A GIS Data Realistic Road Generation Approach for Traffic Simulation

Yacine Amara, Abdenour Amamra, Yasmine Daheur, and Lamia Saichi
Ecole Militaire Polytechnique, Bordj El-Bahri BP 17, Algiers, Algeria
Email: amara.yacine@gmail.com

Abstract—**Road networks exist in the form of polylines with attributes within the GIS databases. Such a representation renders the geographic data impracticable for 3D road traffic simulation. In this work, we propose a method to transform raw GIS data into a realistic, operational model for real-time road traffic simulation. For instance, the proposed raw to simulation-ready data transformation is achieved through several curvature estimation, interpolation/approximation, and clustering schemes. The obtained results show the performance of our approach and prove its adequacy to real traffic simulation scenario as can be seen in this video[1].**

Index Terms—**Road interpolation, Road modeling, Traffic simulation, Vehicle virtual navigation.**

I. INTRODUCTION

In recent years, applications of road traffic simulation have become ubiquitous in everyday life: driving simulation or racing games are increasingly attracting the attention of developers. However, the results obtained are not always consistent with reality. When one wants to reproduce a realistic behavior, the developer must consider the real parameters of the road.

The realization of a real-life road simulation would make it possible to forecast the traffic generated in a given road network. This realism, modeled on a computer, would help managers to detect the problems present in the network, namely congestion, accidents, user stress, an insufficient number of channels, etc. Besides optimizing traffic on the roads without making urban changes, such as location of traffic lights, number, and width of lanes.

The shape of the road is complex, particularly at mountainous segments, severe turns, etc. Therefore, finding the best mathematical function that will fit the shape of these sections while taking into account compressing the number of data to be stored is challenging.

Geometric processing applications rely on the geometric properties of curves such as torsion, curvature, and tangents. In our study, we chose the curvature metric in the road's modeling surface, since the latter embeds the information on the shape of the road. The actual data is extracted from the Geographic Information System (GIS), the latter known for providing the ability to manipulate geographical information laid out on multiple layers. The metric being chosen; our approach is then organized as follows: extraction of the road layer as polylines in the GIS data; curvature estimation at the neighborhood of the points; a grouping of points in clusters according to

the sign of their curvature, and finally the approximation of clusters by continuous mathematical functions.
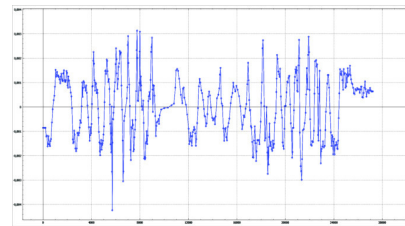

Fig. 1

II. RELATED WORKS

Most of the work that has been proposed on road reconstruction is based on procedural modeling. Roads are generated by empirical rules often based on observation. We will classify the methods proposed in two categories: the methods taking the input of calculated roads (procedural approaches) and the models taking as input the data resulting from Geographic Information Systems (GIS).

A. Procedural approaches

These approaches take as input roads derived or calculated by mathematical or optimization methods. Both, Parish et al. [1] and Jing et al. [2] generated models of cities in which road networks played a key role. Their roads were built from the central lines that are generated either by grammar rules named *L-Systems* [1], or model-based methods [2]. However, the defining laws or models to regenerate exactly the structure of urban roads are not available. Hence, their roads often drift from their real respective counterparts. In addition, their methods were not designed for highways and suburban roads.

Some of these limitations have been resolved by Chen et al. [3], where they used a tensor field from which a graph representing the road network was calculated. This can be modified to control locally the generated road profile. Their method permits creating large areas but does not allow the reconstruction of existing networks because of the problem of scale limitation. In addition, only urban roads were considered.

Another procedural model was proposed by Galin et al. [4, 5] based on a platform for generating mapped roads that contain many types of features such as trees, rivers, and lakes. Their roads resulted from a short path algorithm instead of the real data. As a result, we obtain fictitious roads that do not

---

[1]https://youtu.be/t8eyphcFYHc

satisfy the constraints of civil engineering. The generated roads are erroneous and do not match the shape of the terrain. This lack of realism is because of the shortcomings of the proposed approaches. The major shortcoming of these works is not to have considered the actual data as input of the problem.

