Securing the Cultural Heritage via Geometric Programming and Modeling

Alberto Paoluzzi[†] Giorgio Scorzelli[†,‡] Michele Vicentino[‡]

[†]Dipartimento Informatica e Automazione, Università "Roma Tre", Via della Vasca navale 79 —- 00146 Roma, Italy
[‡]TRS (Technology and Research for Security), Via della Maglianella 65/E —- 00166 Roma, Italy

Figure 1: Geometric modeling of the Leaning Tower of Pisa, a flagship element of the Italian cultural heritage, using a streaming progressive techology developed at Roma Tre University [22].

Abstract

In this paper we outline a geometric programming approach to the solid modeling of monuments and the virtual reconstruction of large-scale historical sites. The main concept at the base of this research project is the integration of advanced modelling, simulation, virtual and augmented reality, serious gaming and tracking techniques in one centralized framework, where the security personnel is provided with decision-making tools integrating environment and behavior modeling, knowledge bases and sensor systems. In particular, this paper discusses the current state of the geometric language PLaSM and its current interfacing to scripting and concurrent languages as Python and Erlang.

Keywords: **VR systems and toolkits** (primary keyword); Cultural heritage; Archeology; Security and protection.

1 Introduction

TRS (Technology & Research for Security) is a spin-off company of Roma Tre University, with entrepreneurship provided by Theorematica spa and Invent sas. The spinoff has the mission of developing an innovative Platform for Intelligent Security of Critical Infrastructures, to assure high levels of awareness and security management of facilities and infrastructures, like government buildings, historical sites, museums, bridges, railways hubs, airports, etc. TRS is developing ICT systems of a new generation, integrating preexisting security with novel capabilities beyond the present state-of-the-art, and including: behaviour and situation modelling, counter-measure assessment and evaluation, and security personnel training. Advanced modelling and simulation using virtual-reality interfaces is the reference paradigm [2]. The spin-off project is funded by the Italian Ministry of University and Research (MiUR).

Fouders and shareholders of TRS are: *"Roma Tre" University*, established in 1992 as the third state university of the capital city of Italy, that enters TRS through the CAD Lab of the Dept. of Informatics and Automation and the Modelling and Simulation Lab of the Dept. of Studies on Structures. *Theorematica* spa operates management solutions for

Public Administration, via integration of Decision Support Systems, management models, organisational consulting, design and delivery of education and training programs. Theorematica has realized Situation and Control Rooms at the national level for the Italian state police and other national police forces. It has a solid expertize in building modeling-based awareness using data from different sources (field sensors, factual information, behavioural descriptions, previous events, etc.). *Invent* sas is a seed-capital company specializing in technology transfer services, which assists industrial and research organizations in the implementation of strategies for the valorization of R&D results. Through its network it has promoted more than 200 international projects of technology transfer actions in Europe.

This paper is organized as follows. Section 2 introduces the vision of using virtual reality models of historical sites and buildings as the interface to the ICT infrastructure supporting complex security systems. Section 3 discusses the embedding of the geometric language developed at Roma Tre within both main-stream languages like Python and others, rapidly gaining wider acceptance as Erlang. Section 4 shortly recalls some basic concepts of PLaSM programming and its internal data structures. Section 5 deals with the strategies for development of complex models and large-scale virtual environments. The Conclusion section summarizes the paper and the new challenges of 3D interfaces to next-generation security and protection systems.

2 Security of cultural heritage

Terrorist attacks and citizens concern for security, urge for the development of novel security technologies as well as for the identication of new standards for the security of infrastructures. There is a sound demand for the innovation of security systems based on intelligent technologies to better prevent threatening acts and to manage possible emergencies. The threat must be considered as particularly severe with respect to museums, art-gallery expositions, and artistic historian places, i.e. the cultural worth-valued sites, generally denoted here as "cultural heritage", where large masses of people—often coming from abroad—convey daily, normally for some short time intervals, and with limited possibility of security enforcing and checking in advance.

A new generation of security systems must be developed, able to integrate the videosurveillance with novel capabilities beyond the state-of-the-art, including: virtual-reality user interfaces, behaviour and situation modelling, and behavioural bases, in our vision making extensive use of 3D modelling as the reference integration paradigm. Next generation security systems will interpret and recognise the weak signals of high-risk situations, in particular in densely populated areas, in order to promptly activate the needed countermeasures.

