Participatory Architectural Design and Fabrication with Natural Materials in Native Forms

Figure 1: *Left: an overview of the workflow: 1. fix branches on plates. 2. scan the plates and upload the model. 3. play the game with scanned branches. 4. fabricate joineries by a CNC (Computer Numerical Control) router. Right top: branch layouts designed by authors and end users. 6 and 7 were fabricated in a workshop with participants. Right bottom: the fabricated 2D fence (2000 mm × 900 mm). Each pair of branches is connected with rigid lapped joinery.*

Abstract

Diverse natural materials such as stones and woods have been used as architectural elements preserving their native forms since primitive shelters, however, the use of them in modern buildings is limited due to their irregular properties. In this paper, we take the diversity as playful inputs for design task, and present our game-based design-fabrication platform for customized architectural elements. Taking tree branches as a material with native forms, a game *BranchConnect* enables end users to design 2D networks of branches and fabricate it by a CNC router. The game considers fabrication constraints such as limitations with ordinal 3-axis CNC routers. Each connection has a customized unique joinery adapted to the native forms of branches. The scoring system of the game guides users to design structurally sound solutions with given branches. Together with low-cost mobile scanning devices, users with diverse contexts can contribute to design and fabrication process not only by playing the game, but also by collecting branches around their physical environments and uploading them to our online platform. For validating our process, we conducted a workshop with children and their parents from a local community. They collected branches in a nearby forest and contributed to design and fabricate a 2D fence with our system.

Keywords: Fabrication, Collaborative Design, Human-in-the-Loop

Concepts: •**Computing methodologies** → **Image manipulation;** *Computational photography;*

1 Introduction

Modern buildings are characterized by its uniformity; built upon the same principle of construction system, consisting of standardized building component and its assembly process. The standardized construction system is favored because of its efficiency in design and production. Each component satisfies specified structural performance, thus the assembled resulting structure can be analyzed systematically. On the other hand, excessive standardization converged buildings into similar materials and details, resulting in the detached design from built environment. Reacting on the issue, designers and architects actively use local materials not only as inspirational sources, but also as a catalyst of their design to the local context [Oliver 1997].

Since primitive shelters and huts, traditional construction has used locally found natural materials directly in their native forms [Weston 2003]. Such a direct use can not compete with highly standardized materials and construction system, however, the uniqueness of native forms is a valuable quality which is lacking in standardized materials. As the material is locally obtained, building and living get much closer, thus people using the building can easily commit design and fabrication, fostering the sense of belonging to the community. Locally obtained materials can easily connect design and the context of built environment in this way, but their irregular properties limit the use of them in modern buildings. Traditionally, craftsman has taken care irregular natural materials varying their native forms and dynamically design the global design by considering individual material properties [Pye 1968]. Such a task is difficult to be automated and these skills are developed through years of training, thus the use of native forms typically inaccessible for end users and costs more than

SIGGRAPH 2016 Posters, July 24-28, 2016, Anaheim, CA
ISBN: 978-1-4503-ABCD-E/16/07
DOI: http://doi.acm.org/10.1145/9999997.9999999

standardized construction system.

This paper aims to make the above-mentioned qualities of materials in native forms more accessible for end users by leveraging digital technologies. We use locally obtained branches which can be found almost everywhere not only in countryside but also in urban environment such as parks and along streets. Public service takes care of these branches: annual pruning, storing, and chipping or burning with some costs. The size of branches (from 50 -300 mm in diameter) is too small for furniture or other structural applications as building components. It is a challenge for digital design and fabrication to utilize the diverse branches in meaningful ways. High-precision but low-cost scanning devices and personal digital fabrication machines make it possible to analyze and control natural materials in diverse native forms.

The key technical difficulty of materials with natives forms is design. A possible approach is optimization: minimize an energy function which integrates structural and fabrication costs. This approach, however, is limited to particular design scenario with specific materials. Furthermore, the concept of optimum solution is well suited to goals such as efficiency or low-cost, but these goals are not the qualities materials in natives forms can compete with standardized construction systems. Instead, we take humans in our scan-design-fabricate workflow not only to solve the design problem, but also to provide an opportunity for people to participate in the workflow. Traditionally, in case of constructing public and symbolic buildings, such as church, people in a community took initiatives from fund raising to design, or even in construction process.

