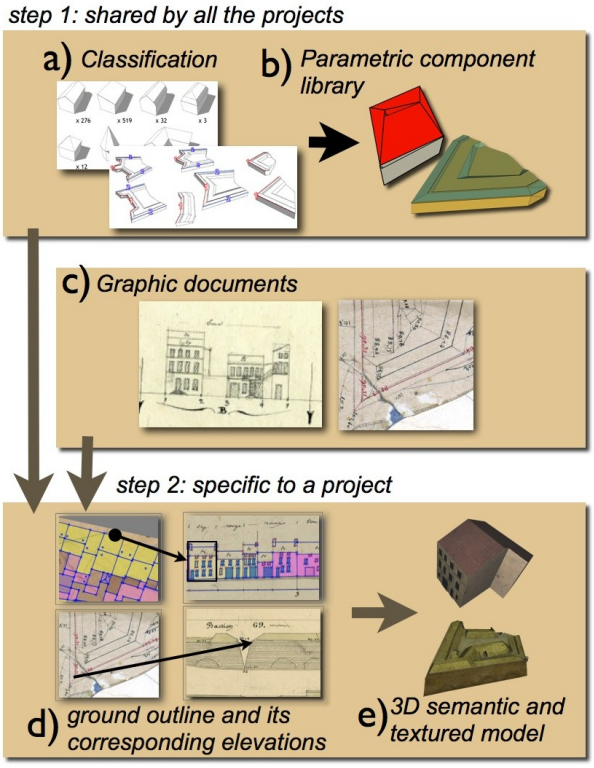

Figure 2: *Principle of the method*

5. 2D drawing of polygons in the plans

We developed a specific graphical user interface to capture polygons and segments. This step is manual for the ground plan (some tests carried out by specialists of this domain research did not produce exploitable results [Cha14]). For the elevations, simple algorithms allow us to automatically detect simple facades and their openings.

5.1. Ground plans

In the 2D ground plan of each city block the user draws points, lines and closed polygons corresponding to parametric entities of the library (*Figure 3*). The ground plan of the city block must be entirely covered by closed polygons in order to have no holes in the 3D model that will be computed from these polygons.

The following entities are drawn:

- *Buildings*: outline of each building is depicted as a closed polygon. The ridge, roof breaks, valleys and angles are represented by segments inside the outline.
- *Chimneys*: chimneys are represented by points inside the building outline.
- *Ground*: streets and courtyards are represented by polygons.
- *Walls*: outline of walls are represented by a thin polygon with parallel sides.
- *Stairs*: outline of stairs are represented by a polygon, the steps of the stairs are not drawn.
- *Vegetation*: vegetation is depicted as points and polylines.

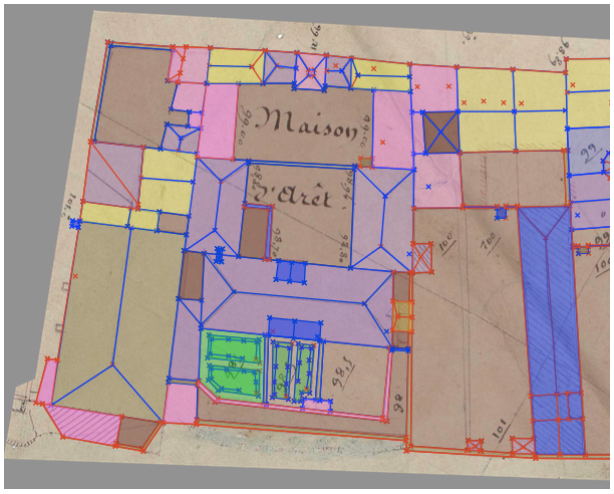

Figure 3: *sol*

For each element drawn on the plan, we must only indicate the type of parametric entity (building, walls, vegetation, stairs). The segments inside the building outline allow automatically the retrieval of the kind of building as explain in [Che15]. Similar rules for walls allow the retrieval of the kind of wall according to the outline. Polygons on the ground plans are used to define position, orientation and dimensions in x and z axis. Axis Y is the vertical axis.

5.2. Elevations

In the 2D elevation plans, must be drawn (*Figure 4*):

- closed polygons corresponding to facades with their openings and cornices, walls, stairs, chimneys. Openings are not used for the creation of the 3D model of the buildings but are used for the automatic process of creation of the textures (*Section 7*).
- lines corresponding to ridges.

A reference altitude is also chosen for each city block (the lowest altitude of the city block) and then on each elevation plan, a line corresponding to this reference altitude must be drawn. The 3D model will be then correctly build and positioned in height (altitude). Elements on the elevation plans are used to define dimensions in the vertical axis (y).

Automatic detection of facades produces good results with clear drawings (*Figure 4c*) but bad results with complex drawings (*Figure 6*). The algorithm detects first the outline of the facade (from a user picked point). Then it detects openings and cornices. Then ridge is detected and finally chimneys. Algorithms are simple and could be improved with better competence in image analysis process. The process is as follows (*Fig. 4 and 5* illustrate the process):

- A white and black picture is computed from the original plan with a threshold color of (160, 160, 160) for our plans.
- In this picture, dilatation process is computed from the picked point in white area (*Figure 5*) green color).