B. GIS data approaches

To overcome the problems mentioned and to build more realistic roads, Bruneton and Neyret [6] proposed a method of generating large road surfaces from GIS data. They represented the roads with Bezier curves to join the sampled points and then mapped them on the ground. Although this model helps to perceive actual road coordinates, current roads are no longer built from Bezier curves. Again, these models are affected by the lack of realism. The details and constraints related to civil engineering discipline are not taken into account. Road networks must be defined by simple forms such as straight lines, arcs of circles, and clothoids. This drawback had been addressed by the following work:

1) LSGA algorithm: This work [7] introduces a new approach to construct smoothed curve pieces representing realistic roads. Given a GIS database of road networks, where the sampled points are organized as 3D polylines, this method creates horizontal and verticals curves, then it combines them to generate the roads. The major contribution of this work is a tree traversal algorithm that extends the sequences of the best fit primitives and a fusion process of these primitives. The latter must respect a certain grammar according to an automaton. This approach offers more realistic results than those that preceded it, and the errors are proportional to the noisiness of input data.

2) Construction by clothoids: In [6], the algorithm for adjusting a sequence of G_2 polylines into clothoid segments takes place in two steps: first a piecewise linear approximation is applied, then a sequence of rigid 2D transformations is applied in order to align the in one consistent result. Although this method respects civil engineering constraints and models the transition between a circular arc and a straight line with a clothoid, its first pass through the linear segments loses the precision when estimating the radius of curvature.

3) Automatic generation of 3D roads: This method, presented in [8], is based on a set of civil engineering rules. It proposes a new approach for the automatic 3D generation of high-fidelity roads. It transforms GIS data that only contains 2D information from the central axis of the road into a 3D model of the road network. In the proposed approach, basic road elements such as road segments, road intersection are generated automatically to form sophisticated road networks. But in the modeling of the axe of the road, the segments were connected by Hermite curves, which satisfy only G_1 continuity, hence the civil engineering constraints were not respected.

The proposed model must be realistic, that is to say, that it must meet the constraints of civil engineering and vehicle dynamics, for this reason, our study was based on an essential criterion which is *curvature*. Curvature, the inverse of the radius of the circle tangent to the curve, is defined as the norm of the acceleration vector of a body traveling the curve at unit speed. It is the second derivative with respect to the curvilinear abscissa of the body position.

III. CURVATURE ESTIMATION

A. Definition

A parametric curve is a function $r : I \subset \mathbb{R} \to \mathbb{R}^n$, when $n = 2$ it is called a *plane curve*. The curvilinear abscissa s from a point $r(t_0)$, $t_0 \in I$, at a given point $r(t_1)$, $t_1 \in I$, is defined by:

$$s(t_1) = \int_{t_0}^{t_1} \|r'(u)\| \, du \tag{1}$$

The vector $T(s) = r'(s)$ is called the tangent vector. The normal vector $N(s)$ is obtained by a rotation of $90°$ anticlockwise. The vectors $T'(s)$ and $N(s)$ are collinear. That is, there is a function $k(s)$ such that:

$$T'(s) = k(s) \times N(s) \tag{2}$$

called the *curvature* of the curve at the point $r(s)$. The curvature also corresponds to the variation of the direction of the tangent vector respectively to the curvilinear abscissa: $k(s) = \theta'(s)$, such that :

$$\theta'(s) = \measuredangle(\overrightarrow{T(s)}, \overrightarrow{(1,0)}) \tag{3}$$

Since our initial data is in discrete form, we performed a local estimation using a sliding window. The latter is centered around a point allowing to approximate all it neighbors within the window by a second order polynomial. The interest of such an operation is to deduce the first and the second derivatives, to then calculate the curvature. Several approximation methods called *implicit parabola fitting* proposed in [9], which have a good performance and a fair simplicity of implementation.

