A Tool for Collaborative Decision Making on Service Information Linked to 3D Geometry of Complex Hierarchical Products

Daniel González-Toledo[1], María Cuevas-Rodríguez[1], Carlos Garre[1], Luis Molina-Tanco[1], Arcadio Reyes-Lecuona[1]

[1]Universidad de Málaga; Málaga, Spain;

Abstract

During the lifecycle of many industrial products, different services are applied over the different parts or sub-assemblies of the product geometry, from design and validation tests to maintenance operations. After a service is run, a decision-making process usually starts from the analysis of the service results. This paper presents a tool for analysis and decision-making based on service results linked to the geometry of complex products organized in a hierarchy of sub-assemblies. The tool is based in the use of separate models for the product, the services and the analysis of results, and the generation of different views of the models including a virtual representation of the product with overlapped service result information. The virtual representation of the product consists in a render of a 3D scan of the product where techniques to handle visual occlusion between parts are applied. An application case is presented for collaborative discussion of inspection results of a power plant steam turbine.

Categories and Subject Descriptors (according to ACM CCS): I.3.8 [Computer Graphics]: Applications; J.2 [Physical Sciences and Engineering]: Engineering.

1. Introduction

The lifecycle of many industrial products includes situations in which several stakeholders have to analyse complex information that leads to important decisions. Examples of such situations are the evaluation of alternatives during the design stage or the maintenance actions following inspections during the operation stage. The information analysis process often involves a collaborative discussion between the different actors implied in the process. The analysis may imply decision-making regarding the potential actions to take (change or keep design, replace or repair part, etc).

In many cases, the product consists in a physical assembly of parts defining a product geometry; for complex products, this assembly is usually hierarchical, with several levels of sub-assemblies. In this type of products, most services are linked in some way with the geometry of the product. Design alternative tests are usually focused in one sub-assembly or one specific part, and different maintenance services may be run over different parts of the product or over an area or volume defined within the product geometry.

In this paper we present a tool for analysis and decision-making based on service results linked to the geometry of a complex hierarchical product. Most ideas in this paper can be generalized for many types of products and services, but the tool is presented around an application case in which the product is a power plant steam turbine and the service consists in a full inspection of the turbine for maintenance. The analysis of the inspection results is achieved through a discussion between power plant operators and inspection service engineers, leading to a final decision, which may range from re-scheduling future inspections to scheduling maintenance activities to repair or replace a part.

This application case is one of the cluster cases of EU funded project *Use-it-Wisely*, in collaboration with Tecnatom S.A, a Spanish company working in inspection and maintenance of power plant machinery. The preliminary work leading to these results was described in [RTG_14].