In this paper, we report our case study to design and fabricate an architectural element out of irregularly shaped branches, using our humans-in-the-loop system. We developed an online platform where users can post branches found at hand, and design with them through the online game *BranchConnect*. The game system itself helps users to design feasible branch layouts and enable them to fabricate customized joints to connect them together without screws and adhesives. The design of our joint extends the traditional orthogonal lap-joints to various angles within a range, freeing the diverse forms of woods from orthogonal connections. Physically collected branches are digitally scanned and stored in a cloud database *BranchCollect*, and offline application *Branch Importer* analyzes forms of branches and upload them to the database. The simple visual feedback and scoring system of the game guide users to feasible solutions, which are further inspected by an offline application *G-Code*[1] *Generator* for CNC milling process. The game system and developed import/export applications are currently limited to branches, however, the principle of human-in-the-loop with design is applicable for other kinds of materials with diverse irregular forms. We hope our method sheds lights on materials such as waste from demolition of buildings for various design applications.

In summary, our contributions are

- a workflow enabling to take natural materials with native forms as design components.
- an online game-based approach to participatory architectural design and fabrication.
- a method to design and fabricate customized non-orthogonal joints using high-resolution contours.

[1] G-Code is the generic name for a control language for CNC machines.

2 Related Work

3D printers and CNC routers made digital fabrication more accessible, and pre-fabricated customized building components are often used in buildings nowadays [Knaack et al. 2012]. According to the theory by Pye [1968], these components are processed from highly standardized material, thus its digital fabrication process is "workmanship with certainty"; a batch process of reading G-Code and strict execution of the code. On the other hand, as "workmanship with risks" in digital fabrication, interactive fabrication enables machines to pick up uncertain happenings and react on it [Willis et al. 2011]. Mueller and her colleagues developed interactive laser cutting, taking user inputs and recognizing placed objects in a fabrication scene [Mueller et al. 2012]. While their system interprets objects as simple platonic geometry, our work interacts with the native forms of irregularly shaped branches. Crowdsourced Fabrication project took advantage of humans-in-the-loop in their fabrication system [Lafreniere et al. 2016]. On the other hand, our work puts emphasis on crowd-sourced design as a socially networked fabrication. As a crowdsourced design system, Talton et. al. developed a platform for light users to design trees and plants [Talton et al. 2009]. Our work also developed online collaborative design platform but directly linked to the real-world.

There are few works that take natural materials in native forms as design components. Schindler and his colleagues used digitally scanned wood branches and used them for furniture and interior design elements [Schindler et al. 2014]. Monier and colleagues virtually generated irregularly shaped branch-like components and explored designs of large scale structure [Monier et al. 2013]. Using larger shaped forked tree trunks, *Wood Barn* project designed and fabricated custom joints to construct a truss-like structure [Mairs 2016]. *Smart Scrap* project digitally measured lime stone leftover slates from a quarry and digitally generated assembly pattern of slates [Greenberg et al. 2010]. In industry, recognition of irregularly shaped objects is essential for waste management. *ZenRobotics* developed a system that sorts construction and demolition waste by picking objects on a conveyor belt using robotic hands [Lukka et al. 2014]. For factory automation purpose, there is a system that recognizes irregularly shaped objects and sort them into a container [Sujan et al. 2000]. While these projects demonstrated the capability of digital fabrication processes to handle irregularly shaped materials, design process with native natural materials is still dependent on experts.

Cimerman discussed architectural design practices that took computer-mediated participatory (architectural) design [Cimerman 2000]. He mentioned three motivations of digital participatory design: 1. including stakeholders in creation of one's environment. 2. experimenting diverse design tastes from multiple point of views. 3. solving complex design tasks with full of diverse solutions.

Database of available local materials allows people with various backgrounds to involve in design process, which could lower the design cost with natural materials in native forms. We took inspiration from existing gamification systems. For example, *NanoDoc* took gamification approach to search valid nano-particle designs against tumors out of infinite design space [Hauert et al.]. *DrawAFriend* has developed an online game to collect big-data for drawing applications which assist humans with auto-stroke assistance [Limpaecher et al. 2013]. While these works developed games for collecting valuable data for solving medical or engineering problems, our game is served as a collaborative design platform, aiming to solve the socio-cultural issues in modern buildings such as generic design and detached context.