- close to the detected area, we look for vertical separation black segments that will be the limits of the facade: for each column of pixels inside de bounding box of the area (*Figure 5*) light blue box), near the right limit then near the left limit, we count the number of black pixels. The column with the maximum number is kept to be the searched segment.
- close to the detected area, we look for the bottom segment (not necessary horizontal) of the facade. We look for the longest segment near the bottom part of the bounding box.
- For the top part of the facade, we look for a flat segment. If no flat segment is found we look for a pinion composed of one or two inclined segments.
- then openings are searched inside the facade (lintel can be rounded), cornices are also sometimes detected (*Figure 5*) pink color).
- If the top of the facade is flat, we look for an horizontal segment above to detect the roof ridge. We also look for squares that represent chimneys. They are difficult to detect because features are not dark enough.

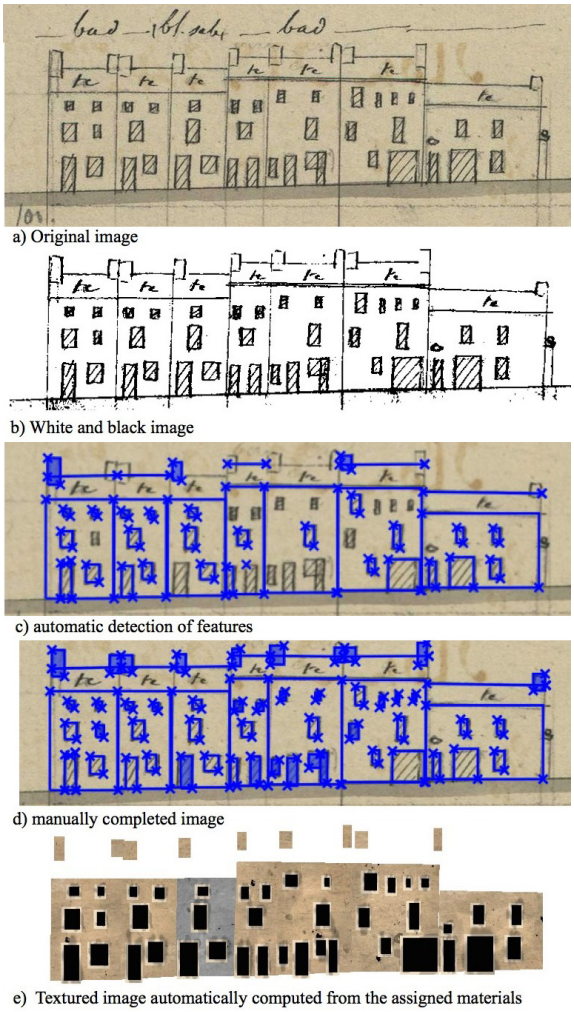

Figure 4: *Steps of feature detection process.*

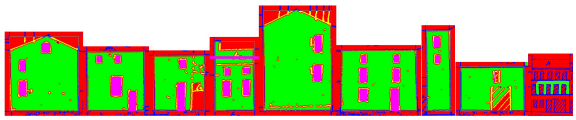

Figure 5: *examples of automatic detection with false colors for areas.*

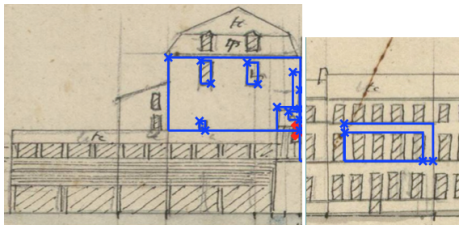

Figure 6: *Problems with automatic detection due to a large concentration of features. In some parts nothing is detected (hatching parts).*

6. Relations between the ground polygons and the elevations

Once the drawings are done, one must now define the relations between the ground outline of a building and its elevations (*Figure 7*). We develop a specific user interface allowing to specify which segment of the building outline corresponds to a facade. One can also have the drawing of only a part of a facade but not of the whole facade. We can then specify which part of the ground segment corresponds to the part of the facade. This will be taken into account in the automatic creation of the building.

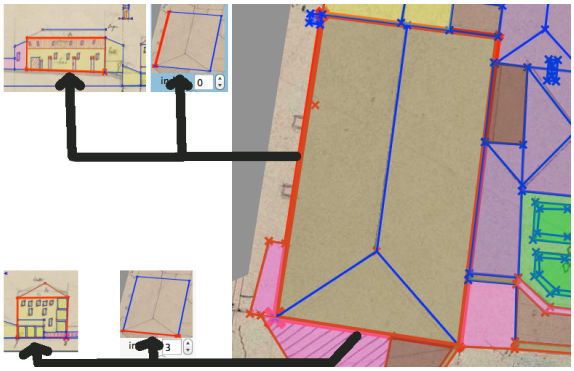

Figure 7: *Relations between the edge of the ground polygon with their corresponding facade.*

Interior segments require also relations with elevation features: the ridge segment on the ground plan must be related to the ridge segment on the elevation plan. Points corresponding to chimneys on the ground plans must be related to the corresponding polygon in the elevation plan.