In particular, following the London and Madrid attacks, the EC supports the fight against terrorism and advanced a directive proposal aiming at dening the European critical infrastructures (ECI), in order to settle higher levels of protection. ICT Technologies provide the adequate response to such a need. In particular, computer modelling and simulation of sites, behaviours, situations and events are needed to ensure effective continuous surveillance, protection and security. For the development of the technologies discussed here, we capitalize on a novel parallel framework for high-performance solid and geometric modeling, that (i) compiles the generating expression of the large-scale geometry model into a dataflow network of concurrent processes, and (ii) splits the model into fragments to be distributed to computational nodes and generated and rendered independently.

3 Embedded Geometric Programming

PLaSM, (the Programming LAnguage for Solid Modeling) is a *geometric language*, developed by the CAD Group at the University "Roma Tre" and previously at the University of Rome "La Sapienza". The language introduced a very high-level approach tn "constructive" or "generative" modeling, where geometrical objects are generated by evaluating some suitable language expressions. Because generating expressions can be easily combined, the language also extends the variational geometry approach by supporting classes of geometric objects with varying topology and shape.

The generated objects are always geometrically consistent because the validity of geometry is guaranteed at a syntactical level. The language is strongly influenced by FL (programming at Function Level), an advanced approach to functional programming [4, 5, 1] developed by the Functional Programming Group leaded by John Backus and John Williams at the IBM Research Division in Almaden (California) in the first nineties.

The C kernel of the PLaSM language—named XGE (eXtreme Geometric Environment)—was recently rewritten and optimized to a large extent, in order to support very large-scale applications, such as sensor fusion for monitoring critical infrastructures (Assogna et al. 2008). This work has produced the novel PlasmX v.0.9, able to deal with meshes of approximately 200K 3-cells in a single processing node. A complete rock-solid implementation of the geometric kernel is on the way, that can be easily interfaced with very different embedding languages.

3.1 Python 2D/3D geometry packages

Python is a well-known multi-paradigm language with efficient built-in data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas. Several packages and modules exist por both 2D and 3D graphics modeling and rendering. We have recently embedded the PLaSM functional kernel and the geometric library XGE in Python, in order to make easier to embed its powerful modeling semantics in interactive environments, with the aim of supporting the development of protection applications for the cultural heritage and the security of critical infrastructures. Another aim was to gain easy and direct access from PLaSM models to the several game and rendering engines and GIS applications already interfaced with Python, and shortly recalled in the following of this section.

Shapely [9] is a 2D Python package for geometric programming based on GEOS (Geometry Engine Open Source) [23], the C++ port of the Java Topology Suite (JTS) [15], the API for modelling and manipulating 2D polygonal geometry. Shapely includes the Simple Features for SQL spatial predicate functions and spatial operators published by the OpenGIS Consortium [15]. CGAL-Python [13] provides Python bindings for the CGAL library, the Computational

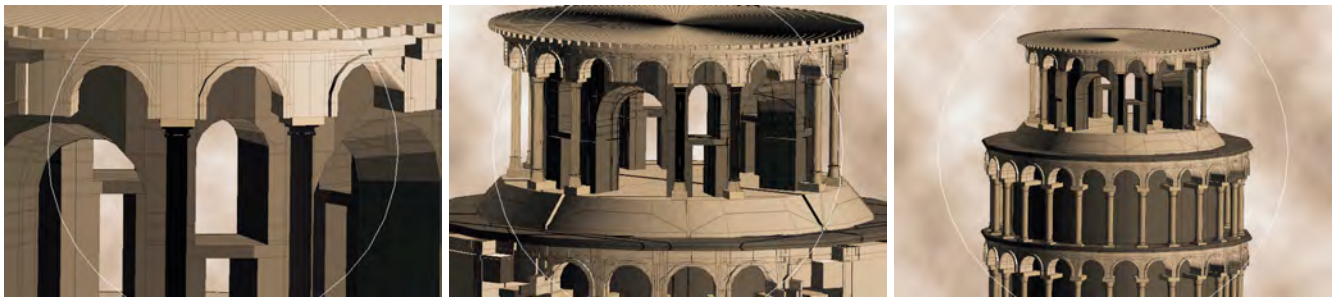

Figure 2: Progressive generation of geometric detail at run-time.