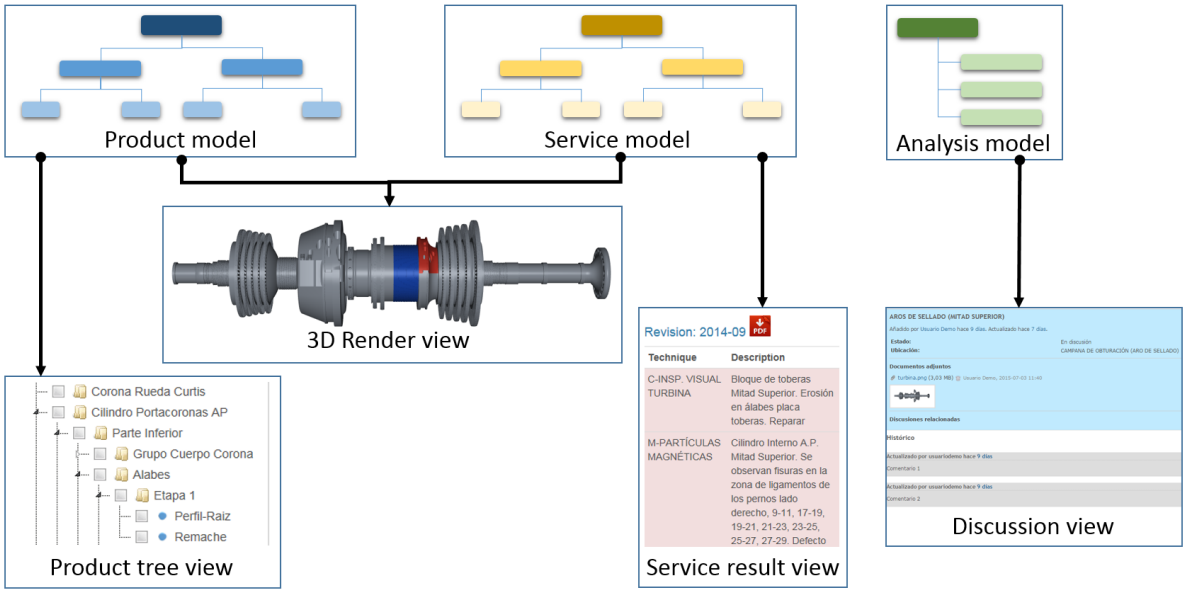

Fig. 1. Model-view architecture of the tool.

The rest of the paper is organized as follows. First, we present a summary of related work and an overview of our tool, which consists in a set of views generated from a set of models. Details on the models are given in Section 4, while Section 5 presents the detail of the views. We conclude by presenting specific implementation details of our application case and a discussion including future work plans.

2. Related work

Previous work has acknowledged the need to offer 3D mediated communication tools for industrial collaborative online decision making [SV13], [NLHK15]. The emphasis is in real time collaboration, including text chatting and video conferencing. Integrating synchronous communication in desktop applications, however, limits in practice their scope to simple 3D structures. In contrast, a power plant turbine is a complex structure with several thousand parts to which inspections can be performed. We share with such applications the goal of capturing the information that supports the decision process, and linking such information to the 3D product structure. However maintenace does not neccesarily require synchronous collaboration between the actors involved, allowing us to put the focus on managing such structural complexity to support maintenance decision making.

Communication and visualization of 3D structures is fundamental for facilitating decisions regarding maintenance of complex products –e.g. what parts of the product have or have not been inspected or where exactly has a problem been detected. Information associated with complex assemblies is difficult to visualize as some parts occlude others. There are multiple approaches to facilitate 3D structure understanding via occlusion handling techniques. These include using transparency, cutaways [Bur11] or exploded views [LACS08], [KTS09]. In our tool we extend these techniques to interactive web-based visualization. To the best of our knowledge, TeamPlatform 3D online 3D viewer is the only product available with goals similar to ours [Tea15]. However their global transparency and exploded view approaches do not take into account the point of view nor the selected parts in the product. Moreover our choice of Unity 3D Web Player Plugin as 3D rendering framework provides much faster interactive performance.

3. Overview of the tool

Our tool consists in a set of interactive views generated from three different data models. The product model contains the geometrical and hierarchical representation of the product (the steam turbine). The service model contains the results of the service (turbine inspection) linked to different areas of the product geometry (the turbine inspection consists in a set of inspections done to different inspection areas). The analysis model consists in a set of collaborative discussions, one for each inspection result.

The different views generated from the models allow the users to navigate through the product to access the service results and run the appropriate analysis. The typical use case of the tool can be seen as a three-step sequence:

1. The user navigates through the product model.
2. This navigation allows the user to access the different results contained in the service model.
3. For each accessed service result, the user may access the analysis model and start or contribute to the analysis of that result.

From these models, we generate four different views, as seen in Fig 1.

We have followed a user-centered design methodology, with two final users directly involved in the whole design, development and test process and with regular prototype test trials run with a representative group of stakeholders. This allowed us to extract the main components of the user mental model, which includes a set of relations between the

product parts: hierarchical relations (*this* is part of *that*), assembly relations (*this* is assembled on top of *that*), geometrical relations (*this* is beside *that*) and service relations (this result includes the inspection of *this* and *that*).

To navigate through the product, the user can use either the product tree or the 3D render views. The product tree view is used to navigate through the hierarchy of the product, while the 3D render view is more oriented to navigation through the geometry. The 3D render view shows basic service results information overlapped with the product geometry. To access more detailed information of a service result, when users reach an inspection area in the geometry (navigating through the product views), they may open the service result view associated to that area of the product geometry. From the service result view, the user may open the discussion view to collaborate in the analysis of that service result.

While the implemented product model with its two views can be used in many different applications involving complex physical products, the service and analysis models and their related views are application-dependent. In our case, the service model consists in a set of inspection results and the service result view shows textual information of a selection of inspection results. Our analysis model consists in a set of discussions, in which each discussion consists in a list of (textual and graphical) contributions from different actors. The discussion view shows the historic of contributions of all actors to a single discussion and allows registering new contributions.