Buildings and walls work in a same manner. Stairs can have one or two relations in the elevation plan. Vegetation has no relation because it is exceptionally represented in the elevation plans.

However, inconsistencies, errors and omissions are common and make it necessary to use *a priori* knowledge for addressing interpretation problems. Spending more time to understand and to interpret documents can sometimes solve this problem otherwise a solution is chosen between the various possible hypothesis. The most common problems we have encountered are:

- More or less facades between two numbers in the elevation plans than in the corresponding part in the ground plan.
- Lack of information about facades on small courtyards.
- Facades of a building are not consistent in height or for the roof ridge or for the kind of roof. They also can be incoherent with the roof building shape in the ground plan.
- Some corrections have been made several years later and modifications are difficult to understand or become inconsistent with facades. Crossing-out on the drawings are not clear (*Figure 8*).

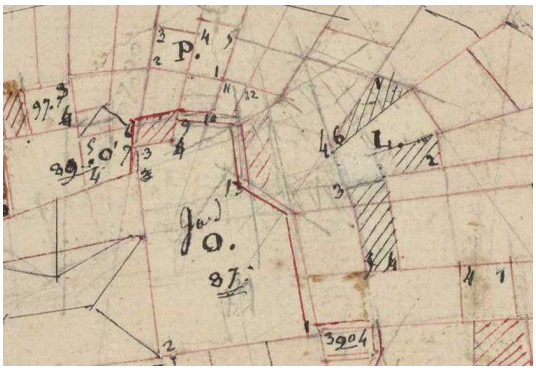

Figure 8: *Parasitic lines in the drawing make it no clear.*

7. Generating the textures from the 2D plans

In order to have more realistic textures for the 3D elements, we compute for each 2D plan a corresponding texture that will be used when creating the 3D models (*Figure 4e*).

7.1. Material assignment

Information about materials are written on the elevation plans for the material of the facades and for the kind of tiles (*Figure 4a* above the facades). We assign to polygons a material from a library. This library has been predefined with the most common materials encountered in the *plans-reliefs*. Transparency can be used for the materials. For ground plans, every polygon must have a material: grass, sand, soil, cobbled for ground / slate, terra cotta for tiles / ochre and other hues for walls. For elevation plans, all facades, cornices, walls and chimneys must have a material assignment. Every non material assigned polygon on elevation plans (like openings) will be black with white edges.

7.2. Automatic computing of the textures

From the material assignment, the texture image of the plan is computed by filling each polygon with the texture of the material (*Figure 9*). For elevation plans, the process begins with the polygons not enclosed in others (it means beginning with the facades and ending with the openings). For ground plans, textures of tiles are automatically oriented to follow the main sewer of the roof so that the tiles are correctly positioned. In courtyards, various polygons inside the outline of the courtyard allow to draw a composite texture (for example, brown soil and grass).

Along each edge of facades, the color of the textured polygon is automatically darkened. For each edge of openings, some dirt is automatically added to imitate the traces of glue of the physical scale models and add more realism (*Figure 4 e*).

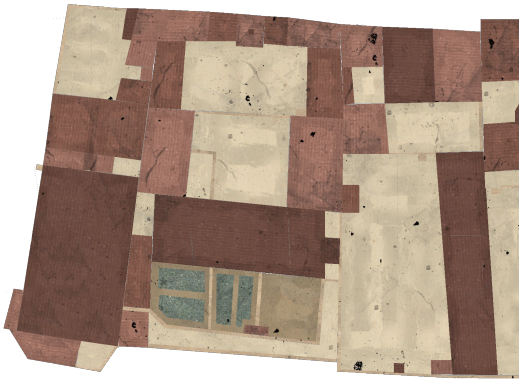

Figure 9: *Automatically textured ground plan*

We now have all required data to automatically compute the 3D parametric elements. Let's first see how we describe the parametric models with graphical scheme before computing the 3D elements.

8. Graphical schemes for the definition of parametric models

The shape of the bodies are mainly of four main types (I, L, U, T) but the roofs can present more variations. The openings are not modelled, they are present only on textures. So only the shape of the building is modelled, making quite simple the drawing of the ground outline, the facades and the roof pans. As explained in [Che15], we use parametric models for the buildings. However the description of the parameters and of the creation method is quite complex and especially for non-computer scientists. That's why we decided to simplify this task by using graphical schemes. The description had become then very simple. An exemple for the *I-shaped body two-slope roof* can be found in *Figure 10*. This section presents the graphical description of a parametric architectural element and algorithms for the automatic creation of the 3D models from these schemes.

After an overview of the kinds of buildings present on scale models (section 8.1), the various kinds of parameters we use to describe a building are then exposed (section 8.2). Section 8.3 allows the presentation of the various schemes we use to describe a parametric element and Section 8.4 details the algorithms to automatically create the 3D model from the parameters and to retrieve the value of the parameters from the 3D points present in the input data.