B. Second-order curve approximation

In the implicit parabola fitting method, proposed in [9], the curve is described by a function such that: $y = f(x)$ or $x = f(y)$. The variation of x and y inside the window determines the parameterization to adopt, in fact if the variation of x is greater than that of y, the algorithm will select the case $y = f(x)$ and vice versa . In order to simplify the notation, we will consider that $p_0 = (0,0)$. The goal is to find f'_0 and f''_0, which minimize:

$$E_x(f'_0, f''_0) = \sum_{i=-q}^{q} (y_i - f'_0 x_i - \frac{1}{2} f''_0 x_i^2)^2 \tag{4}$$

The solution of this problem of least squares gives:

$$f'_0 = \frac{cg - bh}{ac - b^2} \quad f''_0 = \frac{ah - bg}{ac - b^2} \tag{5}$$

such that :

$$a = \sum_{i=-q}^{q} x_i^2 \quad g = \sum_{i=-q}^{q} x_i y_i \quad b = \frac{1}{2} \sum_{i=-q}^{q} x_i^3,$$
$$h = \frac{1}{2} \sum_{i=-q}^{q} x_i^2 y_i \quad c = \frac{1}{4} \sum_{i=-q}^{q} x_i^4 \tag{6}$$
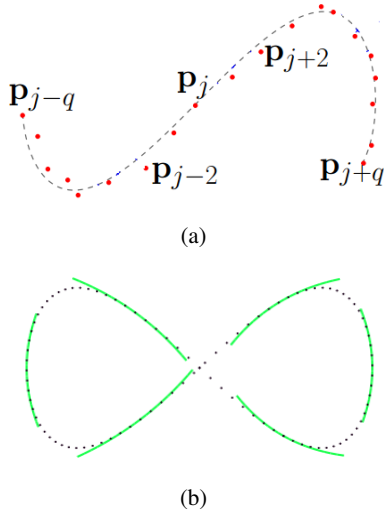

(a)

(b)

Fig. 2: (a) Points window, (b) Approximation by least squares

C. Curvature computation

1) Model and notation: Consider a set of points p_i of a flat smooth curve r, in this study the curve is parameterized by the arc length, the estimate of the curvature for a plane curve requires an approximation of the first and second derivatives of $r(s)$. Let a point p_0 be chosen . The derivation of r in p_0 will be estimated from a window of size $2q + 1$ around $p_0 : p_{-q}, p_{-q+1}, \ldots, p_q$ (see Fig 2a). We set $p_0 = r(0)$ as the origin, the approximation of degree two can be written in the form:

$$r(s) = r'(0)s + \frac{1}{2} + r''(0)s^2 \tag{7}$$

2) The least squares approach: The estimate of $r'(0)$, $r''(0)$ is obtained by the least squares approach (Fig 2b). The weighting w_i of the point p_i must be positive, relatively important for small values of $|s_i|$ and relatively small for large values of $|s_i|$.

The arc length s_i can be estimated as follows: Δl_k is the arc length of the vector $p_k p_{k+1}$, where k varies from $-q$ to $q - 1$. The arc length between p_0 and p_i can be approximated by:

$$\begin{cases} l_i = \sum_{k=0}^{i-1}, i > 0 \\ l_i = -\sum_{k=0}^{-1}, i < 0 \end{cases} \tag{8}$$

D. Curvature computation for a plane curve

For the case of plane curves, the idea of the Independent coordinates method [9] is to construct a parametric curve $(x(s), y(s))$ that approaches the curve locally, by quadratic functions as a function of arc length.

$$\begin{cases} \hat{x}(s) = x_0 + x_0's + \frac{1}{2}x_0''s^2 \\ \hat{y}(s) = y_0 + y_0's + \frac{1}{2}y_0''s^2 \end{cases} \tag{9}$$

The derivatives x_0' and x_0'' are estimated by minimizing:

$$E_x(x_0', x_0'') = \sum_{i=-q}^{q} w_i (x_i - x_i'l_i - \frac{1}{2}x_0''l_i^2)^2 \tag{10}$$