Figure 3: Some exploded views of the "embarrassingly-parallel" decomposition of the "progressive BSP-tree" data structure to computing nodes.

Figure 4: (a) View, seen from the *Circus Maximum*, of the Imperial Palace on the Palatine Hill; (b) an archeological drawing of the ruins site, with the *Stadium* on the rigth, the *Domus Flavia* on the bottom and the *Domus Augustana* on the top.

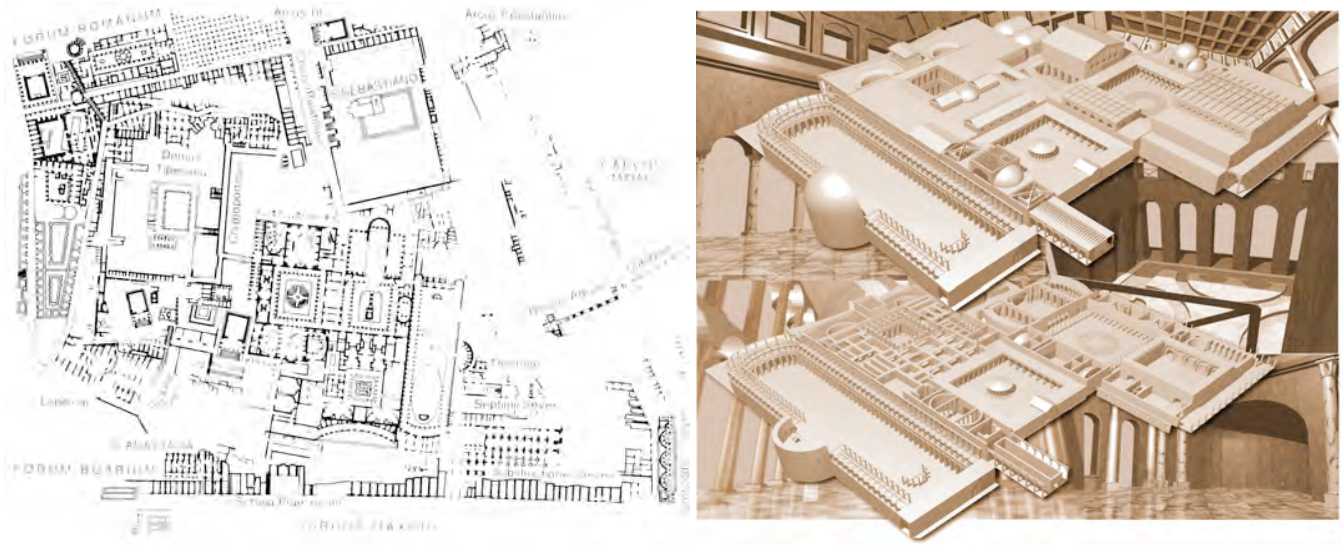

Figure 5: (a) 2D map of the Palatino hill in Rome; (b) the 3D Plasm reconstruction of the emperor's palace (Domus Flavia, Domus Augustana and Hippodrome of Domitian).

Geometry Algorithms Library, of generic, efficient and robust geometric 2D algorithms. Blender [24] is the free open source 3D content creation suite, that provides a set of tools for 3D modeling/animation, that are widely scriptable from Python. The Computer Graphics Kit (cgKit) is a collection of modules [3] with basic types and functions to rendering realistic images, which mainly focuses on Pixar's RenderMan interface. Several Python bindings exist for popular game engines, including the Object-oriented Graphics Rendering Engine (OGRE) [11], and the Pivy [8] wrapper for the Coin/Inventor Scenegraph Library. Of course standard Python binding to both the OpenGL, GLU, GLUT and GLE libraries, and to the Microsoft DirectX libraries are also largely in use.

