Web-to-Real Application for Artistic Robotic Surface Processing of Natural Stone

Sven Stumm[a,b], Matthias Neuhaus[a], Johannes Braumann[a,c], Jan Brüninghaus[a,b], Sigrid Brell-Cokcan

[a]Chair of Individualized Building Production RWTH Aachen University, Germany
[b]Association of Robots in Architecture, Austria
[c]UfG Linz, Austria

Abstract –

Stonemasonry is a traditional industry with a long history. Due to increasingly competitive markets in a hazardous work environment with high physical strain on the stonemasons, a system was developed to automate natural stone surface processing in order to diminish repetitive movements in manual labor. This is achieved using a hammering mechanism and traditional mason's chisels, mimicking traditional techniques and surface appearance. The producible patterns can be informed by a wide range of inputs, from the sampled brightness values of images to 3D attractor points. In order to mediate between the complex possibilities and end-user friendly design techniques, we developed a Web-to-Real Application prototype. This allows for a streamlined process from design to production. For a person without training in stone masonry the results of the chiseling process are very unpredictable. Hence the goal of this application is to give the user the opportunity to see how a certain pattern or image would look like when chiseled 3-dimensional into various stones. The user can select the type of natural stone and structuring pattern or upload an own image. From this information, a first rough estimation of the chiseled surface is calculated and shown to the user, directly within the browser. If the user requires a more accurate prediction of the resulting stone surface, the information is transferred to a remote computer where robot paths and final surface appearance are calculated within an offline programming system.

Keywords –

Natural stone, surface processing, robotic application, Web-to-Real

1 Artistic Stone Structuring - AROSU

The usage of natural stone and the processing surface processing by stone masons has a history of many centuries in Europe. The history of this craft can be seen on old buildings like palaces, churches and castles. Nevertheless, due to cheap transport costs and the globalization of markets, the stone industry in Europe is under increasing pressure from low-wage countries.

In the FP7-SME project AROSU (Artistic Robotic Surface Processing for Stone) new software strategies and hardware tools that enable industrial robots to efficiently structure natural stone surfaces were developed. While one of the most immediate uses of the strategy is to utilize it for large, repetitive patterns, we can use the same, parametric control strategies to create irregular structures, e.g. based on sampling the brightness of raster images, and thus transferring images into stone patterns. We believe that this opens up a large market not only for new facades and interior design, but also for end customers. In order to streamline the process and make such products as accessible as possible, we have created a Web-to-Real interface that allows the user to customize, evaluate, and order individualized stone panels. Figure 1 shows a first workstation where natural stone can be processed and customized patterns are created.

Figure 1. Workstation for natural stone processing

and two resulting stone surface patterns.

Based on the CAD software Rhino which is extended by the software Grasshopper for graphical parametric programming, a simulation environment was created. Grasshopper was in turn extended by a model for stone processing which was developed in accordance with the analysis of processing different stone types in an AROSU workstation. The resulting groove patterns from different strokes with the tool were analyzed [1] and a process model was created. Using KUKA|prc the robot path can be programmed in the same grasshopper environment. By integrating both robot path planning and the stone processing analysis, an AROSU simulation environment was created.

2 Towards Web-to-Real Interfaces

For modern CNC machines it does not matter if a machine performs the same procedure 1000 times, or 1000 individual procedures. The challenge of providing the customer with the possibility of customizing an object lies mostly in the programming of that machine, towards automatically creating programs that are safe and lead to aesthetically pleasing results. Thus, for a long time the easiest way to provide customization was through modular designs such as in the automotive sector, but also for prefabricated housing in architecture. Rather than customizing individual parts, the entire object was customized by replacing entire modules, such as e.g. the interior lining of a car. Among the first large-scale and publicly known services to allow full customization were 3D-printing providers such as Shapeways that allowed users to upload any 3D-geometry and have it printed. The big advantage of other technologies was that the requirements of 3D printing were very clearly defined and enforceable mostly through automated checks: As long as an object fits into a given volume, has a sufficient wall thickness and no geometric defects, it can most likely be printed.

Today, advances in CAD technology allow new approaches such as Tylko (.com) where the user can custom-design furniture by adapting a series of parameters and immediately, virtually place the furniture in the room through augmented reality.

