VLSS

Journal of
Visual Languages and Sentient Systems

Journal of

Visual Languages and Sentient Systems

Volume 1, 2015

Table of Content

PREFACE

The Journal of Visual Languages and Sentient Systems (VLSS) is intended to be a forum for researchers, practitioners and developers to exchange ideas and research results, for the advancement of visual languages and sentient multimedia systems. The success of visual languages especially iconic languages is evident to everyone because most smart phones these days use iconic languages to communicate with the end user. Ironically the success of visual languages in practice has led to doubt and uncertainty about the future of visual languages research. However the advances of sentient systems can motivate more research on visual languages, both at the practical level and at the theoretical level.

Sentient systems are distributed systems capable of actively interacting with the environment by gathering, processing, interpreting, storing and retrieving multimedia information originated from sensors, robots, actuators, websites and other information sources. In order for sentient systems to function efficiently and effectively, visual languages may play an important role. To stimulate research towards that goal, the Journal of Visual Languages and Sentient Systems is born.

VLSS publishes research papers, state-of-the-art surveys, review articles, in the areas of visual languages, sentient multimedia systems, distributed multimedia systems, sensor networks, multimedia interfaces, visual communication, multi-media communications, cognitive aspects of sensor-based systems, and parallel/distributed/neural computing & representations for multimedia information processing. Papers are also welcome to report on actual use, experience, transferred technologies in sentient multimedia systems and applications. Timely research notes, viewpoint articles, book reviews and tool reviews, not to exceed three pages, can also be submitted to VLSS.

Manuscripts shall be submitted electronically to VLSS. Original papers only will be considered. Manuscripts should follow the double-column format and be submitted in the form of a pdf file. Page 1 should contain the article title, author(s), and affiliation(s); the name and complete mailing address of the person to whom correspondence should be sent, and a short abstract (100-150 words). Any footnotes to the title (indicated by *, +, etc.) should be placed at the bottom of page 1.

Manuscripts are accepted for review with the understanding that the same work has not been and will not be nor is presently submitted elsewhere, and that its submission for publication has been approved by all of the authors and by the institution where the work was carried out; further, that any person cited as a course of personal communications has approved such citation. Written authorization may be required at the Editor's discretion. Articles and any other material published in VLSS represent the opinions of the author(s) and should not be construed to reflect the opinions of the Editor(s) and the Publisher.

Shi-Kuo Chang
Editor-in-Chief
Journal of Visual Languages and Sentient Systems

Generative Interface Structure Design for Supporting Existing Objects

Nurcan Gecer Ulu
Carnegie Mellon University

Levent Burak Kara
Carnegie Mellon University

Abstract—Increasing availability of high quality 3D printing devices and services now enable ordinary people to create, edit and repair products for their custom needs. However, an effective use of current 3D modeling and design software is still a challenge for most novice users. In this work, we introduce a new computational method to automatically generate an organic interface structure that allows existing objects to be statically supported within a prescribed physical environment. Taking the digital model of the environment and a set of points that the generated structure should touch as an input, our biologically inspired growth algorithm automatically produces a support structure that when physically fabricated helps keep the target object in the desired position and orientation. The proposed growth algorithm uses an attractor based form generation process based on the space colonization algorithm and introduces a novel target attractor concept. Moreover, obstacle avoidance, symmetrical growth, smoothing and sketch modification techniques have been developed to adapt the nature inspired growth algorithm into a design tool that is interactive with the design space. We present the details of our technique and illustrate its use on a collection of examples from different categories.

Fig. 1. Example problem to generate a phone stand (a) given object and environment configuration (b) user defined target and root points (c) generated interface structure (d) 3D printed result.

I. INTRODUCTION

The customization and personalization of products started to compete with traditional mass production principles with the contribution of maker movement and DIY (Do-It-Yourself) culture. DIY commonly refers to any fabrication, modification or repair event that is outside of one's professional expertise [1]. With the rise of DIY culture, there is a growing interest for design and fabrication tools tailored towards non-expert users.

Recent advances in 3D design and manufacturing technologies now have made content creation accessible to novice users. Besides the basic consumer level 3D printers, online on-demand 3D printing services (e.g. Shapeways, i.Materialise) have enabled ordinary people to access high quality machines. 3D modeling software, such as Autodesk 123D and Tinkercad, allow consumers to create 3D shapes using simplified geometric interaction methods. However, current commercial design software do not take advantage of capabilities of 3D printing. While *almost anything* can be fabricated using 3D printing, these design software limit potential design outputs by mimicking features of traditional manufacturing and assembly methods. In this work, we extend the design possibilities by taking a generative design approach to create organic looking branching shapes that would be challenging to design and fabricate with traditional methods.

We propose a framework that automatically generates interface structures under prescribed constraints. The input to our algorithm is a surface mesh for the object to support and a mesh to represent the ground surface with target and root points to create a shape in between (Fig.1). Then, automatic interface structure generation is achieved by a nature inspired growth mechanism. Users can control the design by changing target-root combinations at the input phase as well as by using sketch modifications after the shape is created. Moreover, the stochastic nature of the growth algorithm lets users design one of a kind pieces by generating different outputs for the same problem on each run. The main contribution of this work is the novel application of a nature inspired growth algorithm for automatic product generation. This is accomplished by the introduction of target attractor and pruning concepts, embedding product design considerations and user interaction.

II. RELATED WORK

Design Tools for Non-Expert Users have recently received significant attention. In [2], a chair design tool is proposed to create balanced chairs from extruded 2D profile sketches. To enable informed exploration, Umetani et al. [3] proposed a suggestive design tool for plank-based furniture. In that work, the user adds planks and edits their positions, orientations or size. A data-driven approach to interactive design of model
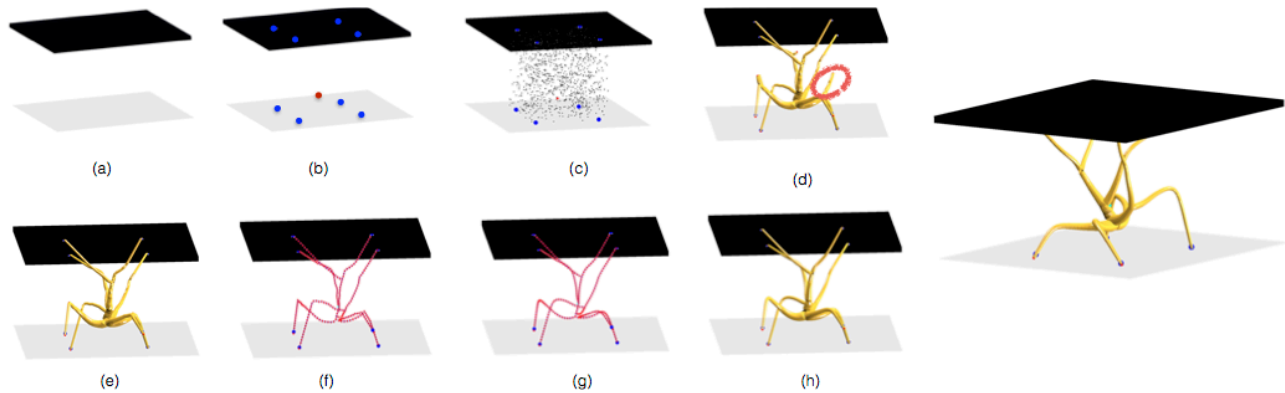
Fig. 2. Interface structure generation process. First, user defines the environment configuration (a) and selects the root and target points shown as red and blue, respectively (b). Attractors are generated randomly in the design space (c) and interface structure is grown automatically (d). Then, unnecessary branches are removed automatically (d-e) and the skeleton of the interface structure is smoothed as desired (f-h). Resulting shape is shown on the right.

airplanes is proposed in [4] where the user creates free-flight gliders with 2D sketches. In this work, we focus on creating a large range of products instead of one specific group such as chairs or gliders. While all three systems are notable interactive tools, components of the resulting designs are limited to 2D laser cut pieces. Our system generates organic 3D geometries that can take advantage of the opportunities in 3D printing.

Much of recent research on design for 3D printing addresses modifications of existing digital models by optimizing physical properties, such as balance and structural strength. For this purpose, inner carving with deformations [5, 6] and thickening of thin sections [7] have been used. Here, we focus on geometric shape *generation* whereas their focus is on shape modification.

Generative Design methods are recognized as significant technologies to rapidly generate different design alternatives. Fabrication of generatively produced designs have been examined illustrating how geometrically complex shapes can be physically created in [8, 9]. In computer graphics, generative design methods have been used to create architectural models such as buildings [10], virtual cities [11] and trees [12, 13]. In this paper, we are inspired by a specific generative design method developed to simulate the tree growth process for automatic shape generation.

Tree Growth methods have been widely used in computer graphics for urban modeling and computer animation. As such, tree growth has been vastly investigated in the literature. L-systems have been used to generate trees in [12]. Runions et al. [13] proposed a space colonization algorithm to mimic open and closed venation process in the leaf formation. Later, the space colonization algorithm has been extended to grow trees in [14] and [15]. Tree generation inside various geometries is investigated using spatial attractor distribution in [16]. In this paper, our automatic interface generation process has been inspired by the space colonization algorithm. We use its spatial attractor distribution feature to enable the interaction with the design space. Moreover, interaction of the tree model with
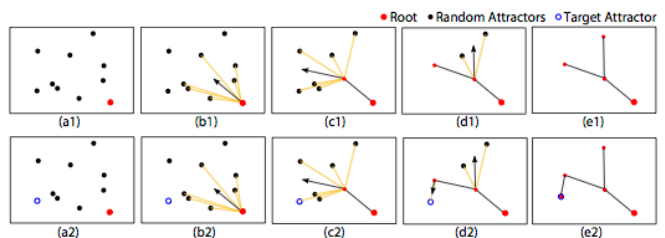
Fig. 3. Space colonization algorithm with (a1-e1) and without (a2-e2) target attractors.

the obstacles in the environment has been studied in [17]. In that, a fully grown tree model is placed around an object and the colliding branches are removed. For our purposes, this approach cannot be used since the grown structure has to be connected to the target points and a branch connected to a target can not be removed. Hence, we utilize obstacle avoidance during the growth process.

III. AUTOMATIC SHAPE GENERATION

In this work, the aim is to automatically create an interface structure between given objects. This process is illustrated in Fig.2 starting with the user input and the steps of the automatic shape creation performed on the background. First, the user supplies the input geometries as 3D mesh models (a). Then, a set of target points are selected by the user to define where the interface structure should be in contact with the input models (b). Then, a root point or points are provided by the user to start the growth process (b). Attractors are randomly generated inside the design space (c). The structure is generated akin to a tree originating at its root and growing in 3D space to reach the targets (d). Branches that are not connected to the input objects through target points are removed from the structure (e). We also refer to this step as pruning or unnecessary branch removal. Finally, the skeleton of the structure (f) is smoothed (g-h). In the following sections, the details of these steps are described.

6
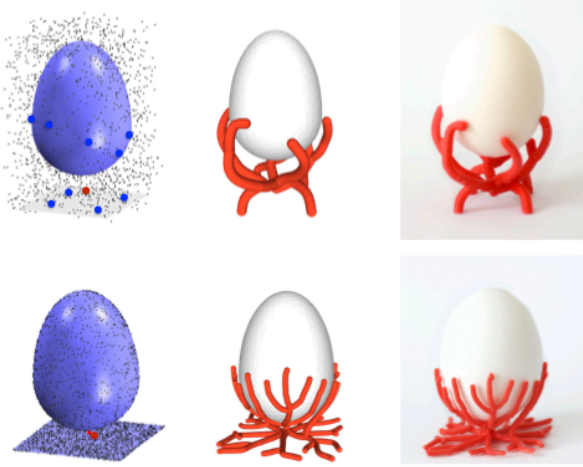
Fig. 4. Egg holder generated using volume (top) and surface (bottom) hormones. Left: problem setup, middle: digital model, right: 3D printed result.

Fig. 5. Example projection growth. Generating the interface structure .Left: problem setup with root & target points illustrated, middle: digital model, right: 3D printed result. Orthogonal (top) and top (bottom) views of the same part are given.

A. Growth Algorithm

The proposed method uses an attractor based growth approach of space colonization algorithm given in [13]. Space colonization algorithm creates a branching tree structure in space as demonstrated in Fig.3.a1-e1. The tree structure grows without the guarantee that it would touch any specific point in the design environment. In this work, we need to create shapes between objects ensuring that the generated interface would be in contact with the target objects to support them. For this reason, we introduce a novel target attractor concept to create branching structures that grow to the required target positions (Fig.3.a2-e2).

The target based growth process starts with the definition of the design space, e.g. rectangle in (a2) and target-root point selections. Then, random attractors are sampled uniformly inside the design space. These random attractors have an *influence distance* that they can pull a branch to themselves as well as a *kill distance* that makes them inactive when they get too close to a branch in the growing structure. At every growth step, depending on the influence and kill distance, each attractor is associated with the tree node that is closest to it (yellow lines) if the node is within the influence distance. Then, normalized vectors from the node to the attractors are created and their average (black arrow) is calculated and used as the growth direction for the node (b2). The new node is added in the growth direction in the distance of branch length. All attractors are checked if they are in the kill distances of nodes. In other words, an attractor is killed if it is close enough to the tree (c2). This process iterates until all attractors are killed. While target attractors also pull the branches towards them, they are a special type of attractor with *zero* kill distance. If an attractor is a target attractor, it does not get *killed* until a tree node reaches it (notice difference in d1 and d2). The position of a new node is calculated as follows

$$\vec{v'} = \begin{cases} \vec{t}, & \text{if reaching target} \\ \vec{v} + L\hat{n}, & \text{otherwise} \end{cases} \quad (1)$$

$$\vec{n} = \frac{\vec{a} - \vec{v}}{||\vec{a} - \vec{v}||} \quad (2)$$

$$\hat{n} = \frac{\vec{n}}{||\vec{n}||} \quad (3)$$

$\vec{v'}$, $\vec{v}$, $\vec{t}$, L, $\hat{n}$, $\vec{a}$ are the position vector of a new node, the position vector of node in the tree set, the position vector of the target attractor, branch length, unit growth direction vector and position vector of attractor in the set, respectively.

B. Random Attractor Placement

Placement of random attractors is a crucial step in our algorithm, especially to create variations for the same problem. Since the placement of the attractors defines the virtual design space in which our structure can grow, how attractors are placed in 3D drastically affects the resulting geometry. This effect is demonstrated for three distinct cases in Fig.4 and Fig.5. Figure 4 compares the use of volume and surface attractors to generate an interface structure between the same objects. In the first one, we use the bounding box volume of the two objects to generate the attractors inside of the volume. On the other hand, in the second one, attractors are sampled on the surface of these objects. From this figure, it can be seen that the resulting interface geometries with very distinct characteristics can be obtained by only changing the distribution of the attractors even for the same problem setting. Here, an important distinction between these two cases is that we do not require target attractors for the surface growth case simply because we are guaranteed to touch the surfaces of both objects in this case. Another distinct case for attractor distribution is illustrated in Fig. 5. Here, the aim is to generate an interface structure that would give a desired 2D profile

7

Fig. 6. Obstacle avoidance is used to restrict growth in specific parts in the space. The restricted regions may be the objects represented as octrees or user defined spheres.

Fig. 7. Effect of Laplace and Biharmonic operators on smoothing is illustrated for skeleton (top) and skin (middle-bottom) of the resulting geometries. Left: the original, middle: after smoothing with Laplace & Biharmonic operators together, right: after smoothing with Laplace operator only. Note that the use of the combined Laplace & Biharmonic operators allows smoothing without significant shrinkage.

when viewed from a certain direction. For this purpose, we sample the attractors on a surface that is created by extruding the desired 2D profile in the viewing direction. Hence, this spe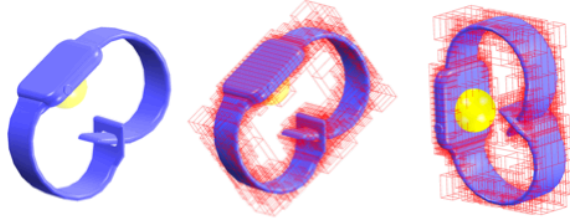cific attractor generation case can be classified as a subset of the surface growth explained previously. Moreover, any 3D swept/curved surface can be used to create 3D profiles. However, the main distinction here is that we require target attractors to be defined in this case, to ensure the resulting structure touches and supports the objects in the problem setup. This is mainly because the surface which the attractors generated on is a virtual one rather than the actual surface of the objects.

Apart from the aforementioned cases, we also enable symmetry in the resulting geometries. In product design, symmetry is considered to be a critical feature for everyday objects [18]. In our shape creation algorithm, we can ensure symmetry by simply placing the attractors in the design space symmetrically. Hence, the addition of a symmetry feature does not add any computational cost in our algorithm. However, the only case that may need special attention is the one where the root point is placed on the symmetry plane. In such cases, growth only happens on the symmetry plane because of the equal attraction from both sides. We solve this problem by moving the user defined root point slightly in both directions orthogonal to the symmetry plane by duplicating the root point.

C. Obstacle Avoidance

When designing an object, its interaction with the environment is important. For this reason, structure growth may be restricted in some parts of the design space. First of all, the interface structure should not intersect with the objects that it is intended to support. For this purpose, we utilize mesh representations of the objects for collision detection. In addition, users may define geometric obstacles in the form of spheres to restrict the growth. As an example for the use of spheres for functional purposes, a sphere is placed under the smart watch to limit the growth of the interface structure on the magnetic touch charging area in Fig 6.

During growth, the intersection of the new branch and obstacles are tested at each step. If there is a collision between the obstacle and the branch, a random direction is chosen for growth until collision is eliminated or maximum number of trials is reached. Intersection tests are conducted using a parametric representation of a line segment and an implicit representation of spherical and triangular objects. Details of the intersection test can be found in [19]. To increase the efficiency of collision detection for triangular meshes, we use octree representation [20].

D. Smoothing

While jagged transitions between biological branches look realistic for trees, smooth transitions are usually more appealing in product design. For this purpose, we apply curve smoothing to the tree structure as shown in Fig.7. Here, the position of each node on the tree skeleton is updated based on the positions of the neighboring nodes using Laplacian and Biharmonic operators as follows [21, 22].

$$\vec{v}_i = \vec{v}_i + \lambda_1 \triangle \vec{v}_i + \lambda_2 \triangle (\triangle \vec{v}_i) \qquad (4)$$

where Laplace and Biharmonic operators can be defined as:

$$\triangle \vec{v}_i = \nabla^2 \vec{v}_i = \frac{1}{2}(\vec{v}_{i+1} - \vec{v}_i) + \frac{1}{2}(\vec{v}_{i-1} - \vec{v}_i) \qquad (5)$$

$$\triangle (\triangle \vec{v}_i) = \nabla^4 \vec{v}_i \qquad (6)$$

$\vec{v}_{i-1}$ and $\vec{v}_{i+1}$ denote two neighbors of the node, $\vec{v}_i$.

In order to achieve smooth curves, we linearly combined Laplace and Biharmonic operators. Although it is possible to accomplish smoothing with only the Laplacian term, Biharmonic term is included to suppress the shrinking behavior arising from the Laplace operator when used alone (Fig.7). In this paper, we select the coefficients λ_1, λ_2 as 0.2 and 0.1, respectively.

E. Branch Pruning

As can be observed from Fig.2 and Fig.3, our approach creates many branches that may not serve a structural function on the interface (i.e., branches that do not touch a target point). Hence, branches that are not connected to the target object or ground are automatically detected and removed form the tree graph (Fig.2(d)-(e)).

F. Modifications and Variation In The Design

Although we produce the interface structures automatically, we enable users to control many aspects of the geometry generation. The user control starts by importing 3D models of the objects and the selection of root-target attractor configurations. Then, another significant control comes from the placement of random attractors as explained previously. In addition to these inputs, there are four factors that affect the growth process (1) influence distance, (2) kill distance, (3) branch length and (4) number of random attractors. These factors are very important to generate variations in the space colonizations algorithm for tree generation such as trees with dense or sparse branch structures [14]. On the other hand, results of our proposed growth algorithm are not affected by the changes in those parameters primarily due to the target attractor and branch removal concepts. As long as the parameters are *suitable*, results do not change significantly. There is a wide range of suitable parameters for a given problem. Any suitable parameter set has the following properties:

- Influence distance is greater than kill distance.
- Branch length, which can be considered as step size, is small compared to the environment dimensions but it is large enough to facilitate efficient growth computation.
- The number of random attractors is high enough to create uniform distribution in the design space, we used 2000 attractors for the examples in the paper.
- Influence distance is high enough to enable attraction of a node for the created uniform distribution.

In this work, we choose default values using the given guidelines. For each problem setup, we use the default values by scaling them with the dimensions of the bounding box of the system.

Another set of important controls comes into play after the interface structure is generated automatically. At this point, users can control the radius variation in the branches of the interface structure as well as modify the skeleton of the structure by sketched strokes. Now that the skeleton of the structure is obtained, a 3D skin is created by covering each branch with a truncated cone and taking the union of all cones. The radius at each node is calculated based on its age as

$$r = r_{min} + (r_{max} - r_{min}) * e^{-k\alpha} \tag{7}$$

where r, r_{min}, r_{max}, α, k is the radius, minimum radius, maximum radius, age and decay of radius, respectively. Here, age of each node is determined in such a way that every node starts with age of 0 and the age increases by 1 at each growth step. Decay of radius defines how fast the radius changes from

(a1) (b1) (c1)

(a2) (b2) (c2)
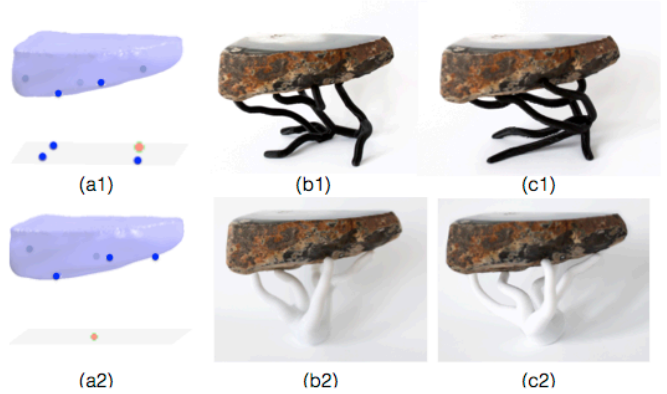
Fig. 8. Effect of target-root selection and stochastic growth demonstrated on two different target-root configurations (a1-c1, a2-c2). For the same problem setup (a), various stochastic growth results (b and c) can be obtained.

the root to the targets between maximum radius and minimum radius and is set by the user. Also, r_{min}, r_{max} are set by the user.

Sketch modifications are performed through modifier sketches performed by the user to specify the new shape of the skeleton curve as it would occur from the current viewpoint. To do this, a surface is created by the rays emanating from the user's eyes, passing through the strokes and extends into the page. In theory, there are infinitely many candidate solutions on this surface. The best 3D configuration is thus found by computing the minimum distance projection of the original curve onto the surface. For the details of the sketch modifications please refer to [23].

IV. RESULTS AND DISCUSSION

Our approach enables the automatic generation of interface structures for 3D printing. The user may control the geometry generation through target-root configurations, random attractor placement, skin radius selection and sketch modifications. We applied our algorithm to a variety of examples including gadget accessories, decoration and restoration of existing objects and furniture. In order to transform the existing objects into the digital design environment, we utilized 3D scanning using a Kinect device. We downloaded the 3D digital models through the stock 3D model websites like GrabCAD and Google 3D Warehouse for the common objects.

The latest trends in decorating and modern furniture design include hybrid design approaches where natural materials with imperfections are combined with machine-made parts to create innovative and original designs. In Figure 8, an example hybrid design created using our system is shown. Here, we take a natural rock piece and design a support structure that complements its organic geometrical features. One important control that our system provides is the target-root placement. We generated two different target-root configurations (a1, a2) for the same problem to demonstrate the significant variance in the resulting geometries (b1, b2). Moreover, we would like to draw attention to the stochastic nature of our algorithm

Fig. 9. Use of sketch modifications for a functional purpose. Top part of the planter holder is enlarged to be able to insert the planter. Left: Sketch modification steps, Right: 3D printed result with the inserted planter.

Fig. 10. Form completion: A broken vase is restored using a scanned model. Left: problem setup, middle: digital model, right: 3D printed result.

that comes from the random sampling of attractors inside the design space where different results are obtained for distinct set of random attractors. However, the effect of the stochastic nature on the resulting geometries (b1-c1 and b2-c2) are subtle compared to the effect of target-root selection.

Another direction for craft, arts and design is the restoration of broken objects through 3D printing to obtain new artistic expressions rather than restoring the original object [24]. In that, the motivation is not to restore the initial function of the object, but rather use it to function as a memorial. In Fig.10, we show that our method can be used for similar purposes. Here, a missing part of a broken vase is restored with the generated interface shape. For this, we first scanned the broken vase and placed desired target-root points. Then, the resulting piece to complete the broken part is grown using our algorithm. We also 3D printed and assembled the resulting part to the broken vase. Another alternative interface structure for this example can also be seen in Fig.12.a.

In addition to the aesthetic needs, the need for sketch modifications may arise from functional requirements. Use of sketch modifications for a functional purpose is illustrated in Fig.9. In this example, a hanger is designed to suspend the planter. However, the planter can not be inserted into this automatically generated structure. For this reason, sketch modifications are applied on the skeleton of the structure to enlarge the top part of the hanger so that the planter can be inserted.

In some configurations, users might need to control the growth process more strictly to achieve a geometry with particular desired properties. In such cases, our geometry creation process can be guided by progressively manipulating the problem setup. To do this, instead of defining all target points at once, we start with a subset of targets and progressively add the remaining ones as we grow the structure. Figure 11 demonstrates this on an example to attach the phone to a baseball cap for first person view camera shots. Here, the
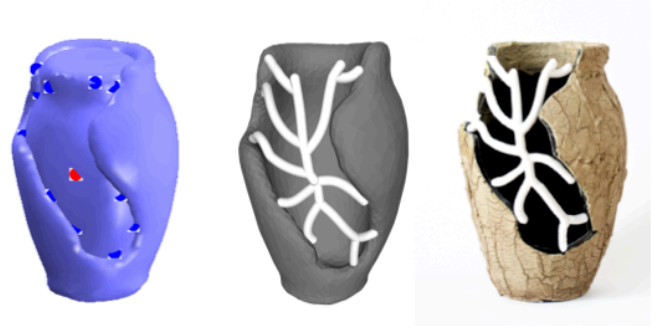
aim is to guide the growth on the side of the cap instead of any other possible outcome. First, only five of the targets are defined (a) and the growth process is completed (b). Then, a new target point is added (b) and another growth process is accomplished. This process is iterated (c) until the final desired shape is created (d). Since we are using a consumer level 3D printer with a limited build volume, we partitioned the resulting object into smaller pieces to be able to 3D print. For the assembly, we manually added dovetail structures on the assembly surfaces (f).

There are many communities that promote reuse of materials through community engagement, resource conservation and creativity e.g. Pittsburgh Center for Creative Reuse and Lancaster Creative Reuse. Since our design framework is developed to work with existing objects, users can easily utilize our algorithm for creative reuse purposes. A virtual example of material reuse is shown in Fig. 12.e. Here, the usage of a seat and back from a broken chair to design a new support and legs is demonstrated. In the example, while we have virtual models for the elements to be reused, as mentioned earlier any object can be scanned and used to create the interface structures. Although for the previous examples, we focus on creating attachment structures that hold the object in place without fixing or gluing, this example requires the interface structure to be fixed to the supported objects.

Since our framework is tailored towards non-expert users, we fabricated all our examples with a consumer level low-cost 3D printer, PrintrBot Simple 1405, to study the printability of our results. However, more advanced 3D printers can be used to fabricate resulting geometries with higher qualities using various material options.

We recorded the computation time for automatic shape generation for a number of examples. Since our method has a stochastic nature, computation time changes as the random attractor set changes. Thus, the results are reported for three different random attractor sets for each example in Table I. One reason computation time changes for each example is the change in the complexity of the objects that increases the time for collision checks. Another, important factor is how easy or difficult it is to reach the targets inside the design space.

Fig. 11. Guided progressive growth (top) is shown on a baseball cap example to attach a phone for first person view camera shooting. Use of assembly structures to 3D print larger designs (bottom) have been demonstrated with the zoomed in dovetail joint detail (g).

TABLE I
COMPUTATION PERFORMANCE OF OUR GENERATIVE DESIGN ALGORITHM

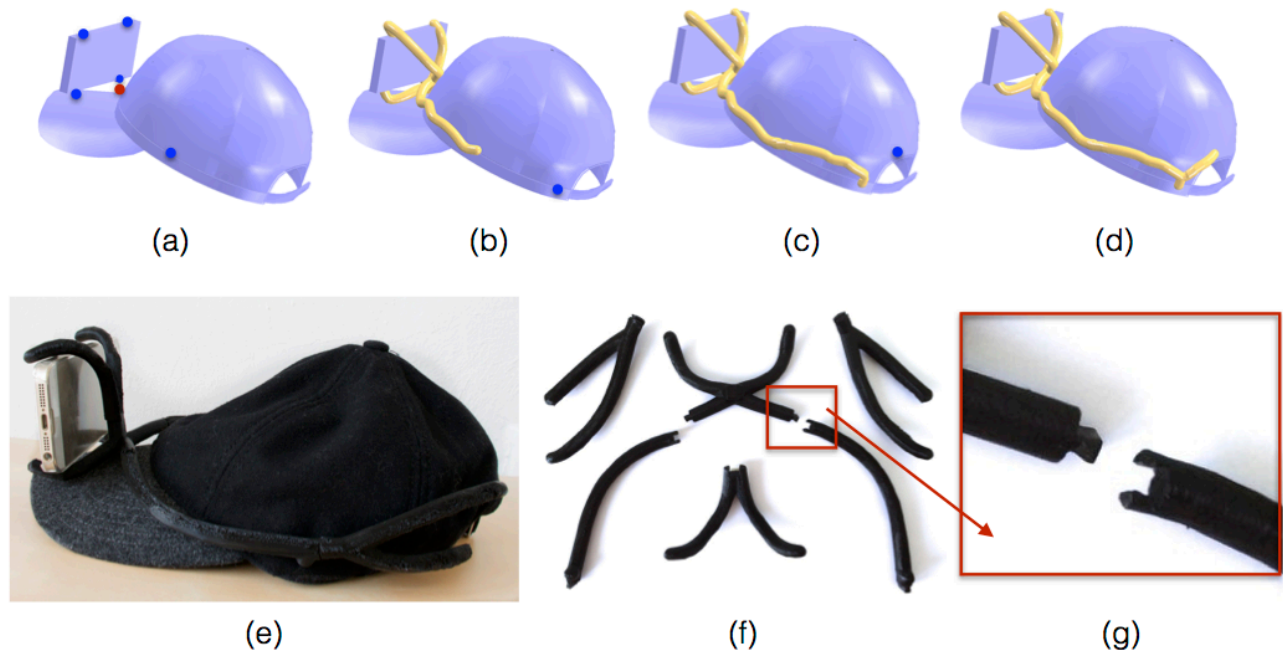|	Total Run Time [s]				
	Run 1	Run 2	Run 3	Run 4	Run 5
Fig.12.a	11	12	10	10	9
Fig.12.b	< 1	< 1	< 1	< 1	< 1
Fig.12.c	< 1	< 1	< 1	< 1	< 1
Fig.12.d	2	2	2	2	2
Fig.12.e	4	2	4	3	2

V. LIMITATIONS AND FUTURE WORK

Our focus has been on generating tree-like skin structures on the skeletons we grow that makes the resulting designs resemble trees. We expect the proposed formulation to be readily applicable with different building blocks instead of our current truncated cones to achieve a richer variation in form. Moreover, since our obstacle avoidance is achieved through random search directions, our algorithm might not converge to a solution within predefined maximum number of trials. While we have observed this issue rarely, increasing maximum trial number for complex problem settings may be required. Finally, in this work, we do not consider structural performance of the resulting shapes. Yet, our algorithm can be extended to ensure structural soundness for a given problem configuration. This may require utilization of finite element analysis during the shape generation process.

VI. CONCLUSION

We present a generative design framework to create interface structures to support existing objects. The proposed method enables novice users to automatically generate geometries and edit as well. Our approach introduces a novel application of a nature inspired growth algorithm with embedded product design considerations. Our current studies indicate that the approach works well for a variety of design problems with the presented actual 3D printed results alongside their digital models.