3.2 Erlang: concurrent geometry embedding

Erlang is an (open source and multiplatform) concurrent functional programming language and runtime system, characterized by strict evaluation, single assignment and dynamic typing. It is purely functional, without mutable state (induced by multiple assignment), easy to understand and to debug. Type checking is performed at run-time. Erlang was developed by Ericsson to support distributed, fault-tolerant, soft real-time, non-stop applications. Even hot swapping of programs is supported, so that code can be changed without stopping a system. The Erlang language seems to fit very well with multicore CPUs and SMP architectures, since processes belong to the programming language, and not to the operating system. The language has several useful features (Armstrong 2007a,b) for concurrent computation: (i) creating and destroying processes is very fast, (ii) sending messages between processes is very fast, (iii) processes behave in the same way on all operating systems, (iv) a very large numbers of processes may coexistup to hundreds of thousands, (v) processes share no memory and are completely independent, and (vi) processes interact only through message passing.

An Erlang embedding is being currently used for the development of Proto-Plasm, a computational environment to produce libraries of executable, combinable and customizable computer models of natural and synthetic biosystems, aiming to provide a supporting framework for predictive understanding of structure and behaviour of virtual human models through multiscale geometric modelling and multiphysics simulations [6]. Computational modelling of human biosystems has to face great challenges. We believe that description languages are not sufficient, and that totally new progressive and adaptive methods for terascale geometric modelling are needed, combined with novel adaptive methods for multiphysics and multiscale simulation, working on parallel and distributed supercomputers. Both symbolic and hierarchical characterizations of the various components are needed, as well as shape reconstruction from high-resolution nuclear magnetic resonance (NMR) and other volume imaging techniques.

In particular, the Erlang-based geometric package prototype translates each generating function of geometric values—hence, both the associated symbol and the tuple of formal parameters—into an Erlang process that encapsulates the chain complex and its properties within the local store, and which natively interacts via message passing.

4 Background

4.1 PLaSM basics

PLaSM can be seen as a geometric extension of FL [4, 5], allowing for a powerful algebraic calculus with dimension-independent geometric objects, that include polyhedral complexes and parametric polynomial and rational manifolds (curves, surfaces, curved solids, and higher-dimensional objects). The language is a natural environment for geometric computations, where a complex shape is generated as an assembly of component shapes, highly dependent from each other, and where (a) each part may result from computations with other parts, (b) a generating function is associated to each, (c) geometric expressions may appear as actual parameters.

Language semantics *Primitive objects* are characters, numbers and truth values. *Expressions* are primitive objects, functions, applications and sequences. *Sequences* are

expressions separated by commas and contained within a pair of angle brackets, e.g. `<5,fun>`

Every PLaSM program is a function. When applied to some input argument, a program produces some output value. Two programs are usually connected by using *functional composition*, so that the output of the first program is used as input to the second program. According to the FL semantics, an arbitrary PLaSM script can be written by using only three programming constructs:

application of a function to the actual value of its input parameters, elements of the function domain, producing the corresponding output value in the function codomain. An application expression `exp1:exp2` applies the function resulting from the evaluation of `exp1` on the argument resulting from the evaluation of `exp2`. Binary functions can also be used in infix form. For example:

$$\texttt{+:<1,3>} \equiv \texttt{1 + 3} \equiv \texttt{4}$$

composition of two or more functions allowing the pipelined execution of their ordered sequence. The binary composition `~` of functions `f` and `g` has the standard mathematical definition:

$$\texttt{(f} \sim \texttt{g):x} \equiv \texttt{f:(g:x);}$$

The n-ary composition `COMP` of functions is also allowed:

$$\texttt{COMP:<f,h,g>:x} \equiv \texttt{(f} \sim \texttt{h} \sim \texttt{g):x} \equiv \texttt{f:(h:(g:x));}$$

construction of a *vector function* allowing the parallel execution of its component functions. The construction combining form `CONS` applies a sequence of functions to an argument, producing the sequence of applications:

$$\texttt{CONS:< f1,...,fn >:x} \equiv \texttt{[f1,...,fn]:x} \equiv \texttt{< f1:x,...,fn:x >}$$