The production of artistic stone surfaces requires nowadays the expertise of a skilled stone mason. However the different manufacturing techniques and variety of stones make it difficult for a non-mason to predict and successfully communicate the intended design. In the AROSU project we have developed new and innovative workflows that allow users to program stone structuring toolpaths quickly and intuitively, but also to predict the visual effect of a toolpath as well as to check it for collisions, unreachable positions, etc. and that can be directly executed on the robot [2]. As such, the base part of a Web-to-Real application was already created, however the technical challenge of transitioning from a workstation-based PC software to a web application remains.

3 Initial Prototypes

The initial idea of the Web-to-Real AROSU was to build upon the game development platform Unity 3D, the main reason being the possibility of deploying Unity software on a wide range of platforms, from mobile to the browser. As such, a first prototype (Figure 2) realized within Unity used KUKA|prc's kinematic solver to simulate the robotic arm. For the preview of the material effect, each chisel stroke was rendered to a bitmap as a line and slightly blurred. The resulting texture was then used as a heightmap, setting the height deformation of a plane. Finally, a texture relating to the chosen kind of stone was blended over, creating a sufficiently realistic preview of the AROSU result.

Figure 2. Initial Web-to-Real AROSU prototype running within the browser using Unity 3D

However, implementing the different structuring strategies proved to be complicated, as the previously Grasshopper-based visual code had to be entirely rewritten. Another significant issue was that Unity 3D required a separate plugin to display applications within the browser. While during the first planning stage of the project, browser plugins were very prominent, safety concerns have since greatly reduced the acceptance of plugins or even prohibited their use. So much so that even Unity themselves announced that in March 2016 the Unity web player will become an unsupported product [3]. The use of the WebGL export is recommended however in 2015, while Unity's WebGL support was still

in beta.

In order to create a powerful and extensible platform it was decided to rebuild the Web-to-Real application using current web technologies as a "headless" AROSU client.

4 Designing for Mass-Customization

One of the main concepts of Web-to-Real is to give the user the ability to design and create individual objects without requiring any expert knowledge or specialized, expensive software. Thus, the newly developed application basically acts as a client that sends information to the server and displays the information that is returned. In order to be able to update and customize the server in parallel to the regular software framework, the AROSU server was set up to work within Grasshopper, using the regular AROSU toolpath strategies, managing the individual data inputs and outputs. As such, any parameter available to regular users would therefore also be available within the Web-to-Real application. It was therefore highly important to find a middle ground that on the one hand provides the user with a large range of inputs towards allowing very diverse and individualized products, while at the same time providing a clean and easily readable interface. Unlike current, "static" design, where the designer creates a single, aesthetically pleasing and functional object, designing for mass customization requires the creation of an entire range of designs, while still fulfilling all functional and aesthetic requirements.

We have identified a range of core inputs and outputs which are always provided to the user:

- **Material**: A very large number of different natural stones exists. They differ in mechanical properties as well as in appearance and haptic. The particular stone also has a significant impact on the price.
- **Size**: The stone size in mm is one of the key price factors. For practical reasons, the selection has to be constrained within reasonable panel sizes. This is mainly influenced by the underlying machine setup and the available block size for the specific material.
- **Thickness**: While a minimum of thickness reduces costs, it increases the risk of splitting the stone. The minimum thickness is set by the combination of a particular stone and AROSU structuring strategy.
- **Strategy**: The strategy has got the largest impact on the visual result, while also influencing the machine time factor of a given element [4].

As the output, the user will require at least the following data points:

- **General Feasibility**: Machine reachability, material availability
- **Design Preview**: A reliable, possibly abstract, preview of the individual design
- **Costs**: Costs based on material, machine time, other surcharges

Depending on the strategy, significantly more parameters may be required. The first proof of concept provided basic functionality, by allowing custom inputs and outputs, resulting in a custom KUKA *.src toolpath file that can be executed at the robot.