REFERENCES

[1] C. Mota, "The rise of personal fabrication," in *Proceedings of the 8th ACM Conference on Creativity and Cognition*, ser. C&C '11. New York, NY, USA: ACM, 2011, pp. 279–288. [Online]. Available: http://doi.acm.org/10.1145/2069618.2069665

[2] G. Saul, M. Lau, J. Mitani, and T. Igarashi, "Sketchchair: An all-in-one chair design system for end users," in *Proceedings of the Fifth International Conference on Tangible, Embedded, and Embodied Interaction*, ser. TEI '11. New York, NY, USA: ACM, 2011, pp. 73–80. [Online]. Available: http://doi.acm.org/10.1145/1935701.1935717

[3] N. Umetani, T. Igarashi, and N. J. Mitra, "Guided exploration of physically valid shapes for furniture design," *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2012)*, vol. 31, no. 4, 2012.

[4] N. Umetani, Y. Koyama, R. Schmidt, and T. Igarashi, "Pteromys: Interactive design and optimization of free-formed free-flight model airplanes," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 65:1–65:10, Jul. 2014. [Online]. Available: http://doi.acm.org/10.1145/2601097.2601129

[5] R. Prévost, E. Whiting, S. Lefebvre, and O. Sorkine-Hornung, "Make It Stand: Balancing shapes for 3D fabrication," *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, vol. 32, no. 4, pp. 81:1–81:10, 2013.

[6] M. Bächer, E. Whiting, B. Bickel, and O. Sorkine-Hornung, "Spin-It: Optimizing moment of inertia for spinnable objects," *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, vol. 33, no. 4, 2014.

[7] O. Stava, J. Vanek, B. Benes, N. Carr, and R. Měch, "Stress relief: Improving structural strength of 3d printable objects," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 48:1–48:11, Jul. 2012. [Online]. Available: http://doi.acm.org/10.1145/2185520.2185544

[8] Y. Wang and J. P. Duarte, "Automatic generation and fabrication of designs," *Automation in Construction*, vol. 11, no. 3, pp. 291 – 302, 2002, rapid Prototyping. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0926580500001126

[9] L. Sass and R. Oxman, "Materializing design: the implications of rapid prototyping in digital design," *Design Studies*, vol. 27, no. 3, pp. 325 – 355, 2006, digital Design Digital Design. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0142694X05000864

[10] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool, "Procedural modeling of buildings," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 614–623, Jul. 2006. [Online]. Available: http://doi.acm.org/10.1145/1141911.1141931

[11] Y. I. H. Parish and P. Müller, "Procedural modeling of cities," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '01. New York, NY, USA: ACM, 2001, pp. 301–308. [Online]. Available: http://doi.acm.org/10.1145/383259.383292

[12] P. Prusinkiewicz, M. James, and R. Měch, "Synthetic topiary," in *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '94. New York, NY, USA: ACM, 1994, pp. 351–358. [Online]. Available: http://doi.acm.org/10.1145/192161.192254

[13] A. Runions, M. Fuhrer, B. Lane, P. Federl, A.-G. Rolland-Lagan, and P. Prusinkiewicz, "Modeling and visualization of leaf venation patterns," in *ACM SIGGRAPH 2005 Papers*, ser. SIGGRAPH '05. New York, NY, USA: ACM, 2005, pp. 702–711. [Online]. Available: http://doi.acm.org/10.1145/1186822.1073251

[14] A. Runions, B. Lane, and P. Prusinkiewicz, "Modeling trees with a space colonization algorithm," in *Proceedings of the Third Eurographics Conference on Natural Phenomena*, ser. NPH'07. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2007, pp. 63–70. [Online]. Available: http://dx.doi.org/10.2312/NPH/NPH07/063-070

[15] W. Palubicki, K. Horel, S. Longay, A. Runions, B. Lane, R. Měch, and P. Prusinkiewicz, "Self-organizing tree models for image synthesis," in *ACM SIGGRAPH 2009 Papers*, ser. SIGGRAPH '09. New York, NY, USA: ACM, 2009, pp. 58:1–58:10. [Online]. Available: http://doi.acm.org/10.1145/1576246.1531364

[16] S. Longay, A. Runions, F. Boudon, and P. Prusinkiewicz, "Treesketch: Interactive procedural modeling of trees on a tablet," in *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*, ser. SBIM '12. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2012, pp. 107–120. [Online]. Available: http://dl.acm.org/citation.cfm?id=2331067.2331083

[17] S. Pirk, O. Stava, J. Kratt, M. A. M. Said, B. Neubert, R. Měch, B. Benes, and O. Deussen, "Plastic trees: interactive self-adapting botanical tree models," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 50:1–50:10, Jul. 2012. [Online]. Available: http://doi.acm.org/10.1145/2185520.2185546

[18] B. De Mozota, *Design Management: Using Design to Build Brand Value and Corporate Innovation*. Skyhorse Publishing Company, Incorporated, 2003. [Online]. Available: https://books.google.com/books?id=jpy_JBhZ7nUC

[19] P. Shirley and S. Marschner, *Fundamentals of Computer Graphics*, 3rd ed. Natick, MA, USA: A. K. Peters, Ltd., 2009.

[20] J. Revelles, C. Urea, and M. Lastra, "An efficient parametric algorithm for octree traversal," in *Journal of WSCG*, 2000, pp. 212–219.

[21] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. uno Levy, *Polygon Mesh Processing*. AK Peters, 2010.

[22] E. B. Arisoy, G. Orbay, and L. B. Kara, "Free form surface skinning of 3d curve clouds for conceptual shape design," *Journal of Computing and Information Science in Engineering*, vol. 12, p. 031005, 2012.

[23] L. B. Kara and K. Shimada, "Sketch-based 3d-shape creation for industrial styling design," *IEEE Computer Graphics and Applications*, vol. 27, pp. 60–71, 2007.

[24] A. Zoran and L. Buechley, "Hybrid reassemblage: an exploration of craft, digital fabrication and artifact uniqueness," *Leonardo*, vol. 46, no. 1, pp. 4–10, 2013.
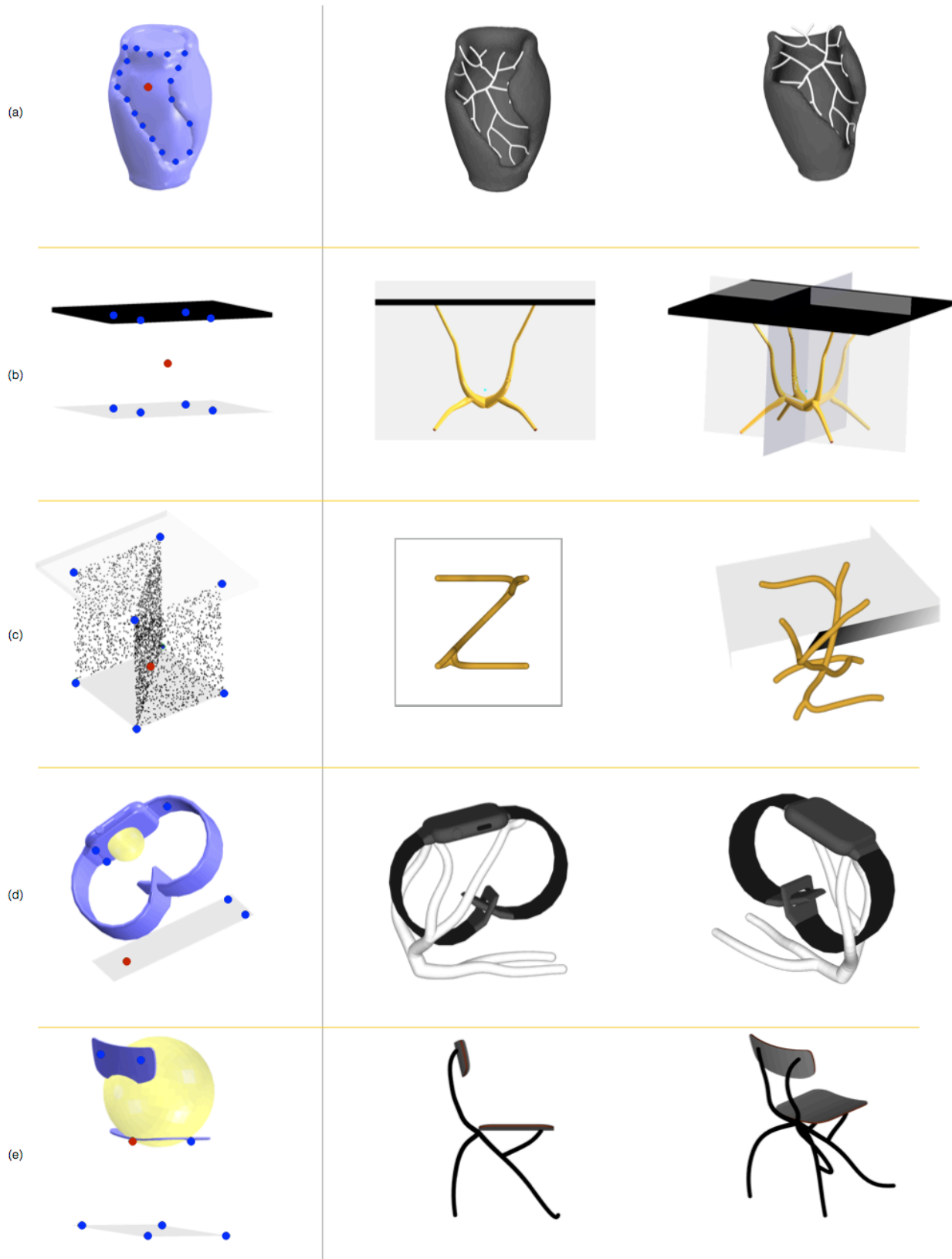
Fig. 12.   Designs created with our system. Left: problem setup, right: resulting design from two different views.

Fast prototyping of visual languages using local context-based specifications

Gennaro Costagliola, Mattia De Rosa, Vittorio Fuccella
Dipartimento di Informatica, University of Salerno
Via Giovanni Paolo II, 84084 Fisciano (SA), Italy
{gencos, matderosa, vfuccella}@unisa.it

Abstract

In this paper we present a framework for the fast prototyping of visual languages exploiting their local context based specification.

In previous research, the local context specification has been used as a weak form of syntactic specification to define when visual sentences are well formed. In this paper we add new features to the local context specification in order to fully specify complex visual languages such as entity-relationships, use case and class diagrams. One of the advantages of this technique is its simplicity of application and, to show this, we present a tool implementing our framework. Moreover, we describe a user study aimed at evaluating the satisfaction and effectiveness of users when prototyping a visual language.

Keywords: *local context, visual languages, visual language syntax specifications.*

1 Introduction

Due to the ever growing evolution of graphical interfaces, visual languages are now a well established form of digital communication in many work and research environments. They are being used extensively by engineers, architects and others whenever there is the need to state and communicate ideas in a standardized way. This is traditionally happening, for example, with software engineering UML graphical notations but it is also catching on in other research fields, such as, for example, synthetic and system biology, [21, 16].

In the 1990s and the early 2000s, the formalization and implementation of visual languages have excited many researchers and many proposals have been defined. In particular, formalisms were defined mainly based on the extension of textual grammar concepts, such as attributed grammars [13, 17, 9, 23] and graph grammars [3, 14], and on meta-modeling [4].

Lately, a new proposal for the specification and interpretation of diagrams from the only syntactic point of view has been given in [7]. This research is motivated by the need to reduce the complexity of the visual language syntax specification and originated while working on the recognition of sketch languages and on the difficulty of recognizing the symbols of the language. In order to disambiguate sketched symbols, the more knowledge is given on each symbol and on its possible interactions with the others, the better. The methodology, known as *local context-based visual language specification* requires the designer to define the *local context* of each symbol of the language. The local context is seen as the interface that a symbol exposes to the rest of the sentence and consists of a set of attributes defining the local constraints that need to be considered for the correct use of the symbol.

At first, this approach is, then, more lexical than syntactic: the idea is to provide a very detailed specification of the symbols of the language in order to reduce complexity when specifying the language at the sentence level. On the other hand, due to the easy-to-use and intuitiveness requirements, many practical visual languages have a simple syntax that can be completely captured by the local context specification as defined in [7]. To show this, the syntax of the binary trees and of a Turing complete subset of flowcharts have been completely specified through local context in [7].

When considered as a syntax specification, however, the simplicity of the approach is counterbalanced by its low expressiveness, especially with respect to the more powerful grammars formalisms.

In this paper, we define new features to be added to the original local context definition in order to push the expressiveness of the methodology and to allow the complete syntactic specification of complex visual languages such as entity-relationships, use case and class diagrams. We present the tool LoCoMoTiVE (Local Context-based Modeling of 2D Visual language Environments) implementing our framework which basically consists of a simple inter-

face allowing a user, in one screen, to define the symbols of the language and their local context. Moreover, we demonstrate the usability of the tool in a user study, in which participants are asked to define and test a visual language after a short introduction to the tool. Besides the participants' ability to define the visual language, we also administered a System Usability Scale (SUS) [5] questionnaire to evaluate their satisfaction with the tool.

The rest of the paper is organized as follows: the next section refers to related work; Section 3 gives the background information on the "local context-based visual language specification", sketches the three new features and presents a complete syntax specification for the use case diagrams; Section 4 is devoted to describe the tool; the experiment is presented in Section 5; some final remarks and a brief discussion on future work conclude the paper.

2 Related Work

In recent years many methods to model a diagram as a sentence of a visual language have been devised. A diagram has been represented either as a set of relations on symbols (the *relation-based approach*) [22] or as a set of attributed symbols with typed attributes representing the "position" of the symbol in the sentence (the *attribute-based approach*) [12]. Even though the two approaches appear to be different, they both consider a diagram (or visual sentence) as a set of symbols and relations among them or, in other words, a spatial-relationship graph [3] in which each node corresponds to a graphical symbol and each edge corresponds to the spatial relationship between the symbols.

Unlike the relation-based approach, where the relations are explicitly represented, in the attribute-based approach the relations must be derived by associating attribute values.

Based on these representations, several formalisms have been proposed to describe the visual language syntax, each associated to ad-hoc scanning and parsing techniques: Relational Grammars [23], Constrained Set Grammars [17], and (Extended) Positional Grammars [10]. (For a more extensive overview, see Marriott and Meyer [18] or Costagliola et al. [8].) In general, such visual grammars are specified by providing an alphabet of graphical symbols with their "physical" appearance, a set of spatial relationships generally defined on symbol position and attachment points/areas, and a set of grammar rules in context-free like format.

A large number of tools exist for prototyping visual languages. These are based on different types of visual grammar formalisms and include, among others, VLDesk [9], DiaGen [19], GenGed [2], Penguin [6], AToM3 [11], and VL-Eli [15].

Despite the context-free like rule format, visual grammars are not easy to define and read. This may explain why there has not been much success in transferring these techniques from research labs into real-world applications. We observe that several visual languages in use today are simple languages that focus on basic components and their expressive power. These languages do not need to be described with complex grammar rules. Our approach provides a simpler specification for many of them, making it easier to define and quickly prototype visual languages.
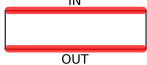
3 Local Context Specification of Visual Languages

According the to the attribute-based representation, a visual sentence is given by a set of symbols defined by a set of typed attaching points linked through connectors. In [7], the local context specification of a visual language consists in the definition of the sets of graphical elements (named *symbols*) and spatial relations (named *connectors*) composing the language and, for each of them, their attributes. These are:

- Name of the graphical symbol;
- Graphical aspect (for example, specified through an svg-like format);
- Maximum mumber of occurrences of the symbol in a sentence of the language;
- Attachment areas: an attachment area is a set of points (possibly one) through which symbols and connectors can be connected. For each area the following attributes have been identifed:
  - Name of the attachment area;
  - Position: the location of the attachment area on the symbol or connector;
  - Type: an attachment area of a symbol can be connected to an an attachment area of a connector only if they have the same type;
  - Local constraints: such as the max number of possible connections for an attachment area.

As a further and sentence level constraint, a local-context based specification may assume that the underlying spatial-relationship graph of any sentence of the specified language is connected.

As an example, Table 1 shows the attributes of the statement symbol STAT and the connector ARROW when considered as alphabetical elements of a particular set of flowcharts. The specification states that STAT has the graphical aspect of a rectangle; it has two attaching areas named IN and OUT corresponding to the upper and lower sides of the rectangle, respectively; it may occur zero or more times in a flowchart; the attachment area IN can only be connected to an attachment area of a connector with type *enter*, while the attachment area OUT can only be connected to an attachment area of a connector with type *exit*. Moreover, IN may participate in one or more connections, while OUT may participate in only one connection. As re-

Symbol name	Graphics	Symbol occurrences	Attachment areas name	type	constraints
STAT	IN / OUT	≥ 0	IN	enter	$connectNum \geq 1$
			OUT	exit	$connectNum = 1$

Connector name	Graphics	Attachment areas name	type	constraints
ARROW	TAIL●———▶HEAD	$HEAD$	enter	$connectNum = 1$
		$TAIL$	exit	$connectNum = 1$

Table 1: Attribute specification for the symbol STAT and the connector ARROW.

gards the connector ARROW, its graphical appearance is given by a line with a head and the attachment areas are located to the head (HEAD) and tail (TAIL) of the arrow itself. An arrow can be connected to only one symbol through its head and only one symbol through its tail. In [7], complete local context specifications for a particular set of Turing complete flowcharts and for binary trees are given to show how local context can be used to fully specify the syntax of visual languages.

3.1 New Local Context Features

In order to capture as much as possible of the syntax of complex languages other than flowcharts and binary trees and to keep simplicity, new local features need to be added to the original definition of local context. In particular, we introduce the possibility of

- defining *symbol level* constraints involving more than one attaching area of a symbol/connector as opposed to constraints on individual attaching areas;
- assigning multiple types to attaching areas;
- defining constraints to limit connector self loops.

These features allow us to give complete local context specifications of complex languages such as the entity relationship diagrams, class diagrams, and use case diagrams. In the following, we show the practical usefulness of this extension by referring to the local context specifications of the use case diagrams and the entity-relationship diagrams.

Symbol level constraints. Table 2 shows the binary version of the *relation* symbol of the well-known entity-relationship (E-R) graphical notation. Each vertex of the symbol has one attaching point (Up, Down, Left or Right) of type *enter*. In order to force its use as an E-R binary relation (as opposed to ternary) the constraints need to set the sum of all their connections to two, apart from requiring that the number of connections to each attaching point be at most one. In this case, the feature simplifies the specification by avoiding that a designer define all the possible ways a binary relation symbol can be attached to the other symbols.

Multiple types and Connector self loop constraints. Table 3 shows the complete specification of the use case graphical notation, while Figure 1 shows some examples of correct and incorrect sentences. In the table, the language symbols and connectors are specified in the first and second part while, for sake of completeness, the last row declares, if present, any requirement at sentence level. It can be noted that the attachment area *Border* of the symbol ACTOR (first table row) has two types GenA and AssOut. By considering the Connector part of the table, this means that an ACTOR can be connected through its border to both the head and tail of the GENERALIZATION_A connector (through GenA) and also to the attaching point *P1:P2* of the connector ASSOCIATION (through AssOut). Moreover, because of the constraint $numLoop = 0$, a GENERALIZATION_A connector cannot be connected to the border of an ACTOR with its head and tail, simultaneously.

In Figure 1, case (b) shows the correct use of connector GENERALIZATION_Uc, while case (d) shows the correct use of connector GENERALIZATION_A.

The use of these new features allow the language designer more flexibility in the definition of the language. However, multiple types must be carefully used when dealing with connectors with the same graphical aspect since they may introduce ambiguities in the language.

It is not difficult to see that Table 3 completely specifies the syntax of the use case graphical notation as presented in *http://agilemodeling.com/artifacts/useCaseDiagram.htm* but without the optional "System boundary boxes".

With respect to a grammar definition, the new specification is basically distributed on the language elements instead of being centralized.

As a final note, the selection of which language elements are symbols and which are connectors is completely left to the language designer. Moreover, connectors may not have a graphical representation (such as the relationships "touching", "to the left of").
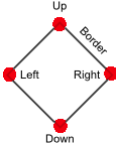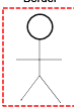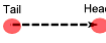
Symbol name	Graphics	Symbol occurrences	Attachment points name	type	constraints
BIN_REL		≥ 0	Up	ConA	$connectNum(X) \leq 1$ for $X = Up, Down, Left, Right \land (connectNum(Up) + connectNum(Down) + connectNum(Left) + connectNum(Right) = 2) \land (connectNum(Border) \geq 0)$
			Left	ConA	
			Right	ConA	
			Down	ConA	
			Border	ConB	

Table 2: ER binary relation specification.

Symbol name	Graphics	Symbol occurrences	Attachment points name	type	constraints
ACTOR		≥ 1	Border	GenA	$connectNum \geq 0 \land numLoop = 0$
				AssOut	$connectNum \geq 0$
USE_CASE		≥ 1	Border	GenUc	$connectNum \geq 0 \land numLoop = 0$
				AssIn	$connectNum \geq 0$
				Dep	$connectNum \geq 0 \land numLoop = 0$

Connector	Graphics	Attachment points name	type	constraints
ASSOCIATION		P1:P2	AssOut	$connectNum = 1$
		P2:P1	AssIn	$connectNum = 1$
GENERALIZATION_A		Head	GenA	$connectNum = 1$
		Tail	GenA	$connectNum = 1$
GENERALIZATION_UC		Head	GenUc	$connectNum = 1$
		Tail	GenUc	$connectNum = 1$
DEPENDENCY		Head	Dep	$connectNum = 1$
		Tail	Dep	$connectNum = 1$

Non local constraint
the spatial-relationship graph must be connected

Table 3: Use case diagrams language specifications.

4   The tool LoCoMoTiVE

The current implementation of the local context methodology includes the new set of constraints defined in the previous section and is composed of two different modules:

- LoCoModeler: the local context-based specification editor, and
- TiVe: a web-based visual language environment for editing and checking the correctness of the visual sentences.

4.1   LoCoModeler

The LoCoModeler module allows designers to create and edit visual language specifications based on local con-

text, quickly and easily. Its output is the formal definition in XML format of the language that will be used during the disambiguation and the recognition of diagrams. Once the language designer has completed the specification, s/he can compile it into a web-based environment (the TiVE module) to allow users to draw sentences and verify their correctness. During language definition, this feature also allows the designer to check the correctness of the specification.

The main view of the LoCoModeller GUI is shown in Figure 2. Its main components are:

- A textbox containing the name of the language and a checkbox to enable/disable the option that diagrams must necessarily be connected;
- A table reporting the main information of symbols and connectors included in the language. It is possible to

(a) (incorrect)   (b) (correct)   (c) (incorrect)

(d) (correct)   (e) (incorrect)

Figure 1: Simple instances of syntactically correct and incorrect use case diagrams.

(a) Language name selection

(b) Symbol selection

(c) Connector selection

Figure 3: Wizard.

interact with the widgets in the selected row to edit and/or delete it. The user can add new symbols or connectors by using the buttons below the table.

- A panel (on the right) showing a graphical preview of the symbol/connector selected in the table. It is possible to change the graphical representation of the symbol by using the button *Change*.
- A table (in the center) showing the information related to the selected symbol/connector. Each row specifies the attachment points and their constraints. It is possible to add new rows by using the buttons above the table;
- A textarea to specify the symbol/connector level constraints through C language-like expressions.

4.1.1 Wizard

A new language can be defined by using a wizard interface. Through a sequence of three different views, the user chooses the name of the language, its symbols and connectors (see Figure 3). Symbols and connectors are chosen from a hierarchical repository and their definition already includes some attachment points having default types/constraints. The user can choose whether to keep the default values or modify them.

4.2 TiVE: the visual language environment

Once the language is defined, the diagrams can be composed by using the symbols and the connectors defined in its specification. This can be done through the graphical editor TiVE, which is a web application enabling diagram composition directly in the web browser and which is created by and can also be launched from the LoCoModeler.

Figure 4 shows the environment. The central component is the working area where the diagrams are composed. The symbols and connectors used for diagram composition are displayed in the sidebar, which contains only those elements included in the definition of the language. An element can be selected and dragged in the central working area.

The upper toolbar provides shortcuts to features such as zoom manipulation, changing fonts, checking diagram correctness, etc.

The correctness of a diagram can be checked at any point of the diagram composition. The diagram in Figure 5 represents an ER diagram with two entities interconnected by
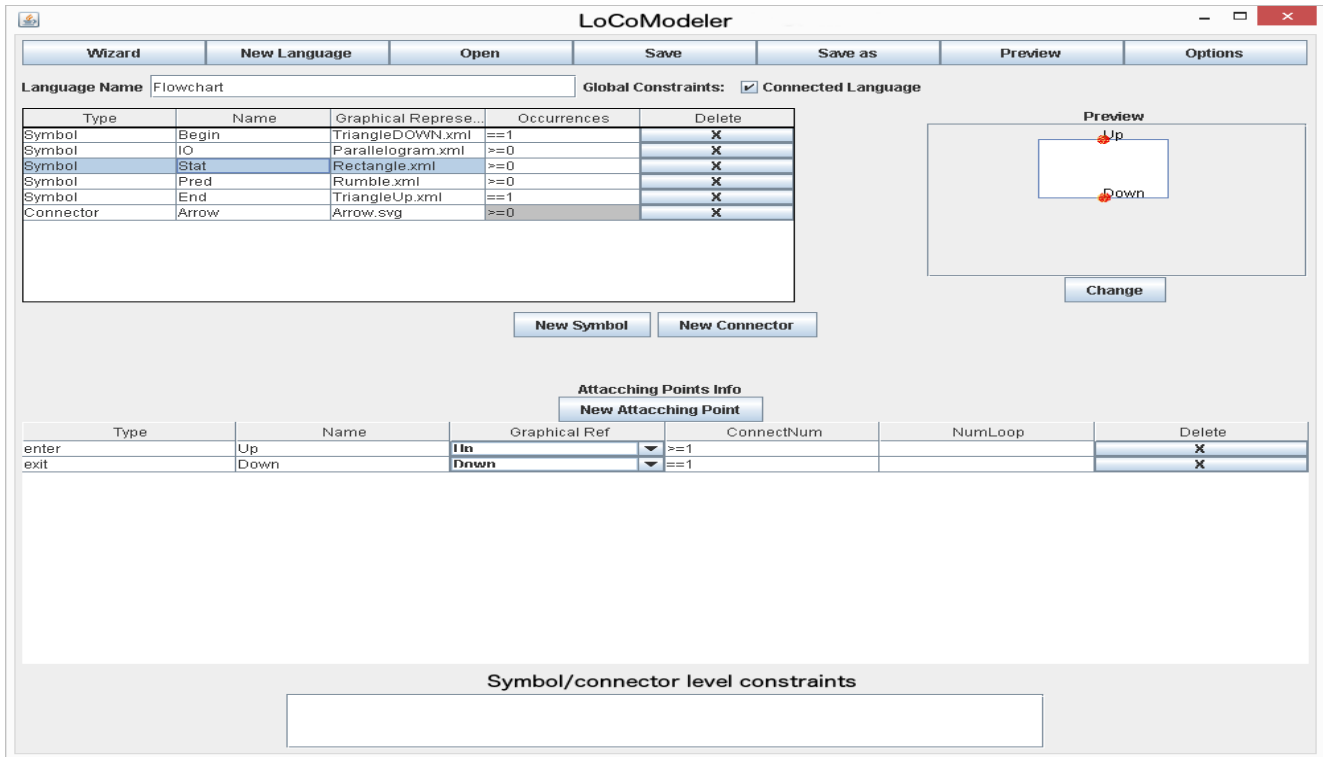
Figure 2: The Local Context-based Modeler.

a binary relation. The diagram is correct and, by launching the check, an alert reports the positive result of the verification. The diagram in Figure 6, instead, represents an incorrect ER diagram, as the relation (the diamond symbol) is connected to a single entity (the rectangle). This violates the local constraints defined for the relation symbol in the language. In fact, in this language, relations must be connected to two or three entities through the diamond's vertices.

4.3 Implementation

The LoCoModeler allows the user to produce the language specification in XML format. The specification is used during the removal of ambiguities and the recognition of symbols and connectors.

TiVE is based on Draw.io (https://www.jgraph.com), which is a free web application that allows users to create charts directly from the browser and integrates with Google Drive and Dropbox to store data. Draw.io is in turn based on the mxGraph library, which renders the diagram and stores its structure in the form of a graph, where symbols and connectors are its vertices. We modified the library to handle attaching points of symbols and connectors.

In a typical thin-client environment, mxGraph is divided into a client-side JavaScript library and a server side library in one of the two supported languages: .NET and Java. The
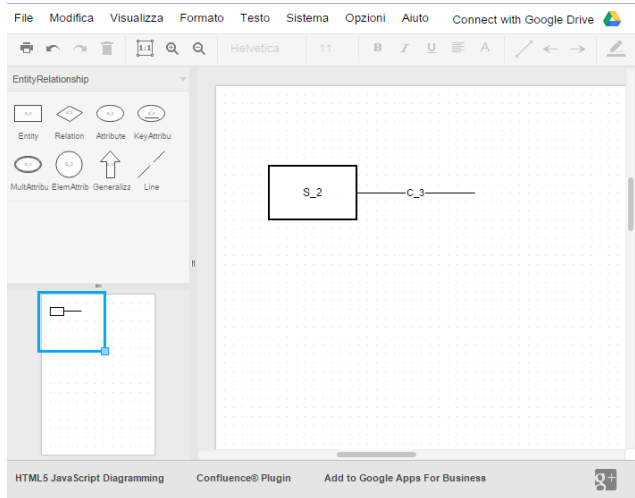

Figure 4: The TiVE home page.

JavaScript library is a part of a larger web application. The JavaScript code uses vector graphics to render the chart. The languages used are SVG (Scalable Vector Graphics) for standard browsers and VML (Vector Markup Language) for Internet Explorer.

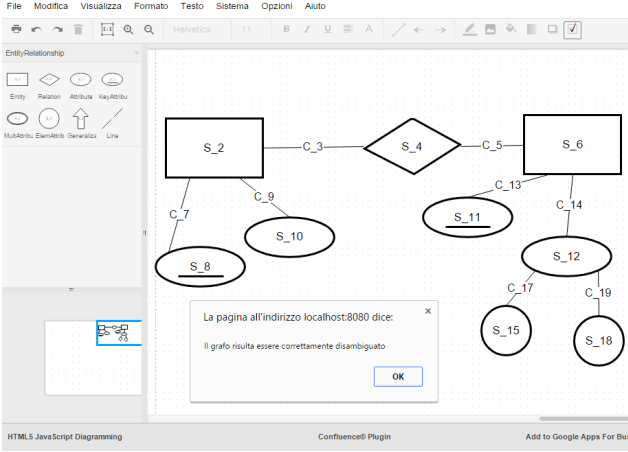An implementation of the tool can be downloaded at the address http://weblab.di.unisa.it/locomotive.

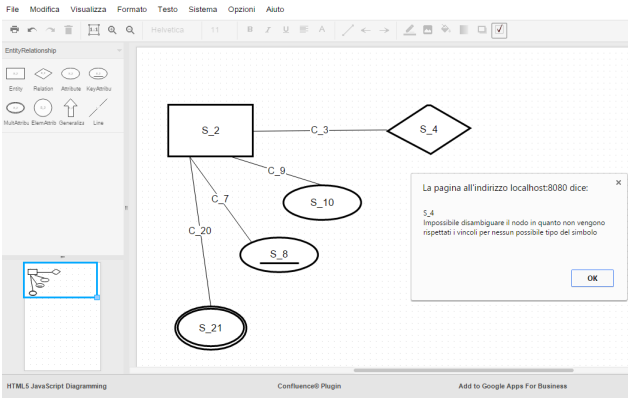Figure 5: Successful diagram verification (no error found).

Figure 6: Failed diagram verification (one error found).

5 Evaluation

We ran a user study aimed at measuring users' capacity to define visual languages using our tool. Furthermore, we recorded the perceived usability of the system through a questionnaire.

5.1 Participants

Our participants were six male and four female Italian university students in computer science (six master students and four phd students), aged between 22 and 45 ($M = 26.8$, $SD = 6.9$), with no previous experience with the system.

Participants were asked to evaluate, with a 5-point Likert scale, their prior knowledge in programming, diagrams, compilers, formal languages, flowchart and other visual languages. The average and standard deviations of the responses are reported in Table 4.

Knowledge	Avg.	St. dev.
Programming	4.40	0.70
Diagrams	3.40	0.84
Compilers	2.90	1.10
Formal languages	3.40	1.07
Flowchart	3.40	0.97
Other visual languages	2.80	1.14

Table 4: Participants prior knowledge evaluation with a 5-point Likert scale.
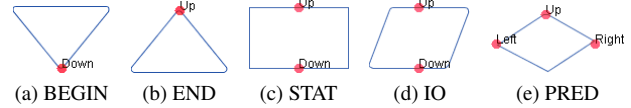
Figure 7: Flowchart symbols.