E.g. `CONS:<+,->`, written also `[+,-]`, when applied to the argument `<3,2>` gives the sequence of applications:

$$\texttt{[+,-]:<3,2>} \equiv \texttt{<+:<3,2>,-:<3,2>>} \equiv \texttt{<5,1>;}$$

PLaSM provides the full power of a Turing-complete programming language with support for conditional, recursion, higher-level functional abstraction, etc. Furthermore, the language is multidimensional by design, and this choice greatly increases its expressive power, based on dimension-independent geometric primitives and operators, and so allowing very terse definitions of highly complex models. The generated objects and assemblies can then be exported in several formats and rendered by a VRML browser or imported into Java3D applications. The language may currently export in VRML1, OpenInventor, VRML2, XML, and PovRay formats.

Geometric operators Several geometric operators are natively available. They include: the dimension-independent affine operators `T`, `S`, `R` for translation, scaling, and rotation, respectively; the `STRUCT` operator for scene graphs and hierarchical transformations from local to global coordinates, according to the PHIGS semantics; the Cartesian product `*` of geometric objects; the Boolean operators `+`, `-`, `&`, and `^` for union, difference, intersection, and symmetric difference, respectively. `Q` is used to transform either a signed number or a sequence of numbers into a 1D polyhedral complex. `JOIN` is used to compute the convex hull of a set of objects; `EMBED` to insert an object into an higher-dimensional space; `MAP` to apply a set of transformations to the coordinates of vertices of an object; `TOP`, `BOTTOM`, `LEFT` and `RIGHT` are used for relative positioning of objects. `MKPOL` and `UKPOL` transform a symbolic form into the internal representation of an object and viceversa. The higher-level functional character of the language is exploited to produce amazingly simple *transfinite parametric* geometry, based on the interpolation or approximation in functional spaces (i.e. the spaces of curves, surfaces and multivariate manifolds).

Programming tools A rich set of libraries provides more than 600 predefined functions and predicates for dimension-independent geometric modeling, documented in the book [16] by Wiley, and displayed in color by PLaSM source editors. Several multi-platform programming tools are available, including an easy-to-use Integrated Development Environment (Xplode), a functionally equivalent IDE plugin for the industrial-strength open development platform Eclipse, a visual programming framework based on only *two* graphical tokens, and producing high-level *executable code* (Visual PLaSM). Support is also given for Knuth's Literate Programming with LaTeX (via the *listings* package). Further support for documenting the code is given by the integrated AsciiDoc rules, that may convert AsciiDoc/PLaSM documents to HTML, PDF, LaTeX, DocBook and other formats.

4.2 The geometric kernel

In this section we shortly outline the dimension-independent representations used by the geometric kernel of PLaSM. Progressive BSP trees are used for adaptive and parallelizable streaming dataflow evaluation of geometric expressions [22], and associated to the polyhedral cells of the HPC (Hierarchical Polyhedral Complex) data structure [21] used by the language. Hasse graphs are used to maintain a complete representation of the model topology. A novel tensorial representation of the chain complex discrete mock-up of the model, that we called *Hasse matrix* [7], should in future support both the geometric modeling and the physical simulations.

BSP trees *Binary Space Partition* (BSP) is a method [14] for recursively subdividing a space into convex sets by hyperplanes, i.e. by affine sets of codimension 1. This subdivision gives rise to a representation of the scene by means of a tree data structure known as BSP tree. This one is a binary tree with partitioning hyperplanes in the inner nodes and with either IN or OUT labels in the leafs. A solid cell of the space partition is labeled IN; an empty cell is labeled OUT. A node of a BSP tree represents the convex set generated by the intersection of all the subspaces on the unique path from the node to the root. The convex set of a node equals the disjoint union of the convex sets associated to its child nodes. BSP trees are largely used in graphics and computational geometry applications as gaming and robotics. *Progressive BSP trees*, supporting Boolean operations [19], were adopted with the new geometric kernel.