4.1 Technical Implementation

As described, the feasibility of the Web-to-Real approach employing a basic integration into the cross-platform 3D engine Unity was verified. However, after this first test a scalable server client structure and a more accurate prediction of the surface processing needed to be designed. In order to create a fully integrated user experience we decided on a browser based cross platform approach employing WebGL. The only requirement for the user is therefore a browser supporting HTML5 (client). In order to improve upon the prediction quality we integrated the existing simulation software, incorporating the prediction algorithm of the project partner XXX, as the server backend. The material removal simulation is based on a conclusive analysis of the process [1] and its influential parameters [5] to generate a model of the future design object close to reality. The user has the possibility to load a chosen image file (jpeg, png, bmp), which is in turn used to extract the groove pattern in accordance with the images contrast variations. After a first a priori on client visualization by extracting a grayscale height map from the image, the image and other parameters can be sent to the server where they are processed to create a more realistic rendering of the stone structure. In order to avoid the feeling of delay to the user and design a more immersive experience during the calculation and simulation of the robotic work process, in the future it would be possible to show the robot movement within the client application. In order to implement these features in a user friendly way the following structure and workflow was designed.

The implementation consists of a part running in the user's browser, a web server and a server backend, which redirects simulation requests to the AROSU simulation environment. The client is used as the user interface, i.e. to handle all the user input and to display the results. The user loads an image with a geometric pattern from a file, sets a value for the desired maximum depth of a chisel hit and selects the type of stone. Then a coarse prediction of the result is calculated locally in the browser. If the user is satisfied with the result, he can send it to the remote

server to get a better prediction. On the server a better prediction is calculated using the AROSU prediction algorithm, which was trained using actual processing parameter and results. The prediction of the surface is sent back to the user's browser and displayed.

The client part is implemented using HTML5 and JavaScript. The user interface consists of a 3D view of a stone tile and various control elements. The THREE.js library is used for creating the 3D view. For creating a visualization of the processed surface, a heightmap is generated from the user's pattern file using the red, green and blue color values. This heightmap is used to structure the top side of a cuboid, mapping the intensity of the image to heights and depths of the surface. After that a texture is applied to this cuboid, creating a realistic view of the selected stone type.

After this, the user is able to request a better prediction of the result by uploading his pattern image to a server. The server backend contains a Grasshopper component in a running Rhino instance. The connection between server and client is realized using the WebSocket protocol. When the image and chisel depth were uploaded to the server, a prediction of the result from processing the stone tile is calculated. For this, a path of the robot for processing the tile is calculated based on the uploaded pattern image. On this path the result of the chisel hits is calculated. The resulting surface is transformed into a heightmap, which is in turn saved as an image and transferred back to the client using the WebSocket connection. In the client browser, the image is used as a heightmap for displaying the result in the same way as the initial image uploaded by the user. Figure 3 illustrates the flow of information within the Web-to-Real Application in a sequence diagram.

The environment is divided in a frontend (client) and backend (webserver and server simulation backend) and therefore provides two different layers of the Web-to-real application. The frontend shows initially a simplified form of the scene within the THREE.js browser interface. When requesting a more accurate simulation the required parameters are sent to the backend where the result of the resulting groove pattern is predicted. This is done using the same toolpath calculation as can be used within offline programming of the real robot installation. For this reason the same conditions and calibration of the fabrication process is used. Therefore the resulting robot program can directly be used for the processing of the stone. As an extension, a combination with an ordering or shop system could be done. This would give the possibility to save the program together with the price quotation when a customer decides to buy a specific stone using the Web-to-Real application. In this scenario the Web-to-Real application would be able to reduce workload by already providing all required information for the fabrication.

It should however be noted that the interface is currently only a prototype implementation. That can be used as a proof of concept for possibility and feasibility of such an interface. A number of further developments on the application for deployment of the software is still required before using it in a productional environment. A better handling of multi-client access for scalability, or an integration of chisel movement visualization within the interface for example. For an easier approach two major components of the application where analyzed separately. One of these parts was the heightmap calculation to visualize the stone processing. The other part was the simulation of robot motion.