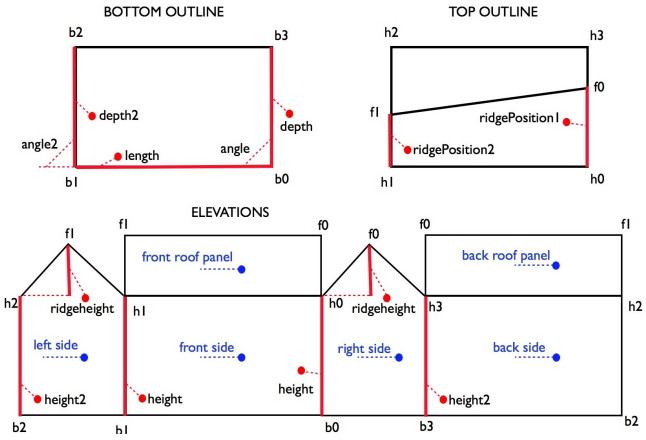

Figure 10: *Description scheme of the I-shaped body two-slope roof building. b_i is a bottom point, h_j is a point of the roof outline and f_k is a point inside the roof outline.*

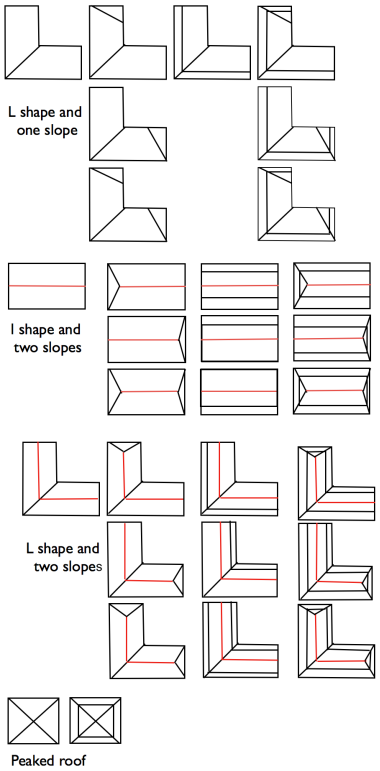

Figure 11: *The I and L types of roofs encountered in the scales models are mainly peaked, one-slope or two-slope roofs. They are combined with the body shapes and also with breaks and/or hips.*

8.1. Buildings on scale models

Buildings on scale models present few details as shown in *Figure 1*. Only the global shape of the building is represented. The openings and other facade features are only represented with textures. There is sometimes roof overhangs. The various shapes for the building bodies are I, L, U, T and O shape. More than 90 % of the buildings are of I shape. Most of the rest are of L shape. Shape for the roof pans can be more complicated with many unique case. Most of the cases are one or two pans with break or hip pans (*Figure 11*). See [Che15] for more details.

Some roof pan configurations may be found once or only several times in a scale model. However, with our method it is simple to describe them and fast to model as the process of reconstruction is quite automatic (full if we work with 3D textured meshes [Che15] and partially with 2D plans).

We currently have about sixty various parametric models for the buildings (found in the *plans-reliefs* of Toul and Verdun. The use of graphical schemes allow the creation of the 3D models from the parameters and also the determination of the parameters' value in a reserve process. We then do not need anymore the rules for the creation of the 3D parametric model of the buildings (*section 4.2.1 in [Che15]*) nor the rules for an automatic retrieval of the parameter values (*section 4.2.3 in [Che15]*). All the useful information for the creation of the 3D model or of the retrieval of the parameters' value are included in the schemes : points, facets, parameters and constraints. Some models have curved facets. We will explain how we handle this particular case also in the followings sections.

8.2. The various kind of parameters are:

- **lengths** (length, depth or other distance): to define a length parameter, two points are necessary. The second point will be computed from the first point.
- **heights** : to define a height parameter, two points are necessary. The second point will be computed from the first point.
- **angles**: to define a angle parameter, two points are necessary. The second point will be computed from the first point.
- **proportion** (of a segment relatively to another segment): three points are used to define a proportion parameter: $[p_1,p_2] = value * [p_1,p_3]$. Example $[h_0,f_0] = value * [h_0,h_3]$ in *Figure 10*.
- **curvature** (for curved facet): to define a curvature parameter, two points are necessary. A list of points will then replace the segments to create a curved line.
- **parallelism constraints**: to define a parallelism constraint four points are necessary. The second point is placed according to the other points. $\overrightarrow{p_1,p_2} \,//\, \overrightarrow{p_3,p_4}$.
- **alignment constraints**: to define an alignment constraint three points are necessary. The second point is placed according to the other points.

The **aim** is to position the parameters on the schemes in order to allow the computation of the 3D points that will compose the 3D model of the parametric object.

8.3. The various schemes

Three schemes are used : the bottom scheme (*Section 8.3.1*), the top scheme (*Section 8.3.2*), and the elevation scheme (*Section 8.3.3*). In the bottom and top schemes, the x and z coordinates of the 3D points can be computed from the parameters present in the

schemes. In the elevation scheme, the y coordinate of the 3D points can be computed from the parameters.

8.3.1. Bottom scheme (*Figure 12*a).

This scheme allows defining the kind of body: I, L, T, U, O or other configuration. On this scheme we have:

- A polygon representing the outline of the body.
- Each point of the polygon must have a name: we choose the following convention: b_i is a bottom point, h_j is a roof contour point and f_k is a point inside the roof contour.
- All the useful parameters for the creation of the outline of the body (y coordinates are null): lengths, angles and curvatures.
- If a segment of the polygon has a curvature parameter, the segment must also be named with a point list that will be used for the creation of the 3D model.
- Constraints (aligment or parallelism) can also be defined here.