The minimization of this equation can be written in the following matrix form

$$\begin{bmatrix} a_1 & a_2 \\ a_2 & a_3 \end{bmatrix} \begin{bmatrix} x_0' \\ x_0'' \end{bmatrix} = \begin{bmatrix} b_{x,1} \\ b_{x,1} \end{bmatrix} \tag{11}$$

such that :

$$a_1 = \sum_{i=-q}^{q} w_i l_i^2 \quad a_2 = \frac{1}{2} \sum_{i=-q}^{q} w_i l_i^3 \quad a_3 = \frac{1}{4} \sum_{i=-q}^{q} w_i l_i^4$$
$$b_{x,1} = \sum_{i=-q}^{q} x_i w_i l_i \quad b_{x,2} = \frac{1}{2} \sum_{i=-q}^{q} w_i l_i^2 x_i \tag{12}$$

Algorithm 1: Weighted least square variables setting

$I[] = a_1 = a_2 = a_3 = b_{x,1} = b_{x,2} = b_{y,1} = b_{y,2} = 0$
for *i = -q; i ≤ q; q++* **do**
  $\quad | \quad I[i] \leftarrow I[i-1] + \|p_i p_{i-1}\|$
end
$m = I[0]$
for *i = -q; i leqq; q++* **do**
  $\quad I[i] \leftarrow I[i] - m$
  $\quad w \leftarrow weight(I[i])^2$
  $\quad a_1 \leftarrow a_1 + w(I[i])^2$
  $\quad a_2 \leftarrow a_2 + \frac{w}{2}(I[i])^3$
  $\quad a_2 \leftarrow a_2 + \frac{w}{4}(I[i])^4$
  $\quad b_{x,1} \leftarrow b_{x,1} + w(I[i])(x_i - x_0)$
  $\quad b_{y,1} \leftarrow b_{y,1} + w(I[i])(y_i - y_0)$
  $\quad b_{x,2} \leftarrow b_{x,1} + \frac{w}{2}(I[i])^2(x_i - x_0)$
  $\quad b_{y,2} \leftarrow b_{y,1} + \frac{w}{2}(I[i])^2(y_i - y_0)$
end
$d \leftarrow a_1 a_3 - a_2^2$

The same procedure is applied for the calculation of y_0' and y_0''. The tangent T is obtained by the normalization of the vector $r_0' = (x_0', y_0')$, while the normal vector is obtained by a rotation of 90° of T.

IV. CUTTING ACCORDING TO CURVATURE

This step will aim to cut the road into a set of primitives consisting of straight lines, left/right turns based on the value of curvature estimated at each point as explained in the previous step.

Algorithm 2: Coefficient computation

$x_0' \leftarrow (a_3 b_{x,1} - a_2 b_{x,2})/d$
$y_0' \leftarrow (a_3 b_{y,1} - a_2 b_{y,2})/d$
$x_0'' \leftarrow (a_1 b_{x,2} - a_2 b_{x,1})/d$
$y_0'' \leftarrow (a_1 b_{y,2} - a_2 b_{y,1})/d$
$\kappa \leftarrow (x_0' y_0'' - y_0' x_0'')/ \left\| (x_0', y_0') \right\|^3$
$T \leftarrow (x_0', y_0')/ \left\| (x_0', y_0') \right\|$
$N \leftarrow sign(\kappa)(-T_y, T_x)$

For this, we will execute the following processes in the order indicated. We first start with a classification in two primitives only, that is to say left/right turn, we continue by extracting the sections which are straight lines independently of the primitives obtained previously and we thus finish by eliminating the isolated points, which are of the right type because a primitive of the right type must have a minimum number of two points. In what follows we will detail the procedure for filtering the curvature values according to the order of execution of the steps.

A. Preprocessing

In order to facilitate the different treatments, an initial marking of all the points of the curve has been carried out. For this, we assigned to each point whose curvature is positive the number +1, and -1 for those whose value of the curvature is negative, and we stored the result in a table named "ids" which will be subsequently updated by the results of the different processing.

B. Left and right turns detection

Our approach was based on a local estimate of the curvature sign near the point considered. A sliding window of size w has been used to estimate the global sign of the values inside the window by summing the values previously assigned, so the classification of the types of sections will be carried out as following :

- The sum is positive, so it is a right turn and the point is marked +1.
- The sum is negative, so it is a left turn and the point is marked -1.
- The zero sum is not a possible case because the size of the window around the point is odd.