4. Models

The tool allows exploration of information contained in the different models. In this section we present a general description of the different type of models with specific implementation details for our application case.

4.1. Product model

Our product model consists of two main parts: the product tree and the geometry data. The geometry data is a set of files containing the description of the geometry of different parts. In our case, our files contain 3D meshes generated through a 3D scan of a real steam turbine and a post-process with 3D modelling software. An alternative would be using CAD representations (either 3D or 2D).

The product tree consists in a tree structure where each node contains: a) Hierarchy information: parent and children nodes; b) Geometry information: associated geometry data and 3D transformations; and c) Assembly information: order in the assembly sequence of the product and direction in which it is assembled.

In our implementation, each leaf node is associated to one or more geometry areas (3D meshes, in our case) from the geometry data. While the product node represents a semantically atomic part of the product, we store separate meshes for different areas of a single part to allow connection with the service model. For each associated geometry area, the node stores also a 3D transform with the position and orientation of the mesh in the global (product) reference system. This allows to build products containing many instances of the same geometrical parts without the need to store separate meshes for each instance.

While the geometry information specifies where each part area is located in the final product (through the 3D transforms), it does not contain information about the assembly procedure of each part. This information is necessary for the generation of exploded-views of the product, allowing to expose internal parts which are completely occluded in the 3D render view. The type of assembly information to store depends on the type of exploded-views to generate, but typically includes an assembly sequence (sequence of explosion) and the assembly direction (explosion direction) of each part.

4.2. Service model

The service model is application-dependent. In our case, turbine inspections are organized in a tree structure based on the different types of elements and areas to inspect rather than on product geometry. Fig. 2 shows an example, in which the turbine/product tree has a node called *Rotor*, with children nodes corresponding to the different *Stages* of the rotor, and each stage node with children for the different blade *Rings*. Instead, the inspection/service tree has a single node for *Blades* inspection, with children for the different *Stages* and each stage node with children corresponding to the different areas of the blade to inspect (*Profile*, *Root*).

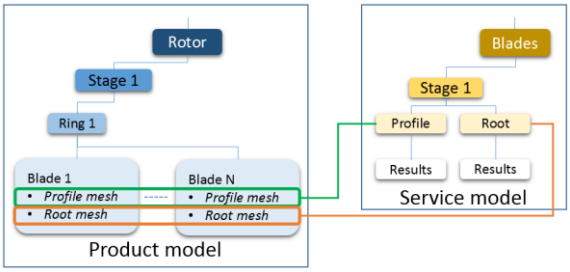

Fig. 2. Example of link between the product and service models.

The *Profile* node of the service tree contains a link to all profile meshes of *Blade* nodes in all *Rings* of *Stage 1*. This link between product areas/meshes and service tree nodes allows to access the service results when navigating through the product. There exist three types of nodes in our service model tree: a) Inspection results: leaf nodes, containing the final information we want to access. Each leaf node contains the result of one inspection technique applied over one inspection point in one date; b) Inspection points: are in the second-lowest tree level (leaf-parent) and contain the link with the product areas; and c) Other nodes: all nodes above the inspection point level are used to organize and give a structure to the inspection procedure.

4.3. Analysis model

The analysis model depends directly on the service model and thus is application-dependent. Typically, the analysis model will consist in a data structure linked to the service model, so that different analysis procedures can be run for different service results. In our case, the analysis model consists in a set of *Discussion* data structures, where each discussion is linked to one inspection point node of the service model. We constrain the analysis model so that one inspection point may be linked to one discussion only if there is at least one inspection result for that inspection point.

Each *Discussion* data structure consists of: a) Discussion state, which may be: *empty* (discussion for this inspection point has not yet started), *open* (discussion currently in progress) or *closed* (a decision was taken and no more contributions are accepted); b) Link to the corresponding inspection point in the service model (one-to-one); c) The discussion information itself: a set of contributions from different actors, which may include textual annotations, and/or attached images or documents; and d) List of related discussions from other inspection points. The possibility to establish relations between discussions allows the decision making process to be less dependent of the service model structure.

5. Model views

In this section we present the different views generated from our models. Most views are generated from data of a single model, while the 3D render view is generated combining data from the product and service models.