The computation starts from the first point b_0 that is set to (0,0,0) at the beginning of the process. If there are some induced parameters, one can declare some additional variables with a computing expression and use these variables as supplementary parameters for the schemes. These additional variables are not considered as parameters for the parametric object, they are only variables for the computation.

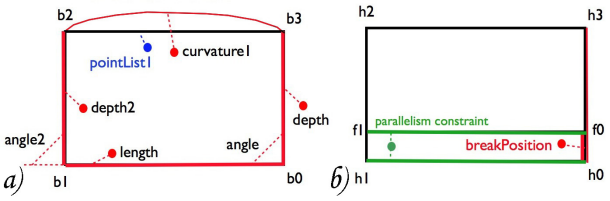

Figure 12: *a) Bottom scheme of a curved building. there is a curved facade between the point b_2 and the point b_3: parameter curvature1. The list of points used here is pointList1 in blue color. b) Top scheme of a I-Shape body break roof building. There is a proportion parameter breakPosition (in red color) from the segment $[h_0 f_0]$ relative to the segment $[h_0 h_3]$. There is also a constraint parameter (parallelism in green color) between the segment $[f_0 f_1]$ and the segment $[h_0 h_1]$.*

8.3.2. Top scheme (*Figure 12*b).

On this scheme we have:

- A polygon representing the outline of the roof. The top outline is the same as in the bottom scheme.
- Segments representing ridge, roof breaks, valleys and angles.
- Each point of the scheme must have a name: h_j and f_k.
- All the parameters useful for the computation of the 3D points (y coordinates are null at this step): curvature and proportion parameters.
- Some constraints, if required (parallelism or alignment).

An example of the use of proportion and parallelism constraints can be seen in *Figure 12*b. The x and z coordinates of the h_j are the same as their corresponding b_j. The parameters on the top scheme allow the computation of the f_k points.

8.3.3. Elevation scheme (*Figure 10*).

The elevation scheme corresponds to the developed of the facades. This scheme allows also creating developed drawings of 3D models for paper toys for example. On this scheme we have:

- Points with a name corresponding to those given in the bottom and top schemes.
- Polygons corresponding to the facets of the building (facades and roof slopes) with a name. Each facade can be created in 3D thanks to the named points.
- All the parameters useful for the computation of the y coordinate of the 3D points: height (height between bottom points and top points, roofing height, break height, hip height), curvature and proportion parameters. A same parameter can appear several time in the scheme: each y-point has to be defined from another point (except for the bottom points that are set to 0 for y coordinate). These repetitions of parameters will also be useful for the reverse process in case of lack of information (for example, we do not dispose of all the facade drawings in the 2D plans).

8.4. Algorithms

Two algorithms are used to handle the 3D parametric models:

- The automatic creation of the 3D model from the parameter values and the schemes (*Section 8.4.1*)
- The automatic adjustment of the value of parameters from the points and the schemes (*Section 8.4.2*)

8.4.1. Automatic creation of the 3D model from the parameters and the schemes

The algorithm *1* allows the computation of the points of the 3D model from the parameters' value. The value of the parameters have been assigned by the user in the graphical user interface or assign automatically from the reverse process. Once the 3D points are computed, the 3D model can be created from these points and the graphical description. The process to compute the 3D points is: first, we compute x and z coordinates of the bottom points: *Algorithm1*(ground plan, bottom scheme). Then we compute x and z coordinates of the top points: *Algorithm1*(ground plan, top scheme) and finally we compute y coordinate of the points: *Algorithm 1*(elevation plan, elevation scheme).

8.4.2. Automatic adjustment of the value of parameters

We assume that points' coordinates are assigned from input data (2D plans (*section 9.1*) or 3D textured mesh) by the reverse process. Some points may not be assigned because of lack of information in the input data (facade not described in a 2D plan or not captured in a 3D mesh). The algorithm *2* allows now the computation of the parameters' value from the points' coordinates.

9. Automatic generation of the 3D parametric objects

First of all, we correctly position the 2D ground plans in 3D, the 3D objects will be superimposed on them. The ground plans allow determining the positions and dimensions in a horizontal plan, while the elevation plans allow determining the positions and dimensions along the vertical axis. On one side we have a ground outline and