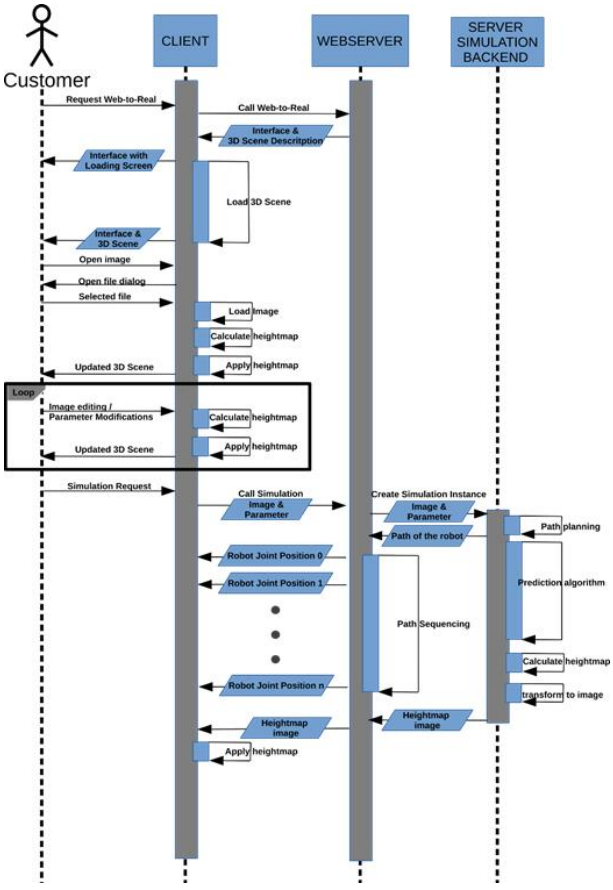

Figure 3. Illustrates the flow of information between the different parts of the tool

After finishing the proof of concept and verifying the possible information flow, a new design for the interface was planned. Figure 4 illustrates the new interface design, which employs a WebGL & Javascript based 3D Scene as well as different menus using HTML and Cascading Style Sheets (css) based overlays. The interface combines the necessary functionality to support the required user

interactions.

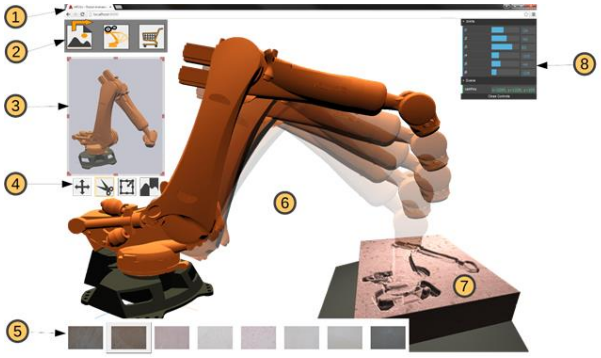

Figure 4. Design of the Web-to-Real interface including elements of the current prototype implementation

In the following we will describe the interface menu in more detail as illustrated in Figure 4. Element 1 shows a standard browser search bar. The menu item number 2 illustrates the main menu, which is still very similar to the first Unity implementation. Therefore the main menu adds the functionalities of loading an image from the computer's hard drive, starting the simulation as well as requesting a quotation and ordering. Interface part 3 shows the image preview which can be used for some basic editing functionalities. Menu 4 contains the image editing functions. The editing allows for alignment (translation and rotation), cutting, scaling, as well as inversion of the image. Element 5 illustrates the material selection giving a preview of the stone texture as well as allowing for advanced settings to set the stone size. Part 6 shows the 3D WebGL environment including the stone with an applied heightmap of the image (element 7). Section 8 shows additional information for the simulation and can include messages about possible manufacturing problems as well as allowing to set other parameters for the fabrication. Figure 5 shows the resulting prototype, which still implements the heightmap calculation and the simulation of robot motion in separate from each other.

5 Challenges

One of the main challenges of Grasshopper is that it is fundamentally single threaded (though allowing individual components to work with multiple threads). As such, further testing will be required to determine how Grasshopper can cope with a large number of clients uploading/changing data in parallel. A custom schedule may be necessary to buffer any incoming data and process it in series, so that a high demand only leads to larger processing times, rather than lost requests or error messages. A container based approach can also improve the scalability of the backend.