5.2 Apparatus

The experiment was executed on a *Dell Precision T5400* workstation equipped with an *Intel Xeon* CPU at 2.50 GHz running *Microsoft Windows 8.1* operating system, the *Java Run-Time Environment 7*, and the Firefox browser.

5.3 Procedure

Participants were asked, as a single task of the session, to define a visual language. In particular, they were asked to create a simplified version of flowcharts, as defined in [7], with the following features:

- The language only includes a small set of blocks (start, end, I/O, decision, processing - shown in Figure 7) and an arrow as a connector;
- The handling of text within blocks or arrows is not required;
- An arrow is always directed at the top of a block and comes out from its bottom;

Participants were asked to specify all the constraints necessary to ensure that only well formed flowcharts would pass the correctness check. Furthermore, they were required to carefully check they had defined the language correctly before submitting the task. The time limit for task completion was half an hour.

Before the experimental session, each participant had a brief tutorial phase where an operator (one of the authors) explained him/her the purpose and operation of the system and instructed him/her about the experimental procedure and the task. While showing the operation of the system, the operator also showed participants how to use our tool to define a simple visual language, in this case a simplified version of ER diagrams.

A post-test questionnaire in the form of System Usability Scale (SUS) [5] was administered to participants at the end of the experiment. SUS is composed of 10 statements to which participants assign a score indicating their strength of agreement in a 5-point scale. The final SUS score ranges from 0 to 100. Higher scores indicate better perceived usability. We also gathered some participants' freeform comments.

5.4 Results

Two participants out of ten completed the experiment defining the language perfectly, seven completed the experiment with minor inaccuracies in the language definition, while only one of them completed the experiment with major inaccuracies. Here, for minor inaccuracies, we mean small errors that allow user to compose at least one invalid diagram which however satisfied the user's language specification. Typical errors are inaccuracies in defining attachment points cardinality. The participant who committed major errors was unable to compose and correctly compile any diagram. The average task completion time was 25.5 minutes.

The responses given by participants to the statements in the SUS questionnaire are reported in Table 5. In particular the responses to statements 1, 3, 5, 7 and 9 show that participants appreciate the system, that they considered it simple to use and easy to learn even for non-experts of visual languages. Moreover the responses to questions 2, 4, 6, 8 and 10 show that participants did not feel they need support to use the system and did not found the system complex, intricate or inconsistent.

The scores of the questionnaire calculated on the responses of the participants range from 37.5 to 95, with an average value of 80.0, which value indicates a good level of satisfaction [1]. As it can be seen from the data in the table, only a single participant (the one who committed major errors) expressed a negative judgment on the tool.

In addition, participants provided some freeform suggestions for improving the system: most of the criticism was expressed on the editor for diagram composition tool, which was not felt to be very user-friendly. In particular, participants noticed that some basic operations for diagram composition, such as the insertion of connectors, are surprisingly uncomfortable. Furthermore, one participant pointed out that the editor is not well integrated with the VLDE.

6 Conclusions

In this paper we have presented a framework for the fast prototyping of visual languages exploiting their local context based specification. We have shown how to define a visual language by extending the local context with three new features and have presented a simple interface for its implementation LoCoMoTiVE. Moreover, we have described a user study for evaluating the satisfaction and effectiveness of users when prototyping a visual language. At the moment, the user study has been limited to the simpler version of the local context methodology in order to provide us with a first feedback. Given the encouraging results, we are now planning to test the usability of LoCoMoTive with more complex applications.

The local context approach may then greatly help visual language designers to prototype their languages very easily. However, more studies are needed to investigate the computational borders of the approach. Our intention is not too push local context features more than needed, keeping simplicity as a priority. More complex language constructs should then be left to the following phases of the recognition process as it is the case for programming language compiler construction.

As a final goal, we are working on the integration of the local context approach in frameworks for the recognition of hand drawn sketches, as shown in [7].

References

[1] A. Bangor, P. T. Kortum, and J. T. Miller. An Empirical Evaluation of the System Usability Scale. *International Journal of Human-Computer Interaction*, 24(6):574–594, 2008.

[2] R. Bardohl. Genged: a generic graphical editor for visual languages based on algebraic graph grammars. In *Visual Languages, 1998. Proceedings. 1998 IEEE Symposium on*, pages 48–55, Sep 1998.

[3] R. Bardohl, M. Minas, G. Taentzer, and A. Schürr. Handbook of graph grammars and computing by graph transformation. chapter Application of Graph Transformation to Visual Languages, pages 105–180. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1999.

[4] P. Bottoni and G. Costagliola. On the definition of visual languages and their editors. In M. Hegarty, B. Meyer, and N. Narayanan, editors, *Diagrammatic Representation and Inference*, volume 2317 of *Lecture Notes in Computer Science*, pages 305–319. Springer Berlin Heidelberg, 2002.

[5] J. Brooke. Sus: A quick and dirty usability scale. In P. W. Jordan, B. Weerdmeester, A. Thomas, and I. L. Mclelland, editors, *Usability evaluation in industry*. Taylor and Francis, London, 1996.

[6] S. S. Chok and K. Marriott. Automatic construction of intelligent diagram editors. In *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology*, UIST '98, pages 185–194, New York, NY, USA, 1998. ACM.

[7] G. Costagliola, M. De Rosa, and V. Fuccella. Local context-based recognition of sketched diagrams. *Journal of Visual Languages & Computing*, 25(6):955–962, 2014. Distributed Multimedia Systems {DMS2014} Part I.

	Question	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	Avg. resp.	St. dev.
S1	I think that if I needed to define a visual language I would use this system	4	5	4	4	4	3	5	3	5	5	4.2	0.79
S2	I found the system unnecessary complex	1	1	1	2	4	2	2	4	1	2	2.0	1.15
S3	I found the system very easy to use	5	5	5	4	4	4	4	2	5	4	4.2	0.92
S4	I think I would need the support of a person who is already able to use the system	2	1	1	1	1	2	2	4	1	1	1.6	0.97
S5	I found the various system functions well integrated	4	5	4	3	3	4	4	3	5	3	3.8	0.79
S6	I found inconsistencies between the various system functions	2	1	1	2	1	1	1	2	1	2	1.4	0.52
S7	I think most people can easily learn to use the system	5	4	5	4	5	4	5	3	5	4	4.4	0.70
S8	I found the system very intricate to use	2	1	1	1	2	2	2	4	1	2	1.8	0.92
S9	I have gained much confidence about the system during use	4	4	4	5	5	4	3	2	4	5	4.0	0.94
S10	I needed to perform many tasks before being able to make the best use of the system	1	1	2	1	1	1	3	4	2	2	1.8	1.03
Score		85	95	90	82.5	80	77.5	77.5	37.5	95	80	80	16.33

Table 5: SUS questionnaire results (5-point Likert scale).

[8] G. Costagliola, V. Deufemia, and G. Polese. A framework for modeling and implementing visual notations with applications to software engineering. *ACM Trans. Softw. Eng. Methodol.*, 13(4):431–487, Oct. 2004.

[9] G. Costagliola, V. Deufemia, and G. Polese. Visual language implementation through standard compiler–compiler techniques. *Journal of Visual Languages & Computing*, 18(2):165 – 226, 2007.

[10] G. Costagliola and G. Polese. Extended positional grammars. In *Proc. of VL '00*, pages 103–110, 2000.

[11] J. de Lara and H. Vangheluwe. Atom3: A tool for multi-formalism and meta-modelling. In R.-D. Kutsche and H. Weber, editors, *Fundamental Approaches to Software Engineering*, volume 2306 of *Lecture Notes in Computer Science*, pages 174–188. Springer Berlin Heidelberg, 2002.

[12] E. J. Golin. Parsing visual languages with picture layout grammars. *J. Vis. Lang. Comput.*, 2(4):371–393, Dec. 1991.

[13] E. J. Golin and S. P. Reiss. The specification of visual language syntax. *J. Vis. Lang. Comput.*, 1(2):141–157, June 1990.

[14] R. J. and A. Schurr. Defining and parsing visual languages with layered graph grammars. *Journal of Visual Languages & Computing*, 8.1:27–55, 1997.

[15] U. Kastens and C. Schmidt. Vl-eli: A generator for visual languages - system demonstration. *Electr. Notes Theor. Comput. Sci.*, 65(3):139–143, 2002.

[16] N. Le Novere et al. The systems biology graphical notation. *Nature Biotechnology*, 27:735–741, 2009.

[17] K. Marriott. Parsing visual languages with constraint multiset grammars. In M. Hermenegildo and S. Swierstra, editors, *Programming Languages: Implementations, Logics and Programs*, volume 982 of *Lecture Notes in Computer Science*, pages 24–25. Springer Berlin Heidelberg, 1995.

[18] K. Marriott and B. Meyer. On the classification of visual languages by grammar hierarchies. *Journal of Visual Languages & Computing*, 8(4):375 – 402, 1997.

[19] M. Minas and G. Viehstaedt. Diagen: A generator for diagram editors providing direct manipulation and execution of diagrams. In *Proceedings of the 11th International IEEE Symposium on Visual Languages*, VL '95, pages 203–, Washington, DC, USA, 1995. IEEE Computer Society.

[20] I. Plauska and R. Damaševičius. Design of visual language syntax for robot programming domain. *Information and Software Technologies Communications in Computer and Information Science*, 403:297–309, 2013.

[21] J. Quinn et al. Synthetic biology open language visual (sbol visual), version 1.0.0, 2013. http://sbolstandard.org/downloads/specification-sbol-visual/ [Online; accessed 4-June-2015].

[22] J. Rekers and A. Schurr. A graph based framework for the implementation of visual environments. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 148–155, Sep 1996.

[23] L. Weitzman and K. Wittenburg. Relational grammars for interactive design. In *Visual Languages, 1993., Proceedings 1993 IEEE Symposium on*, pages 4–11, Aug 1993.

Incremental indexing of objects in pictorial databases

G. Castellano, A.M. Fanelli, M.A. Torsello
Computer Science Department
University of Bari A. Moro
Via Orabona, 4 - 70126 Bari, Italy
(giovanna.castellano, annamaria.fanelli, mariaalessandra.torsello)@uniba.it

Abstract

Object indexing is a challenging task that enables the retrieval of relevant images in pictorial databases. In this paper, we present an incremental indexing approach of picture objects based on clustering of object shapes. A semi-supervised fuzzy clustering algorithm is used to group similar objects into a number of clusters by exploiting a-priori knowledge expressed as a set of pre-labeled objects. Each cluster is represented by a prototype that is manually labeled and used to annotate objects. To capture eventual updates that may occur in the pictorial database, the previously discovered prototypes are added as pre-labeled objects to the current shape set before clustering. The proposed incremental approach is evaluated on a benchmark image dataset, which is divided into chunks to simulate the progressive availability of picture objects during time.

1. Introduction

The extensive use of image digital capturing systems in several fields has generated massive amount of digital images that are typically collected in pictorial databases [1]. Most of the past projects on pictorial databases focus on content-based approaches searching images that are visually similar to the query image [2]. Such approaches do not have the capability of assigning textual descriptions automatically to pictures, i.e. they do not perform linguistic indexing.

Linguistic indexing is a difficult task due to the semantic gap problem, i.e. the lack of coincidence among the visual content of images represented by automatically extracted features and the human visual interpretation of the picture content typically expressed by high-level concepts [3]. Learning concepts from images and automatically translating the content of images to linguistic terms can bridge the semantic gap thus resulting in one of the most influential factors in successful image retrieval [4], [5] consequently

broadening the possible usages of pictorial databases.

Different machine-learning methods have been applied to learn associations between the low-level features and the linguistic concepts in a pictorial database [15]. In particular, learning techniques can be used to annotate objects clearly identifiable by linguistic cues. A common approach is to perform classification on the collection of picture objects [6], [14], so that visually similar objects are grouped into the same class and a textual label is associated to each class. Thus, each object is indexed by classifying it into one of the identified classes.

Classification of picture objects can be performed by means of supervised or unsupervised learning methods. Supervised techniques require a lot of training data, and providing these data is a very tedious and error-prone task, especially for large image database. Unsupervised learning techniques overcome these limitations but often they generate inconsistent classes including objects that, although having a similar shape, actually represent different linguistic cues. The presence of objects with ambiguous shapes motivates the use of semi-supervised clustering algorithms that can improve classification by using a combination of both labeled and unlabeled data. In [11] we proposed the use of a semi-supervised clustering algorithm called SSFCM (Semi-Supervised Fuzzy C-Means) to create object classes and prototypes useful for indexing images in a database. However, when new images are added to the database, this static indexing scheme requires rebuilding the prototypes starting from scratch by reprocessing the whole set of objects, i.e. it does not take advantage of the previously created prototypes.

To overcome this limitation, in this paper we propose the use of an incremental version of the SSFCM clustering algorithm, that we call Incremental SSFCM (ISSFCM). The ISSFCM applies SSFCM to chunks of picture objects that are periodically added to the database, thus providing an incremental scheme for picture object indexing.

The paper is organized as follows. Section 2 describes the proposed indexing scheme for pictorial object annota-

tion. In section 3 we provide some preliminary simulation results on a benchmark data set containing picture objects of different shapes. Finally, section 4 concludes the paper.

2. Incremental scheme for object indexing

We assume that a collection of pictorial objects is available. Each object is described by the contour of its shape. Different shape descriptors could be used to represent object shapes. In this work, each object shape is represented by means of Fourier descriptors that are well-recognized to provide robustness and invariance, obtaining good effectiveness in shape-based indexing and retrieval [8]. The shape of each pictorial object is described by means of M Fourier descriptors and denoted by a numerical vector $\mathbf{x} = (x_1, x_2, ..., x_M)$.

The proposed scheme for incremental indexing of objects is based on the assumption that sets of object shapes belonging to different semantic classes are available during time and processed as chunks, that is, a chunk of N_1 object shapes is available at time t_1, a chunk of N_2 shapes is available at t_2 and so on. We denote by X_t the chunk of picture objects available at time t. For a correct application of the proposed incremental scheme, all semantic classes should be represented in the early chunks. The chunks of objects are processed as they are added to the database, by applying incrementally the Semi-Supervised FCM (SSFCM) algorithm [11] described in section 2.1. The resulting scheme, called ISSFCM (Incremental SSFCM), is shown in fig. 1. It enables the update of previously derived prototypes when new shapes are continuously available over time. Each time a new chunk of shapes is available, previously created cluster prototypes are used as pre-labeled shapes for the new run of SSFCM. At the end of each SSFCM run, the derived labeled prototypes are used to index all available shapes accumulated in the pictorial database.

The overall scheme of the proposed incremental indexing approach is summarized in algorithm 1. Each time a chunk is available, it is clustered by SSFCM and the resulting clusters are represented by K prototypes that are manually annotated by textual labels (step 4-7). Then each object is added to the cluster corresponding to the best matching prototype and labeled with the related label (step 8-9). Matching is based on computing Euclidean distance between the Fourier descriptors of the object and the descriptors of prototypes. We chose the Euclidean distance since it is one of the most popular distances in literature that permits to obtain accurate results when matching shapes represented by Fourier descriptors with a low-cost and simple computation [16]. To take into account the evolution of the database, the prototypes discovered from one chunk are added as pre-labeled objects to the next chunk (step 10, step 4).

Precisely, when the first chunk of pictorial objects is
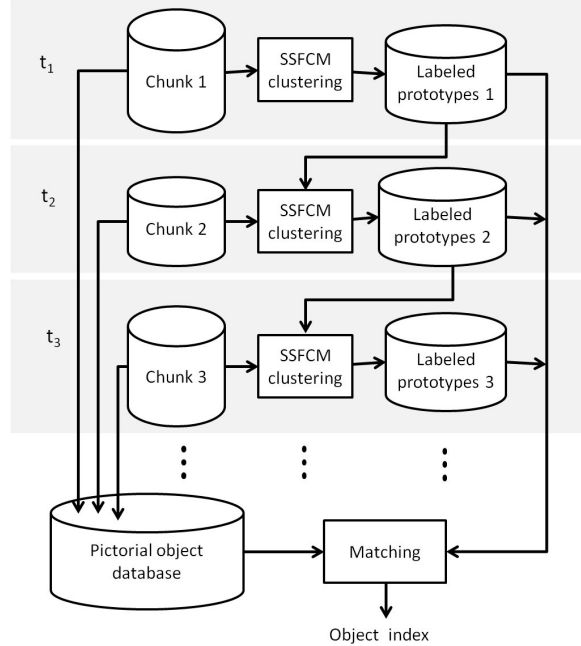
Figure 1. The scheme of the incremental indexing approach

available, the algorithm will cluster the chunk into K clusters and it will derive a set of K object prototypes that are manually labeled. When a second or later chunk of objects is available, it will be clustered with the labeled prototypes derived from the previous clustered chunks[1].

Summarizing, our incremental indexing scheme generates a structure of clusters on the basis of chunks which capture the availability of new picture objects during time and reflect physical evolution of the database. The indexing mechanism is incremental in the sense that the cluster prototypes derived from one chunk are used not only for current indexing but also as a starting point for the clustering of successive chunks. The derived prototypes offer an intermediate indexing mechanism that enables automatic linguistic indexing of pictorial objects by requiring manual annotation of a very limited number of objects (namely the prototypes).

2.1. Clustering by SSFCM

The SSFCM algorithm works in the same manner as FCM (Fuzzy C-Means) [9], i.e. it iteratively derives K clusters by minimizing an objective function. Unlike FCM, that performs a completely unsupervised clustering, SSFCM performs a semi-supervised clustering, i.e. it uses

---

[1]How many chunks of history to use for clustering with a new chunk is predefined by the user.

Algorithm 1 Incremental SSFCM (ISSFCM)

Require: Chunks of unlabeled objects $X_1, X_2, ...$

Ensure: P: set of labeled prototypes; X: set of annotated objects

1: $H \leftarrow \emptyset$ /* Initialization of history */
2: $t \leftarrow 1$ /* Initialization of time step */
3: **while** $\exists$ non empty chunk X_t **do**
4: $\quad X_t \leftarrow X_t \cup H$ /* Add history to current chunk */
5: $\quad$ Cluster X_t using SSFCM
6: $\quad$ Derive the set P of prototypes
7: $\quad$ Annotate manually each prototype in P
8: $\quad$ Annotate each object in $\bigcup_{\tau=1}^{t} X_\tau$ using the best-matching prototype in P
9: $\quad$ Update X with annotated objects
10: $\quad$ Update H with P
11: $\quad t := t + 1$
12: **end while**
13: **return** P, X

---

a set of pre-labeled data to improve clustering results. To embed partial supervision in the clustering process, the objective function of SSFCM includes a supervised learning component, as follows:

$$J = \sum_{k=1}^{K}\sum_{j=1}^{N_t} u_{jk}^m d_{jk}^2 + \alpha \sum_{k=1}^{K}\sum_{j=1}^{N_t}(u_{jk} - b_j f_{jk})^m d_{jk}^2 \quad (1)$$

where

$$b_j = \begin{cases} 1 & \text{if object } \mathbf{x}_j \text{ is pre-labeled} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

f_{jk} denotes the true membership value of the pre-labeled object $\mathbf{x}_j$ to the cluster k, d_{jk} represents the Euclidean distance between the object shape $\mathbf{x}_j$ and the center of the k-th cluster, m is the fuzzification coefficient ($m \geq 2$) and α is a parameter that serves as a weight to balance the supervised and unsupervised components of the objective function. The higher the value of α, the higher the impact coming from the supervised component is. The second term of J captures the difference between the true membership f_{jk} and the membership u_{jk} computed by the algorithm. The aim to be reached is that, for the pre-labeled objects, these values should coincide.

As described in [12], the problem of optimizing the objective function J is converted into the form of unconstrained minimization using the standard technique of Lagrange multipliers. By setting the fuzzification coefficient m equal to 2, the objective function is minimized by updating membership values u_{jk} according to:

$$u_{jk} = \frac{1}{1+\alpha}\left[\frac{1 + \alpha(1 - b_j \sum_{l=1}^{K} f_{lk})}{\sum_{l=1}^{K} d_{jk}^2/d_{lk}^2}\right] + \alpha b_j f_{jk} \quad (3)$$

and the centers of clusters according to:

$$\mathbf{c}_k = \frac{\sum_{j=1}^{N_t} u_{jk}^m \mathbf{x}_j}{\sum_{j=1}^{N} u_{jk}^m} \quad (4)$$

The clustering process ends when the difference between the values of J in two consecutive iterations drops below a prefixed threshold or when the established maximum number of iterations is achieved.

Once the clustering process is completed, a prototype is identified for each cluster by selecting the object shape belonging with the highest membership to that cluster. Then, each prototype is manually associated to a label corresponding to a specific linguistic cue or semantic class.

Summarizing, the result of SSFCM applied to each chunk is a set of K labeled prototypes $P = \{p_1, p_2, ..., p_K\}$ that are used to index objects in the database. Namely, all objects belonging to cluster k are associated with the text label assigned to prototype p_k.

3. Experimental results

To assess the suitability of the proposed incremental indexing approach, we considered the MPEG-7 Core Experiment CE-Shape-1 data set [8] containing 1400 binary images of object shapes grouped into 70 different classes with each class including 20 samples. Fig. 2 shows a sample image for each class of the considered data set. In order to apply ISSFCM, all images were processed to extract boundaries of shapes and compute Fourier descriptors. Each object shape was represented by a vector of 32 Fourier coefficients (this number was set in our previous experiments on the same dataset).

To evaluate the clustering results we used the average purity error, as in [10], defined as follows:

$$pur = 1 - \frac{1}{K} \times \sum_{k=1}^{K} \frac{|C_k^d|}{|C_k|}$$

where K denotes the number of clusters, $|C_k^d|$ denotes the number of objects with the dominant class label in cluster k and $|C_k|$ denotes the total number of objects in cluster k. Intuitively, the purity error measures the purity of the clusters with respect to the true cluster (class) labels that are known for the MPEG-7 dataset.

We performed a suite of experiments in order to analyze the behavior of ISSFCM when varying the percentage p of pre-labeled shapes ($p = 20\%$ and $p = 30\%$) and following two different pre-labeling schema:

- scheme A: we assume that each chunk contains a percentage p of pre-labeled shapes;
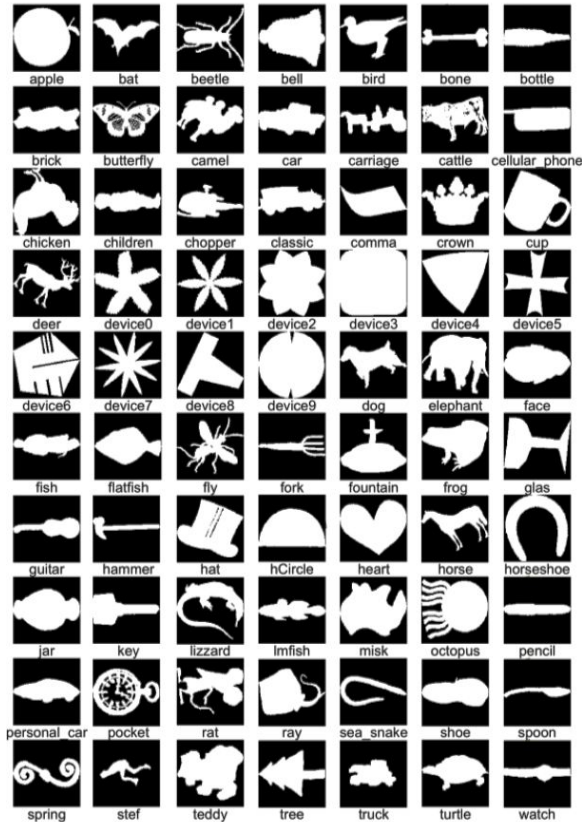
Figure 2. Sample images from the MPEG-7 Core Experiment CE-Shape-1 data set

Figure 3. Prototypes derived in each time step for three semantic classes

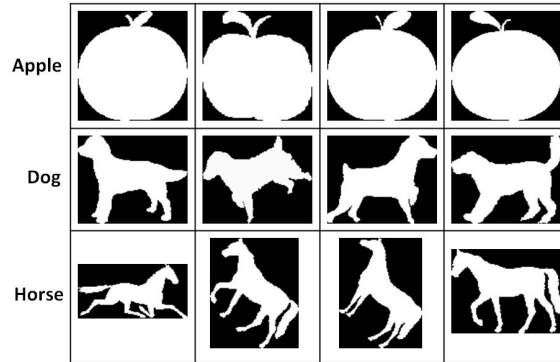**Table 1. Average purity error values**

| scheme | percentage of pre-labeled shapes | |
	20%	30%
A	0.29	0.18
B	0.28	0.17

• scheme B: we assume that only the first chunk contains a a percentage p of pre-labeled shapes; in the next chunks the previously derived prototypes represent the pre-labeled shapes.

In all the experiments, the parameters of the ISSFCM algorithm were set as follows: the number of cluster K was set to the number of classes in the dataset (i.e. $K = 70$), the size of a chunk was set to 280 shapes (hence 5 chunks were built from the whole dataset), the history was set to 1, meaning that only the prototypes extracted from the previous chunk were considered as pre-labeled shapes in the current chunk. Since SSFCM is not deterministic (due to the random initialization of the cluster centers) 10 different runs were performed and the average results are presented.

At the first time step, the SSFCM was applied to the union of the first two chunks in order to obtain more stable and significant initial prototypes to be exploited in the next steps of the incremental clustering process. In this way 4 different time steps were simulated. After clustering a chunk, 70 prototypes were derived and each prototype was manually annotated by a label descriptive of a semantic class. These prototypes were used to annotate all shapes included in the previous chunks on the basis of a top-matching score. To perform matching we computed the Euclidean distance between descriptors of each shape and descriptors of each prototype. Each shape was annotated with the label of the best-matching prototype. As an example, in fig. 3, we show the prototypes derived for three semantic classes at the end of each time step by applying ISSFCM with the 30% of pre-labeled shapes following scheme B.

The annotation results were evaluated by computing the average purity error. Table 1 reports the average values of the purity error obtained by varying the percentage of pre-labeled shapes and the pre-labeling scheme. It can be seen that, as expected, when the percentage of pre-labeled shapes increases, the quality of the obtained clusters improves. Regardless the pre-labeling percentage, the two pre-labeling schema provide comparable values of the purity error.

The effectiveness of the proposed incremental approach was evaluated by comparing the average purity error obtained in the last step of ISSFCM and the average purity error obtained by applying the SSFCM algorithm in a one-shot way (following the experimental setting described in [11]). To apply the SSFCM in one-shot way the data set was divided into a training set (composed of the shapes included in the first 4 chunks) and a test set (including the 280 remaining shapes). The training set was used to derive the shape prototypes whilst the test set was used to perform annotation by exploiting the derived prototypes. Figure 4a compares the average purity error obtained by applying ISS-

FCM (varying the pre-labeling scheme and the percentage of pre-labeled shapes) and the one-shot SSFCM. We observe that ISSFCM obtains results that are comparable to those obtained by the static one-shot SSFCM with the additional advantage to exploit and update the knowledge discovered in the previous time steps. Finally, we evaluated the annotation accuracy in terms of Precision and Recall and we compared the results obtained by applying the incremental SSFCM the static SSFCM. Figures 4b and 4c show the comparative values of precision and recall, respectively. It can be seen that the incremental indexing approach achieves better annotation accuracy with respect the static one-shot approach thus confirming the benefit of exploiting previously acquired knowledge whenever new picture objects have to be added to the pictorial database.

4. Conclusions

In this paper an incremental scheme for pictorial object indexing has been proposed. The approach exploits a semi-supervised fuzzy clustering algorithm to derive a set of prototypes representative of a number of semantic categories. The derived prototypes are manually annotated by attaching labels related to semantic categories. The use of shape prototypes, which represent an intermediate level of visual signatures, facilitates the annotation process, since only a reduced number of objects need to be manually annotated. Moreover, the use of prototypes simplifies the search process in a pictorial database by reducing time needed to retrieve similar shapes. Indeed, a query is matched only with shape prototypes, thus avoiding unnecessary comparisons with all objects in the database. Annotation results on the MPEG-7 benchmark dataset show that our incremental scheme obtains results which are very similar to those obtained by the one-shot approach with the additional advantage to exploit the previously discovered prototypes thus avoiding the reprocessing of the whole database. These preliminary results encourage the application of the proposed approach to real-world contexts requiring the indexing of evolving collections of pictorial objects.

References

[1] S.K. Chang and T.L. Kunii. Pictorial data-base systems, IEEE 23. G. Strang, Linear Algebra and Its Applications. Harcourt, Brace, and Computer 14, pp. 13-21, 1981.

[2] Y. Rui, T. Huang, and S. Chang, Image retrieval: current techniques, promising directions and open issues, J. Visual Commun. Image R. 10(4):39-62, 1999.

[3] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, Content-based image retrieval at the end
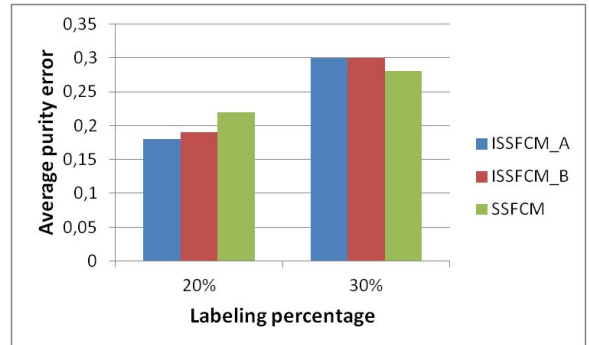
(a)
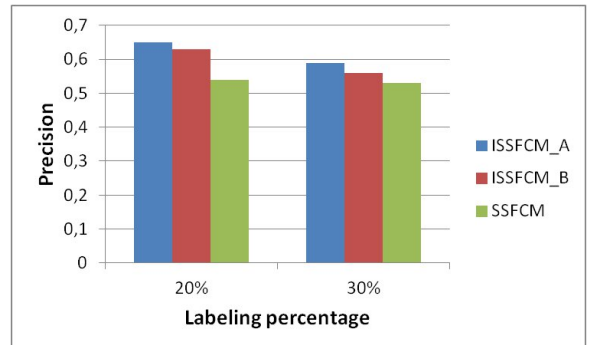
(b)
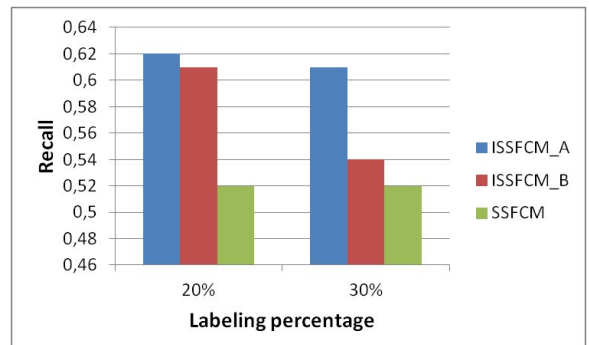
(c)

Figure 4. Comparison between Incremental SSFCM and one-shot SSFCM

of the early years, IEEE Trans. Pattern Analysis and Machine Intelligence. 22:1349-1380, 2000.

[4] W.I. Grosky, and R. Mehrotra. Index-based object recognition in pictorial data management. Computer Vision, Graphics, and Image Processing, 52(3):416-436, 1990.

[5] J. Li, and J.Z. Wang. Automatic linguistic indexing of pictures by a statistical modeling approach. IEEE Transactions on Pattern Analysis and Machine Intelligence 25(9):1075-1088, 2003.

[6] H. Nezamabadi-Pour, and S. Saryazdi. Object-based image indexing and retrieval in DCT domain using clustering techniques. In Proc. of World Academy of Science Engineering and Technology, pp. 207-210, 2005.

[7] S. Aghabozorgi, M.R. Saybani, and T.Y. Wah. Incremental clustering of time-series by fuzzy clustering. Journal of Information Science and Engineering 28(4):671-688, 2012.

[8] I. Bartolini, P. Ciaccia, and M. Patella. WARP: Accurate retrieval of shapes using phase of Fourier descriptors and Time warping distance. IEEE Trans. on Pattern Analysis and machine Intelligence, 27(1): 142-147, 2005.

[9] J.C. Bezdek, *Pattern recognition with fuzzy objective function algorithms*, Plenum Press, New York, 1981.

[10] F. Cao, M. Ester, W. Qian and A. Zhou. Density-based clustering over an evolving data stream with noise. In 2006 SIAM Conference on Data Mining, pp. 328-339, 2006.

[11] G. Castellano, A.M. Fanelli and M.A. Torsello. Shape annotation by semi-supervised fuzzy clustering. Information Sciences, 289(24):148-161, 2014.