Algorithm 1 Compute the points from the parameters

Input: *plan*: ground plan or elevation plan
Input: *scheme*: bottom, top or elevation scheme
p_1 *to* p_4: points in the parameter representation (b_i, h_j or f_k).
Put all the parameters of the *scheme* in an *array*
while there is parameter p in *array* **do**
  switch (p)
  case *length parameter*:
    if p_1 coordinates are assigned (x and z if we treat the bottom or top plan, y if we handle the elevation plan) **then**
      Look for an *angle* parameter that uses p_1 and p_2
      Compute vecteur v from the *value of p* and the *angle*
      Compute $p_2 = p_1 + v$
    end if
  case *height parameter*:
    if p_1 are assigned **then**
      p_2.y = p_1.y + *value of p*
    end if
  case *proportion parameter*:
    if parameter points p_1 *and* p_3 are assigned **then**
      Compute p_2 with the relation: $\overrightarrow{p_1,p_2} = value\ of\ p * \overrightarrow{p_1,p_3}$
    end if
  case *parallelism constraint*:
    if parameter points p_1, p_3 *and* p_4 have their x and z coordinates assigned **then**
      Compute p_2 with the relation: $\overrightarrow{p_1,p_2} = k * \overrightarrow{p_3,p_4}$
    end if
  case *alignment constraint*:
    if parameter points p_1 *and* p_3 have their x and z coordinates assigned **then**
      Compute p_2 with the relation: $\overrightarrow{p_1,p_2} = k * \overrightarrow{p_1,p_3}$
    end if
  end switch
  if a point has been computed, remove p (and *angle*) from *array*
end while
Create the curved segments:
for each segment s_{scheme} of the outline **do**
  Look for a *curvature* parameter associated to s_{scheme} and get the associated *pointList*
  Compute the list points by creating an arc with *curvature* from the first point to the second point of the segment..
end for

its relations (facades), on the other side we have the description of a parametric object. We have to create an instance of that object and to determine the parameters' value from the ground outline and its relations. We also have to place that objet correctly in 3D. The first step is to determine the coordinates of the 3D parametric object from the 2D plans (*Section 9.1*) and then we will be able to assign the values to the parameters (*Section 9.2*).

9.1. Determine the 3D object coordinates from the 2D plans.

We assume that the bottom outline on the 2D plan and the bottom outline on the scheme begin with the same point (b0) and turn in the same way. If not, we pre-treat the polygon in the ground plan as in [Che15]. The process is to compute:

Algorithm 2 Compute the parameters' value from the 3D points

p_1, p_2, p_3, *and* p_4 : points of the parameter
array : all parameters in the schemes (bottom, top and elevation)
while there is a parameter p in *array* **do**
  If required points for the computation are assigned the *value* of p is computed according to the parameter kind:
  value = distanceBetween(p_1, p_2) for a length
  value = angleBetween($\overrightarrow{p_1,p_2}$, $\overrightarrow{X\text{-}axis}$) for a angle
  value = p_2.y - p_1.y for a height
  value = lengthOf($\overrightarrow{p_1p_2}$) / lengthOf($\overrightarrow{p_1p_3}$) for a proportion
  if p is assigned **then**
    Remove it from the *array*
  end if
end while
create the curved segments (*algo 5*).

- the ground coordinates of the bottom and top points with *algo 3*,
- the y coordinate of the top points with *algo 4*,
- the coordinates of the points inside the roof outline with *algo 5*,
- the chimneys with *algo 6*.

Algorithm 3 Compute the x and z coordinates of the points

outline: bottom/top outline in the scheme
polyg: polygon in the 2D ground plan
p_{plan}: point in the 2D plan ; p_{scheme}: point in the scheme
p_{object}: point in the object (points we want to compute)
for i=1 to nbPointsInPolyg **do**
  p_{scheme} = outline.points[i] (for example b_0)
  p_{object}: point with the same name in the parametric object
  p_{plan} = polyg.points[i]
  $p_{object}.x$ = p_{plan}.x (affect the coordinate of the plan point to the object point)
  $p_{object}.z$ = p_{plan}.z
end for

Algorithm 4 Compute the y coordinate of the top points

bottomOutline: bottom outline in the ground plan
for each *relation* in *bottomOutline*.relations **do**
  $facade_{plan}$ = facade of the *relation*
  $facade_{scheme}$: scheme facade with the same points (b_i, h_i or f_i)
  for i=1 to facade.numberOfPoints **do**
    p_{scheme} = $facade_{scheme}$.points[i] (for example h_1)
    $p_{object}[i]$: point with the same name in the object
    p_{plan} = $facade_{plan}$.points[i] no name for that point
    $p_{object}[i].y$ = height measured from the reference in the elevation plan of $facade_{plan}$.
    memorize the bottom-points' height to assign automatically the correct height to the street points.
  end for
end for

We now have computed the coordinates of all points represented the 3D parametric object. We must then determine the position, orientation and value of the parameters of the parametric object.

Algorithm 5 Compute the inside points' coordinates

By comparing the drawings of $topOutline_{plan}$ and $topOutline_{scheme}$, we can assign the x and z coordinates of the inside points of the parametric object.
With the relations on the ground inside segments (ridges), we can compute the y coordinate of the inside points.

Algorithm 6 Compute the chimneys

Get non-connected points inside the outline in the ground plan.
Chimneys are rectangular boxes computed with the process of 9.1.
Cast a vertical ray in the building 3D model to find the intersection between the chimney and the house: the chimney can be correctly positioned in height.

9.2. Determining the parameters' value and computing the textures

Section 8.4.2 has presented the algorithm for the retrieval of the parameters' value from the 3D points. Adjustment in the parameters value or in the textures can be done in the user interface. Finally, we have to compute the textures from the facades and ground polygons in the 2D plans. Two kinds of texture are automatically computed: from the plans and more realistic textures with the images computed in *Section 7*. The case of the streets, rivers and courtyards is particular: these objects are created as 3D triangulated polygons. As there is no relations on the 2D polygons on the ground plans, only (*algo 3*) is processed. The height of each point was memorised during the computation of the buildings and they are used now to affect the points of the street, courtyard or river points.