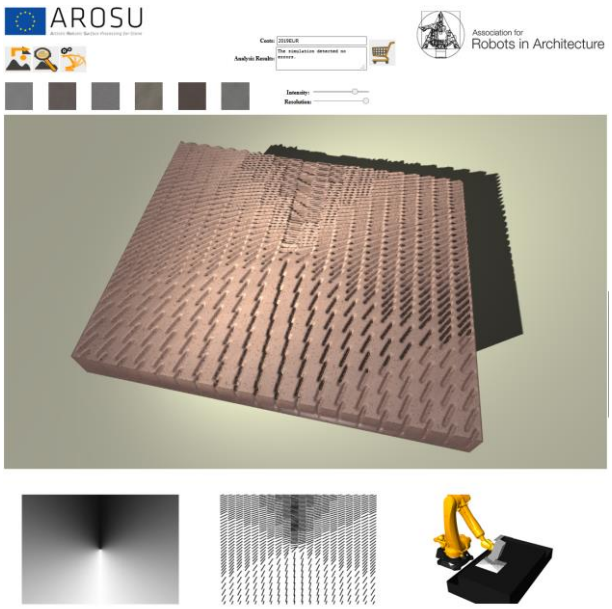

Figure 5. Current prototype of the Web-to-Real interface

6 Web-to-Real for a traditional industry

Skilled stonemasons have performed traditional techniques of natural stone surface processing for centuries. Within the project AROSU repetitive manual labor was automated and an easy to use interface for mass customization was created. Through the implementation of a Web-to-Real platform new possibilities for commissioning were created. Designers are not only able create and commission their own surface design in a mass customization approach, but are even able to verify feasibility and quality through a preprocess simulation.
The used approach ensures that current simulation is up to date and the results can even be employed for actual robot programming. Any optimization done to the process has a direct effect on the result seen by the customer. The customer gets a direct connection to the product created and has clear influence on how it turns out. Simultaneously the usability of the interface was kept simple through a direct upload of a pattern as an image file, allowing the customer to create a design in whatever software they are used to. The pre-process simulation allows for a direct verification of the design via the web. This adds direct feedback to the idea of a streamlined process from design to production. Thus a

new type of commerce was introduced to the natural stone processing industry. The possibilities and results of this new customized commerce with direct integration of the customer will be analyzed further in future work.

7 Conclusion

The developed Web-to-Real application shows how a custom browser interface can make a specialized, flexible process accessible to the wide public. Rather than having to duplicate the capabilities of CAD software, data is simply exchanged, requiring just the underlying CAD software Rhinoceros 3D. As Rhinoceros 3D can use software to render the OpenGL viewport, it would even be possible to deploy a large number of virtual machines, without requiring special 3D-hardware, e.g. via Azure or Amazon AWS.
While a dedicated solution building solely upon web technology would be preferable over the current "headless-CAD" approach, it is considerably more expensive and less flexible, requiring a programmer to change the inner workings of the application. For the current interface, changes can be performed by any Grasshopper user.
We believe that the Web-to-Real workflow presented in this work package will open new distribution channels for SMEs, quickly promoting the ideal of customized, robotically-fabricated stone panels.

A demonstrator of the Web-to-Real application will be presented as a part of the exhibit by the European Robotics Association (euRobotics) at Automatica in Munich, the 7th International Trade Fair for Automation and Mechatronics.

8 Acknowledgments

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n°606453.

References

[1] Steinhagen, G. and Brüninghaus, J. and Kuhlenkötter, B. Robotergestützte künstlerische Steinbearbeitung, In: Fachtagung Mechatronik, pages 103–108, Dortmund, Germany, 2015.
[2] Braumann, J. and Brell-Cokcan, S. Adaptive Robot Control - New Parametric Workflows Directly from Design to KUKA Robots, In: Real Time - Proceedings of the 33rd eCAADe Conference, pages 243–250, volume 2, 2015.
[3] Jonas Echterhoff Unity Technologies: Unity Web Player Roadmap October 8, 2015 in Technology http://blogs.unity3d.com/2015/10/08/unity-web-player-roadmap/, Accessed: 15/03/2016.
[4] Steinhagen, G. and Braumann, J. and Brüninghaus, J. and Neuhaus, M. and Brell-Cokcan, S. and Kuhlenkötter, B. Path planning for robotic artistic stone surface production, In: Robotic Fabrication in Architecture, Art and Design 2016, Sydney, Australia, 2016.
[5] Steinhagen, G. and Kuhlenkötter, B. Analysis of the material removing mechanism for an automated chiselling approach, In: 3rd International Conference on Stone and Concrete Machining, 2015, pages 61–71, Bochum, Germany, 2015.