[12] W. Pedrycz and J. Waletzky. Fuzzy clustering with partial supervision. IEEE Transactions on System Man and Cybernetics, 27(5): 787-795, 1997.

[13] S. Guha, A. Meyerson, N. Mishra, R. Motwani and L. O'Callaghan. Clustering data streams: Theory and practice, IEEE Trans. on Knowledge and Data Engineering, 15(3):515-528, 2003.

[14] D. Stan, and I.K. Sethi. Mapping low-level image features to semantic concepts. In Proc. of the SPIE, pp. 172-179, 2001.

[15] D. Zhang, M.M. Islam, and G. Lu, A review on automatic image annotation techniques, Pattern Recogn. 45(1):346-362, 2011.

[16] D. Zhang and G. Lu, Shape-based image retrieval using generic Fourier descriptor, Signal Processing: Image Communication, 17(10):825-848, 2002.

28

RankFrag: A Machine Learning-Based Technique for Finding Corners in Hand-Drawn Digital Curves

Gennaro Costagliola*, Mattia De Rosa*, Vittorio Fortino†, Vittorio Fuccella*

*Dipartimento di Informatica, University of Salerno, Via Giovanni Paolo II, 84084 Fisciano (SA), Italy
†Unit of Systems Toxicology, Finnish Institute of Occupational Health (FIOH), Helsinki, Finland
{gencos, matderosa, vfuccella}@unisa.it, vittorio.fortino@ttl.fi

Abstract

We describe RankFrag: a technique which uses machine learning to detect corner points in hand-drawn digital curves. RankFrag classifies the stroke points by iteratively extracting them from a list of corner candidates. The points extracted in the last iterations are said to have a higher rank and are more likely to be corners. The technique has been tested on three different datasets described in the literature. We observed that, considering both accuracy and efficiency, RankFrag performs better than other state-of-art techniques.

Keywords: *corner finding, stroke segmentation, fragmentation, sketch recognition, machine learning, RankFrag*

1 Introduction

The research on hand-drawn sketch recognition has had a recent boost due to the diffusion of devices (smartphones and tablets) equipped with touch screens. Sketched diagrams recognition raises a number of issues and challenges, including both low-level stroke processing and high-level diagram interpretation [11]. A low-level problem is the *segmentation* (also known as *fragmentation*) of input strokes. Its objective is the recognition of the graphical primitives (such as lines and arcs) composing the strokes. Stroke segmentation can be used for a variety of objectives, including symbol [16, 4] and full diagram [3] recognition.

Most approaches for segmentation use algorithms for finding corners, since these points represent the most noticeable discontinuity in the graphical strokes. Some other approaches [1] also find the so called *tangent vertices* (smooth points separating a straight line from a curve or parting two curves). Besides stroke segmentation, the identification of corners has other applications, including gesture recognition [9] and gestural text entry [6, 5].

A high accuracy and the possibility of being performed

in real time are crucial features for segmentation techniques. Tumen and Sezgin [26] also emphasize the importance of the adaptation to user preferences and drawing style and to the particular domain of application. Adaptation can be achieved by using machine learning-based techniques. Machine learning has also proven to improve accuracy. In fact, almost all of the most recent segmentation methods use some machine learning-based technique.

The technique presented here, which we call *RankFrag*, uses machine learning to decide if a candidate point is a corner. Our technique is inspired by previous work. In particular, the work that mostly influenced our research is that of Ouyang and Davis [20], which introduced a *cost* function expressing the likelihood that a candidate point is a corner. We adopt their cost function, but our corner finding procedure is different. The technique works by iteratively removing points from a list of candidate corners. At each iteration, the point minimizing the cost function is classified and, in the case it is not a corner, it is removed. As a point is removed from the list, it is assigned a *rank*, which is a progressively decreased integer value. Points with a higher rank (a lower integer value) are more likely to be corners. Another important characteristic of RankFrag is the use of a variable "region of support" for the calculation of some local features, which is the neighborhood of the point on which the features are calculated. Most of the features used for classification are taken from several previous works in the literature [23, 15, 27, 21, 20]. Four novel features are introduced.

We tested our technique on three different datasets previously introduced and already used in the literature to evaluate existing techniques. We compared the performance of RankFrag to other state-of-art techniques [28, 26].

Summarizing, this paper introduces and evaluates:

1. a novel iterative procedure for finding corners in digital curves;

2. the use of four previously untested features for corner classification.

The rest of the paper is organized as follows: the next section contains a brief survey on the main approaches for

sketch segmentation; in Section 3 we describe our technique; Section 4 presents the evaluation of the performance of our technique in comparison to those of existing techniques, while the results are reported in Section 5; lastly, some final remarks and a brief discussion on future work conclude the paper.

2 Related Work

According to a widely accepted classification [24], the methods for corner detection in digital curves can be divided in two categories: those that perform a classification of the points and those that compute a piecewise approximation of the curves.

The former methods evaluate some features on the points of the stroke, after they have possibly been resampled, e.g. at a uniform distance. Curvature and speed are the features that have been used first. In particular, the corners are identified by looking at maxima in the curvature function or at minima in the speed function. Lately, methods based on machine learning have begun to consider a broader range of features.

One of the first methods proposed in the literature is [24], which assesses the curvature through three different measures. The authors also propose an advanced method for the determination of the region of support for local features. One of the first methods based on the simple detection of speed minima is [14]. Given the inaccuracy of curvature and speed taken individually, it was decided to evaluate them both in combination: [22] uses a hybrid fit by combining the set of candidate vertices derived from curvature data with the candidate set from speed data.

A method introducing a feature different from curvature and speed is *ShortStraw* [27]. It uses the *straw* of a point, which is the segment connecting the endpoints of a window of points centered on the considered point. The method gave good results in detecting corners in polylines by selecting the points having a straw of length less than a certain threshold. Subsequently, the method has been extended by Xiong and LaViola [28] to work also on strokes containing curves.

One of the first methods to use machine learning for corner finding is the one described in [20]. It is used to segment the shapes in diagrams of chemistry. A very recent one is *ClassySeg* [13], which works with generic sets of strokes. The method firstly detects candidate segment windows containing curvature maxima and their neighboring points. Then, it uses a classifier trained on 17 different features computed for the points in each candidate window to decide if it contains a corner point.

The approaches for computing a piecewise approximation of digital curves try to fit lines and curves sequentially in a stroke; the dominant points then correspond to the intersections of adjacent substrokes. The problem of finding the optimal subset of the n points of the stroke has an exponential complexity. Nevertheless, almost all the algorithms that implement this approach use dynamic programming to reduce the exponential runtime complexity to $O(n^2)$. The first work [2] dates back to 1961. This algorithm fixes the number of segments and finds the solution minimizing the error. An algorithm proposed later [8] fixes the error and minimizes the number of segments. The algorithms also differ for the norm they use to measure the approximation error. A recent method, called DPFrag [26] learns primitive-level models from data, in order to adapt fragmentation to specific datasets and to user preferences and sketching style.

Lastly, there are hybrid methods, which use both the approaches mentioned above. *SpeedSeg* [12] and *TCVD* [1] are examples of such methods. *TCVD* is also able to find both the corners and the points where there is a significant change in curvature (referred to as "tangent vertices" in [1]). In order to detect corners, the former method mainly relies on pen speed while the latter uses a curvature measure. Tangent vertices are found through piecewise approximation by both methods.

3 The Technique

Our technique segments an input stroke in primitives by breaking it in the points regarded as corners. As a preliminary step, a *Gaussian smoothing* [10] is executed on the raw points in order to reduce the resampled stroke noise. Then, the stroke is processed by resampling its points to obtain an equally spaced ordered sequence of points $P = (p_1, p_2, \ldots, p_n)$, where n varies depending on a fixed space interval and on the length of the stroke.

In order to identify the corners, the following three steps are then executed:

1. Initialization;

2. Pruning;

3. Point classification.

The initialization step creates a set D containing n pairs (i, c), for $i = 1 \ldots n$ where c is the *(initial) cost* of p_i and is calculated through Eq. 1 derived, through some simplification steps, from the *cost* function defined in [20].

$$Icost(p_i) = \begin{cases} [dist(p_i; p_{i-1}, p_{i+1})]^2 & \text{if } i \in \{2, \ldots, n-1\} \\ +\infty & \text{if } i = 1 \text{ or } i = n \end{cases} \quad (1)$$

In the above equation, the term $dist(p_i; p_{i-1}, p_{i+1})$ indicates the minimum distance between p_i and the line segment formed by (p_{i-1}, p_{i+1}). Since p_1 and p_n do not have a preceding and successive point, respectively, they are treated as special cases and given the highest cost.

The pruning step iteratively removes $n-u$ elements from D in order to make the technique more efficient. The value u is the number of candidate corners not pruned in this step and depends on the complexity of the strokes in the target dataset. Its value has no effect on the accuracy of the method, provided that it is conservatively chosen so that no corner is eliminated in the pruning step. However, too high a value for this parameter may affect its efficiency.

At each iteration, the element m with the lowest cost is removed from D and the costs of the closest preceding points p_{pre} in P and the closest successive point p_{suc} in P of p_m, with pre and suc occurring in the set $\{i : (i,c) \in D\}$, are updated through Eq. 2 derived from the *cost* function defined in [20].

$$Cost(p_i) = \begin{cases} \sqrt{mse(S; p_{ipre}, p_{isuc})} \times dist(p_i; p_{ipre}, p_{isuc}) \\ \qquad\qquad\qquad\qquad \text{if } i \in \{2, \ldots, n-1\} \\ +\infty \qquad\qquad\qquad\quad\; \text{if } i = 1 \text{ or } i = n \end{cases} \quad (2)$$

In the above equation,

- the points p_{ipre} and p_{isuc} are, respectively, the closest preceding and successive points of p_i in P, with $ipre$ and $isuc$ occurring in the set $\{i : (i,c) \in D\}$;

- $S = \{p_{ipre}, \ldots, p_{isuc}\}$ is the subset of points between p_{ipre} and p_{isuc} in the resampled stroke P;

- $mse(S; p_{ipre}, p_{isuc})$ is the mean squared error between the set S and the line segment formed by (p_{ipre}, p_{isuc});

- the function *dist* is defined as for Eq. 1.

The point classification step returns the list of points recognized as corners by further removing from D all the pairs with indices of the points that are not recognized as corners. This is achieved by the following steps:

1. find the current element in D with minimum cost (if D contains only pairs with indices 1 and n, return an empty list);

2. calculate the features of the point corresponding to the current element and invoke the binary classifier, previously trained with data.

    - if the classifier returns false, delete the element from D, make the necessary updates and go to 1.

    - if the classifier returns true, proceed to consider as current the next element in D in ascending cost order. If the corresponding point is one of the endpoints of the stroke, return the list of points corresponding to the remaining elements in D (except for 1 and $|P|$), otherwise go to 2.

In Fig. 1, the function DETECTCORNERS() shows the pseudocode for the initialization, pruning and point classification steps. In the pseudocode, D is the above described set with the following functions:

- INIT(L) initialize D with all the (i,c) pairs contained in L;

- FINDMINC() returns the element of D with the lowest cost;

- PREVIOUSI(i) returns j such that (j,c') is the closest preceding element of (i,c) in D, i.e., $j = \max\{k \mid (k,c) \in D \text{ and } k < i\}$;

- SUCCESSIVEI(i) returns j such that (j,c') is the closest successive element of (i,c) in D, i.e., $j = \min\{k \mid (k,c) \in D \text{ and } k > i\}$;

- SUCCESSIVEC(i) returns the successive element of (i,c) in D with respect to the ascending cost order;

- REMOVE(i) removes (i,c) from D;

- UPDATECOST(i, c) updates the cost c' to c for (i,c') in D.

DETECTCORNERS() calls a CLASSIFIER(i, P, D) function that computes the features (described in Section 3.2) of the point $P[i]$, and then uses them to determine if $P[i]$ is a corner by using a binary classifier previously trained with data (described in Section 3.3).

3.1 Complexity

The complexity of the function DETECTCORNERS() in the previous section depends on the implementation of the data structure D. We will base our calculation by implementing D with an array and a pointer: the ith element of the array refers to the node that contains the pair (i,c) (or nil if the node does not exist) while the pointer refers to the node with the minimum c. Each node has 3 pointers: one that points to the successive node in ascending c order, one that points to the successive node in ascending i order and one that points to the previous node in ascending i order. Based on this implementation, the FINDMINC(), PREVIOUSI(), SUCCESSIVEI(), SUCCESSIVEC() and REMOVE() functions are all executed in constant time, while the UPDATECOST() function is $O(|D|)$ (where $|D|$ is the number of nodes referred in D) and the INIT(L) function is $O(|L| \log |L|)$ (by using an efficient sorting algorithm). In the following we will show that the DETECTCORNERS() complexity is $O(n^2)$, where $n = |P|$.

It is trivial to see that: the complexity of the ICOST() function is $O(1)$; the complexity of COST() is $O(n)$ in the

Input: an array P of equally spaced points that approximate a stroke, a number u of not-to-be-pruned points, and the CLASSIFIER() function.

Output: a list of detected corners.

```
 1: function DETECTCORNERS(P, u, CLASSIFIER)
 2:     # initialization
 3:     for i = 1 to |P| do
 4:         c ← ICOST(i, P)            # computes Eq. 1
 5:         add (i, c) to TempList
 6:     end for
 7:     D.INIT(TempList)

 8:     # pruning
 9:     while |D| > u do
10:         (i_min, c) ← D.FINDMINC()
11:         REMOVEANDUPDATE(i_min, P, D)
12:     end while

13:     # point classification
14:     while |D| > 2 do
15:         (i_cur, c) ← D.FINDMINC()
16:         loop
17:             isCorner ← CLASSIFIER(i_cur, P, D)
18:             if isCorner then
19:                 (i_cur, c) ← D.SUCCESSIVEC(i_cur)
20:                 if i_cur ∈ {1, |P|} then
21:                     for each (i, c) in D such that
22:                                     (i ≠ 1 ∧ i ≠ |P|)
23:                         add P[i] to CornerList
24:                     return CornerList
25:                 end if
26:             else
27:                 REMOVEANDUPDATE(i_cur, P, D)
28:                 break loop
29:             end if
30:         end loop
31:     end while
32:     return ∅
33: end function

34: procedure REMOVEANDUPDATE(i, P, D)
35:     i_pre ← D.PREVIOUSI(i)
36:     i_suc ← D.SUCCESSIVEI(i)
37:     D.REMOVE(i)
38:
39:     c ← COST(i_pre, P, D)          # computes Eq. 2
40:     D.UPDATECOST(i_pre, c)
41:
42:     c ← COST(i_suc, P, D)
43:     D.UPDATECOST(i_suc, c)
44: end procedure
```

Figure 1: The implementation of the initialization, pruning and corner classification steps.

worst case and, consequently, the complexity of REMOVEANDUPDATE() is $O(n)$; and the complexity of CLASSIFIER() is $O(n)$ since some features need $O(n)$ time in the worst case to be calculated.

The complexity of each of the three steps is then:

1. Initialization: ICOST() is called n times and D.INIT() one time, consequently the complexity of the initialization step is $O(n \log n)$.

2. Pruning: D.FINDMINC() and REMOVEANDUPDATE() are called $n - u$ times each, consequently the complexity of this step is $O(n(n - u))$.

3. Point classification: the *while* loop (in line 14) will be executed at most $k = |D| - 2 \le u - 2$ times. In the loop (in line 16), CLASSIFIER() will be called at most k times, D.SUCCESSIVEC() at most $k - 1$ times, and REMOVEANDUPDATE() at most once. Thus, in this step, they will be called less than or equal to k^2, k^2 and k times, respectively.

   The complexity of the CLASSIFIER() calls can be calculated by considering that for each point, if none of its features changes, the result of CLASSIFIER() can be retrieved in $O(1)$ by caching its previous output. Since the execution of the REMOVEANDUPDATE() function involves the changing of the features of two points, CLASSIFIER() will be executed at most $3k$ times in $O(n)$ (for a total of $O(k \times n)$) and the remaining times in $O(1)$ (for a total of $O(k^2)$), giving a complexity of $O(k \times n)$.

   Furthermore, the complexity of the D.SUCCESSIVEC() calls is $O(k^2)$, while the complexity of the REMOVEANDUPDATE() calls is $O(k \times n)$.

   Thus, since $k < n$, the point classification step is in the worst case $O(k \times n)$, or rather $O(n \times u)$.

It is worth noting that the final $O(n^2)$ complexity does not improve even if a better implementation of D providing an $O(\log |D|)$ UPDATECOST() function is used.

3.2 Features

Most of the features used in our classification are derived from previous research in the field. In particular, we have three different classes of features:

- *Stroke features*: features calculated on the whole stroke;

- *Point features*: local features calculated on the point. These features are calculated using a fixed region of support and their values remain stable throughout the procedure;

- *Rank-related features*: dynamically calculated local features. The region of support for the calculation of these features is the set of points from the predecessor p_{pre} and the successor p_{suc} of the current point in the candidate list. Their value can vary during the execution of the *Point classification* step.

Some features are parametric. In particular, they can adopt two different types of parameters:

- An integer parameter w, defining the width of the (fixed) region of support used to calculate point features;

- A boolean parameter $norm$, indicating whether a normalization is applied in the calculation of the feature.

3.2.1 Stroke Features

The features calculated on the whole stroke can be useful to the classifier, since a characteristic of the stroke can interact in some way with a local feature. For instance, the length of a stroke may be correlated to the number of corners in it: it is likely that a long stroke has more angles than a short stroke. We derived two stroke features from [20]: the length of the stroke and the diagonal length of its bounding box. These features are called $Length$ and $Diagonal$, respectively. In Figure 2a the bounding box (light gray) and the diagonal (dark gray) of a hand drawn diamond (black) are shown. Furthermore, we added a feature telling how much the stroke resembles an ellipse (or a circle), called $EllipseFit$. The use of this feature prevents that corners are accidentally inserted in strokes resembling circles or ellipses. It is calculated by measuring the average Euclidean distance of the points of the stroke to an ideal ellipse, normalized by the length of the stroke. Figure 2b shows the $EllipseFit$ calculation for a hand-drawn diamond. In particular, the figure shows the segments (dark gray) connecting the diamond (black) and the ellipse (light gray), of which we calculate the average measure.

3.2.2 Point Features

The *point features* are local characteristics of the points. The speed of the pointer and the curvature of the stroke at a point have been regarded as very important features from the earliest research in corner finding. Here, the speed at p_i is calculated as suggested in [23], i.e., $s(p_i) = \|p_{i+1}, p_{i-1}\| / (t_{i+1} - t_{i-1})$, where t_i represents the timestamp of the i-th point. We also have a version of the speed feature where a min-max normalization is applied in order to have as a result a real value between 0 and 1; the $Curvature$ feature used here is calculated as suggested in [15].

A feature that has proven useful in previous research is the *straw*, proposed in [27]. The straw at the point p_i is the length of the segment connecting the endpoints of a window of points centered on p_i. Thus we define $Straw(p_i, w) = \|p_{i+w}, p_{i-w}\|$, where w is the parameter defining the width of the window. An example of straw is shown in dark gray in Figure 2d.

A simple feature to evaluate if a point is a corner, is the magnitude of the angle formed by the segments (p_{i-w}, p_i) and (p_i, p_{i+w}), defined here as $Angle(p_i, w)$. An example is shown in Figure 2e. A useful feature to distinguish the curves from the corners is what we call *AlphaBeta*, derived from [28]. Here we use as a feature the difference between $alpha$ and $beta$, the magnitudes of two angles in p_i using different segment lengths, one three times the other: $AlphaBeta(p_i, w) = Angle(p_i, 3w) - Angle(p_i, w)$. An example of the two angles is shown in Figure 2f.

Lastly, in this research we introduce two point features that, as far as we know, have never been tested so far for corner detection. One feature is the position of the point within the stroke. Its use tends to prevent that corners are inserted in uncommon positions of the stroke. The position is calculated as the ratio between the length of the stroke from p_0 to p_i and the total length of the stroke. We call this feature $Position(p_i)$. The other feature is the difference of two areas: the former is the one of the polygon delimited by the points $(p_{i-w}, \ldots, p_i, \ldots, p_{i+w})$ and the latter is the one of the triangle (p_{i-w}, p_i, p_{i+w}). The rationale for this feature is that its value will be positive for a curve, approximately 0 for an angle and even negative for a cusp. We call it $DeltaAreas(p_i, w)$. Figure 2g shows an example that highlights the difference between the two areas.

3.2.3 Rank-Related Features

The *rank-related features* are local characteristics of the points. The difference with the *point features* is that their region of support varies according to the rank of the point: the considered neighborhood is between the closest preceding and successive points of p_i, which we have called p_{ipre} and p_{isuc}, respectively. The $Cost$ function defined in Equation (2) is an example of feature from this class. It tends to assume higher values at the corners. A distinguishing feature of our approach, strictly related to the $Cost$, is the $Rank$. We define the $Rank$ of a point $p = P[i]$ with respect to D, as the size of D resulting from the removal of (i, c) from D. As already explained, this feature is a good indicator of whether a point is a corner and it is useful to associate it to the cost function, to improve classification.

A simple feature derived from [20] is $MinDistance$, representing the minimum of the two distances $\|p_{ipre}, p_i\|$ and $\|p_i, p_{isuc}\|$, respectively. We also used a normalized version, obtained by dividing the minimum by $\|p_{ipre}, p_{isuc}\|$.

Feature	Class	Parameters	Ref.
$Length(S)$	Stroke	/	[20]
$Diagonal(S)$	Stroke	/	[20]
$EllipseFit(S)$	Stroke	/	
$Speed(p, norm)$	Point	$norm = T, F$	[23]
$Curvature(p)$	Point	/	[15]
$Straw(p, w)$	Point	$w = 4$	[27]
$Angle(p, w)$	Point	$w = 1, 2$	[28]
$AlphaBeta(p, w)$	Point	$w = 3, 4, 6, 15$	[28]
$Position(p)$	Point	/	
$DeltaAreas(p, w)$	Point	$w = 11$	
$Rank(p)$	Rank-Related	/	
$Cost(p)$	Rank-Related	/	[20]
$MinDistance(p, norm)$	Rank-Related	$norm = T, F$	[20]
$PolyFit(p)$	Rank-Related	/	[21]
$CurveFit(p)$	Rank-Related	/	[21]

Table 1: The features used in our classifier. Features without a reference are defined for the first time in this paper.

As in previous research, we try to fit parts of the stroke with beautified geometric primitives. The following two features are similar to the ones defined in [21]: $PolyFit(p_i)$ fits the substroke $(p_{ipre}, \ldots, p_i, \ldots, p_{isuc})$ through the polyline $(p_{ipre}, p_i, p_{isuc})$, while $CurveFit(p_i)$ uses a bezier curve to approximate the points. The return value is the average point-to-point euclidean distance normalized by the length of the stroke. Examples of the two aforementioned features are shown in Figures 2i and 2c, respectively.

Table 1 summarizes the set of features used by RankFrag in the CLASSIFIER function. The table reports the name of the feature, its class, the values of the parameters (if present) with which it is instantiated and the reference paper from which we derived it. The presence of more than one parameter value means that some features are used multiple times, instantiated with different parameter values. The set of features has been chosen by performing a two-step feature selection method. In the first step, bootstrapping along with RF algorithm was used to measure the importance of all the features and produce stable feature importance (or rank) scores. Then, all the features were grouped into clusters using correlation, and those with the highest ranking score from each group were chosen to form the set of relevant and non-redundant features.

3.3 Classification method

The binary classifier used by RankFrag in the CLASSI-FIER function to classify corner points is based on *Random Forests* (RF) [17]. Random Forests are an ensemble machine learning technique that builds forests of classification trees. Each tree is grown on a bootstrap sample of the data, and the feature at each tree node is selected from a random subset of all features. The final classification is determined by using a voting system that aggregates the classification results from all the trees in the forest. There are many ad-

vantages of RF that make their use an ideal approach for our classification problem: they run efficiently on large data sets; they can handle many different input features without feature deletion; they are quite robust to overfitting and have a good predictive performance even when most predictive features are noisy.

3.4 Implementation

RankFrag was implemented as a Java application. The classifier was implemented in *R* language, using the *randomForest* package [18]. The call to the classifier from the main program is performed through the *Java/R Interface* (JRI), which enables the execution of *R* commands inside Java applications.

4 Evaluation

We evaluated RankFrag on three different datasets already used in the literature to evaluate previous techniques. We repeated 30 times a 5-fold cross validation on all of the datasets. For all datasets, the strokes were resampled at a distance of three pixels, while a value of $u = 30$ was used as a parameter for pruning. Since there is no single metric that determines the quality of a corner finder, we calculated the performance of our technique using the various metrics already described in the literature. The results for some metrics were averaged in the cross validation and were summed for others.

The hosting system used for the evaluation was a laptop equipped with an *Intel^{TM} Core^{TM} i7-2630QM* CPU at 2.0 GHz running *Ubuntu 12.10* operating system and the *Open-JDK 7*.

4.1 Model validation

Here we describe the process of assessing the prediction ability of the RF-based classifiers. The accuracy metrics were calculated by repeating 30 times the following procedure individually for each dataset and taking the averages:

1. the data set DS is randomly partitioned into 5 parts $DS_1, \ldots, DS_5$ with an equal number of strokes (or nearly so, if the number of strokes is not divisible by 5);

2. for $i = 1 \ldots 5$: $DSt_i = DS \setminus DS_i$ is used as a training set, and DS_i is used as a test set.

   - RankFrag is executed on DSt_i in order to produce the training data table. In DS, the correct corners had been previously marked manually. For each point extracted from the candidate list the input feature vector is calculated, while the

(a) Bounding box and diagonal for a hand-drawn diamond.

(b) EllipseFit calculation for a hand-drawn diamond.

(c) CurveFit calculation for point p_i.

(d) Straw (in gray) for point p_i ($w = 4$).

(e) Angle for point p_i ($w = 2$).

(f) AlphaBeta calculation for point p_i ($w = 2$).

(g) DeltaAreas (highlighted in black) for point p_i ($w = 6$).

(h) MinDistance for point p_i (the minimum between $\|p_{ipre}, p_i\|$ and $\|p_i, p_{isuc}\|$).

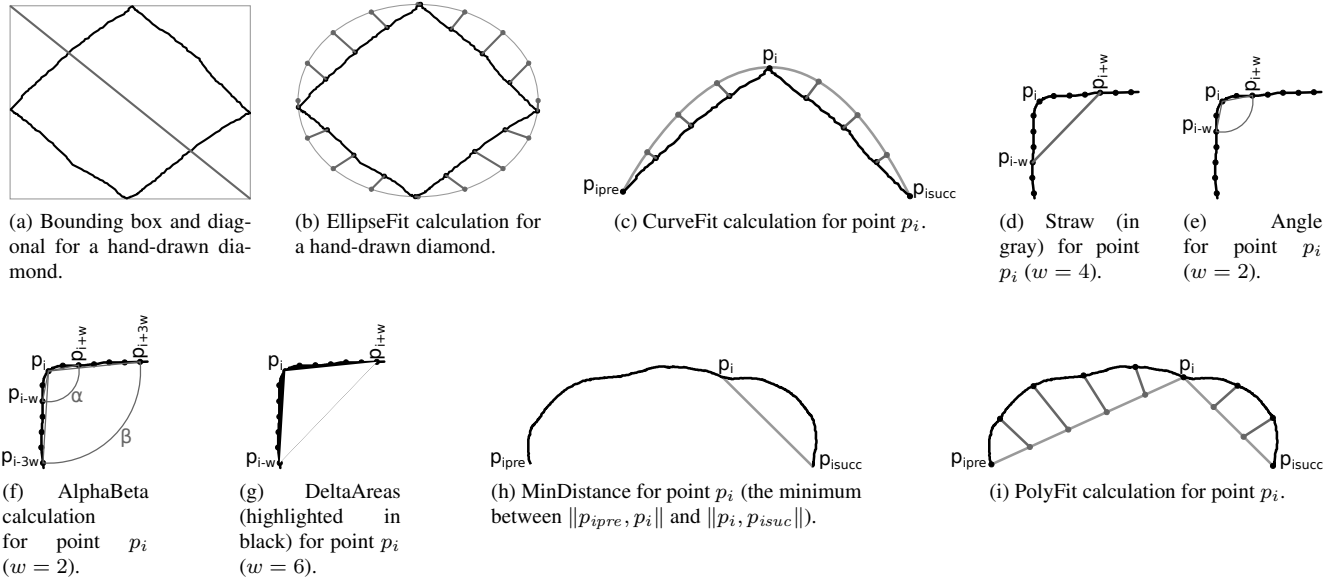(i) PolyFit calculation for point p_i.

Figure 2: Examples for the features used in our classifier.

output parameter is given by the boolean value indicating whether the point is marked or not as a corner. The training table contains both the input and output parameters;

- a random forest is trained using the table;

- RankFrag is executed on DS_i, using the trained random forest as a binary classifier;

- In order to generate the accuracy metrics, the corners found by the last run of RankFrag are compared with the manually marked ones. A corner found by RankFrag is considered to be correct if it is within a certain distance from a marked corner.

3. In order to get aggregate accuracy metrics, for each of them the average/sum (depending on the type of the metric) of the values obtained in the previous step is calculated.

4.2 Accuracy Metrics

A corner finding technique is mainly evaluated from the points of view of accuracy and efficiency. There are different metrics to evaluate the accuracy of a corner finding technique. We use the following, already described in the literature [27, 13]:

- **False positives** and **false negatives**. The number of points incorrectly classified as corners and the number of corner points not found, respectively;

- **Precision**. The number of correct corners found divided by the sum of the number of correct corners and false positives: $precision = \frac{correct\ corners}{correct\ corners + false\ positives}$;

- **Recall**. The number of correct corners found divided by the sum of the number of correct corners and false negatives: $recall = \frac{correct\ corners}{correct\ corners + false\ negatives}$. This value is also called **Correct corners accuracy**;

- **All-or-nothing accuracy**. The number of correctly segmented strokes divided by the total number of strokes;

The presence of the angle is determined by human perception. Obviously, different operators can perform different annotations on a dataset. The task of judging whether a corner is correctly found should also be done by a human operator. In our case, the human judgment is unfeasible due to the very high number of tests. Thus, we just checked whether the found corner was at a reasonable distance from the marked corner. In particular, we adopted as a tolerance the fixed distance of 20 pixels already used in literature for tests on the same datasets [13].

4.3 Datasets

Two of the three datasets used in our evaluation, the *Sezgin-Tumen COAD Database* and *NicIcon* datasets, are associated to a specific domain, while the *IStraw* dataset is not associated to any domain, but was produced for benchmarking purposes by Xiong and LaViola [28]. Some fea-

Dataset	No. of classes	No. of symbols	No. of strokes	No. of drawers	Source
COAD	20	400	1507	8	[25]
NicIcon	14	400	1204	32	[19]
IStraw	10	400	400	10	[28]

Table 2: Features of the three data sets.

tures of the three datasets are summarized in Table 2. The table reports, for each of them, the number of different classes, the total number of symbols and strokes, the number of drawers and a reference to the source document introducing it.

The symbols in the *Sezgin-Tumen COAD Database* (called only COAD, for brevity, in the sequel) dataset are a subset of those used in the domain of *Military Course of Action Diagrams* [7], which are used to depict battle scenarios. A set of 620 symbols was firstly introduced by Tirkaz et al. [25] to measure the performance of a multi-stroke symbol recognizer. Here we use a subset of 400 symbols annotated by Tumen and Sezgin and used to evaluate a technique for finding corners [26].

The *NicIcon Database of Handwritten Icons* [19] is a set of symbols, drawn by 32 different subjects, gathered for assessing pen input recognition technologies, representing images for emergency management applications. Here we use the subset of 400 multi-stroke symbols, annotated by Tumen and Sezgin [26].

The *IStraw* dataset is referred to as an *out-of-context* dataset, i.e., it is not linked to a domain. It was one of the datasets used to test the homonymous technique [28] and DPFrag [26]. It contains both line and arc primitives belonging to 400 unistroke symbols, drawn by 10 different subjects.

Figure 3 shows one random sample from each class of the three symbol set.

5   Results

In this section we report the results of our evaluation. As for the accuracy, we calculated all of the metrics described in the previous section. Furthermore, RankFrag's accuracy is compared to that of other state-of-art methods by using the All-or-nothing metric. It is worth noting that, due to the unavailability of working prototypes, we did not directly test the other methods: we only report the performance declared by their respective authors.

The accuracy achieved by RankFrag on the three datasets is reported in Table 3. The results are averaged over the 30 performed trials.