This step is very important because it allows to assign a point to a given turn even if the sign of its curvature does not correspond, because it is the trend of the whole neighborhood is taken into consideration.

C. Straight line detection

Straight line detection is based on the value of the estimated curvature. Indeed, we know that a line is characterized by a null curvature. Since the values of the estimated curvature are not all accurate and contain noise, a threshold δ has been set. If the curvature norm is below this threshold, the point will be assigned to a straight line primitive. Points satisfying the relationship $|K(s)| < \delta$ will be assigned the number 0.

D. Elimination of isolated right point

Knowing that a marked point of a straight line can not be isolated, the solution is to go through all the points of the line and to modify those which are of marking different from their neighbors according to the signs of the curvature at this point, i.e. +1 marking if the curvature is positive, -1 otherwise.

Algorithm 3: Primitive assignment

while $i < ids.size()$ **do**
    gpts gpt
    gpt.type $\leftarrow$ ids[i]
    gpt.id $\leftarrow$ i
    gpt.nbr $\leftarrow$ 0
    for $i = 0; i < ids.size(); i{+}{+}$ **do**
        if *gpt.type* $\neq$ *ids[i]* **then**
            Break
        end
        gpt.nbr++
    end
end

V. Turns approximation with second order polynomials

A. Preprocessing

In order to interpolate curves with polynomials, a data structure has been created to facilitate processing on the one hand and to save the results obtained on the other hand. This data structure ("gpts") consists of an integer field named "type", which will contain the type of the primitive according to the marking carried out in the previous step. An integer field named "id", which will contain the identifier of the first point of the primitive. An integer field named "nbr", which will contain the number of points belonging to this primitive as well as two other fields "paramX" and "paramY", which will be used to store the coefficients of polynomials associated with turns. The structure being created, the fields were subsequently assigned according to algorithm 3.

For each left-handed or right-turn type primitive, we will proceed to the parameterization according to the length of the arc, so we will have two polynomials according to x and y such that:

$$\begin{cases} x(s) = a_0 + a_1 s + a_2 s^2 + a_3 s^3 \\ y(s) = b_0 + b_1 s + b_2 s^2 + b_3 s^3 \end{cases} \tag{13}$$

The polynomials in question will be of degree 3, this choice can be justified by the absence of inflection points since the primitives have been classified according to their (there is not a transition in the same road segment). They are convex or concave curves hence the choice of the polynomial regression that is a statistical analysis that describes the variation of an dependent random variable, called here x or y, according to an independent random variable, called here s, being the length of the arc. We seek, by regression, to bind the variables by

a polynomial of degree 3. The calculation of the coefficients therefore amounts to solving a system of equations that can be expressed in the following matrix form:

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 1 & s_1^1 & s_1^2 \cdots & s_1^m \\ 1 & s_2^1 & s_2^2 \cdots & s_2^m \\ \vdots & \vdots & \ddots & \vdots \\ 1 & s_n^1 & s_n^2 \cdots & s_n^m \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix} \quad (14)$$

The number $m = 3$ corresponds, in our case, to the cubic polynomial regression. We will treat the case of the polynomial $x(s)$ generated and it is the same for $y(s)$. The problem is therefore to find the vector $\vec{a}$ in the equation:

$$\vec{x} = S\vec{a} \quad (15)$$

The solution is given by:

$$\vec{a} = (S^T S)^{-1} S^T \vec{x} \quad (16)$$

B. Error criterion

Given the needs and requirements of the realism imposed on the generated road, we set ourselves the objective of approximating the GIS input data with lines and polynomials of third degrees such that the maximum deviation between the two curves does not exceed 2.0m, in this part we discuss the calculation of error as well as treatments to be undertaken in the case of exceeding this limit.