Hasse graphs A cell complex K is a collection of compacts subsets of $\mathbb{E}^d$, called *cells*, such that: (a) if $c \in K$, then every *face* of c is in K; (b) the intersection of any two cells is either empty or a face of both. A *d-polytope* is a solid, convex and bounded subset of $\mathbb{E}^d$. A *polytopal d-complex*, or *d-mesh*, is a cell complex of d-polytopes and their k-faces ($0 \leq k \leq d$). A complete representation of a d-mesh is given by its *Hasse*

Figure 6: Top-down site analysis: (a) aerophotogrammetic plan of the archeological site; (b) selection and naming of palace zones; (c) selection and naming of single spaces in each zone.

diagram, the directed graph of the *cover* relation of cells, whose nodes are the members of the complex K, partially ordered by containment, and where there is an arc from node x to node y iff: (a) $x \subset y$ and (b) there is no z such that $x \subset z \subset y$. In this case, we say y covers x, or y is an immediate successor of x. Hasse diagrams are used in the PLaSM kernel as complete representation of the topology of K, that cannot be handled efficiently by BSP trees.

5 Top-Down or Bottom-Up Encoding?

The programming approach at Function Level enforced by PLaSM allows for both top-down development of complex models and for bottom-up modeling of large-scale environments with recurrence of standardized shapes, and even for model development with mixed strategies. Language scripts are used to parametrically generate shape instances and to formalize the sequence of geometric modeling decisions, possibly with the aid of an interactive user interface. In this approach a complicated design may be described hierarchically and developed either top-down or bottom-up. Using a syntactically validated programming approach the design updating becomes easier in both cases: when the changes concern some specific components and when they apply to the overall design organization. In either case it is suficient to update some program units at suitable hierarchical levels. It can be noted that some program units, which are either evaluated more frequently or are formalized more elegantly, may be abstracted as operators and may enter a design knowledge base (i.e., some specialized PLASM package). Later on, they can be reused easily by the same developer or by other persons on the project team during the development of different modeling projects.

PLASM combines geometric objects without performing a consistency checking, rather than adopting some generalized nonmanifold representation. The latter choice would in fact imply: (a) the maintaining of a number of cross-references between the subcomponents of any geometry and, even worse (b) the constant possibility of topology invalidation by inconsistencies due to numeric computations. The assumptions underlying our "weak" representation of polyhedral complexes [Pascucci et al. 1994] are strong enough to guarantee the validity of the representation, that is, the consistency of the represented object. It is only required that an object is generated by some syntactically well-formed expression in the language. The proper combination of well-formed expressions will always give a well-formed result. In other words, it is not possible to construct a geometrically invalid object.

A significant example of the modeling of an historical site, including the virtual-reality reconstruction of the emperor's palace on the Palatine hill in Rome is shown in Figures 4, 6 and 7. The whole top-down reconstruction process can be summarized as follows [12]:

1. development of a library of parametric architectural primitives, including: semicircular and segmental arches; barrel and groin vaults with various opening angles; architraves and colonnades; columns and savast; wooden frames and trusses; exedr; circular, elliptical, octaedral and squared domes, as well halfdomes; panelled ceilings, etc.

2. analysis of the archeological site and the available geometric information, including stereo-photogrammetric maps, and ancient relief drawings;

3. tracing out, zooning and labeling of macro zones and single spaces;

4. importing of geometric information into the PLaSM language through simple definitions whose body is either a polyline or a polygon;

5. introduction of z local information of both floors and ceilings, producing the containment volumes of the spaces;

6. separate development of microzones in local coordinates, and assembling into macrozones;

7. global assembly via either local to global affine transformations or three-linear maps defined by 8 corresponding points.

The development process was pretty quick, and required no more that 1.5 man-month, including the development of the library of architectural elements.