5.1. 3D render view

This view combines information coming from the product and service models. It consists in an interactive 3D render of the whole product allowing the user to navigate through the product geometry. The view allows multiple selection of product parts using the mouse. When the user makes a selection, a context menu is open presenting several options which will be described in the following paragraphs. Since some parts may be occluded by other parts, some mechanisms to deal with occlusion have been implemented.

Navigation

The choice of DoFs for navigation depends on the user mental model of the product. In our case, we found that axial symmetry is what governs the mental model on how to move around the turbine. Navigation based on cylindrical coordinates would be a natural choice for this mental model, but it does not allow all viewpoints to be accessed. We instead implemented navigation along ellipsoidal coordinates, adjusting the forward and up vectors of the camera to keep the turbine shaft horizontal.

When the user makes a selection, the context menu provides a *Focus* option, which makes a smooth transition to an optimal viewpoint for maximum visibility of the selected part/s. The shape of the navigation ellipsoid is then adapted to the bounding box of the selection.

Occlusion handling

Some parts cannot be selected nor even visualized in the default 3D render of the product. Visual occlusion occurs depending on viewpoint, but there may be parts which are always occluded regardless of viewpoint (as in the case of small parts inside bigger container parts). When the user makes a selection, the context menu provides an option to make that selection semi-transparent. This allows the user to see through container parts, but not to access contained parts inside for selection. To mitigate this problem, the interface provides a slider for setting transparency level with a double function: it gradually changes the amount of transparency of the selected parts until it reaches a threshold, after which the part is completely removed from the render, so that it no longer blocks access to the occluded parts.

Manual selection of the parts to make transparent is a slow procedure when users know exactly which part they want to view or access. Moreover, the selection of parts to make transparent changes depending on the viewpoint, so the user should manually change selection while navigating. To assist the user with transparency choice, when the user makes a selection, the interface presents an *adaptive transparency* option, which can be switched on and off. Adaptive transparency continuously changes the degree of transparency of all occluding parts while navigating (changing viewpoint) through the product (See Fig. 3).

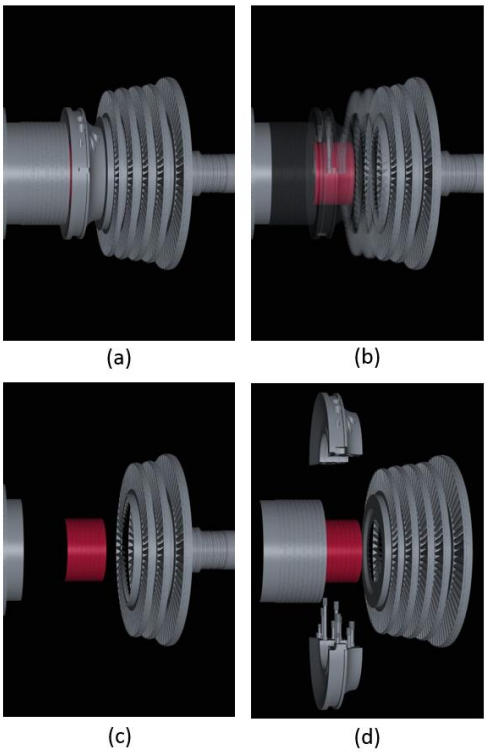

Fig. 3. Occlusion handling in the 3D render view. (a): in the default render, a portion of the turbine shaft is occluded by the crowns. (b): when applying transparency to the occluding parts, a portion of the turbine shaft is revealed in which the red color indicates a flaw result in the service model. (c): hidding the occluding parts allows complete access to the occluded part. (d): the exploded view allows access to the occluded part without visual removal of parts.

The interface provides also options to generate different types of exploded views of the product. We implement the symmetric and focused layouts of [KTS09] and the interactive riffling of [LACS08]. The exploded views allow the user to see and access internal parts while keeping the spatial relations with the rest of the product, without the need to visually remove other parts. The exploded views are generated using the assembly information stored in the product nodes. In our case, the assembly information was manually introduced, but it could be automatically generated using the method of [LACS08]. We store assembly information for all product nodes, not only leaf-nodes, to obtain explosion layouts where the accumulative explosion of each tree level exhibits the hierarchical nature of the product.