1) Error computation: The error d is estimated as the distance between the approximate point $p = (x, y)$ and the entry point $p = (x, y)$, it is deduced according to the following formula:

$$d = \sqrt{(\hat{x}(s) - x(s))^2 + (\hat{y}(s) - y(s))^2} \quad (17)$$

The maximum value of this error must not exceed δ for each of the points of the different right or polynomial primitives.

$$d_{max} < \delta \quad (18)$$

2) Error processing: Primitives whose error exceeds the required threshold must be modified, for this our approach is to generate from the initial primitive two primitives such that the number of points constituting each primitive is equal to half the number of points of the primitive. For a given primitive C_m one generates two primitives $C_{m/2}$ and $C'_{m/2}$, and the same treatment will be executed on the two new primitives in a recursive way, this will assure us:

- Compliance with the error limit required for each final primitive obtained,
- The generation of a minimal number of primitives,

VI. RESULTS

The results obtained when applying our method are summarized in Figure (3). The figures show the stages of our approach. Figure (3a) shows a sample of a road polyline. In Figure (3b), the curvatures were estimated by the least squares method. Note that this representation is more exploitable than that of Figure (1), and that there is less disturbance in the values of the curvatures due to windowing.

A. Curvature estimation by the implicit parabola fitting method

In order to overcome the problems encountered when calculating the curvature by cubic splines, we opted for an estimation of the latter by the least squares. Compared with spline, the quality of curvature values improved significantly. Indeed, most points follow the trend of the turn where they belong as can be seen in Figure (3b). Nevertheless, in the case of more complex turns, outliers appear. Hence, we have proposed to neglect them, and to mark a window by a single sign of curvature relative to reality.

B. Clustering

Figure (3c) shows that the points of the input road were grouped into three basic categories: left turn, right turn, and straight line. We note that the results are more refined, because this step corrects any possible residual error of the previous section. Moreover, it allows noise reduction, by defining the straight line segments from a given threshold, since the SIG data being noisy, we cannot obtain zero curvature values.

This step allowed us to have the same signs of curvature for a given type of cluster. Nevertheless, at this stage the clusters are not connected to each other, a major problem on which the approximation will be based in the following phases.

To overcome the problem of connection, we fix first the ends of the primitives, then we approximate the calculated model of the initial points. In this perspective, the Bezier curves with a least squares approximation prove to be an adequate choice. Indeed, the first and last control point are superimposed on their correspondents in the initial data, then the other points are calculated by minimizing the differences between the model and the initial data. The main disadvantage of this method is that it guaranties only C_0 continuity between two clusters of successive points, which does not satisfy civil engineering constraints. Moreover, in some complex turns, it remains difficult to follow the shape of the cluster by a third degree polynomial.

Figure (3g, 3h) represents the constructed road. The initial data points circled and clustered. Note that the resulting model does not deviate from the input road polyline. The curves in magentas represent the edges of the road, the input data (polylines) are in red, and the model (at the central axis) is in blue. Figure (3h) illustrates the mapping of the road to the geo-referenced satellite image corresponding to the road section. As can be seen, the model provides a smooth representation of the road surface. We see that complex shapes such as turns are well approximated. In addition, our model meets the C_2 continuity (imposed by road civil engineering), along the road without oscillations or other erratic behaviors that would compromise the visual comfort during the simulation.

VII. CONCLUSION AND FUTURE WORKS

The objective of this work was the realistic modeling of the road surface by taking into account a number of civil engineering and vehicle dynamics constraints. We chose curvature

as a parameter describing the shape of the road in order to approach the profile of the latter as closely as possible.

We proceed through the estimation of curvature by a windowing approach because of the discrete nature of GIS data. In order to approach the reality, we grouped the points into clusters according to their neighborhood's curvature to obtain basic forms of the road namely left/right turns and straight lines.

This being done, the next step was to find the mathematical functions as well as the appropriate conditions and constraints to approach the initial data with functions that respect C_2 continuity conditions.

The results obtained are satisfactory insofar as our road reconstruction approach takes into account the real constraints, moreover the model obtained is of C_2 continuity, which reflects the smoothness of the position, speed and the acceleration of the simulated vehicle.

As perspective, we plan to improve and refine this work by considering road intersections, since the latter are frequently encountered in real-life situation.

REFERENCES

[1] Yoav IH Parish and Pascal Müller. "Procedural modeling of cities". In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM. 2001, pp. 301–308.

[2] Jing Sun et al. "Template-based generation of road networks for virtual city modeling". In: *Proceedings of the ACM symposium on Virtual reality software and technology*. ACM. 2002, pp. 33–40.