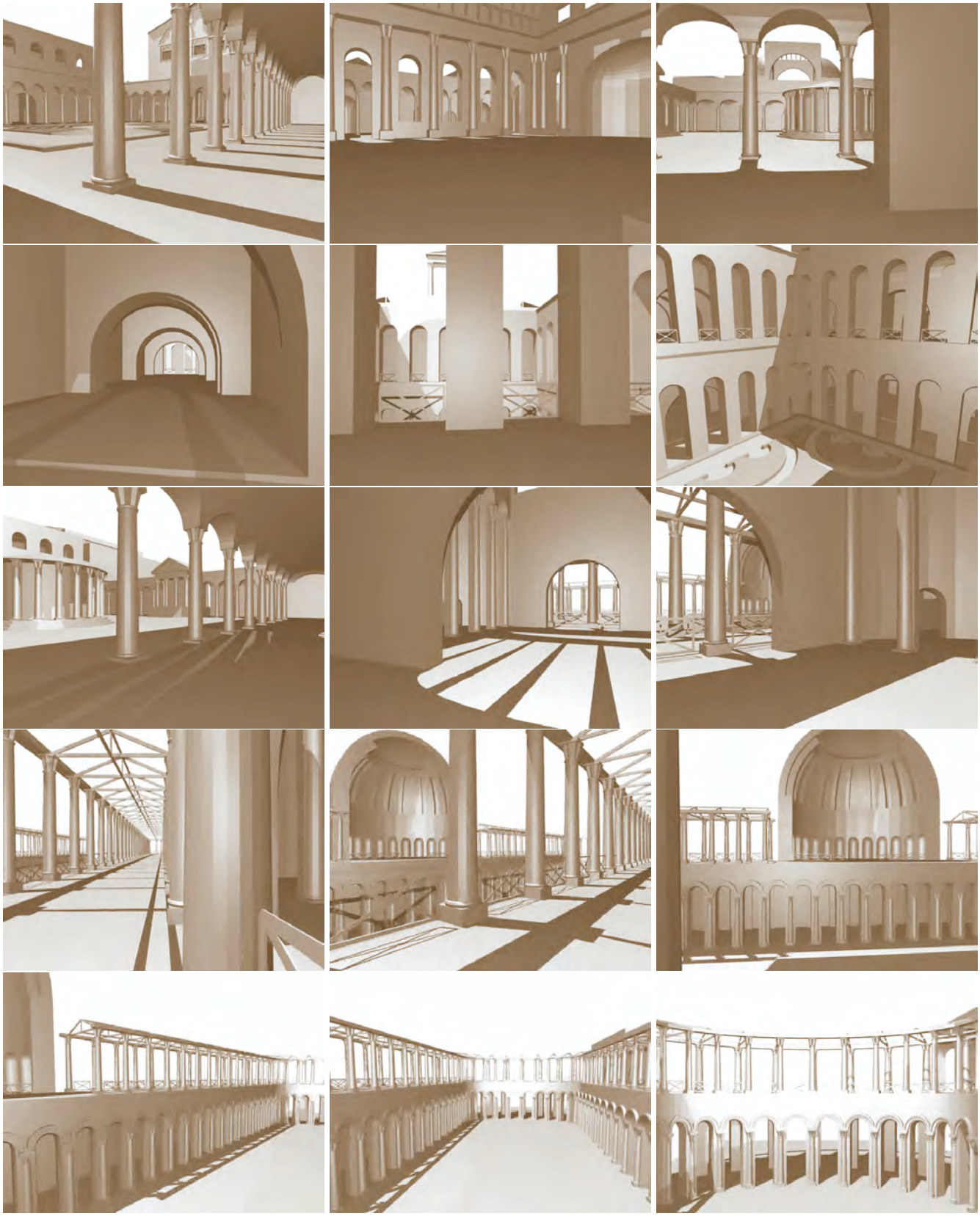

Figure 7: Several frames from the movie of a virtual "walk-through" the Domus Augustana and the Domus Flavia on the Palatine.

6 Conclusion

In this paper we have discussed a geometric programming approach to the 3D modeling of worth-valued historical and cultural sites, where a higher level of security and protection from threatening acts is needed. A streaming approach to progressive generation of geometry, either embedded in high-level scripting languages as Python or in concurrent functional languages as Erlang, was also outlined. TRS, a security spinoff of the University Roma Tre, was recently incorporated with the specific aim to develop this vision into the proper ICT technologies and to market the resulting products, also as plug-ins for existing security platforms. A lot of more work is in front of us, but the first results look very promising.

References

[1] Aiken, A., Williams, J. H., and Wimmers, E. L. The FL project: The design of a functional language, 1991. Unpublished report.

[2] Assogna, P., Bertocchi, G., Paoluzzi, A., Scorzelli, G., Vicentino, M., and Zollo, R. Securing critical infrastructures via geometric modeling and discrete simulation. In *Second Annual IFIP WG 11.10 Int. Conf. on Critical Infrastructure Protection* (George Mason University, Arlington, Virginia, USA, March 2008).