Overlapping service model information

There exist different methods to overlap information from the service model with the geometry of the product, such as using labels [TKGS14] or modifying the render of specific parts or areas. In our implementation, we show most details of the service in the service result view and we just want the 3D render view to show at a glance where the most important results are located. Our intention is to focus the attention of the user rather than providing detailed information of the service results. For this, our interface provides an option to mark in red the parts having at least one inspection result with a detected flaw, which are the results on which decision-making is required.

5.2. Product tree view

As seen in Fig. 1, this view allows the user to navigate through product hierarchy with an interface similar to that of a file explorer. The nodes can be expanded and collapsed until reaching leaf nodes corresponding to atomic parts. Multiple selection can be done at any level of the product tree. When a set of nodes is selected, all corresponding areas/meshes will be highlighted in the 3D render view; in a similar fashion, selecting one mesh in the 3D render view will highlight the corresponding leaf node in the product tree view. This link between the product tree and 3D render views allows a navigation through the product combining the user knowledge on both product hierarchy and geometry.

5.3. Service result view

This view is typically open after the user access a specific area of the product using the product tree and/or 3D render views. If there exists at least one service result for that area, the user has an option to open the corresponding service result view. In our implementation, filter-based search facilities have also been implemented, allowing the user to reach a service result view without the need to navigate through the product. The service result view may show information from more than one inspection point, when the user selects multiple areas or after a filter-based search.

The service result view shows all information contained in one or more leaf nodes of the service model (inspection results) which is completely application-dependent. In our case, we show textual information regarding the inspection technique applied, the inspection date and the description of the potential flaws found (the inspection result itself). The service result view includes a link to the discussion view related with the inspection point to which the inspection result belongs.

5.4. Discussion view

The discussion view is accessed through the link present in the service result view, given the one-to-one relation between service results and discussions. When the discussion state is *empty*, the first access to the discussion view will automatically change its state to *open*.

The layout of the discussion view is different depending on the discussion state. For open discussions, the view provides a set of controls for registering new contributions (in the form of text or attached files), establishing relations with other discussions and a button to change the discussion state to *closed*. For closed discussions, the view consists in a read-only summary with the historic of all contributions, including the final decision if it was annotated.

6. Implementation and results

There exist different choices for data storage and access of the different models. For the product model, we have stored the product tree inside a MySQL database and the geometry data as separate *.blend* files. Since our geometry files are large, we store the file paths in the database rather than using a BLOB approach. A different approach to store the product tree would be to use specifically formatted JSON or XML files. The same applies for the service model, which we implement as separate tables inside the same database. For the analysis model, we have used a customized installation of Bitnami Redmine [Les13]. All discussion data is stored inside Redmine database, except for the link with the service model, which is stored in a table inside our database, relating inspection point IDs with Redmine issue numbers.

To implement the functionality of our application, including the generation of the different views, we used the ASP.NET MVC framework. Most functionalities have been implemented from scratch in C#, except for the 3D render view and the discussion view. Fig 4 shows a final layout of our application combining different views.

The 3D Render View has been implemented using Unity Pro 5 with C# scripting and is embedded in the application using the Unity Web Player plug-in [Gol09]. To generate the Discussion View, we launch a new tab in the browser with the corresponding issue view of our customized Redmine installation (See Fig 5). The layout and aesthetics of the issue view have been highly modified, leaving only the functionalities needed by our application: registering notes, attaching