10. Results and discussion

Results are shown in *Figures 13 and 14* for a city block and for a larger part in Unity game engine with the vegetation. For the vegetation, we process as in [CJP11]: the position and the kind of vegetation is given to Unity via files and the vegetation is planted with scripts from models in a library. Some indication is sometimes found on the 2D plans for the kind of vegetation: the kind of tree or the rough height of the trees are then assigned to their corresponding 3D models.

This method has been used for the 3D modelling of the *plan-relief* of Verdun [CJH*15] and produces good results. Monuments like churches, city halls are composed of several simple parametric elements. It is not necessary to describe them as a whole complex entity. The limits are for very complex buildings. In this case there are manually modelled. If a shape of the building is complex and unique in a scale model, one can wonder if it is worth to describe it with our method. Won't it be quicker to model it with a commercial software ? Based on our experience, we estimate the time each method will take and we decide for each case how we model it.

11. Conclusion and future work

In this paper, we have presented the method we have conceived and developed for the 3D semantic modellisation of old scale models from their historical documents. This method could be improved by automating the drawing of the polygons in the 2D plans. It is currently the most manual time consuming task in the process.

We have also presented the new method for the graphical description of the parametric objects which is more simple and reliable. Our aim is now to describe composed objects (like monuments, churches or bridges) with relative positioning as simply as possible: an element is at the right (left / in front of / behind) of another element, is above or below, is anchored at a specified point of another element...

In our library, there are only several models with curved surfaces. Tests could be done with more complicated surfaces for body shapes but also for roof shapes. We just began to work with the *plan-relief* of Strasbourg which have other kinds of shape with bow windows and dormer windows. We would like also to extend the library with other kinds of parametric objects like the fortification works as studied in [JCH13]. Our method could also be tested with full scale models of town.

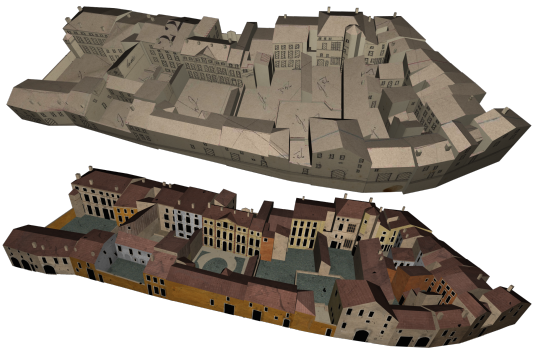

Figure 13: *Result of the 3D modelling on the number 22 city block of the plan-relief of Verdun.*

Figure 14: *Visualisation of the result for a large part of Verdun plan-relief with the vegetation.*

Acknowledgements

Thanks are due to the persons who participate to the project: Kevin Jacquot, Vincent Marchal, Pascal Humbert, Gilles Halin, Senda BenBouheni and Jingfei Zhang.

References

[AF00] AMERI B., FRITSCH D.: Automatic 3d building reconstruction using plane-roof structures. In *ASPRA Conference, Washington, DC* (2000). 2

[Alo11] ALOEST: Projet reef auguste magnin, 2011. http://aloest.com/content/mod(accessed october 2013). 2

[AM13] AUTODESK-MAYA:, 2013. http://usa.autodesk.com/maya (accessed June 2016). 2

[Ble16] Blender software. https://www.blender.org, 2016. Accessed: 2016. 2

[Cha14] CHALFOUH S.: Extraction dÕéléments graphiques pour la modélisation du plan-relief de verdun. In *Technical report Master 2* (laboratoire Image, Informatique et Interaction (L3i) de l'université de La Rochelle, France, 2014). 3

[Che15] CHEVRIER C.: Semi-automatic parametric modelling of the buildings on town scale models. *Journal on Computing and Cultural Heritage (JOCCH) vol. 7*, n. 4 (February 2015). 1, 2, 3, 5, 6, 8

[Cit13] CITYENGINE: City engine, 2013. http://www.procedural.com (accessed June 2013). 2

[CJH*15] CHEVRIER C., JACQUOT K., HUMBERT P., BENBOUHENI S., HALIN G.: Virtual 3d reconstruction of plan-relief from historical document analysis for valorisation applications. In *Digital Heritage* (Grenada, Spain, 2015). 2, 9

[CJP10] CHEVRIER C., JACQUOT K., PERRIN J.: 3d modeling of a town scale model. In *Proceedings of the EuroMed conference* (Limassol, Cyprus, 2010), pp. 99–107. LNCS 6436. 1, 2

[CJP11] CHEVRIER C., JACQUOT K., PERRIN J.: Modeling specificities of a physical town scale model. In *Proceedings of the Digital Media and its Applications in Cultural Heritage (DMACH) conference* (Amman, Jordania, March 13-15 2011), pp. 103–117. 9