[3] Baas, M. Python computer graphics kit. Tech. rep., mbaas@users.sourceforge.net, February 2008.

[4] Backus, J. Can programming be liberated from the von neumann style?: a functional style and its algebra of programs. *Commun. ACM 21*, 8 (1978), 613–641.

[5] Backus, J., Williams, J. H., and Wimmers, E. L. An introduction to the programming language FL. In *Research topics in functional programming*. Addison-Wesley Longman Publ., Boston, MA, USA, 1990, pp. 219–247.

[6] Bajaj, C., DiCarlo, A., and Paoluzzi, A. Proto-plasm: A parallel language for scalable modeling of biosystems. *Philosophical Transactions of the Royal Society A 366*, 1878 (2008), 3045–3065. Issue "The virtual physiological human: building a framework for computational biomedicine I". compiled by Marco Viceconti, Gordon Clapworthy, Peter Coveney and Peter Kohl.

[7] DiCarlo, A., Milicchio, F., Paoluzzi, A., and Shapiro, V. Chain-based representations for solid and physical modeling. *IEEE Transactions on Applied Science and Engineering 5* (2008). (To appear).

[8] Fahmy, T. Pivy - embedding a dynamic scripting language into a scene graph library. Master's thesis, Vienna University of Technology, 2003.

[9] Gillies, S. *The Shapely Manual*. sgillies@frii.com, May 2008.

[10] Holzner, S. *Eclipse*. O'Reilly Media, 2004.

[11] Junker, G. *Pro OGRE 3D Programming*. Apress, 2006.

[12] Lucia, D., Martire, F., Paoluzzi, A., and Scorzelli, G. Top-down archeology with a geometric language. *Computer-Aided Design and Applications 5*, 4 (2008), 483–496. Preliminary version at CAD'08, June 22–27, Orlando, FL, USA.

[13] Meskini, N. *Python Bindings for the CGAL Library*. http://cgal-python.gforge.inria.fr/, 2009.

[14] Naylor, B., Amanatides, J., and Thibault, W. Merging BSP trees yields polyhedral set operations. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1990), ACM Press, pp. 115–124.

[15] OpenGIS. Implementation Specification for Geographic information - Simple feature access. Tech. Rep. 1.1.0 and 1.2.0, OGC–Open Geospatial Consortium, 2006.

[16] Paoluzzi, A. *Geometric Programming for Computer Aided Design*. John Wiley & Sons, Chichester, UK, 2003.

[17] Paoluzzi, A. *Geometric Programming for Computer Aided Design*. John Wiley & Sons, Chichester, UK, 2003.

[18] Paoluzzi, A., Milicchio, F., Scorzelli, G., and Vicentino, M. From 2d plans to 3d building models for security modeling of critical infrastructures. *International Journal of Shape Modeling 14*, 1 (2008), 61–78.

[19] Paoluzzi, A., Pascucci, V., and Scorzelli, G. Progressive dimension-independent boolean operations. In *Proceeding of the 9-th ACM Symposium on Solid Modeling and Applications* (2004), G. Elber, N. Patrikalakis, and P. Brunet, Eds., pp. 203–212. SM 04.

[20] Paoluzzi, A., Pascucci, V., and Vicentino, M. Geometric programming: A programming approach to geometric design. *ACM Transactions on Graphics 14*, 3 (1995), 266–306.

[21] Pascucci, V., Ferrucci, V., and Paoluzzi, A. Dimension-indipendent convex-cell based hpc: Skeletons and product. *International Journal of Shape Modeling 2*, 1 (1996), 37–67.

[22] Scorzelli, G., Paoluzzi, A., and Pascucci, V. Parallel solid modeling using bsp dataflow. *International Journal of Computational Geometry and Applications 18*, 5 (October 2008), 441–467.

[23] Steering Committee. Thread safe CAPI. Tech. Rep. RFC-3, GEOS–Geometry Engine Open Source, 2008.

[24] Van Gumster, J. *Blender For Dummies*. John Wiley & Sons, 2009.