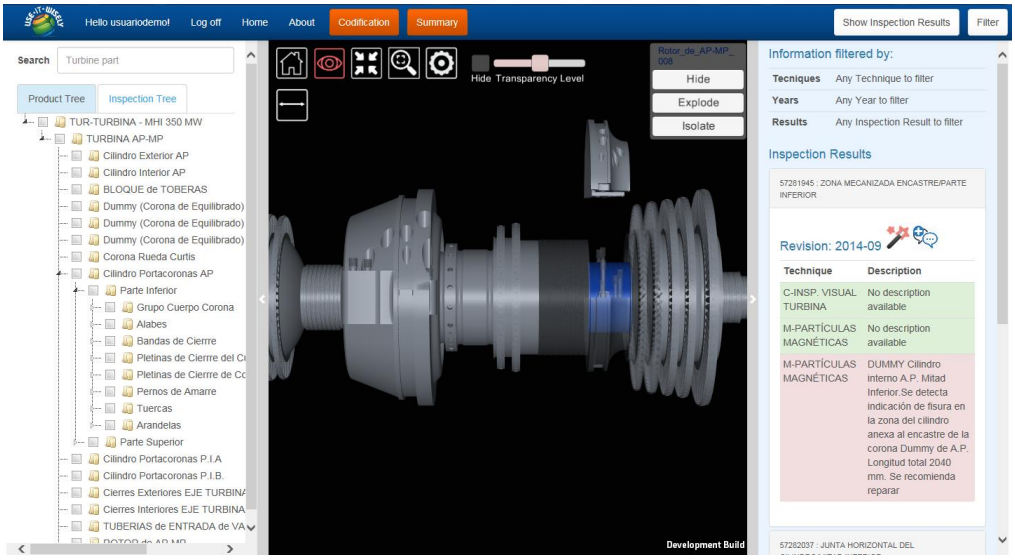

Fig. 4. Composed view layout of our turbine inspection tool. Left: product tree view; Middle: 3D render view; Right: service result view.

files, relating issues and following issues. Discussions (issues) are created and modified from the application using the Redmine REST API. For the *Discussion View* of *closed* discussions, the application downloads the issue PDF generated by Redmine and stores it in the application database.

7. Conclusion

This paper presents an application which allows navigation through a virtual representation of a complex hierarchical product to access information regarding the results of a service run over different parts of the product geometry. The application provides a tool for analysis of the service results. The paper presents general ideas applicable to different types of analysis of different services over different products and its implementation over a specific application case, consisting in the discussion (analysis) of inspection results (service) over a steam turbine (product).

Future work plans include augmenting the amount of service information overlapped in the 3D render view by using labels [TKGS14], running experimental tests with the final users to improve the usability of the interface and improving occlusion handling with more advanced transparency techniques [Bur11] and more interactive exploded views. The 3D scanning and modelling process of the full turbine is still in progress and we are adding more parts to the virtual representation of the product when they are available.

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement No. 609027 (Project Use-it-wisely). We would like to thank Tecnatom S.A. and the Human Factors Research Group of University of Nottingham for their collaboration.

References

[Bur11] BURNS M.S.: Efficient and comprehensible visualization of complex 3-D scenes. PhD Thesis, Department of Computer Science of Princeton University. January 2011.

[Gol09] GOLDSTONE W.: Unity game development essentials. Packt Publishing Ltd 2009. ISBN 978-1-847198-18-1.

[KTS09] KALKOFEN D., TATZGERN M., SCHMALSTIEG D.: Explosion diagrams in augmented reality. In Proc. 2009 IEEE Virtual Reality Conference (pp. 71–78). IEEE. doi:10.1109/VR.2009.4811001.

[LACS08] LI W., AGRAWALA M., CURLESS B., SALESIN D.: Automated generation of interactive 3D exploded view diagrams. ACM Transactions on Graphics, 27(3), 1. doi:10.1145/1360612.1360700.

[Les13] LESYUK, A.: Mastering Redmine. Packt Publishing Ltd 2013. ISBN 978-1-84951-914-4.

[NLHK15] NYAMSUREN, P., LEE, S.-H., HWANG, H.-T., KIM, T.-J. A web-based collaborative framework for facilitating decision making on a 3D design developing process. J. Comput. Des. Eng. 2 (July 2015), 148–156. doi:10.1016/j.jcde.2015.02.001

[RTG_14] REYES-LECUONA A., MOLINA-TANCO L., GONZALEZ-TOLEDO D., FLORES S., FRUTOS E., PATEL H. HOUGHTON R.: Design, Maintenance and Refurbishment of Turbines in a Collaborative Environment. In Proc. AHFE Conference, 2014.

[SV13] SILTANEN, P., VALLI, S. Web-based 3D Mediated Communication in Manufacturing Industry, in: Concurrent Engineering Approaches for Sustainable Product Development in a Multi-Disciplinary Environment (2013). Springer London, pp. 1181–1192.

[Tea15] TeamPlatform. http://www.teamplatform.com. 3D Systems Inc. Retrieved July 2015.

[TKGS14] TATZGERN M., KALKOFEN D., GRASSET R., SCHMALSTEIG D.: Hedgehog labelling: view management techniques for external labels in 3D space. In Proc. IEEE Virtual Reality (April 2014). , pp 27-32. doi: 10.1109/VR.2014.6802046.