Table 4 shows a comparison of the accuracy of RankFrag with other state-of-art methods. The methods considered

Metrics	COAD	NicIcon	IStraw
Corners manually marked	2271	867	1795
Corners found	2260.67	774.03	1790.80
Correct corners	2254.20	730.90	1784.33
False positives	6.47	43.13	6.47
False negatives	16.80	136.10	10.67
Precision	0.9972	0.9441	0.9964
Recall / Correct corners accuracy	0.9926	0.8428	0.9940
All-or-nothing accuracy	0.9870	0.8657	0.9572

Table 3: Average accuracy results of RankFrag on the three datasets.

Dataset	RankFrag	DPFrag	IStraw
COAD	0.99	0.97	0.82
NicIcon	0.87	0.84	0.24
IStraw	0.96	0.96	0.96

Table 4: Comparison of RankFrag with other methods on the All-or-nothing accuracy metric.

here are DPFrag [26] and IStraw [28]. Due to the unavailability of other data, we only report the results related to the All-or-nothing metric. As we can see, RankFrag outperforms the other two methods on two out of three datasets.

As for efficiency, we report that the average time needed to process a stroke is ~390 ms. Our prototype is rather slow, due to the inefficiency of the calls to R functions. We also produced a non-JRI implementation by manually exporting the created random forest from R to Java (avoiding the JRI calls). With this implementation, the average execution time was lowered to ~18 ms, enabling real-time runs.

6   Discussion and Conclusion

We have introduced RankFrag, a technique for segmenting hand-drawn sketches in the corner points. RankFrag has a quadratic asymptotic time complexity with respect to the number of sampled points in an input stroke. This complexity is the same reported in the literature for many other methods and, to the best of our knowledge, there is no method with a lower complexity. The technique was evaluated on three different datasets. The datasets were specifically produced for evaluating corner detection algorithms or were already used previously for this purpose.

We compared the results obtained by RankFrag with those already available in the literature for two different techniques: DPFrag [26] and IStraw [28]. With respect to the latter, our results show a clear advantage in accuracy on two datasets for RankFrag. With respect to DPFrag, our technique has a comparable accuracy, with a slight advantage on two of the three datasets. Nevertheless, compared to DPFrag, our technique has the additional advantage that

(a) COAD
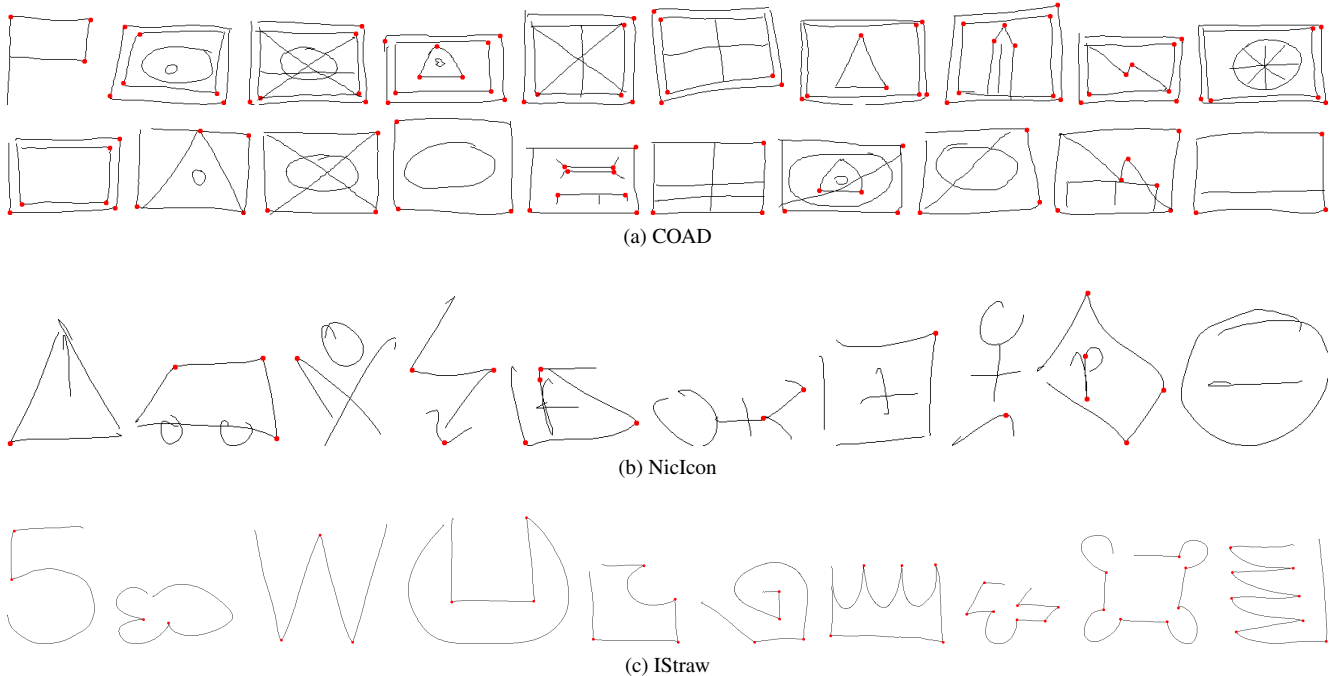

(b) NicIcon

(c) IStraw

Figure 3: One random sample from each class of the three symbol set. The manually annotated corners are highlighted with a red circle.
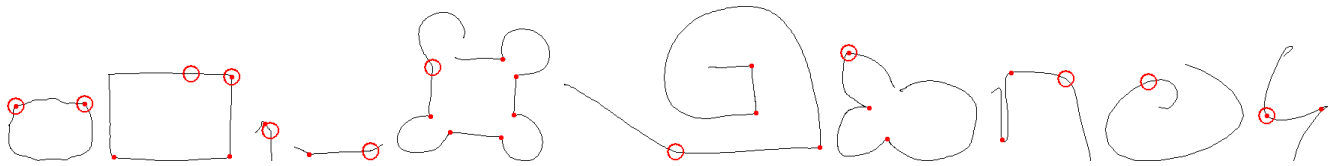

Figure 4: Examples of misclassification by RankFrag. Detected corners are represented through a red dot, while classification errors are represented through rings.

it can be performed in real time on all the tested data, regardless of the complexity of the input strokes. The chart reported in [26] (Figure 9) shows that this is not guaranteed for DPFrag and that its running time grows with the number of corners in the stroke.

RankFrag can be considered a significant improvement to the segmentation technique presented in [20]. In that technique, the classifier is used as a stop function: when the classifier decides to stop, all the remaining points (those with a higher cost) are classified as corners. We found that such a technique is not appropriate for strokes containing curves, since the cost function alone is not a reliable indicator and gives many false positives. Thus, we decided to invoke the classifier within a more complex, but still efficient, procedure, which performs further checks to establish whether a point is an angle. We also more profitably use a larger set of features, some of which have a variable region of support. Lastly, in our analyses the random forest

seemed to have better performance with respect to the other classifiers which we preliminarily tested, such as SVM and Neural Networks.

It is worth noting that, although further accuracy improvements are possible, it is very difficult to get a score close to 100% due to the procedure used in our tests: the decision of the classifier was compared to an earlier annotation made by a human operator. Some decisions are debatable and the annotation process is not free from errors. Figure 4 shows some examples of corner misclassification by Rank-Frag on the three datasets, including both false positives (dots inside a ring) and false negatives (rings). Although annotation errors are evident in some of the strokes reported in the figure, we decided not to alter the original annotation in order to obtain a more faithful comparison with the other methods.

RankFrag has only been tested for finding corner points and not *tangent vertices*, as done by other techniques

37

[12, 1]. It can be directly used in various structural methods for symbol recognition. However in some methods an additional step to classify the segments in lines or arcs may be required.

The non-JRI version of our implementation is able to produce the segmentation of a stroke in real time on a sufficiently powerful device. Future work will aim to achieve further implementation improvements, in order to further reduce the execution time and make the technique applicable in real time on more strokes at once (e.g., an entire diagram) or on mobile devices with low computational power. For testing purposes, our implementation can be downloaded at `http://weblab.di.unisa.it/rankfrag/`.

References

[1] F. Albert, D. Fernández-Pacheco, and N. Aleixos. New method to find corner and tangent vertices in sketches using parametric cubic curves approximation. *Pattern Recognition*, 46(5):1433 – 1448, 2013.

[2] R. Bellman. On the approximation of curves by line segments using dynamic programming. *Commun. ACM*, 4(6):284, June 1961.

[3] G. Costagliola, M. De Rosa, and V. Fuccella. Local context-based recognition of sketched diagrams. *Journal of Visual Languages & Computing*, 25(6):955 – 962, 2014.

[4] G. Costagliola, M. De Rosa, and V. Fuccella. Recognition and autocompletion of partially drawn symbols by using polar histograms as spatial relation descriptors. *Computers & Graphics*, 39(0):101 – 116, 2014.

[5] G. Costagliola, V. Fuccella, and M. D. Capua. Interpretation of strokes in radial menus: The case of the keyscretch text entry method. *Journal of Visual Languages & Computing*, 24(4):234 – 247, 2013.

[6] G. Costagliola, V. Fuccella, and M. Di Capua. Text entry with keyscretch. In *Proceedings of the 16th International Conference on Intelligent User Interfaces*, IUI '11, pages 277–286, New York, NY, USA, 2011. ACM.

[7] T. D.U. Commented APP-6A - Military symbols for land based systems, 2005.

[8] J. Dunham. Optimum uniform piecewise linear approximation of planar curves. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(1):67–75, Jan 1986.

[9] V. Fuccella and G. Costagliola. Unistroke gesture recognition through polyline approximation and alignment. In *Proceedings of CHI '15*, pages 3351–3354, New York, NY, USA, 2015. ACM.

[10] R. Haddad and A. Akansu. A class of fast Gaussian binomial filters for speech and image processing. *Signal Processing, IEEE Transactions on*, 39(3):723–727, Mar 1991.

[11] T. Hammond, B. Eoff, B. Paulson, A. Wolin, K. Dahmen, J. Johnston, and P. Rajan. Free-sketch recognition: Putting the chi in sketching. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '08, pages 3027–3032, New York, NY, USA, 2008. ACM.

[12] J. Herold and T. F. Stahovich. Speedseg: A technique for segmenting pen strokes using pen speed. *Computers & Graphics*, 35(2):250–264, 2011.

[13] J. Herold and T. F. Stahovich. A machine learning approach to automatic stroke segmentation. *Computers & Graphics*, 38(0):357 – 364, 2014.

[14] C. F. Herot. Graphical input through machine recognition of sketches. In *Proceedings of the 3rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '76, pages 97–102, New York, NY, USA, 1976. ACM.

[15] D. H. Kim and M.-J. Kim. A curvature estimation for pen input segmentation in sketch-based modeling. *Computer-Aided Design*, 38(3):238 – 248, 2006.

[16] W. Lee, L. Burak Kara, and T. F. Stahovich. An efficient graph-based recognizer for hand-drawn symbols. *Computers & Graphics*, 31:554–567, August 2007.

[17] B. Leo. Random forests. *Machine Learning*, 45(1):5–32, dec. 2001.

[18] A. Liaw and M. Wiener. Classification and regression by randomForest. *R News*, 2(3):18–22, 2002.

[19] R. Niels, D. Willems, and L. Vuurpijl. The nicicon database of handwritten icons. 2008.

[20] T. Y. Ouyang and R. Davis. Chemink: a natural real-time recognition system for chemical drawings. In *Proceedings of the 16th international conference on Intelligent user interfaces*, IUI '11, pages 267–276, New York, NY, USA, 2011. ACM.

[21] B. Paulson and T. Hammond. Paleosketch: accurate primitive sketch recognition and beautification. In *Proceedings of the 13th international conference on Intelligent user interfaces*, IUI '08, pages 1–10, New York, NY, USA, 2008. ACM.

[22] T. M. Sezgin, T. Stahovich, and R. Davis. Sketch based interfaces: Early processing for sketch understanding. In *Proceedings of the 2001 Workshop on Perceptive User Interfaces*, PUI '01, pages 1–8, New York, NY, USA, 2001. ACM.

[23] T. F. Stahovich. Segmentation of pen strokes using pen speed. In *AAAI Fall Symposium Series*, pages 21–24, 2004.

[24] C.-H. Teh and R. Chin. On the detection of dominant points on digital curves. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(8):859–872, Aug 1989.

[25] C. Tirkaz, B. Yanikoglu, and T. M. Sezgin. Sketched symbol recognition with auto-completion. *Pattern Recognition*, 45(11):3926–3937, 2012.

[26] R. S. Tumen and T. M. Sezgin. Dpfrag: Trainable stroke fragmentation based on dynamic programming. *IEEE Computer Graphics and Applications*, 33(5):59–67, 2013.

[27] A. Wolin, B. Eoff, and T. Hammond. Shortstraw: A simple and effective corner finder for polylines. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*. Eurographics Association, 2008.

[28] Y. Xiong and J. J. J. LaViola. A shortstraw-based algorithm for corner finding in sketch-based interfaces. *Computers & Graphics*, 34(5):513 – 527, 2010.

WiSPY: A Tool for Visual Specification and Verification of Spatial Integrity Constraints

Vincenzo Del Fatto
Faculty of Computer Science
Free University of Bozen-Bolzano
39100 Bolzano, ITALY
vincenzo.delfatto@unibz.it

Vincenzo Deufemia
Department of Computer Science
University of Salerno
84084 Fisciano (SA), ITALY
deufemia@unisa.it

Luca Paolino
Department of Research
Link Campus University
00162 Roma, ITALY
l.paolino@unilink.it

Sara Tumiati
South Tyrolean Municipality Consortium
39100 Bolzano, ITALY
sara.tumiati@gvcc.net

Abstract

Nowadays, most of tools for spatial data manipulation allow to edit information on maps without performing any integrity verification. On the other hand, data repositories such as the DBMS only permit few constraints to be defined by means of their Data Definition Languages and leave programmers to implement procedures for complex constraints. In this work we present the WiSPY system, a plugin of the GIS tool uDig for visually specifying and verifying complex spatial integrity constraints. WiSPY includes a visual environment for defining spatial data models with integrity constraints and for automatically generating the constraint checker. The latter is used by the WiSPY tool to verify the integrity of the data produced during the map editing process. The system has been validated on a real case study concerning the current regulation of the Public Illumination Plan (PIP) managed by an Italian municipality.

1 Introduction

The management of spatial data is one of the fields where companies and researchers have invested much money and time in the last decade. The result is that commercial products such as Autodesk Autocad Map 3D[1], Bentley Microsta-

tion[2], ESRI ArcGIS[3], or free and open source products such as Google Map[4], QGIS[5], GRASS GIS[6], uDig [22], have become part of the daily life not only for GIS users. This is because, they offer a large amount of features for spatial data manipulation, spatial analysis and reasoning functionalities that in many cases may support or simplify our activities. Despite the large amount of available features, these products lack of an adequate control during the editing phase, not allowing a solid constraint check. Also in the Database Management Systems (DBMS) field, commonly used product as Oracle and PostgreSQL, which offer spatial extensions, only allow basic functionalities to support constraint checking. Indeed, their Data Definition Languages (DDL) only support the management of simple topological constraints while advanced controls need to be coded. In this context, it appears to be desirable to provide a significant support in this phase in order to improve the quality of data and minimizing the implementation activities which often are annoying and repetitive.

In order to increase the dataset quality, correction operations can be performed both during the editing phase (on the fly) or after a data manipulation session (*a posteriori*). Both such approaches have pros and cons. On one hand, checking correctness during the editing phase has a direct effect on the data entry process, since the feedback is immediate, but the check can be performed only on a subset

---

[1] http://www.autodesk.it/products/autocad-map-3d/overview

[2] http://www.bentley.com/it-IT/products/microstation/
[3] http://www.esri.com/software/arcgis
[4] https://maps.google.com/
[5] http://www.qgis.org/en/site/
[6] http://grass.osgeo.org/

of data. On the other hand, an a posteriori check is performed on the whole dataset or on a selected subset of data, giving the possibility to apply a complete verification and to globally re-adjust geographic data. However, in this case the system returns feedback to the user just at the end of the manipulation process, making more difficult to handle possible errors.

To guarantee the effective verification of map constraints according to the designer requirements, in [7] a visual language parsing approach for constraint checking of input spatial data during the editing phase is presented. The integrity of data produced during the map editing process is guaranteed by a constraint checker automatically generated from a visual language grammar. In order to reduce the efforts for defining the constraints to be checked, a high-level data model is used to specify the user needs.

In this paper we propose a software system, named WiSPY, which consists of two uDig plugins that allow the user to:

- specify geographic models by means of the OMT-G visual modeling language;

- automatically translate the OMT-G models to the corresponding grammars;

- validate geographic incoming data against the constraint checker generated from the grammars.

The rest of the paper is organized as follows. Section 2 presents the case study based on the Public Illumination Plan which we used to validate the proposed system. Section 3 introduces the OMT-G modeling language used for specifying spatial integrity constraints (SICs, for short). Section 4 describes the WiSPY tool, while Section 5 shows its application to the considered case study. Section 6 discusses the work existing in literature related with our proposal. Conclusions and future work are given in Section 7.

2   Case Study

The case study presented in this section represents the current regulation of the Public Illumination Plan (PIP) managed by the South Tyrolean Municipalities Consortium (STMC), in South Tyrol, Italy. The STMC[7] is a cooperative founded in 1954 that includes among its members all the south Tyrolean municipalities and is mainly focused on legal practice, administrative training, labor legislation, and ICT services. The PIP is a complex set of regulations that can be difficult to interpret and apply correctly. In particular, these regulations define a complex set of lighting categories depending on the type of road the lamp is placed (urban or extraurban roads, pedestrian zone, bicycle paths),

---

[7]http://www.gvcc.net

if the road is heavily busy or not, if there is a pedestrian passage, if there are crossing roads, and so on. The plan must be "safe for people and things" and implemented in order to limit light pollution. Light pollution is considered as a misdirected, excessive or obtrusive artificial light, causing a serious degradation of the natural nocturne light. A public administration must intervene in order to prevent such situations. In addition, a illumination system that involves wrong light bulbs, wrong lamp types or has an overestimation of the lamp power could create economical issues. The proposed WiSPY tool can help domain experts to better understand and effectively manage such a complex real world scenario.

The verification of the PIP is a typical task that a public administration is faced with and it is a complex task for different reasons, such as the difficulty of managing many types of geographic data involved in, as well as, the tight connection to the context they are inserted in. In fact, the simple containment spatial relationship is not sufficient to check a wide range of constraints that could depend on the context on which a lamp is being placed (type of road or area), the type of lamp itself, the type of illumination based on both lumen and lux. In addition, checking the correctness of the geographic data related to a street lamp could be tricky, because a lamp is normally represented as a point in space, while the constraints may need a polygon to be successfully checked. For example, checking the correct distribution of lamps along a road is not sufficient to compute the distance between the points of the lamps, it is also necessary to calculate the amplitude of the radiation given by the lux value.

The case study consists in the automatic verification of real geographic data related to the PIP of the South Tyrolean Municipalities. The data includes basic cartographic data (boundaries, hydrography, vegetation), roads, buildings, and of course the PIP. Based on the current regulations, a set of constraints suited to the validation of the chosen municipality's PIP can be specified. The road types in the municipal boundaries are of type C (secondary extra urban road) and type F (local roads and bicycle paths). The following steps are necessary to determine which configuration is suitable for each road type:

- determine whether the road is of type C or F, including the speed limit;

- determine which technical illumination class is related to the road, in order to have the right luminance values;

- determine the type of lamp;

- determine the height of the poles;

- determine how to place the poles at the side of the road:

    - unilateral;

- bilateral with alternate center;
- bilateral with opposite center;
- double centred (between the two carriageways);

- determine the distance of the poles.

All these factors must be taken into account during the verification process, and they depend on each others; determining the distance of the poles is the step that depends more on the other steps, whilst the first two steps are those that influence more the decision.

3 Visual Modeling Geographic Data embedding Integrity Constraints with OMT-G

Although existing data modeling approaches and tools can be adapted for geographic database design, most of them do not support certain aspects of the modeling process, such as the treatment of SICs. A visual language for modeling geographic data must be able to visualize different aspects of the data structure including numerous types of representations, such as point, line, polygon as well as non-spatial data; conventional as well as geo-referenced classes; different types of spatial relations, spatial constraints, spatial aggregation relationships.

Object Modeling Technique for Geographic Applications (OMT-G) is an object-oriented approach to model data for geographic information. Its notation is based on the classic OMT class diagram notation [5], and further extended to embrace also Unified Modeling Language (UML) concepts and notations [4]. OMT-G provides three types of primitives, based on the UML primitives for class diagrams, to model the geometry and topology of geographic data, providing support for topologic structures, network structures, multiple views of objects, and spatial relationships. These types are classes, relationships and SICs. These primitives allow also for the specification of alphanumeric attributes and associated methods for each class.

OMT-G offers three types of diagrams: the class diagram, that represents the classes involved in the model, as well as their relations; the transformation diagram, that permits the description of the transformation process of a class, if the class diagram indicates the need of multiple representation of it; the presentation diagram which describes how to represent the visual aspects of objects in the visualization.

OMT-G class diagrams are composed of conventional and geo-referenced classes. The first behave as UML classes and have no geographical properties. The latter include a geographical representation alternative, which specializes in two types of representations: discrete, associated with real world elements (geo-objects), or continuously distributed over the space (geo-fields). Geo-objects are represented with points, lines, polygons or network elements, whereas geo-fields correspond to variables such as soil type, relief and temperature. The relationships of a OMT-G class diagrams can be conventional, e.g. UML relationships, or georeferenced. The latter include topological relations (e.g. touch, in, cross, overlap, and disjoint), arc-node network relations and spatial aggregations.

OMT-G class diagram permits the derivation of the set of SICs that must be observed in the implementation. SICs can be classified in: *topological* (the geometrical properties), *semantic* (the semantic of the geographic feature), and *user-defined* integrity constraints, "business rules" and all those controls that are non-spatial. Topological integrity constraints include spatial dependencies, spatial associations, connectivity, and geo-field rules. Semantic integrity constraints include spatial association and disjunction rules. User-defined integrity constraints are obtained from methods that can be associated to the classes.

4 The WiSPY Tool

In this section we present WiSPY (Visual specification & Verification of SPatial integritY constraints), an extension of the uDig GIS tool [22] for enabling users to visually model geographic applications, also embedding SICs, and to verify the correctness of the input geographic data. WiSPY has been implemented by means of two plugins. In the following we present the architecture of the WiSPY tool and provide details about the implemented plugins.

4.1 The Architecture

Figure 1 shows the architecture of the proposed WiSPY tool. It has been implemented on top of uDig, which is a software program based on the Eclipse platform featuring full-layered Open Source GIS. In particular, uDig provides a complete Java solution for viewing, editing, and accessing GIS data. Since it is built on top of the Eclipse "Rich Client Platform", WiSPY has been developed in Java as two uDig plugins, namely OMT-G Editor and Constraint Checker.

The OMT-G Editor provides three different environments, one for each diagram the OMT-G data model provides. In the canvas of the class diagram editor, users specify their schema, adding classes and relationships chosen from the tool palette. The relationships selected from the palette can be inserted by clicking over the source class and dragging a line to the target class. As an example, Figure 2 shows simple OMT-G class diagram modeling *containment* constraints among Municipality, Lamp, and Road objects.

The OMT-G editor includes a function to derive a visual grammar modeling the SICs specified in the OMT-G data model. The Constraint Checker generated from the grammar by using the ANTLR parser generator[8] can be activated
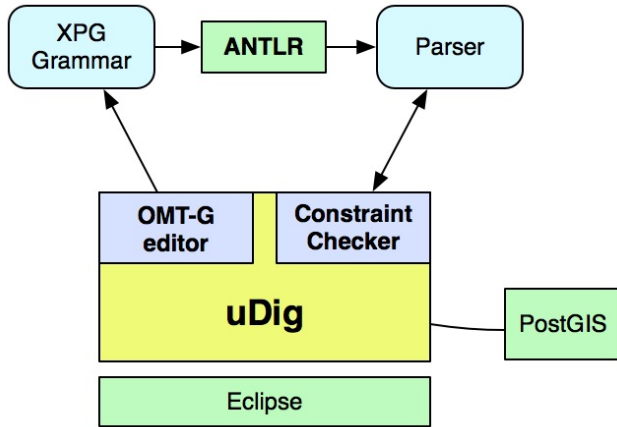
---

[8]http://www.antlr.org/

Figure 1: The architecture of WiSPY tool.

by the user during map editing phase. In particular, the input of WiSPY is a set of geographic data whose type is defined in the OMT-G data model. The output is the validation of the input data with respect to the SICs specified between the classes of the OMT-G data model. If a SIC is violated then a suitable error message is shown to user with information to recover from the violation.

4.2 Constraint Checker Generation

The WiSPY tool automatically generates a constraint checker able to verify the SICs defined by a OMT-G class diagram. In particular, WiSPY exploits the visual language compiler-compiler technique proposed in [6] for deriving a visual language parser through standard compiler-compilers, like YACC [11].

The OMT-G Editor allows users to develop their schema, adding classes and relationships chosen from the tool palette. The relationships can be annotated with SICs, which impose restrictions on the input data. In particular, the classes of the model represents the geographical objects that the user can place on the map, while the relationships specified between two classes define SICs on their instances. Such constraints are defined on the attributes of the involved classes. The editor provides the set of standard OMT-G spatial integrity rules (e.g., *contain relation*, *coincide relation*, *cross relation*, *touch*, *in*) as well as standard processes such as generalization and specialization. Moreover, the users can define new rules by specifying a set of conditions on the classes' attributes.

The constraint checker generation process consists of mapping the OMT-G class model into a visual grammar. To this end, we use the XPG grammar formalism [6], which is similar to context-free string grammars, where more general relations other than concatenation are allowed. In particular, an XPG textually describes a diagram by grammar

productions that alternate (terminal and nonterminal) symbols with relations defined on the symbol attributes. Thus, the idea is to map the classes defined in the OMT-G schema, which represent the spatial objects to be placed on the map, into terminal symbols of the grammar, while the SICs defined between the spatial objects are modeled in terms of spatial relations among them [6]. In this way, the user can analyze the SICs specified for a particular application domain and, eventually, customize some of them interacting with the editor.

For instance, the containment constraint between *Road* and *Lamp* in Fig. 2 is modeled by the production:

Roads $\rightarrow$ ROAD $\langle contains \rangle$ LAMP

where *contains* is an empty production with associated a semantic action that verifies the satisfiability of the relationship [6]. ROAD and LAMP are terminal symbols having associated the set of attributes defined in the corresponding classes of the OMT-G model, e.g., *type* for ROAD. Such attributes are used by the *contains* production to verify the spatial constraint. The nonterminal symbol Roads has associated a set of attributes whose value is synthesized from the attribute values of ROAD and LAMP.

The WiSPY tool provides the implementation of semantic actions for a predefined set of constraints. However, as said above, the tool enables users to define their own constraints. In particular, the user can annotate a OMT-G relationship connecting two classes A and B with a boolean condition on the attributes of A and B. As an example, for the OMT-G diagram in Figure 2, a user could define the following illuminance constraint: *the lightning of lamps associated to urban highways is greater than $40SB^2$*. This constraint is named *illuminates* and is defined by the following boolean expression:

(Road.$type$='Highway' $\wedge$ Lamp.$lightning$ >40).

The constraint checker is obtained by giving as input to a compiler-compiler the grammar automatically generated from the OMT-G model. Since WiSPY uses the ANTLR parser generator to perform this task, we represent the XPG grammar into a format compatible with ANTLR. The use of

Fig. 3 shows the grammar constraint checker editor embedded into the uDig interface. In particular, in the right side of the interface (label D), the palette contains all the suitable tools for grammar checking, and the "Select Feature Set" operator is activated. By using this operator, users can select geographic features into the area of interest by using a simple rectangle selection tool on the standard uDig map view. After this operation the geographic data of interest are selected and highlighted in yellow in the map (see label A). In this example, the highlighted polygons represent areas, while the highlighted points represent lamps. In this case, only the selected geographic features are involved in the constraint check process. At the bottom left side of the interface (label B), the details about the selected features
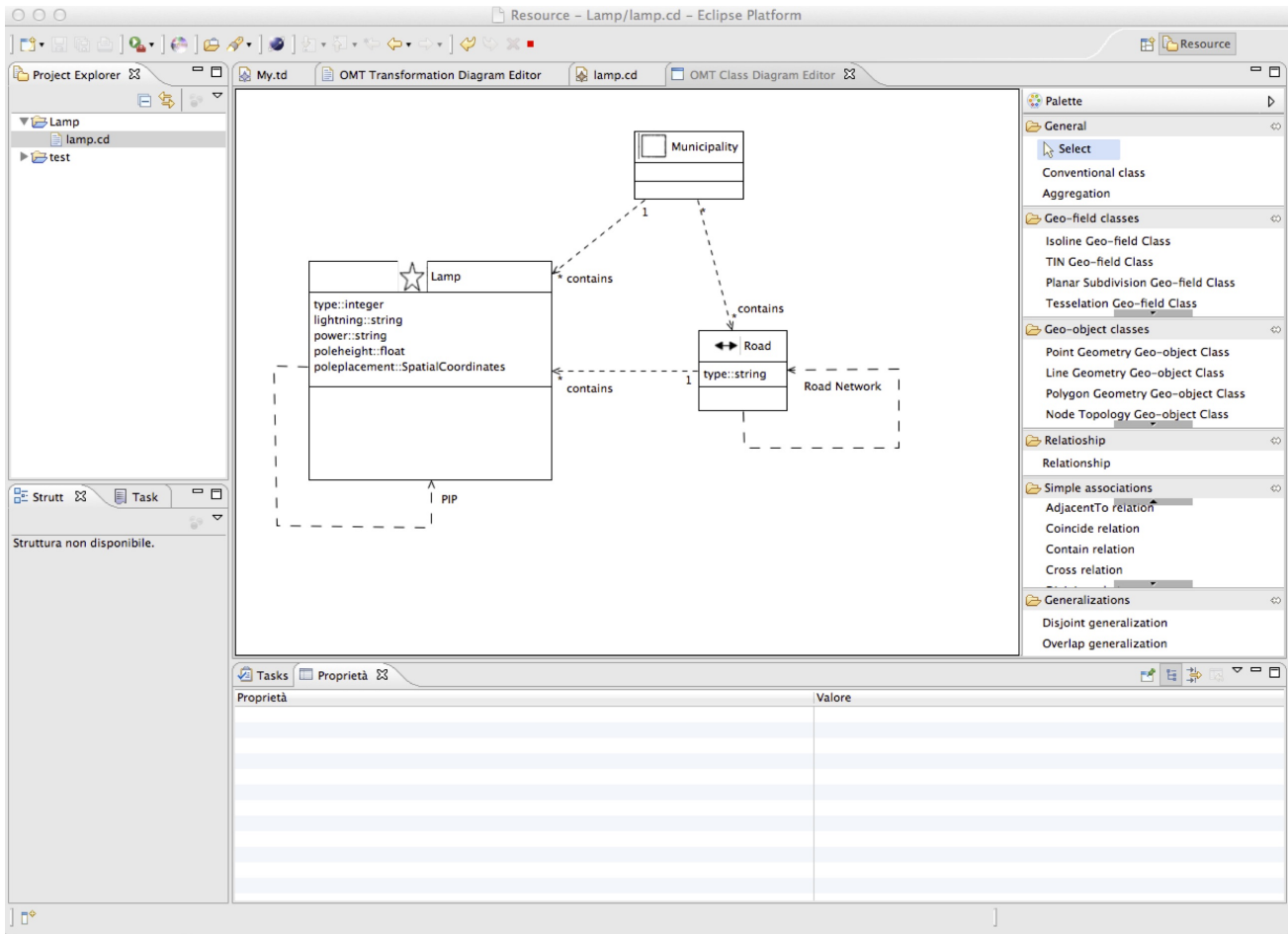
Figure 2: OMT-G interface for specifying the spatial integrity constraints.

are shown. Finally, at the bottom right side of the interface (label C), the output console of the validation process is shown.

4.3 Verification of SICs

The parser generated with ANTLR is used by WiSPY to validate the input spatial data against the SICs specified in the OMT-G model. In particular, the parser analyzes the spatial objects positioned on a map driven by the relationships specified in the grammar. If the spatial objects violates a SIC then it yields a parse error.

Fig. 4 shows the result of the constraint checking process in the WiSPY interface. In this example, points representing lamps are highlighted by using the "Select Feature Set" operator. Executing the constraint checking, the steps performed by the validation process are listed into the console view, located at the bottom of the interface. If an error occurs, it is reported to the user in the console view.