[3] Guoning Chen et al. "Interactive procedural street modeling". In: *ACM transactions on graphics (TOG)*. Vol. 27. 3. ACM. 2008, p. 103.

[4] Eric Galin et al. "Authoring hierarchical road networks". In: *Computer Graphics Forum*. Vol. 30. 7. Wiley Online Library. 2011, pp. 2021–2030.

[5] Eric Galin et al. "Procedural generation of roads". In: *Computer Graphics Forum*. Vol. 29. 2. Wiley Online Library. 2010, pp. 429–438.

[6] Eric Bruneton and Fabrice Neyret. "Real-time rendering and editing of vector-based terrains". In: *Computer Graphics Forum*. Vol. 27. 2. Wiley Online Library. 2008, pp. 311–320.

[7] Hoang Ha Nguyen, Brett Desbenoit, and Marc Daniel. "Realistic road path reconstruction from GIS data". In: *Computer Graphics Forum*. Vol. 33. 7. Wiley Online Library. 2014, pp. 259–268.

[8] Jie Wang, Gary Lawson, and Yuzhong Shen. "Automatic high-fidelity 3D road network modeling based on 2D GIS data". In: *Advances in Engineering Software* 76 (2014), pp. 86–98. DOI: 10.1016/j.advengsoft.2014.06.005. URL: https://doi.org/10.1016%2Fj.advengsoft.2014.06.005.
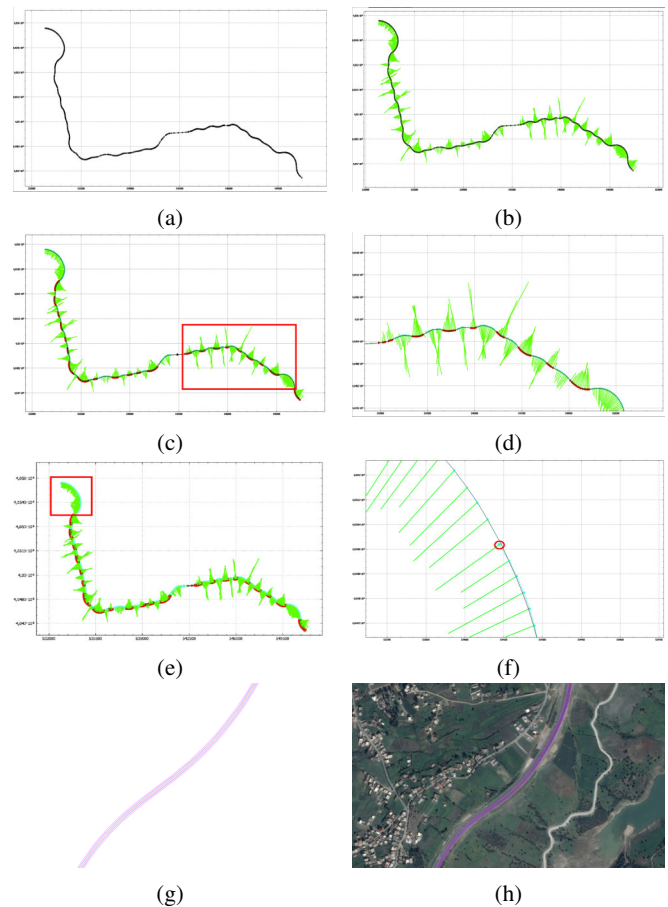
Fig. 3: Processing steps of the proposed algorithms. (a) Raw road in polyline form, (b) Curvature obtained by least squares according to the normal, (c) Cutting of the road (left, right turns and straight lines) according to curvature values, (d) Turns detected by the cutting algorithm, (e) Polynomial fitting of the detected turns, (f) 3rd degree polynomials approximation of a given turn, (g) Result: generated road surface, (h) Mapping of the result on a real road

[9] Thomas Lewiner et al. "Curvature and torsion estimators based on parametric curve fitting". In: *Computers & Graphics* 29.5 (2005), pp. 641–655. DOI: 10.1016/j.cag.2005.08.004. URL: https://doi.org/10.1016%2Fj.cag.2005.08.004.