[DMU*09] DYLLA K., MÜLLER P., ULMER A., HAEGLER S., FRISCHER B.: Rome reborn 2.0: A framework for virtual city reconstruction using procedural modeling techniques. In *Proceedings of Computer Applications and Quantitative Methods in Archaeology (CAA)* (2009). 2

[GFDs*05] GUIDI G., FRISCHER B., DE SIMONE M., CIOCI A., SPINETTI A., CARASSO L., LOREDANA L., RUSSO M., GRASSO T.: Virtualizing ancient rome: 3d acquisition and modeling of a large plaster-of-paris model of imperial rome. *Videometrics VIII vol. 5665* (18-20 January 2005), 119–133. 2

[Gra17] Grasshopper-rhino. http://www.grasshopper3d.com, 2017. Accessed: 2016. 2

[HVF*97] HENDRICKX M., VANDEKERCKHOVE J., FRERE D., MOONS T., GOOL L. V.: 3d reconstruction of house roofs from multiple aerial images of urban areas. In *International Archives of Photogrammetry and Remote Sensing* (Stuttgart, September 17-19 1997), pp. 88–95. vol 32, Part 3-4W2. 2

[Ing13] INGEO: Modélisation 3d du plan-relief d'aire sur la lys, 2013. http://www.ingeo3d.fr. 1

[JC08] JAW J., CHENG C.: Building roof reconstruction by fusing laser range data and aerial images. In *Proceedings of the International Society for Photogrammetry and Remote Sensing* (Beijing, China, 2008), pp. 707–712. 2

[JCH13] JACQUOT K., CHEVRIER C., HALIN G.: Reverse engineering of scale models using dataflow programming: Application to the fortification of plans-reliefs. In *Digital Heritage International Congress* (Marseille, France, 2013), pp. 63–69. 2, 9

[KD07] KOEHL M., DARWISH O.: Construction et intégration de maquettes 3d dans un sig. In *Conférences SIG* (Versailles, France, 2007). 2

[KKSS12] KERSTEN T., KELLER F., SAENGER J., SCHIEWE J.: Automated generation of an historical 4d city model of hamburg and its visualization with the ge engine. In *Proceedings of the 4th international conference on Progress in Cultural Heritage Preservation* (Berlin, Germany, 2012), pp. 55–65. 2

[LKB08] LAROCHE F., KEROUANTON J., BERNARD A.: A case study of capitalization and valorization of our technical heritage. In *CIRP Design conference* (2008). 2

[MFVVG98] MOONS T., FRERE D., VANDEKERCKHOVE J., VAN GOOL L.: Automatic modeling and 3-d reconstruction of urban house roofs from high resolution aerial imagery. In *Proceedings of the European Conference on Computer Vision* (Freiburg, Germany, 1998), pp. 410–420. 2

[OnS14] ONSITU: Modélisation 3d du plan-relief de saint-omer, 2014. http://www.patrimoines-saint-omer.fr/Le-territoire/Le-plan-relief-en-3D. 1

[PCDB09] PFEIFFER M., CARRÉ C., DELFOSSE V., BILLEN V.: Virtual leodium: From an historical 3d city scale model to an archaeological information system. In *Proceedings of the CIPA conference* (Kyoto, Japan, October 11-15 2009). 2

[PRA11] PRAM: Projet relief auguste magnin, 2011. http://www.ville-geneve.ch/fileadmin/public/Departement_3/Documents_d_actualite/relief-magnin-conference-presse-presentation.pdf (accessed october 2013). 2

[Pra13] PRAGUES: Scale model of pragues, 2013. http://www.langweil.cz/index_en.php (accessed June 2013). 2

[Qua16] Quartz composer, mac os x. http://quartzcomposer.com, 2016. Accessed: 2016. 2

[Rev16] REVIT: Autodesk, 2016. http://www.autodesk.fr/products/revit-family/overview (accessed April 2016). 2

[Rom13] ROME: Scale model of rome, 2013. www.romereborn.virginia.edu (accessed June 2013). 2

[RPO15] RIZVIC S., PLETINCKX D., OKANOVIC V.: Enhancing museum exhibitions with interactive digital content: Sarajevo city model interactive. In *XXV International Conference on Information, Communication and Automation Technologies (ICAT)* (2015). 2

[SV02] SUVEG I., VOSSELMAN M.: Automatic 3d building reconstruction. pp. 59–69. Volumes 4657 - 4677. 2

[SZ09] SEDLACEK D., ZARA J.: Graph cut based point-cloud segmentation for polygonal reconstruction. In *Proceedings of the 5th international Symposium on Advances in Visual Computing: Part II* (Las Vegas, Nevada, USA, 2009). 2

[TD04] TAILLANDIER F., DERICHE R.: Automatic buildings reconstruction from aerial images: a generic bayesian framework. In *International Archives of Photogrammetry and Remote Sensing* (2004). Remote Sensing and Spatial Information Sciences 35 (Part B3). 2

[vlp16] Visual programming languages. https://en.wikipedia.org/wiki/Visual programming language, 2016. Accessed: 2016. 2

[WZ02] WERNER T., ZISSERMAN A.: New techniques for automated architectural reconstruction from photographs. In *ECCV02(II: 541 ff.)* (2002). 2