5 Checking SICs for Public Illumination Plans

The best way to illustrate how to apply our system to real problems is through an example on PIP case study. In this domain, lamps are spatial objects having associated the following information: localization of the lamp (municipality, hamlet, street, GPS coordinates), number of light points for every lamp, lamp type, type of light source, number of light sources for light point, electric power for each light source, year, overall electric power of the lamp, mounting typology (wall or pole), pole type, pole height, circuit and electric power panel, road classification and technical illumination classification. The roads are classified as:

- Category A: highways;
- Category B: high-speed extraurban roads;
- Category C: secondary extraurban roads;
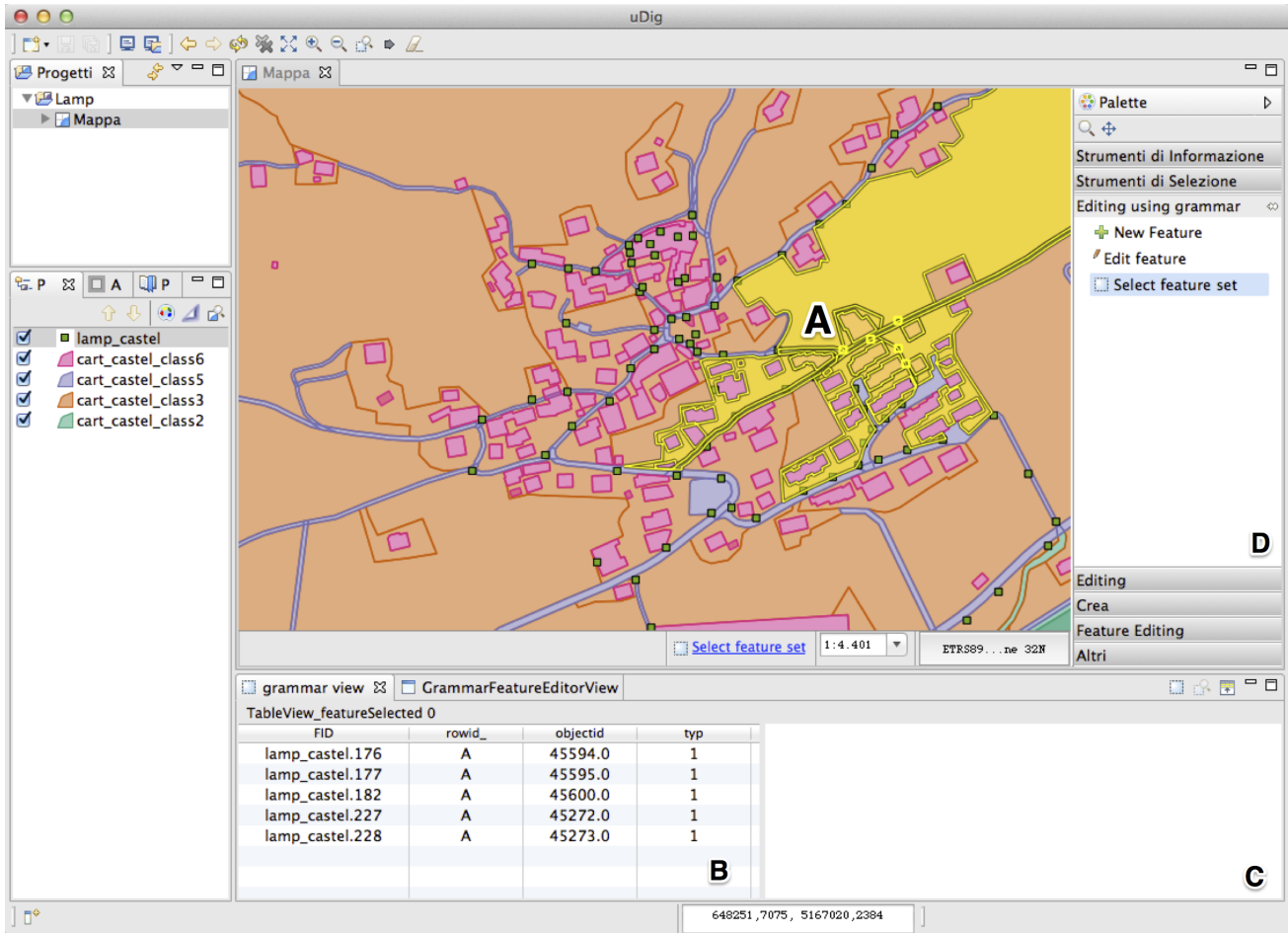- Category D: urban arterial roads;

Figure 3: WiSPY main window with the additional tools for grammar parsing.

- Category E: urban district roads;

- Category F: local roads.

Road illumination varies depending on road classification and on the related technical illumination classification. This classification is the same specified in the UNI EN 13201 European normative, which states that the illumination level is based on the traffic intensity of the road and on the daytime. Therefore the technical illumination classification of a road may vary during the daytime. Since the geographic data used for the prototype are coming from a municipality far away from highways, the classification used in WiSPY is simplified as reported in Tables 1 and 2. The illumination parameters reported in Table 2 refers to:

- $\bar{L}(cd/m^2)$ is the average road surface luminance of a carriageway of a road expressed in candelas per square meter;

- U_o is the overall uniformity of road surface luminance;

- U_l is the longitudinal uniformity of road surface luminance;

- TI is the threshold increment, which measures the loss in percentage of visibility caused by the disability glare of the luminaries of a road lighting installation;

- SB^2 is the surround ratio of illumination of a carriageway of a road

- $\bar{E}$ is the hemispherical illuminance averaged over a road area expressed in lux.

Road Type	Road description	Speed limit (km/h)	Tech. Illum. Class.
C - Secondary extraurban roads	Extraurban road	70-90	ME3a
	Local extraurban roads	50	ME4b
F - Local roads	Local urban roads	30	S3
	Historic town center	-	CE4
F - Pedestrian and bicyle routes	-	-	S3

Table 1: UNI EN 13201 road classification (subset).

In order to illustrate how WiSPY is able to identify violations in the public illumination plan, in the following we
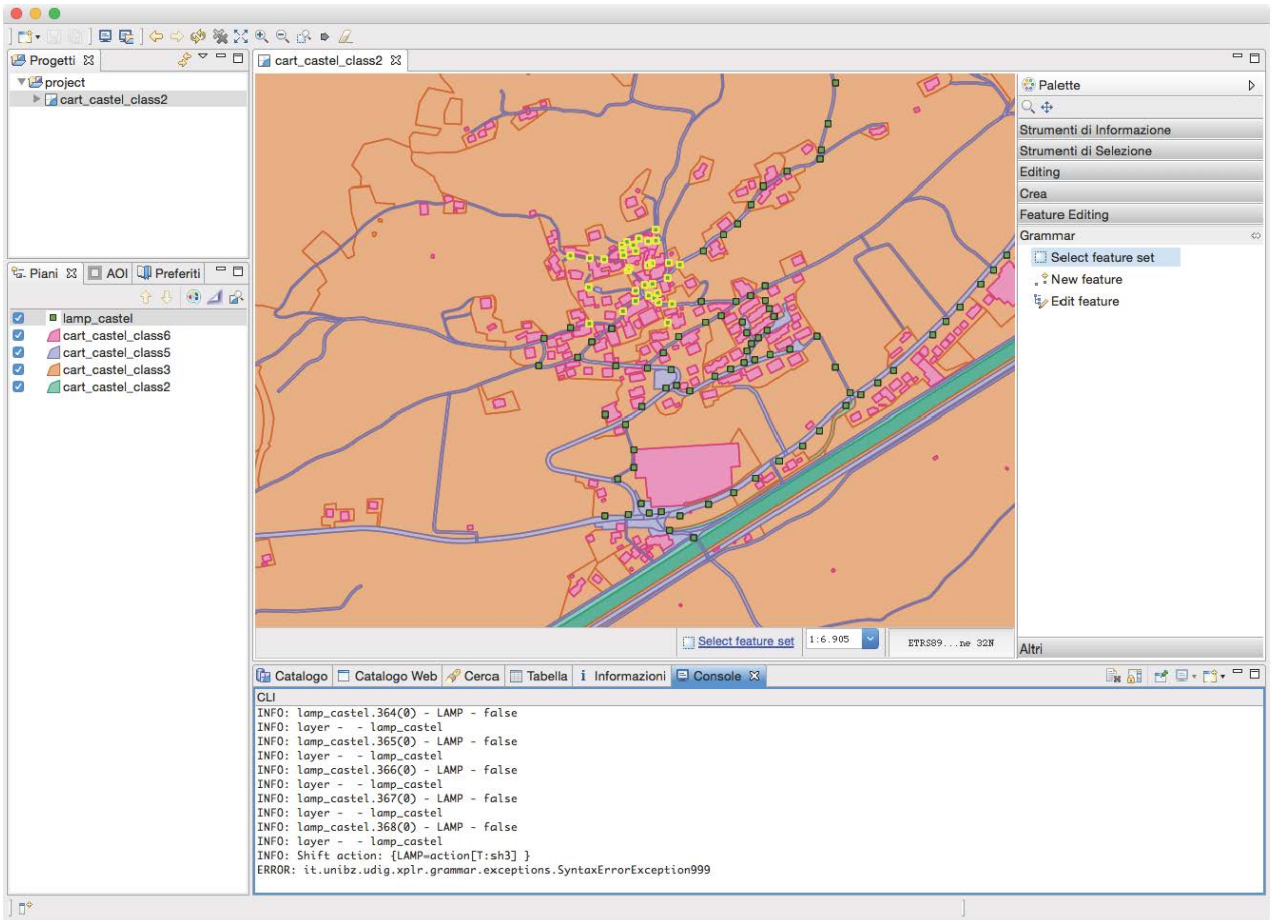
Figure 4: WiSPY interface showing the result of the constraint checking process.

Class	Detail type	Detail values
ME3a	Luminance of the road surface of the carriageway	$1,0\ \overline{L}(\mathrm{cd/m^2})$
		$0,4\ U_o$
		$0,7\ U_l$
	Disability glare	15 TI in %
	Lighting of surroundings	$0,5\ \mathrm{SB^2}$
ME4b	Luminance of the road surface of the carriageway	$0,75\overline{L}(\mathrm{cd/m^2})$
		$0,4\ U_o$
		$0,6\ U_l$
	Disability glare	15 TI in %
	Lighting of surroundings	$0,5\ \mathrm{SB^2}$
S3	Horizontal illuminance	$7,5\overline{E}$ in lx
		$1,5E_{min}$ in lx
CE4	Horizontal illuminance	$10\ \overline{E}$ in lx
		$0,4\ U_o$

Table 2: Considered UNI EN 13201 technical illumination classes.

sketch the grammar derived from the OMT-G model in Figure 2. In particular, the terminal symbols of the grammar correspond to the classes of the model, i.e., MUNICIPALITY, LAMP, and ROAD, while the productions are generated according to the class relationships specified in the OMT-G model:

1. Map $\rightarrow$ MUNICIPALITY $\langle contains \rangle$ Objects;

2. Objects $\rightarrow$ Lamps $\langle union \rangle$ Roads;

3. Lamps $\rightarrow$ Lamp $\langle pip \rangle$ Lamps

4. $pip \rightarrow \epsilon$
   SemanticAction: {
   if *distance*(Lamp,Lamps)$<$MIN_LAMP_DIST
     then
       parse.alert('CONSTRAINT VIOLATION',
       'Lamp'+Lamp.pos+' is too close to other lamps');
   }

5. Lamps $\rightarrow$ Lamp

6. Lamp → LAMP

7. Roads → Road ⟨*touches*⟩ Roads

8. Roads → Road

9. Road → ROAD

10. Road → ROAD ⟨*contains, isIlluminated*⟩ Lamp

11. *isIlluminated* → ϵ
    SemanticAction: {
    if (ROAD.*type*='C' ∧ Lamp.lightning≠0.5) ∨(. . .)
      then
        parse.alert('CONSTRAINT VIOLATION',
        'Lamp'+Lamp.position +' violates the '+
        'illumination of road ' + ROAD.name);
    }

The productions have associated semantic actions that analyze the values of the attributes associated to the spatial symbols and verify whether the PIP constraints are satisfied. In particular, the first production indicates that a map is composed of a municipality symbol (visually defined in Figure 2 with a polygon) containing within its area other objects. The latter can be Lamps and/or Roads as defined in production 2. The set of lamps in the municipality has to satisfy the constraint associated to the *PIP* relationship in Figure 2, which defines the compatibility constraints among lamps. As an example, the semantic action associated to production 4 checks if the lamps are positioned too close. In this case, a message is shown to the user. Thus, productions 3-6 define the nonterminal Lamps as a set of LAMP positioned within a MUNICIPALITY according to compatibility constraints. Similarly, productions 7-9 define the nonterminal Road as a set of ROAD symbols positioned within a MUNICIPALITY and related through a *touch* relationship. Productions 10 and 11 define the compatibility constraint between roads and lamps according to the classifications reported in Tables 1 and 2. In particular, each lamp is associated to road and has to satify the user-defined constraint *isIlluminated*. Notice that, for readability of productions, we have omitted the semantic actions that synthesize the attributes for the LHS nonterminal from the attributes of the RHS (non)terminals.

The parser automatically generated from the previous grammar is able to analyze the municipality, road, and lamp symbols positioned by the user on a map, as shown in Figure 4, and verify whether the previously described SICs are violated. As an example, when the parser analizes the lamps positioned on a map it applies productions 3 and 4 trying to reduce the LAMP terminal symbols into Lamps nonterminal symbols. If a lamp is too close to a lamp already analyzed (the spatial coordinates of the lamps previously analyzed by the parser are associated to Lamps' nonterminal symbol) then violation message is shown to the user.

When a SIC is violated by two or more geographical objects WiSPY shows a message with the information on the objects involved in the violation and the type of violation.

Thw WiSPY approach simplifies the specification and verification of SICs since the geographic application domain can be easily modeled with OMT-G class diagrams, the SICs can be specified as visual relationships between classes and customized using boolean conditions on attribute values, and the constraint checker can be automatically obtained from the annotated OMT-G model. In this way, the user can easily customize/add new SICs and rapidly prototyping new constraint checkers.

6 Related Work

The quality of spatial databases is an open problem in the field of geographic information systems and, in the last few decades, many efforts have been done to deal with implementation and management issues [15, 23]. In the following, we highlight the most important features of these works.

A constraint solver of spatial data based on programming logic has been presented in [1, 2]. The constraint system is able to handle the basic spatial types such as points, lines and polygons as well as the constraints in terms of equalities and inequalities, memberships, metric, topological and structural constraints. The system also provides a suitable theory for managing constraints and a set of transformation rules. The latter handle a special kind of constraints used for consistency checking, enabling an optimized and efficient resolution of spatial constraints.

In [12] a dimension graph representation is used for maintaining the spatial constraints among objects in an Euclidean space. The constraint consistency checking problem is transformed into a graph cycle detection problem on dimension graph.

The process for discovering inconsistencies in geographical dataset described in [19] consists of three steps: error definition, error checking, and error correction. Basically, the first step consists of the execution of some computational geometry algorithms, while the third one is solved by applying the first order calculus predicates.

In [14] a system developed for automatically maintaining topological constraints in a geographic database is presented. This system is based on extending to spatial data the notion of standard integrity maintenance through active databases. Topological relationships, defined by the users, are transformed into SICs, which are stored in the database as production rules. A similar approach is also introduces in [3].

An automated constraint checking procedure has been introduced by Udagepola *et al.* [21] to check constraint violations at compiling time before updating the database. It

is based on a data structure called Semantic Spatial Outlier R-Tree (SSRO-Tree).

In [17] Rigaux *et al.* presented Dedale, a constraint-based spatial database system relied on a linear constraints logical model. This system provides both an abstract data model and a user declarative query language based on SQL in order to represent and manipulate geometric data in arbitrary dimension. A different approach which combines relational and constraint data models is used in [10], where a three-tier constraint database architecture is presented. The latter increases the level of abstraction between the physical data and its semantics by introducing an additional layer to the classical relational data model architecture (logical and physical layer), which allows to manage both constraint-based and geometric data representations in the same layer of abstraction, in opposition to the pure constraint databases, where all data are represented in terms of constraints.

The framework presented in [20] allows the definition of hierarchical descriptions of abstract regions. To this aim, the framework exploits attributed grammars which can be translated by a compiler of compiler to a parser for abstract regions. Once generated, the parsers can be used for evaluating whether the incoming regions are consistent with the specified patterns. Basically, the abstract region candidates that were identified by the parsing rules can be evaluated to check if they conform to the definition provided by the user.

On the commercial side, Oracle® Spatial[9] allows spatial constraint checking by using either the PL/SQL language or by defining the constraint within the table procedure. ArcGIS[10] provides users with a button bar where it is possible to visually define simple constraints. More complex constraints have to be implemented by specific languages.

A significant part of the proposed WiSPY tool concerned with the visual definition of spatial constraints. The choice we made for this purpose is using the OMT-G modelling language. Similar to other approaches, it uses some visual formalisms for describing the spatial objects composing the geodatabase and others for connecting the objects specifying the relationships existing among them. We have chosen OMT-G [4] for its capability of explicitly specifying the constraints in associations and attributes [9], which is a limitation of the models extending UML [18], such as Ext. UML [16] and GeoFrame [8]. Moreover, OMT-G seems to be the most simply and user-friendly notation for non-expert constraint designers. Along this line, in [13] Lizardo and Davis presented a tool which provides various consistency checks on the integrity of the defined schema, and includes a function that maps OMT-G geographic concep-

tual schemas into physical schemas, including the SICs. Although, it seems very similar to our approach, it is based on SQL constraints which considerably limits the power of constraint checking.

7 Conclusions

In this paper we have proposed a system to support users in the automatic verification of SICs in geographic applications by exploiting visual language parsing. We have demonstrated, by implementing the WiSPY tool, that the visual language parsing is suitable for identifying violation in the PIP case study and for solving ambiguities that may arise in their interpretation. We have motivated our choice of having an entirely visual system, and highlighted its advantages. This choice represents the major difference between our proposal and the related work.

Our future work will focus on the extension of the current prototype in a fully functional product. Moreover, we will concentrate our efforts on finding appropriate solutions to present the information provided to the user, feedback and solutions, in a flexible and supportive manner.

References

[1] J. M. Almendros-Jiménez. Constraint logic programming over sets of spatial objects. In *Proceedings of the 2005 ACM SIGPLAN Workshop on Curry and Functional Logic Programming*, WCFLP '05, pages 32–42, New York, NY, USA, 2005. ACM.

[2] J. M. Almendros-Jiménez and A. Corral. Solving constraints on sets of spatial objects. In M. V. Hermenegildo and D. Cabeza, editors, *Practical Aspects of Declarative Languages, 7th International Symposium, PADL 2005, Long Beach, CA, USA, January 10-11, 2005, Proceedings*, volume 3350 of *Lecture Notes in Computer Science*, pages 158–173. Springer, 2005.

[3] A. Belussi, E. Bertino, and B. Catania. Manipulating spatial data in constraint databases. In M. Scholl and A. Voisard, editors, *Advances in Spatial Databases, 5th International Symposium, SSD'97, Berlin, Germany, July 15-18, 1997, Proceedings*, volume 1262 of *Lecture Notes in Computer Science*, pages 115–141. Springer, 1997.

[4] K. A. V. Borges, C. A. Davis, and A. H. F. Laender. OMT-G: an object-oriented data model for geographic applications. *GeoInformatica*, 5(3):221–260, 2001.

[5] K. A. V. Borges, A. H. F. Laender, and C. A. Davis. Spatial data integrity constraints in object oriented geographic data modeling. In C. B. Medeiros, editor, *ACM-GIS '99, Proceedings of the 7th International Symposium on Advances in Geographic Information Systems, November 2-6, 1999, Kansas City, USA*, pages 1–6. ACM, 1999.

[6] G. Costagliola, V. Deufemia, and G. Polese. Visual language implementation through standard compiler-compiler techniques. *J. Vis. Lang. Comput.*, 18(2):165–226, 2007.

---

[9] https://docs.oracle.com/cd/E18283_01/appdev.112/e11830/sdo_intro.htm#insertedID0

[10] https://sites.google.com/site/ochaimwiki/geodata-preparation-manual/how-to-check-topology-using-arcgis

[7] V. D. Fatto, V. Deufemia, and L. Paolino. Map integrity constraint verification by using visual language parsing. *JDIM*, 6(4):332–341, 2008.

[8] J. L. Filho and C. Iochpe. Specifying analysis patterns for geographic databases on the basis of a conceptual framework. In *Proceedings of the 7th ACM International Symposium on Advances in Geographic Information Systems*, GIS '99, pages 7–13, New York, NY, USA, 1999. ACM.

[9] A. Friis-Christensen, N. Tryfona, and C. S. Jensen. Requirements and research issues in geographic data modeling. In W. G. Aref, editor, *ACM-GIS 2001, Proceedings of the Ninth ACM International Symposium on Advances in Geographic Information Systems, Atlanta, GA, USA, November 9-10, 2001*, pages 2–8. ACM, 2001.

[10] D. Q. Goldin. Taking constraints out of constraint databases. In B. Kuijpers and P. Z. Revesz, editors, *Constraint Databases, Proceedings of the 1st International Symposium on Applications of Constraint Databases, CDB'04, Paris, June 12-13, 2004*, volume 3074 of *Lecture Notes in Computer Science*, pages 168–179. Springer, 2004.

[11] S. Johnson. *YACC: Yet Another Compiler Compiler*. Bell Laboratories, Murray Hills, NJ, 1978.

[12] X. Liu, S. Shekhar, and S. Chawla. Consistency checking for euclidean spatial constraints: a dimension graph approach. In *12th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2000), 13-15 November 2000, Vancouver, BC, Canada*, page 333. IEEE Computer Society, 2000.

[13] L. E. O. Lizardo and C. A. D. Jr. OMT-G designer: A web tool for modeling geographic databases in OMT-G. In M. Indulska and S. Purao, editors, *Advances in Conceptual Modeling - ER 2014 Workshops, ENMO, MoBiD, MReBA, QMMQ, SeCoGIS, WISM, and ER Demos, Atlanta, GA, USA, October 27-29, 2014. Proceedings*, volume 8823 of *Lecture Notes in Computer Science*, pages 228–233. Springer, 2014.

[14] C. B. Medeiros and M. Cilia. Maintenance of binary topological constraints through active databases. In *Proceedings of the 3rd ACM International Workshop on Advances in Geographic Information Systems, Baltimore, Maryland, December 1-2, 1995, in conjunction with CIKM 1995.*, page 127, 1995.

[15] L. Plümer and G. Gröger. Achieving integrity in geographic information systems maps and nested maps. *GeoInformatica*, 1(4):345–367, 1997.

[16] R. Price, N. Tryfona, and C. S. Jensen. Extended spatiotemporal UML: motivations, requirements and constructs. *J. Database Manag.*, 11(4):14–27, 2000.

[17] P. Rigaux, M. Scholl, L. Segoufin, and S. Grumbach. Building a constraint-based spatial database system: model, languages, and implementation. *Inf. Syst.*, 28(6):563–595, 2003.

[18] J. Rumbaugh, I. Jacobson, and G. Booch. *Unified Modeling Language Reference Manual, The (2Nd Edition)*. Pearson Higher Education, 2004.

[19] S. Servigne, T. Ubeda, A. Puricelli, and R. Laurini. A methodology for spatial consistency improvement of geographic databases. *GeoInformatica*, 4(1):7–34, 2000.

[20] J. Steinhauer, T. Wiese, C. Freksa, and T. Barkowsky. Recognition of abstract regions in cartographic maps. volume 2205 of *Lecture Notes in Computer Science*, pages 306–321. Springer, 2001.

[21] K. P. Udagepola, L. Xiang, L. H. Wei, and Y. X. Zong. Efficient management of spatial databases by data consistency and integrity constraints. *WSEAS Transactions on Computers*, 5(2):447–454, 2006.

[22] uDig. User-friendly desktop internet gis. `http://udig.refractions.net/`, 2004.

[23] M. F. Worboys. A unified model for spatial and temporal information. *Comput. J.*, 37(1):36–34, 1994.

GO-Bayes Method for System Modeling and Safety Analysis

Guoqiang Cai, Limin Jia, Hui Zhen,

Mingming Zheng, Shuai Feng

State Key Lab of Rail Traffic Control & safety
Beijing Jiaotong University, Beijing, China
guoqiangcai@163.com

MengChu Zhou
Department of Electrical and Computer Engineering
New Jersey Institute of Technology, Newark, NJ 07102,
USA
zhou@njit.edu

Abstract—Safety analysis ensuring the normal operation of an engineering system is important. The existing safety analysis methods are limited to relatively simple fact description and statistical induction level. Besides, many of them enjoy poor generality, and fail to achieve comprehensive safety evaluation given a system structure and collected information. This work describes a new safety analysis method, called a GO-Bayes algorithm. It combines structural modeling of the GO method and probabilistic reasoning of the Bayes method. It can be widely used in system analysis. The work takes a metro vehicle braking system as an example to verify its usefulness and accuracy. Visual implementation by Extendsim software shows its feasibility and advantages in comparison with the Fault Tree Analysis (FTA) method.

Keywords: **Safety analysis, GO-Bayes method, and Reliability**

I. INTRODUCTION

Safety evaluation technologies were originated in the 1930s. In the 1960s, the needs from the US military field's engineering system safety theory and applications promoted their rapid development[1]. As people's awareness of safety continues to grow and system safety engineering becomes a mature discipline, a system safety engineering approach is gradually extended to aviation, nuclear industry, petroleum, chemical, and manufacturing areas. Researchers have proposed new theories and methods, such as safety checklist[2], safety analysis[3] and evaluation methods[4], event trees[5], fault trees[6] and risk assessment techniques[7-8], mode evaluation, six-stage safety and other risk index evaluation method, artificial neural networks and other technologies.

The GO method has commonly been used since 1980s [9]. Several improved methods for quantitative analysis are proposed in signal processing[10]. This work intends to improve the GO algorithm based on Bayes reasoning[11-13] and names the new method as a GO-Bayes algorithm. It has the following innovative characteristics:

First, the structural modular reliability analysis of the GO method is applied to analyze the operational status of a safety analysis assessment system; Second, the Bayes probability theory is used in a safe state probability parameter extraction process to each basic unit of the model; Third, the Bayes inference is integrated into the system GO graph model, reversing fault reasoning analysis and evaluation, thereby achieving simpler quantitative analysis. The proposed GO-Bayes method combines the structural modeling of the GO method and probabilistic reasoning of the Bayes method[14], which can be used in situations where one has a large amount of system fault information. Its use can help one prevent and diagnose faults in a timely fashion, thus ensuring the safe operation of an entire system.

II. GO-BAYES METHOD

The proposed GO-Bayes method is system-unit-component failures oriented. It combines basic unit models and logic analysis models according to a flow chart to establish the analysis model, in accordance with certain rules to calculate reliability parameters. Besides we adopt Bayes methods to deduce system failure and solve inverse probability, in order to achieve a comprehensive system safety evaluation. The GO-Bayes method's operators are shown in Figure 1.

A. Modeling method

The GO-Bayes method inherits graph modeling ideas, e.g., schematic diagrams, flow charts and other drawings. First, we summarize the basic model elements, and explain the unit algorithm. Second, we build a system model according to a system structure and data flows among its units. We use system modeling algorithms to process raw input data and then obtain system outputs according to the working mechanism and fault conditions.

B. Bayes theory based on information fusion

Information fusion research based on the Bayes theory is mainly used for system internal self-monitor and self-test information. The information (hereinafter collectively referred to as detection information) plays a strong role in system operation safety analysis. The GO-Bayes method is based on the description and information of each component and subsystem, and the model is systematically analyzed. Fault information related to the system reliability and safety is integrated for system reliability analysis.
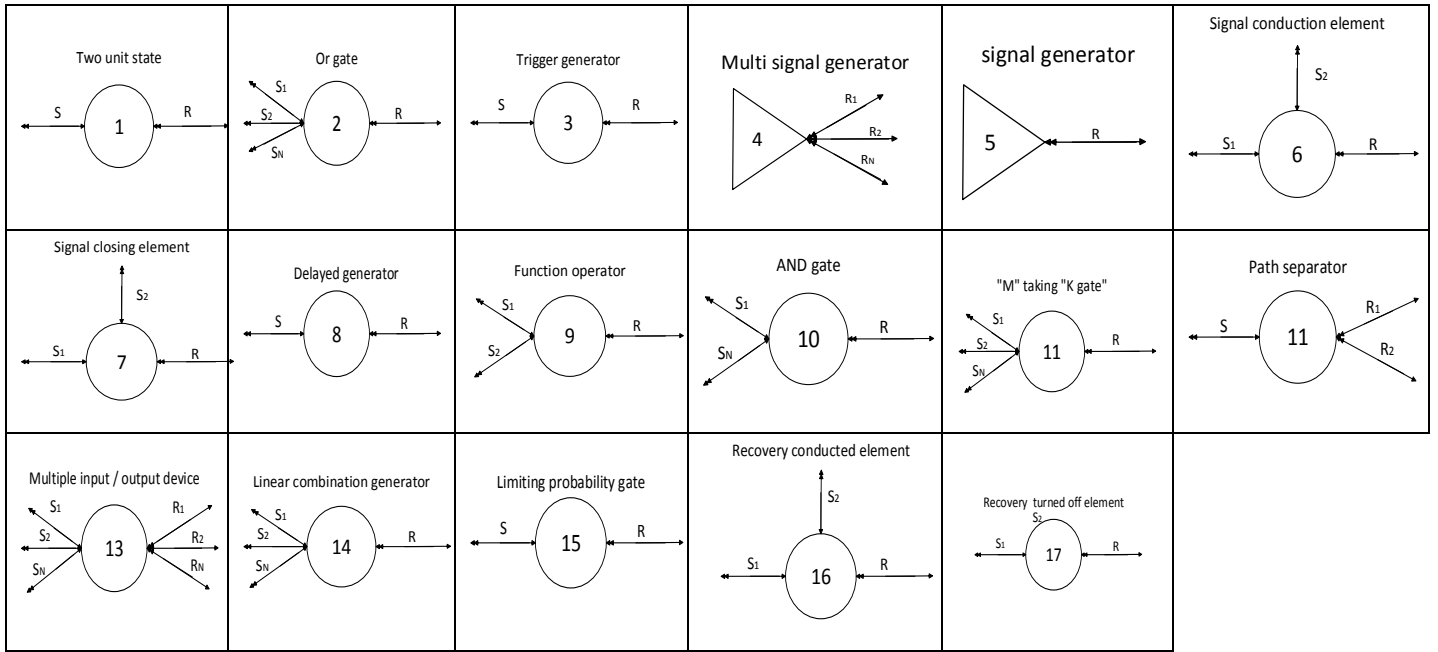
Figure 1. The operators used in the GO-Bayes method

C. Probabilistic Inference based on GO-Bayes

System probabilistic safety evaluation can be realized in two ways. First, from components to systems, based on the probability parameters of component parts, we solve probability parameters of a system, such as normal work probability and failure probability. Second, from the system to the components, based on known system state information and component probability parameters, we reason a system's various safety status probabilities, i.e., "inverse probability".

III. INTRODUCTION TO UNIT MODEL

When a basic unit is described, its probability data follows the following principles [15-16]. We use the following notation: S is the data unit subscript, like R_s, F_s and P_s; I is the input data subscript, like R_I, F_I and P_I; and O is the output data subscript, like R_o, F_o and P_o.

A. Signal generating unit

A signal generating unit means an input to a system, external event or signal independent of the system. It can represent a generator, power, environmental impact and human factors. It has two states, normal or faulty. Its safety probability parameter comprises, Unreliability $F(1)$, inverse probability $P(1)$. Its single arrow output indicates an unreliability output, double arrow indicates an inverse probability input, satisfying:

$$F_O(1) = F_S(1) \qquad (1)$$

$$P_S(1) = P_I(1) \qquad (2)$$

Figure 2 means a signal generating unit model, and Figure 3 means a signal generator unit.
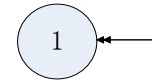

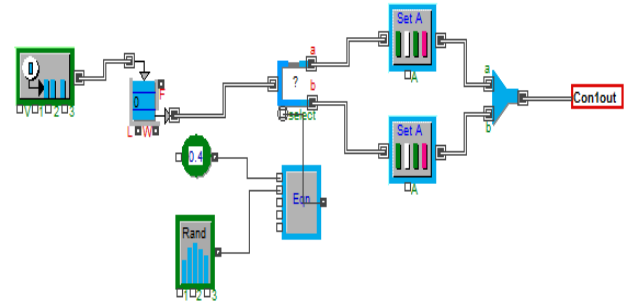Figure 2. A signal generating unit model

Figure 3. A Signal generator unit

B. Two state unit

As shown in Figures 4-5, a two-state unit is the most common unit, whose two states are normal and faulty ones. It has input and output data, and can represent resistors, switches, and valves. Its unreliability value is calculated based on the reliability theory,

$$F_o(2) = 1 - [1 - F_S(2)][1 - F_I(2)] \qquad (3)$$

Two-state unit output failure results from either input fault or its own fault. They form a series logical relationship with the inverse probability

$$P_S(2) = \frac{F_S(2)P_I(2)}{F_o(2)} \qquad (4)$$

50

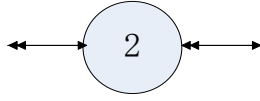$$P_o(2) = \frac{F_I(2)P_I(2)}{F_o(2)} \qquad (5)$$
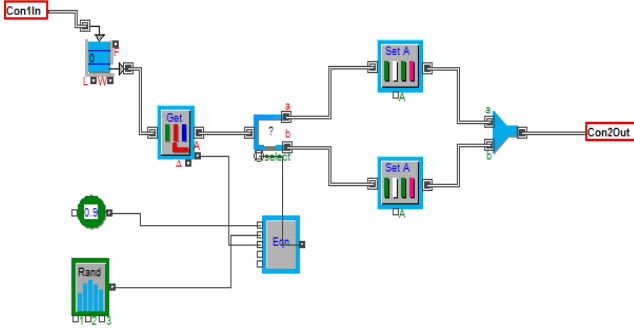

Figure 4. A Two-state unit model

Figure 5. A two-state unit

C. Conditional control unit

A conditional control unit, as shown in Figures 6-7, requires two inputs, the working status input, with a subscript label 1, and the control state input, with a subscript label 2. Its output represents their safety status. A conditional control unit may represent relay and mechanical control valves and so on. Its probability parameter calculation rules as follows:

$$F_o(3) = 1 - [1 - F_{I1}(3)][1 - F_{I2}(3)][1 - F_s(3)] \qquad (6)$$

$$P_s(3) = \frac{F_s(3)P_I(3)}{F_o(3)} \qquad (7)$$

$$P_{O1}(3) = \frac{F_{I1}(3)P_I(3)}{F_o(3)} \qquad (8)$$

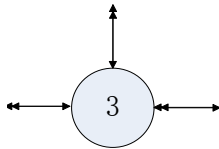$$P_{O2}(3) = \frac{F_{I2}(3)P_I(3)}{F_o(3)} \qquad (9)$$
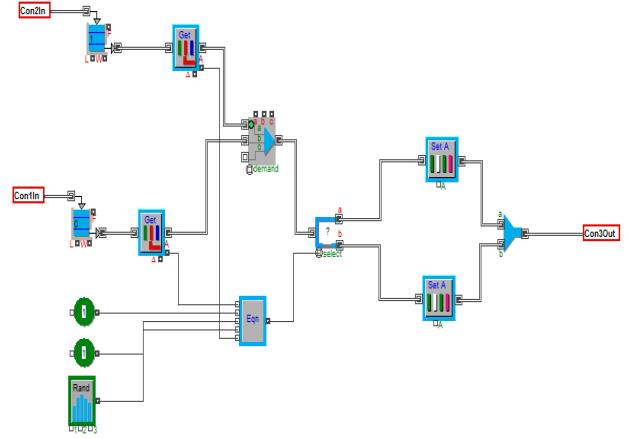

Figure 6. A Conditional control unit model

Figure 7. A Conditional control unit

D. AND gate

An AND gate unit is shown in Figures 8-9. It can rely on several reliability input data items (with subscript labels being 1, 2, 3, …, n), to compute one reliability output. It yields an output only when multiple inputs simultaneously are available. It does not have its own data. It does not stand for an internal system component, but is used to connect different units. Its reverser fault data is expressed as an input and multiple output. Obviously, an AND gate unit represents a parallel logical relationship. Its probability parameters can be computed:

$$F_o(4) = F_{I1}(4)F_{I2}(4)...F_{In}(4) \qquad (10)$$

$$P_{O1}(4) = P_I(4) \qquad (11)$$

$$P_{O2}(4) = P_I(4) \qquad (12)$$
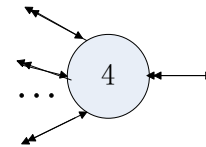
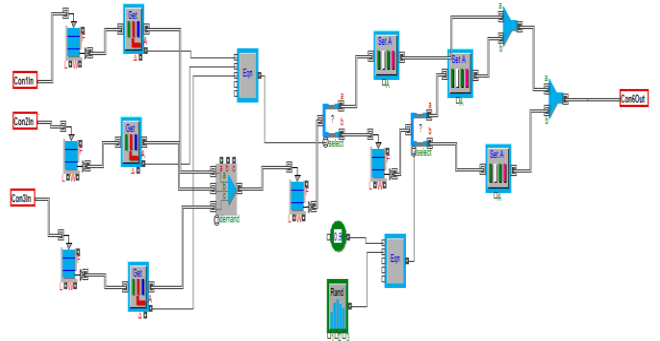$$P_{On}(4) = P_I(4) \qquad (13)$$

Figure 8. An AND gate unit model

Figure 9. An AND gate unit

E. OR gate

An OR gate unit relies on several reliability input data (with a subscript label 1, 2, 3,…, n), to compute reliability output data, as shown in Figures 10-11. When one of the multiple inputs occurs, it can yield an output. It does not have its own data, and stand for no internal system component, but can be used to connect multiple units. Its reverser fault data is expressed as an input and multiple outputs. Probability parameters are calculated as follows:

$$F_o(5) = 1 - [1 - F_{I1}(5)][1 - F_{I2}(5)]...[1 - F_{In}(5)] \quad (14)$$

$$P_{O1}(5) = \frac{F_{I1}(5)P_I(5)}{F_o(5)} \quad (15)$$

$$P_{O2}(5) = \frac{F_{I2}(5)P_I(5)}{F_o(5)} \quad (16)$$

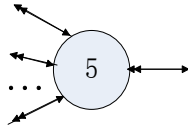$$P_{On}(5) = \frac{F_{In}(5)P_I(5)}{F_o(5)} \quad (17)$$
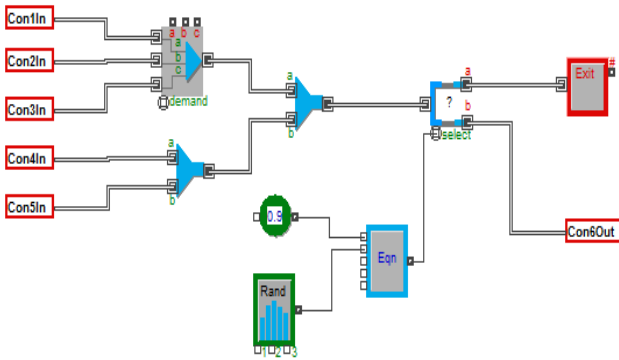

Figure 10. An OR gate unit model

Figure 11. An OR gate unit

F. Voting gate

A voting gate unit as shown in Figures 12-13 has several reliability input data items (with a subscript label 1, 2, 3, …, n), and one output. It produces an output only when more than k inputs are present at the same time. It does not have its own data, and stands for no internal system component, but it can be used to connect multiple units. Its reverse fault data is expressed as an input and multiple outputs. It represents a parallel and series logical relationship. It can be divided into a combination of AND gate units and OR gate units. For example taking 2 from 4 has $C_4^2 = 6$ options, two AND gate units connect to one OR gate unit, meaning that two or more input failures lead to system output failure.
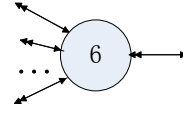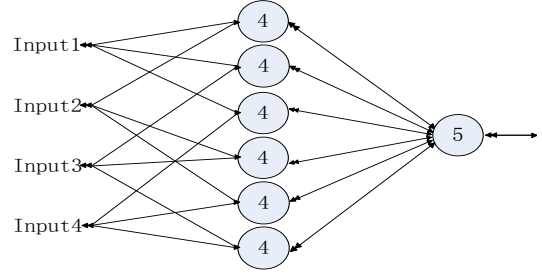

Figure 12. A Voting gate unit model

Figure 13. A 4-input series-parallel voting gate model

Its probability parameters can be derived from those for AND gate and OR gate units. We derive an algorithm for a GO-Bayes basic voting model.

Take 2 from 4 as an example in Figure 13. It can be divided into a combination of 6 AND gate units and 1 OR gate unit.

Assuming that the probabilities of inputs 1-4 are $R_{I1}(6)=x_1$, $R_{I2}(6)=x_2$, $R_{I3}(6)=x_3$, $R_{I4}(6)=x_4$, we can derive the probability formula for the 2/4 voting gate $F_o(6)$,

$$F_o(6) = 1 - [1 - (1 - x_1) \cdot (1 - x_2)] \cdot [1 - (1 - x_1) \cdot (1 - x_3)] \cdot$$
$$[1 - (1 - x_1) \cdot (1 - x_4)] \cdot [1 - (1 - x_2) \cdot (1 - x_3)] \cdot \quad (18)$$
$$[1 - (1 - x_2) \cdot (1 - x_4)] \cdot [1 - (1 - x_3) \cdot (1 - x_4)]$$

thus

$$F_o(6) = 1 - (x_1 \cdot x_2 \cdot x_3 + x_1 \cdot x_2 \cdot x_4 + x_1 \cdot x_3 \cdot x_4 + x_2 \cdot x_3 \cdot x_4 - 3x_1 \cdot x_2 \cdot x_3 \cdot x_4) \quad (19)$$

Since the jointly signal has no effect on the calculation of the reverse probability, according to the reverse probability formula of an AND gate unit (label 4) and OR gate unit (label 5), we can derive the following reverse probability of the gate:

$$P_I(1 \cdot 2) = \frac{F_I(1 \cdot 2)P_I(6)}{F_o(6)} = \frac{(1 - x_1) \cdot (1 - x_2) \cdot P_I(6)}{F_o(6)} \quad (20)$$

$$P_I(1 \cdot 3) = \frac{F_I(1 \cdot 3)P_I(6)}{F_o(6)} = \frac{(1 - x_1) \cdot (1 - x_3) \cdot P_I(6)}{F_o(6)} \quad (21)$$

$$P_I(1 \cdot 4) = \frac{F_I(1 \cdot 4)P_I(6)}{F_o(6)} = \frac{(1 - x_1) \cdot (1 - x_4) \cdot P_I(6)}{F_o(6)} \quad (22)$$

$$P_I(2 \cdot 3) = \frac{F_I(2 \cdot 3)P_I(6)}{F_o(6)} = \frac{(1 - x_2) \cdot (1 - x_3) \cdot P_I(6)}{F_o(6)} \quad (23)$$

$$P_I(2 \cdot 4) = \frac{F_I(2 \cdot 4)P_I(6)}{F_o(6)} = \frac{(1-x_2) \cdot (1-x_4) \cdot P_I(6)}{F_o(6)} \quad (24)$$

$$P_I(3 \cdot 4) = \frac{F_I(3 \cdot 4)P_I(6)}{F_o(6)} = \frac{(1-x_3) \cdot (1-x_4) \cdot P_I(6)}{F_o(6)} \quad (25)$$

$$P_{I1}(6) = \frac{F_{I1}(6)}{(1-x_1) \cdot (1-x_2)} \cdot \frac{(1-x_1) \cdot (1-x_2) \cdot P_I(6)}{F_o(6)} = \frac{F_{I1}(6) \cdot P_I(6)}{F_o(6)} \quad (26)$$

$$P_{I2}(6) = \frac{F_{I2}(6)}{(1-x_1) \cdot (1-x_2)} \cdot \frac{(1-x_1) \cdot (1-x_2) \cdot P_I(6)}{F_o(6)} = \frac{F_{I2}(6) \cdot P_I(6)}{F_o(6)} \quad (27)$$

$$P_{I3}(6) = \frac{F_{I3}(6)}{(1-x_1) \cdot (1-x_3)} \cdot \frac{(1-x_1) \cdot (1-x_3) \cdot P_I(6)}{F_o(6)} = \frac{F_{I3}(6) \cdot P_I(6)}{F_o(6)} \quad (28)$$

$$P_{I4}(6) = \frac{F_{I4}(6)}{(1-x_1) \cdot (1-x_4)} \cdot \frac{(1-x_1) \cdot (1-x_4) \cdot P_I(6)}{F_o(6)} = \frac{F_{I4}(6) \cdot P_I(6)}{F_o(6)} \quad (29)$$

Then we get a 2/4 vote gate algorithm. We can obtain the similar results for other voting gates.

IV. SAFETY ANALYSIS OF VISUAL UNIT METRO VEHICLES BRAKING SYSTEM

We now show how to use the proposed GO-Bayes method to analyze an urban rail transit vehicle air braking part.

A. Basic composition of air braking

Air braking portion of a braking system's basic components include air compressor and filtration device (as shown in A5 of Figure 14), total duct, air spring devices and pneumatic part (beginning with L in Figure 14), parking braking device (B7), braking control section (B13), braking airline (beginning with B), foundation brakes (beginning with C), and electronic anti-skid devices (beginning with G).

We build the visual system for an urban metro vehicle braking system as shown in Figure 14.
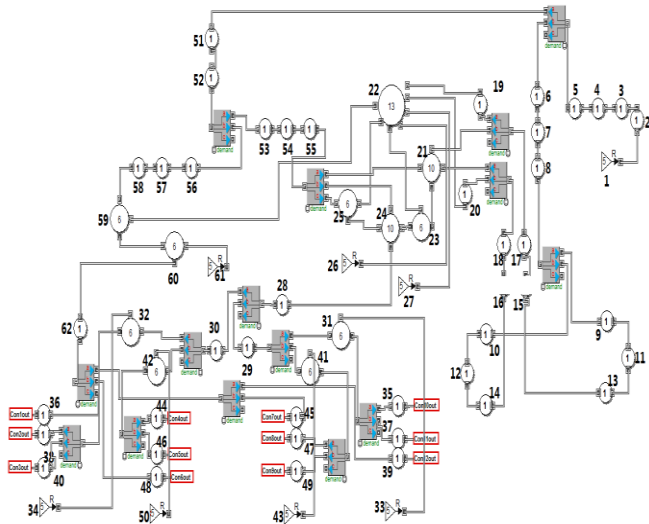

Figure 14. Visual system of an urban metro vehicle braking system

B. GO-Bayes modeling method for a braking system

The braking system has a complex structure and many components. In order to display and analyze it fully, this work uses a hierarchical modeling method by dividing the braking system into two layers. Its first layer has six functional sections as shown in Figure 15.
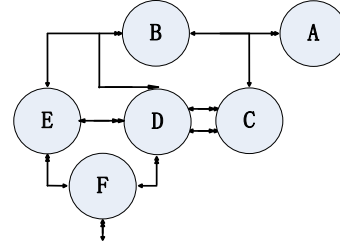

Figure 15. First layer structure model of a braking system

In Figure 15, node A represents an air supply device, B the line along which braking air passes through, C the air spring suspension, D the braking control device, E parking braking control, and F the foundation braking.

In the second layer of the model structure as shown in Figure 16, since the number of components is big, we label them according to the labels in the first parts and the position in the device. Numbers on the left of the dash represents unit types, and those on the right side correspond to the system unit.

(1) An air supply device is shown in Figure 16 and Table 1.
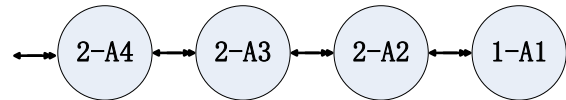
Figure 16. Structure model of an air supply device

TABLE 1 Units in an air supply device model

Code	Corresponding component
1-A1	Drive motor
2-A2	Air compressor
2-A3	Drying tower
2-A4	total air cylinder

(2) A braking air route is given in Figure 17 and Table 2.

Figure 17. Braking air route structure model

TABLE 2 Components in a braking air route structure model

Code	Corresponding component
2-B1	Total air duct
2-B2	Cut-off valve
2-B3	Safety valve
3-B4	Braking reservoir cylinder
1-B5	Exhaust valve

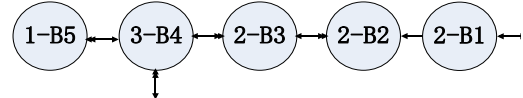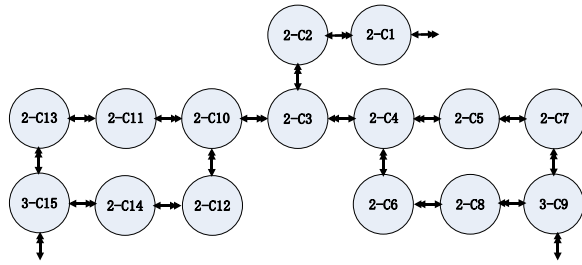(3) An air spring suspension device is shown in Figure 18 and Table 3.

Figure 18. Structure model of an air spring suspension device

TABLE 3 Components in an air spring suspension device model

Code	Corresponding component
2-C1	Cut-off valve
2-C2	Filter
2-C3	Air spring cylinder
2-C4	Cut-off valve
2-C5	Left height valve
2-C6	Right height valve
2-C7	Air spring
2-C8	Air spring
3-C9	Pressure valve
2-C10	Cut-off valve
2-C11	Left height valve
2-C12	Right height valve
2-C13	Air spring
2-C14	Air spring
3-C15	Pressure valve

(4) A braking control device inner has its structure and components shows in Figure 19 and Table 4.
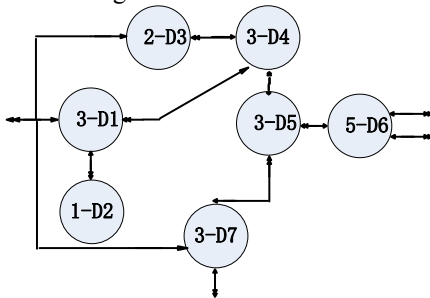

Figure 19. Structure model of a braking control system

TABLE 4 Components in a braking control system model

Code	Corresponding component
3-D1	Analog converter
1-D2	ECU code
2-D3	Emergency solenoid valve
3-D4	Pressure Switch
3-D5	Weighing valve
5-D6	OR gate
3-D7	Relay valve

(5) A parking braking device has its structure and components in Figure 20 and Table 5.
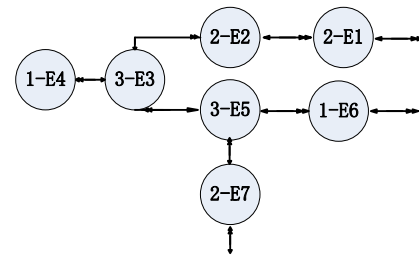
Figure 20. Structure model of a parking braking device

TABLE 5 Components in a braking device model

Code	Corresponding component
2-E1	Cut off valve
2-E2	Pressure Switch
3-E3	Parking braking solenoid valve
1-E4	Parking braking code
3-E5	Pulse valve
1-E6	Two-way valve
2-E7	Check

(6) A Foundation Braking is given in Figure 21 and Table 6.
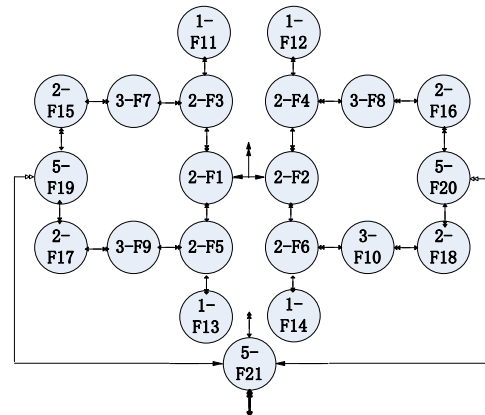
Figure 21.Structure model of a foundation braking device

TABLE 6. Components in a foundation braking device model

Code	Corresponding component
2-F1	Cut off valve
2-F2	Cut off valve
3-F3	Slip solenoid valve
3-F4	Slip solenoid valve
3-F5	Slip solenoid valve
3-F6	Slip solenoid valve
2-F7	Braking air reservoir
2-F8	Braking air reservoir
2-F9	Braking air reservoir
2-F10	Braking air reservoir
1-F11	Speed Sensor
1-F12	Speed Sensor
1-F13	Speed Sensor
1-F14	Speed Sensor
2-F15	Slipper
2-F16	Slipper
2-F17	Slipper
2-F18	Slipper
5-F19	Or gate
5-F20	Or gate
5-F21	Or gate

Another feature of the hierarchical model is that each of its modules can be individually analyzed. During the entire system analysis procedure, the correlations among modules have to be paid attention to.

C. Calculation of probability indicators

Safety probability indicators are computed based on the GO-Bayes model of the braking system. First, obtain the fault parameters for each component by statistically analyzing historical operating cumulative data of the system. Component fault rate is the total number of the system failures divided by the total number of components, and then divided by the time duration. Secondly, we can calculate the component fault probability and normal work probability at time k according to the correlations among failure rate indices.

(1) Original data

Component failure rate data for each component comes mainly from the historical operation statistics. Assume steady-state operation 100h as shown in Table 7,

TABLE 7 Initial data of components in the model

Code	Failure Rate(10E-06/h)	Fault Rate
1-A1	3	2.9996E-04
2-A2	4.5	4.4990E-04
2-A3	6	5.9982E-04
2-A4	0.5	4.9999E-05
2-B1	1	9.9995E-05
2-B2	1.2	1.1999E-04
2-B3	2.5	2.4997E-04
3-B4	0.7	6.9998E-05
1-B5	8	7.9968E-04
2-C1	3	2.9996E-04
2-C2	1	9.9995E-05
2-C3	0.5	4.9999E-05
2-C4	3	2.9996E-04
2-C5	10	9.9950E-04
2-C6	10	9.9950E-04
2-C7	1.5	1.4999E-04
2-C8	1.5	1.4999E-04
3-C9	5	4.9988E-04
2-C10	3	2.9996E-04
2-C11	10	9.9950E-04
2-C12	10	9.9950E-04
2-C13	1.5	1.4999E-04
2-C14	1.5	1.4999E-04
3-C15	5	4.9988E-04
3-D1	2	1.9998E-04
1-D2	42	4.1912E-03
2-D3	9	8.9960E-04
3-D4	4	3.9992E-04
3-D5	0.8	7.9997E-05
3-D7	2	1.9998E-04
2-E1	3	2.9996E-04
2-E2	4	3.9992E-04
3-E3	3.5	3.4994E-04
1-E4	1	9.9995E-05
3-E5	3	2.9996E-04
1-E6	7	6.9976E-04
2-E7	1.2	1.1999E-04
2-F1	3	2.9996E-04
2-F2	3	2.9996E-04
3-F3	3.5	3.4994E-04
3-F4	3.5	3.4994E-04
3-F5	3.5	3.4994E-04
3-F6	3.5	3.4994E-04
2-F7	1	9.9995E-05
2-F8	1	9.9995E-05
2-F9	1	9.9995E-05
2-F10	1	9.9995E-05
1-F11	15	1.4989E-03
1-F12	15	1.4989E-03
1-F13	15	1.4989E-03
1-F14	15	1.4989E-03
2-F15	9	8.9960E-04
2-F16	9	8.9960E-04
2-F17	9	8.9960E-04
2-F18	9	8.9960E-04

(2) Calculation results

Based on the above model structure, we calculate the reliability and unreliability of each component's output, and then reverse reasoning to obtain each component's input probabilities. We can obtain the system output reliability that is 9.7079E-01, unreliability is 2.9205E-02. In Table 8, the inverse probability of the following components is relatively larger. That is to say, (1-D2, 1.4351E-01), (2-D3, 3.0803E-02), (1-F11, 1-F12, 1-F13, 1-F14, 5.1322E-02), (2-F15, 2-F16, 2-F17, 2-F18, 3.0803E-02) indicate the component fault will most likely lead to system fault. Hence, we have to focus on tracking them.

TABLE 8 Safety analysis results of braking system

Code	Output unreliability (Cumulative probability of fault)	Output reliability (Normal work probability)	Component input inverse Probability	Component inverse Probability
1-A1	2.9996E-04	9.9970E-01	1.0271E-02	1.0271E-02
2-A2	7.4972E-04	9.9925E-01	2.5671E-02	1.5405E-02
2-A3	1.3491E-03	9.9865E-01	4.6194E-02	2.0538E-02
2-A4	1.3990E-03	9.9860E-01	4.7903E-02	1.7120E-03
2-B1	1.4989E-03	9.9850E-01	5.1322E-02	3.4239E-03
2-B2	1.6187E-03	9.9838E-01	5.5425E-02	4.1086E-03
2-B3	1.8683E-03	9.9813E-01	6.3970E-02	8.5591E-03
3-B4	2.7362E-03	9.9726E-01	9.3691E-02	2.3968E-03
1-B5	7.9968E-04	9.9920E-01	2.7382E-02	2.7382E-02
2-C1	1.6986E-03	9.9830E-01	5.8160E-02	1.0271E-02
2-C2	1.7984E-03	9.9820E-01	6.1578E-02	3.4239E-03
2-C3	1.8483E-03	9.9815E-01	6.3287E-02	1.7120E-03
2-C4	2.1477E-03	9.9785E-01	7.3538E-02	1.0271E-02
2-C5	3.1450E-03	9.9685E-01	1.0769E-01	3.4224E-02
2-C6	3.1450E-03	9.9685E-01	1.0769E-01	3.4224E-02
2-C7, 2-C8	4.1414E-03	9.9586E-01	1.4180E-01	5.1357E-03

3-C9	4.9378E-03	9.9506E-01	1.6907E-01	1.7116E-02
2-C10	2.1477E-03	9.9785E-01	7.3538E-02	1.0271E-02
2-C11, 2-C12	3.1450E-03	9.9685E-01	1.0769E-01	3.4224E-02
2-C13, 2-C14	4.1414E-03	9.9586E-01	1.4180E-01	5.1357E-03
3-C15	4.9378E-03	9.9506E-01	1.6907E-01	1.7116E-02
3-D1	7.1146E-03	9.9289E-01	2.4361E-01	6.8474E-03
1-D2	4.1912E-03	9.9581E-01	1.4351E-01	1.4351E-01
2-D3	3.6334E-03	9.9637E-01	1.2441E-01	3.0803E-02
3-D4	8.0078E-03	9.9199E-01	2.7419E-01	1.3694E-02
3-D5	1.5056E-02	9.8494E-01	5.1551E-01	2.7391E-03
5-D6	6.6279E-03	9.9337E-01	2.2694E-01	0
3-D7	1.5252E-02	9.8475E-01	5.2226E-01	6.8474E-03
2-E1	3.0354E-03	9.9696E-01	1.0393E-01	1.0271E-02
2-E2	3.4341E-03	9.9657E-01	1.1759E-01	1.3694E-02
3-E3	3.8824E-03	9.9612E-01	1.3294E-01	1.1982E-02
1-E4	9.9995E-05	9.9990E-01	3.4239E-03	3.4239E-03
3-E5	1.7367E-02	9.8263E-01	5.9467E-01	1.0271E-02
1-E6	1.5942E-02	9.8406E-01	5.4585E-01	2.3960E-02
2-E7	1.7485E-02	9.8251E-01	5.9871E-01	4.1086E-03
2-F1,2-F2	1.7780E-02	9.8222E-01	6.0880E-01	1.0271E-02
3-F3, 3-F4, 3-F5, 3-F6	1.9595E-02	9.8040E-01	6.7096E-01	1.1982E-02
2-F7, 2-F8, 2-F9, 2-F10	1.9693E-02	9.8031E-01	6.7432E-01	3.4239E-03
1-F11, 1-F12, 1-F13, 1-F14	1.4989E-03	9.9850E-01	5.1322E-02	5.1322E-02
2-F15, 2-F16, 2-F17, 2-F18	2.0575E-02	9.7942E-01	7.0451E-01	3.0803E-02
5-F19	2.3363E-02	9.7664E-01	7.9996E-01	0
5-F20	2.3363E-02	9.7664E-01	7.9996E-01	0
5-F21	2.9205E-02	9.7079E-01	1.0000E+00	0

D. Experimental Analysis

We can conclude from the above safety analysis:

(1) When a system shows abnormal conditions, we have to obtain real-time inverse probability through the fault backward reasoning method. The inverse probability of components (3-B4, 1.3070E-02), (1-B5, 1.4932E-01), (3-D1, 3.7340E-02), (1-D2, 7.8258E-01), and (1-F11, 4.7087E-01) is significantly larger than the others', which shows that these parts may be abnormal. We should thus focus on tracking them. In addition by using the system diagram model to analyze 3-B4, 1-B5, 3-D1, and 1-D2, which are working parts connected together, the abnormal output of 3-D1 indicates that the failure possibility of these four components is very large, and the failure possibility of (1-D2, 7.8258E-01) is the highest. It represents Electronic Control Unit instruction, error rate of which is higher, because it has many electronic circuit components. While (1-F11, 4.7087E-01) is an independent failure, in fact, it represents the speed sensor with a self-resetting function. Its false detection occurs frequently. If an abnormal event is detected when its probability of failure is less than 1/2, we should check and maintain them.

(2) Traditional fault probability calculation depends on the forward deduction of historic data. By contrast, the GO-Bayes method provides structural models of a system and inverse reasoning probability. The models' output and inverse probability reflect more accurately the system's reliability than traditional fault probability. Figure 22 is a metro train's braking system based on FTA. Table 9 shows GO-Bayes' advantage compared with FTA.
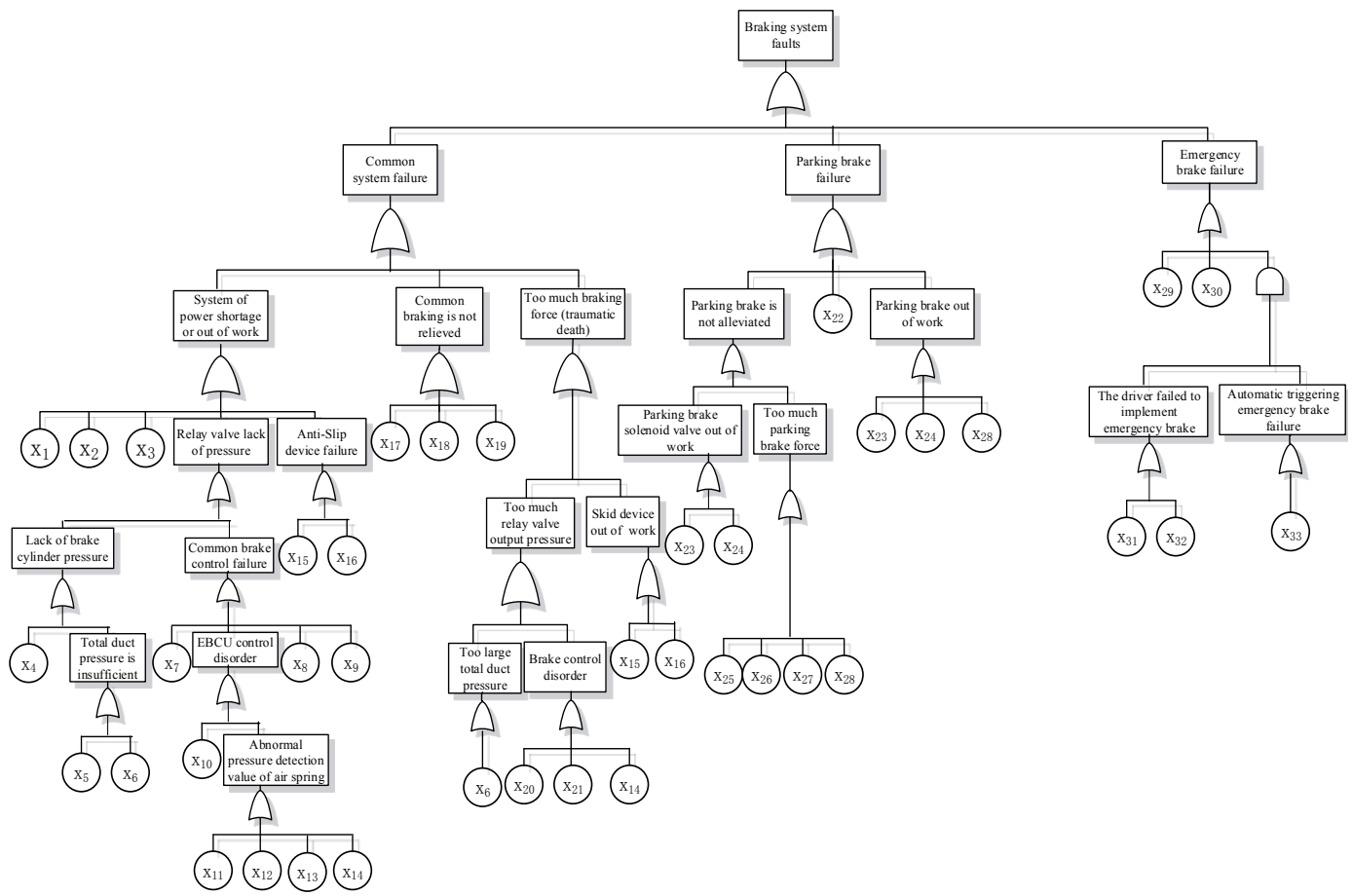
Figure 22 A metro train's braking system based on FTA

TABLE 9 GO-Bayes compared with FTA

Model Feature	GO- Bayes	FTA
Modeling oriented	success	Failure
Modeling method	bi-direction induction	deductive
Modeling consistency	basically identical	differences according to everyone's understanding
Structure	similar principle diagram	hierarchical logic diagram
Volume	compact, small size	multi-layer, huge volume
Elements	component, logic diagram	fault event, logic gate
Description	reflect original system structure	reflect the failure cause and effect
Notation	more operators with rich expression	less operators with poor expression

V. CONCLUSION

This paper presents a new structural GO-Bayes method. It is a comprehensive system safety analysis and evaluation modeling methodology. Using a system diagram model, we can obtain the system's normal work probability output, which is essential for fault backward reasoning. The paper discusses basic components or units and their related analysis results. The application of the proposed method to a metro vehicle braking system shows its contribution to safety analysis and assessment. The results can be used to trace, maintain and improve system components and eventually ensure the entire system's safe operation.

REFERENCES

[1] Yang, J.; Wu, Z.Z.; Chen, T.Z. Review of the urban road traffic safety evaluation methods. Proceedings of the Fourth International Conference on Transportation Engineering.December 2013, pp.2503-2508.

[2] Pugel, A.E.; Simianu, V.V.; Flum, D.R. Use of the surgical safety checklist to improve communication and reduce complications. Journal of infection and public health, 2015, 8(3), pp.219-225.

[3] Long, J.B.; Birmingham, P.K.; De Oliveira, Jr.G.S. Transversus Abdominis Plane Block in Children: A Multicenter Safety Analysis of 1994 Cases From the PRAN (Pediatric Regional Anesthesia Network) Database. Survey of Anesthesiology, 2015, 59(3), pp.139-140.

[4] Saunders, R.P.; Wilcox, S.; Baruth, M. Process evaluation methods, implementation fidelity results and relationship to physical activity and healthy eating in the Faith, Activity, and Nutrition (FAN) study. Evaluation and program planning, 2014, pp.43: 93-102.

[5] Ibánez, L.; Hortal, J.; Queral, C. Application of the Integrated Safety Assessment Methodology to Safety Margins. Dynamic Event Trees,

Damage Domains and Risk Assessment. Reliability Engineering & System Safety, 2015.

[6] Purba, J.H.; Lu, J.; Zhang, G. A fuzzy reliability assessment of basic events of fault trees through qualitative data processing. Fuzzy Sets and Systems, 2014, 243, pp.50-69.

[7] Appelbaum, P.S.; Robbins, P.C.; Monahan, J. Violence and delusions: Data from the MacArthur violence risk assessment study. American Journal of Psychiatry, 2015.

[8] Zhou, L.M.; Cai, G.Q.; Yang, J.W.; Jia, L.M. Monte-Carlo Simulation Based on FTA in Reliability Analysis of Gate System.The 2nd International Conference on Computer and Automation Engineering.February 26-28, 2010, pp.713-717.

[9] Xu, W.Z. Comparative study of the GO method and fault tree reliability modeling. Journal of Beijing institute of light industry, 1999, 17(2).

[10] Shen, Z.P.; Huang, R.X. Z. Shen, R. Huang. Principle and application of GO - a method for reliability analysis of system. Beijing:Tsinghua University Press, 2008.

[11] Takkishi, M.; Michiyuki, K. GO-FLOW: A New Raliability Analysis Methodology. Nuclear science and engineering, January 1988, 98(1), pp.64-78.

[12] Xie, B.L.; Liu, Q. A New Inference Method on Fuzzy Control and Simulation. 2011 International Conference on Mechanical, Industrial, and Manufacturing Engineering (MIME 2011). January 2011, pp.159-161.

[13] Cooper, G.F.; Herskovite, E. A bayesian method for the induction of probabilistic networks from data[J].Machine learning,1992,9(4):pp.309-347.

[14] Holmes, C.C.; Mallick, B.K. Bayesian regression with multivariate linear splines. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 2001, 63(1), pp.3-17.

[15] Wang, H.B.; Wang, Z.W. Study on the method and procedure of logistics system modeling and simulation. 2nd International Conference on Science and Social Research(ICSSR 2013), June 2013, pp.776-780.

[16] Qin, T.H.; Wang, Y.F. Application oriented simulation modeling and analysis with ExtendSim. Beijing Tsinghua University Press, 2009.

Testing a Storytelling Tool for Digital Humanities

Fabio Pittarello
Università Ca' Foscari Venezia
Via Torino 155 - Venezia, Italia
pitt@unive.it

Abstract—**This work presents an evaluation of ToBoA-3D, a social web platform for the annotation of 3D models and the creation of stories. ToBoA-3D was evaluated through a pilot study realized in a real educational context, the 2014 edition of the Fall School in Digital Humanities, held at the Università Ca' Foscari Venezia in collaboration with the École Polytechnique Fédérale de Lausanne (EPFL). The tool was presented during the School's lectures and then used by the students for creating stories during the labs' activities. The results of the study were collected through direct observation and a questionnaire. The answers evidenced positive feedbacks for the core features of the platform and led to define an initial set of guidelines for its educational use.**

Keywords: annotation; cultural heritage; digital humanities; education; pilot study; storytelling; web3d

I. Introduction

The availability of 3D representations for scholars is a great opportunity to support the processes of teaching and learning, especially for those disciplines that are deeply involved in the study of objects that have a 3D shape. A further exploitation of 3D objects can be obtained by their annotation that permits, for example, to search annotated content across a set of different 3D worlds, overcoming the navigation mechanisms provided by the authors of each model. ToBoA-3D [1] is a social web platform that permits to exploit the educational potential of annotated 3D worlds. It can be used for collaborative annotation, navigation and search of 3D worlds; the users can even add new 3D environments to the web repository and share them with the other users. The latest evolution of the platform [2] introduces the possibility to create educational stories on the top of the annotated worlds and to share them on the web. While ToBoA-3D can be personalized for different knowledge domains, so far the development has been focused on art and architecture. In this work we describe how ToBoA-3D was tested in a real educational context, a necessary step for evidencing its points of strength and weaknesses and designing more extensive educational experiences. The occasion for the pilot study came from the Fall School in Digital Humanities, held in Venice in October 2014, in the context of the collaboration between the Università Ca' Foscari Venezia and the EPFL (École Polytechnique Fédérale de Lausanne). Digital Humanities are an area of research and teaching at the intersection of computing and humanistic disciplines. This area combines the methodologies of the traditional humanities with tools and methods provided by computer science. The faculty of the Venice Fall School was compliant with this intersection, being composed by art and architecture historians but also computer scientists, which transmitted to the students complementary knowledge about the School's educational themes. The School was focused on the Venetian Renaissance and included class lessons, visits to historical sites and labs. About 20 students, mainly PhD students in Humanities, were selected for participating to the School, programmed for a full week. During this week a group of students had the opportunity to learn how to use ToBoA-3D and to apply this knowledge to the School's themes. The results of the experience were collected through direct observation and a questionnaire and led to define a set of guidelines for the future use of ToBoA-3D.

II. Related works

The related literature encompasses different fields, from annotation of 3D worlds to storytelling. Because of space limits we'll give only some hints, addressing to [1] [2] for further references. As far as annotation is concerned, there are different proposals for adding high-level semantics to the components of 3D environments, based on specifications such as MPEG-7, Collada or X3D. Most proposals use annotations referred to taxonomies or ontologies, but there are also systems that permit to use free tags [1]. The latter ones offer a more expressive and informal approach suited also to common people. Their benefits for cultural heritage have already been underlined in [3]. Annotation is however a first step towards a more advanced use of 3D worlds. Scopigno et al. [4] underline that the greater challenge for digital technologies is the creation of tools that use 3D models for supporting cultural heritage research. We claim that the introduction of storytelling for annotated 3D worlds can bring further advantages for researchers and pupils. Computer-enhanced storytelling represents an evolution of traditional storytelling. It takes advantage of technology for creating and delivering stories, but also for designing and managing new narrative models and proposing new relations between the narration, the reader and the context. The analysis of narratology, the science that studies the structure of the story, has greatly helped the building of models and architectures for interactive storytelling [5]. Several studies have demonstrated the usefulness of storytelling for educational experiences [6]. In the cultural heritage domain storytelling is used for engaging students during the learning process, associating the narration to multimedia components [7], real scenes [8], augmented [9] and virtual reality [10]. While other tools offer repositories of assets for speeding the creation of stories, ToBoA-3D fully exploits the potential of collaboration and knowledge sharing.

III. THE TOBOA-3D PLATFORM

Each user can contribute to the ToBoA-3D platform in different fashions, uploading 3D models, annotating the components that define the 3D environments or building narrations. All the 3D environments belonging to the platform can be explored both using a first-person navigation style or querying the system for retrieving interesting objects and places contained inside of them. Altough users are not required to perform all the types of activities, a typical session with ToBoA-3D includes a mix of them. As far as the creation of narrations is concerned, the author creates a story starting from a list of personal bookmarks, corresponding to visited 3D viewpoints related to annotated objects and locations, and selects an ordered set of them for defining the stages of a linear story. Each stage is then associated to multimedia content that will be automatically played during the narration. Additional content, such as textual descriptions, images and links to web resources, can be included. The interface for listening to stories is displayed in Fig.1. The story is automatically played synchronizing the delivery of the information associated to each stage with the automatic navigation to the associated 3D locations. If the story is narrated across different environments belonging to the platform, the system takes care of downloading automatically the required environment. The story can be stopped at any time, for allowing the listener to focus on details, exploring the 3D view or accessing the associated hypermedia. Stories can be navigated also selecting the single stage from the list available on the lower panel of the interface. Two key features of ToBoA-3D are that the results of all the activities are shared and that their authors are easily identifiable. The first feature enables all the users to take advantage of the work done by the other members of the community (e.g., upload of 3D models, annotations), avoiding to start from scratch for authoring. The second feature offers interesting scenarios for research and teaching. For example ToBoA-3D permits students to annotate a set of 3D architecture components after a lesson focused on classical orders and then the teacher to check the annotations made by each student, marked with the student's ID. Further details about the ToBoA-3D functionalities are available in [2].

IV. TOBOA-3D AT THE FALL SCHOOL

The goal of the Fall School was to investigate, through a set of coordinated lectures and visits to Venetian palaces and collections, new ways of visualizing the evolution of architecture and artwork display. The School was focused in particular on investigating the evolution of the Grimani Palace in Venice and of the artwork collection contained in its main room, the so-called Tribuna. During the preparatory work we discussed with the other organizers how to present and use the annotation platform during the School's activities. A skilled 3D modeler created a simplified model of the Grimani Palace and its Tribuna that was used as the scenario for an educational narration built with ToBoA-3D. The narration was created with the contribution of Cristiano Guerneri, one of the architecture historians involved in the School. It was organized

as a self-paced linear story, guiding the students with a virtual camera through the locations of the palace. The first part of the School included lectures held by art historians and focused on the main School's theme, but also talks related to the use of new technologies for cultural heritage, among which ToBoA-3D. The last two days of the school were dedicated to technological labs, held in parallel for small groups of students. For this reason, only four students out of twenty had the chance to attend the ToBoA-3D lab. In spite of the low number of students, the results were interesting. The initial phase of the ToBoA-3D lab was dedicated to a tutorial illustrating the different features of the platform. The students were then invited to try the techniques acquired on some test 3D environments prepared for the School. Finally the students listened to the introductory story by the art historian that ended with the presentation of two tasks to accomplish:

- the creation of a narration describing a tour through the rooms of the palace, enriched with the snapshots taken during the visit to the real building;
- the creation of a a story related to an hypothesis of reallocation of the artworks of the Tribuna Grimani, which are currently placed in a different site.

The students interpreted both the themes proposed, altough with some simplifications due to time constraints. The first story was an humorous interpretation where one of the students played the part of a Venetian nobleman and guided the listeners through the rooms of the palace (see Fig.1 on the left). The second story was a more serious narration, focused on the hypothesis of reallocation of the artworks in the Tribuna Grimani (see Fig.1 on the right; the red dots in the 3D scene represent the original position of the artworks).

V. RESULTS OF THE PILOT STUDY

The four students were PhD candidates representatives of different research domains: visual arts and architecture (two students), computer science and interaction design, literature. While we should not consider this study as exhaustive, their answers have a great value for identifying the points of weakness and strength of the platform from different facets. Only two of them (the computer scientist and one of the art historians) had a fair knowledge of modeling techniques and interactive 3D environments. None of them had previous experience related to the annotation of 3D worlds. The results of the pilot study were collected through discussions with students and a final questionnaire, composed of closed and open questions, and articulated in 6 sections focused on annotation, search, storytelling, sharing, workflow and usability/engagement. We obtained positive results for most of the features of ToBoA-3D. While we don't have the space to analyze the single results, we underline how the pilot study led to define more precisely the profile of the students interested in using this platform and the set of tasks that should define a complete experience. These results will be useful for designing a more advanced study or more extensive and complete educational experiences based ToBoA-3D:
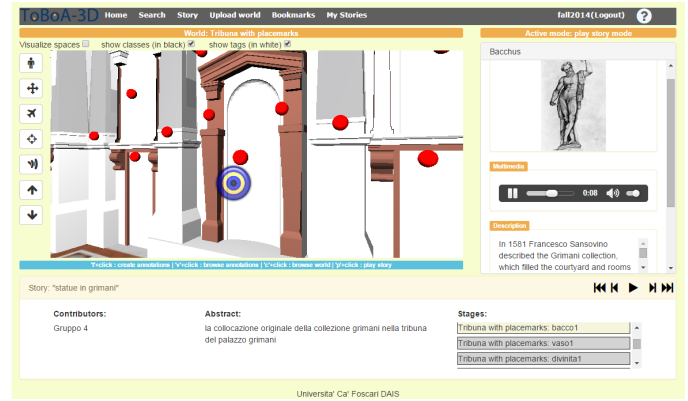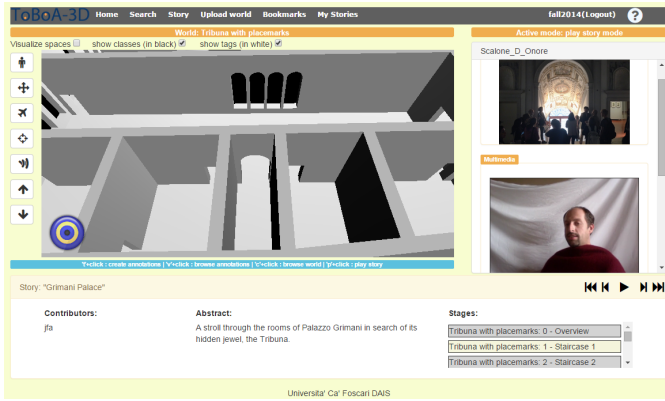
Fig. 1. The two stories created by the students at the Digital Humanities Venice Fall School

- *design educational experiences for students interested in visual representation*; the study revealed that ToBoA-3D resulted more appealing to students with a background focused on visual representation rather than on literature; we suggest, as a complementary guideline, that for longer term experiences such as a master course, the introduction of a basic 3D modeling course would be useful for giving a basic knowledge of 3D representation and attracting students with different cultural backgrounds;

- *include search and information sharing among the activities of the educational experience*; in the pilot study students were shown the functionalities related to search and information sharing and declared their interest for them; however, starting also from the observations of the students, we estimate that it would be interesting to include specific tasks focused on these activities in the structure of the educational experience, proposing for example to search an annotated set of 3D worlds in relation to a given goal or to ask a group students to annotate individually the same set of objects and then check the annotations of their fellows for identifying different point of view; probably these activities would augment the awareness of the potential of these techniques and would stimulate their application to individual research;

- *include knowledge checking as part of the educational experience*; while the educational experience proposed in this experience included a set of goals defined by a teacher, for time constraints it was not possible to fulfill all the teacher's requests and to check the results; a full educational experience should include a final check by the teacher of the individual work and a feedback to students.

Other suggestions came from the feedback related to the ToBoA-3D functionalities. While we obtained positive judgments for the core features of ToBoA-3D, most of the students complained about the quality of the 3D models, realized under heavy time constraints. This is an issue that we'll take into account for the future educational experiences. The students suggested improvements to the platform as well. While the students with an humanistic background focused more on content and structure, suggesting for example the possibility to

create stories with branching structures, the computer science student focused on interaction design issues, suggesting ways to refine the interface or adding additional functionalities. While the positive findings encourage us to propose the use of the platform in real educational contexts, the future development will take care of all the points of weaknesses underlined by the users and improve further its features.

ACKNOWLEDGMENTS

REFERENCES

[1] F. Pittarello and I. Gatto, "ToBoA-3D: an architecture for managing top-down and bottom-up annotated 3d objects and spaces on the web," in *Proc. of Web3D '11*, 2011, pp. 57–65.

[2] I. Gatto and F. Pittarello, "Creating web3d educational stories from crowdsourced annotations," *Journal of Visual Languages & Computing*, vol. 25, no. 6, pp. 808–817, 2014.

[3] J. Trant, "Exploring the potential for social tagging and folksonomy in art museums: Proof of concept," *New Review of Hypermedia and Multimedia*, vol. 12, no. 1, pp. 83–105, 2006.

[4] R. Scopigno, M. Callieri, P. Cignoni, M. Corsini, M. Dellepiane, F. Ponchio, and G. Ranzuglia, "3D Models for Cultural Heritage: Beyond Plain Visualization," *Computer*, vol. 44, no. 7, pp. 48–55, Jul. 2011.

[5] M. Cavazza and D. Pizzi, "Narratology for interactive storytelling: A critical introduction," in *Proc. of TIDSE'06*. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 72–83.

[6] J. Ohler, *Digital Storytelling in the Classroom: New Media Pathways to Literacy, Learning, and Creativity*. Corwin Press, 2008.

[7] K. Kwiatek and M. Woolner, "Let me understand the poetry: Embedding interactive storytelling within panoramic virtual environments," in *Proc. of EVA '10*, 2010, pp. 199–205.

[8] J. Halloran, E. Hornecker, G. Fitzpatrick, M. Weal, D. Millard, D. Michaelides, D. Cruickshank, and D. De Roure, "The literacy fieldtrip: using ubicomp to support children's creative writing," in *Proc. of IDC '06*, 2006, pp. 17–24.

[9] Z. Zhou, A. D. Cheok, J. Pan, and Y. Li, "An interactive 3d exploration narrative interface for storytelling," in *Proc. of IDC '04*. New York, NY, USA: ACM, 2004, pp. 155–156.

[10] S. Mystakidis, N. Lambropoulos, H. M. Fardoun, and D. M. Alghazzawi, "Playful blended digital storytelling in 3d immersive elearning environments: A cost effective early literacy motivation method," in *Proc. of IDEE '14*. New York, NY, USA: ACM, 2014, pp. 97–101.

A Quick Survey on Sentiment Analysis Techniques: a lexical based perspective

Flora Amato[1], Francesco Colace[2], Luca Greco[2] , Vincenzo Moscato[1], Antonio Picariello[1]

[1]DIETI- Department of Electrical Engineering and Information Technology
Università degli Studi di Napoli "Federico II", Napoli - Italy
{flora.amato, vmoscato, picus}@unina.it
[2]DIEM - Department of Information Engineering, Electrical Engineering and Applied Mathematics
Università degli Studi di Salerno, Fisciano (Salerno) - Italy
{fcolace,lgreco}@unisa.it

Abstract— **With the advent of *World Wide Web* and the widespread of on-line collaborative tools, there is a increasing interest towards automatic tools for *Sentiment Analysis* to provide a quantitative measure of "positivity" or "negativity" about opinions or social comments. In this paper, we provide an overview of the most diffused techniques for sentiment analysis based on the lexical-based approaches as a quick reference guide in the choice of the most suitable methods for solving a specific problem in the sentiment analysis field.**

Index Terms— **Sentiment analysis, Computational linguistics, Text Classification .**

I. INTRODUCTION

People's opinion has always driven human choices and behaviors, even before the diffusion of Information and Communication Technologies. With the advent of *World Wide Web* and the widespread of on-line collaborative tools such as blogs, focus groups, review web sites, forums, social networks (e.g *Facebook*, *Twitter*, *MySpace*, etc.), users more and more use to make available to everyone their tastes and liking, and in general, their opinions and sentiments about an event, a topic, a public person, a political faction, a TV program, etc.

In such a context, there is an increasing need to have available automatic tools for *Sentiment Analysis* (or *Opinion Mining*) and *Tracking* in order to provide a quantitative measure of "positivity" or "negativity about opinions (*polarity*) or comments related to a certain topic of interest and to track along the time such information.

More in details, sentiment analysis aims at finding the opinions of authors (thought leaders and ordinary people) about specific entities, by analyzing a large number of documents (in any format such as PDF, HTML, XML, etc.).

It can be considered as a sub-discipline of Computational Linguistics, indeed it is a *Natural Language Processing* and *Information Extraction task* [14], or challenged by the use of classical *Machine Learning* based approaches.

The most studied languages in the opinion mining field are English and Chinese, but there are several researches on other languages like Italian, Thai and Arabic [12].

Opinion mining allows to identify problems by listening, rather than by asking, ensuring an accurate reflection of reality [14].

The analyzed textual information can be of two types: *facts* and *opinions*. The facts are objective expressions that describe entities, conversely the opinions deal with people's emotions, sentiments and feelings and so they are subjective.

Generally, we can see an opinion (or a sentiment) as a quintuple: $<o, f, s, h, t>$, where o is the object evaluated by the opinion holder h, f is a feature of the object o, t is the time when the opinion has been expressed and s is the value of the opinion (for example positive or negative) [1][14].

Sentiment analysis techniques have as main goal the automatic extraction of the polarity measure "attached" to an object and can adopt several methods and techniques derived both from Computational Linguistics and Machine Learning theory. Here, we focus our attention on *lexical-based* techniques belonging to the branch of Computational Linguistics approaches.

The paper is organized as follows. Section II contains a review of the most diffused lexical-based approaches. Finally, Section III reports some conclusions and final considerations about our study.

II. AN OVERVIEW OF LEXICAL BASED SENTIMENT ANALYSIS TECHNIQUES

In lexical-based approach a predefined list of words is used to determine a specific sentiment. A relevant problem regards ambiguity of natural language: sentiment value for a given word depends on the specific context.

There are several approaches to sentiment lexicons' creation. A manual construction is often difficult and very time consuming. In the literature, the most used methods can be classified as Corpus-based and Dictionary-based.

i. Corpus-based Approach

In this approach, a set of seed words grows by using a corpus of documents of a specific domain. Therefore a specific domain lexicon is constructed on the basis of a labeled corpus of documents.

One of the first works in this field is [6] where, given some seed adjectives, a corpus is used to identify additional sentiment adjectives. A key point regards the presence of conjunctions: for example the conjunction 'and' between two adjectives can refer to the same sentimental polarity. A graph with same or different orientation links between adjectives is created. These adjectives are then separate with a clustering algorithm into two subsets.

Another example is [8] where a corpus of 10000 blog posts from LiveJournal.com is used; the posts are labeled "happy" or "sad". A happiness factor is assigned to words by calculating their frequency: the ratio between the number of occurrences of a word in the happy blogposts and its frequency in the entire corpus.

Among the most recent studies there is the work in [4]. The key of this approach is searching the connotative polarity between a conative predicate and its semantic argument. It is done by using a graph-based algorithm that use PageRank [9] and HITS [7] that collectively learn connotation lexicon together with connotative predicates.

ii. Dictionary-Based Approach

In this approach a small set of seed words is first manually collected and then is expanded with words synonyms and antonyms. This is done by using online resources (dictionaries). The most well-known example is *Wordnet* that is an online lexical database for English language.

A great disadvantage of this approach is that the lexicon acquired is independent from a specific domain.

> ### WordNet-Affect

WordNet-Affect [11] is a linguistic resource, composed by 2,874 synsets and 4,787 words, developed considering WordNet Domains, that is a multilingual extension of Wordnet.

It aims at providing correlations between affective concepts and affective words by using a synset model.

A subset of synsets, which are able to represent affective concepts, is derived from WordNet. Then, these synsets are labeled with one or more affective categories.

The Core of WordNet Affect is created by considering a lexical database, called Affect, composed by 1,903 words that are mostly adjectives and nouns.

Lexical and affective information are associated to each term; they includes parts of speech, definitions, synonyms and antonyms.

In order to assign an affective category to terms, an attribute called Ortony is used. Terms can be classified in emotional terms, non-emotional affective terms, non-affective mental state terms, personality traits, behaviors, attitudes etc.

Ortony information is projected on the subset selected from Wordnet but doesn't cover all Affect items and for this reason some labels are manually assigned. When the subset is completely labeled, WordNet-Affect Core is defined and can be extended exploiting WordNet relations.

> ### SentiWordNet

SentiWordNet is a lexical resource proposed in [2].

SentiWordNet is built with a ternary classification, indeed each synset (set of synonyms) is labeled as positive, negative or objective by using a set of ternary classifiers. If all of them will give to the synset the same label, therefore that label for that synset will have the maximum score; otherwise this score will be proportional.

Each classifier follows a semi-supervised approach that is a learning process where the training set $Tr = L \cup U$ so that: L is a small subset of manually labeled training data, and U is a subset of training data labeled by the process by using L, and other available resource, as input.

In [2] L is divided into: L_P, L_n, that are two small synsets respectively for positive and negative training data, and L_o for the objective ones.

L_p and L_n are expanded with K iterations obtaining the following result for the i-th iteration:

Tr_p^i (resp Tr_n^i) will contain, in adding to Tr_p^{i-1} (resp Tr_n^{i-1}), all the synsets that are related to synsets in Tr_p^{i-1} (resp Tr_n^{i-1}) by WordNet lexical relations and have the same Positive(resp. Negative) polarity, and the synsets that are related to synsets in Tr_n^{i-1} (resp Tr_p^{i-1}) and have the opposite polarity.

Tr_o^K coincides with L_o and it consists of 17,530 synsets that doesn't belong either to Tr_p^K or to Tr_n^K. To each synset is associated a vectorial representation by applying a cosine-normalized tf*idf to its gloss, that is a textual representation of its semantic.

Hence now the training synset, for a class c_i, can be given to a standard supervised learner that generates two binary classifiers. One of these will distinguish *positive* and *not_positive* terms, and takes $Tr_p^K \cup Tr_o^K$ in the training phase, the other one will classify terms as *negative* or *not_negative*, and takes $Tr_n^K \cup Tr_o^K$ in the training phase.

It produces a resulting ternary classifier that will classify the entire WordNet.

SentiWordNet has been developed in several versions, but the most significant is SentiWordNet 3.0 that, in the automatic annotation of WordNet, adds to the semi-supervised learning step a random-walk step for refining the scores. This version is compared with the previous one, and an improvement in accuracy of about 20% is found.

> ### Context Dependent Opinion Observer (CDOO)

CDOO is a system implemented in C++ and it is based on a method that tries to infer the semantic orientation of opinion sentences by associating contextual information to opinion words obtained from WordNet.

This approach goes through four steps.

In the first step, after a preprocessing phase, opinion sentences are extracted from the inputs by using feature keywords directly.

In the second step Context independent opinions that don't

require any contextual information are analyzed to determine the semantic orientation. In this step opinion words from Wordnet are simply considered and in particular are utilized adjective synonym set and antonym set.

In the third step distinct-dependent opinions are analyzed: adjacent sentences are needed to define the semantic orientation by using Linguistic rules, especially conjunction rule.

In the fourth and final step Context indistinct opinions that need contextual information from other reviews are analyzed. In order to collect contextual segments sets for given features, a large number of online reviews are considered. Subsequently, contextual information is extracted from the segment sets by using Emotional-ATFxPDF to compute weight of terms in text segment set. Then the orientation of the opinion is calculated using semantic similarity.

➤ *SenticNet*

SenticNet is inspired by SentiWordNet but it assigns to each concept c only one value p_c belonging to [-1,1].

The polarity of a concept c is defined in the following way:

$$p_c = \frac{Plsn(c) + |Attn(c)| - |Snst(c)| + Aptt(c)}{9}$$

where *Plsn* is *Pleasantness*, *Attn* is *Attention*, *Snst* is *Sensitivity*, *Aptt* is *Aptitude*.

They start from Hourglass model and for example, in order to find positive concepts correlated with Pleasantness, they begin to search concepts semantically correlated to words like "joy", "serenity" and uncorrelated to words like "sadness".

Two different techniques are used: Blending and Spectral Assumption. When polarity is assigned, SenticNet is encoded in RDF triples using a XML syntax.

The current version of SenticNet contains almost 15,000 concepts.

In recent studies SenticNet is often associated to WordNet-Affect. For example in [10] researchers assign to SenticNet concepts, which are not present in WordNet-Affect, emotion labels. It is actually an expansion of WordNet-Affect based on SenticNet. By analyzing several features and utilizing a SVM framework for classification, they obtain an accuracy of 85.12% in their best result.

➤ *Panas-t*

The original PANAS is created by Watson and Clark and they analyzed 10 moods on a 5-point scale [13].

They also expanded it in PANAS-x where eleven specific affects are considered: Fear, Sadness, Guilt, Hostility, Shyness, Fatigue, Surprise, Joviality, Self-Assurance, Attentiveness, and Serenity. To each affect a list of adjectives is associated.

In [5] it is expanded in Panas-t which is an adaptation that analyzes short text from Online Social Media and in particular from Twitter.

They consider a dataset composed by tweets from all the public accounts registered before August 2009. First tweets that explicitly contain feelings (and hence tweets that contain words like "I am", "feelings", "myself") are identified.

Then a preprocessing phase is performed where individual terms are isolated, using white-space boundaries, and punctuation and other non-alphanumeric characters are removed.

It is assumed that a tweet can be mapped to the first sentiment s that appears in the tweet. This can be done by verifying the position of the adjectives.

III. CONCLUSIONS

The paper provided an overview of the most diffused techniques for sentiment analysis based on the lexical-based approaches and the related systems.

The paper wants to be a quick reference guide in the choice of the most suitable lexical-based approaches for a specific problem of sentiment analysis.

REFERENCES

[1] F. Colace, L. Casaburi, M. De Santo, L. Greco, "Sentiment detection in social networks and in collaborative learning environments", Computers in Human Behavior, Available online 27 December 2014, ISSN 0747-5632, http://dx.doi.org/10.1016/j.chb.2014.11.090.

[2] A.Esuli, and F.Sebastiani – "Sentiwordnet: A publicly available lexical resource for opinion mining", Proceedings of LREC. Vol. 6. 2006.

[3] R.Feldman – "Techniques and Applications for Sentiment Analysis", Magazine - Communication of the ACM (April,2013).

[4] S.Feng, B.Ritwik, and C.Yejin – "Learning general connotation of words using graph-based algorithms", Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2011.

[5] P.Gonçalves, F.Benevenuto, and M.Cha - "Panas-t: A pychometric scale for measuring sentiments on twitter", arXiv preprint arXiv:1308.1857 (2013). 14

[6] V.Hatzivassiloglou and K.R.McKeown – "Predicting the semantic orientation of adjectives", Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the european chapter of the association for computational linguistics. Association for Computational Linguistics, 1997.

[7] J.M.Kleinberg – "Authoritative sources in a hyperlinked environment", Journal of the ACM (JACM) 46.5 (1999): 604-632. 1999.

[8] R.Mihalcea, and H.Liu – "A Corpus-based Approach to Finding Happiness", AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs. 2006.

[9] L.Page, S.Brin, R.Motwani and T.Winograd – "The PageRank citation ranking: Bringing order to the web" 1999.

[10] S.Poria, A.Gelbukh, E.Cambria, P.Yang, A.Hussain and T.Durrani - "Merging SenticNet and WordNet-Affect emotion lists for sentiment analysis", Signal Processing (ICSP), 2012 IEEE 11th International Conference on , vol.2, no., pp.1251,1255, 21-25 Oct. 2012.

[11] C.Strapparava, and A.Valitutti - "WordNet Affect: an Affective Extension of WordNet", LREC. Vol. 4. 2004.

[12] G.Vinodhini, R.M.Chandrasekaran – "Sentiment Analysis and Opinion Mining: A Survey", International Journal of Advanced Research in Computer Science and Software Engineering Volume 2,Issue 6 (June 2012).

[13] D.Watson, L.A.Clark, and A.Tellegen - "Development and validation of brief measures of positive and negative affect: the PANAS scales", Journal of personality and social psychology 54.6 (1988).

[14] M.Shelke, S.Deshpande, V.Thakre – "Survey of Techniques for Opinion Mining", International Journal of Computer Applications (0975-8887) Volume 57-No.13 (November 2012).