3 Workflow

As shown in Figure 1 left, our workflow starts from physically collecting branches. The collected branches are uploaded to cloud database by *Branch Importer*, and served to the online game-based design application *BranchConnect*. The game system uses skeletons for its joint detection process, which works on browsers on laptop computers or mobile touch devices. As shown in Figure 1 right, users can explore a global design with multiple branch layouts. Once the global design is fixed, designed layouts are further inspected by *G-Code Generator*, which generates customized joints for CNC milling. After finishing the milling process, users physically assemble branches and complete the fabrication process. The pipeline of the workflow is illustrated in the Figure 2. In this section, we introduce two steps in the pipeline: Digital Model Acquisition and Fabrication. As for the game system, please refer Section 4.

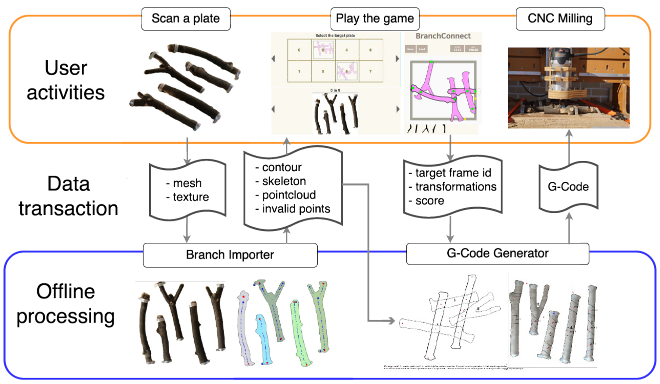

Figure 2: *A pipeline from model acquisition to fabrication.*

3.1 Digital Model Acquisition

Our system takes textured mesh models as input. There are various methods and software available for scanning 3D models. We describe the scanning setup in Section 5. Taking mesh model with colored texture, our *Branch Importer* provides functions such as object detection, skeleton extraction, branch type classification, and fixture point setting. The scanned result is a mesh model representing branches with a base plate. The system first idendifies branches by applying simple height threshold, and then applies contour detection. The obtained 2D contours are used for extracting skeletons and clustering verticies in the mesh model. Contours are triangulated and skeleton points are extracted from middle points on edges of triangles. These middle points are compared with top view image. If the point is inside of a contour, the middle point is counted as a skeleton point. After extracting skeleton points, the connectivity of skeletons is analyzed. In case grafting branch (Y-shaped branch) is detected, a new skeleton sub-branch is added. The result is shown in Figure 3. Metal fixture locations are confirmed by simple mouse-clicks and marked as invalid, meaning that joints should not be placed on these points. The acquired information is stored in a cloud database.

3.2 Fabrication

After a design is selected for fabrication, the fabricatability of the design is further inspected by a high-resolution model. The *G-Code Generator* displays joints and milling paths on scanned orientations. If it identifies an invalid joint in the high resolution model, a layout can be easily modified with simple mouse inputs

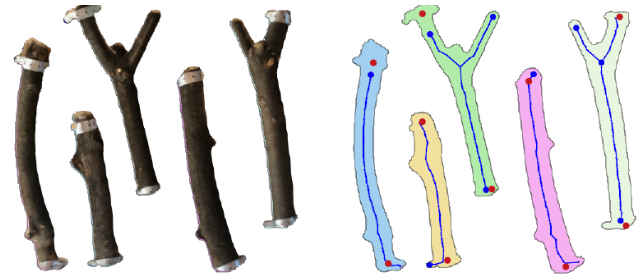

Figure 3: *An interface of Branch Importer. Left: a top orthoview image of textured mesh model. Right: Extracted skeletons are shown with blue dots. The beginning of skeletons is shown bigger dots, and the red dots are invalid points defined by a user.*

(Figure 12.1 and 12.2). Users can also change milling parameters such as offset ratio of milling paths, milling bit diameter, depth of joints, cutting speed, moving height and so forth. After confirming the fabrication settings and milling paths, it generates G-Code.

Some fabrication factors such as invalid points due to metal fixtures and flipped (further described in Section 4.1.1) are already considered by *Branch Importer* and the game system respectively. Here, we describe the process to calculate joint geometry and for fabrication. The *G-Code Generator* searches a set of four closest points from high-resolution contours (Figure 12.1). Trimming contours of eash branch at the corner points, we get *side cuts* and *center cuts* (Figure 12.4). *Side cuts* have wedged corners for smooth assembly process. The depth of the *center cuts* is half of the top height at the joint position from a mesh model (Figure 12.4). The bottom height is usually the height of base plate, however, in case of under-cuts with incomplete mesh model, we calculate a half of the diameter from the 2D contour and subtract it from the top height. The resulting geometry creates rigid joints with irregularly shaped sections of branches.

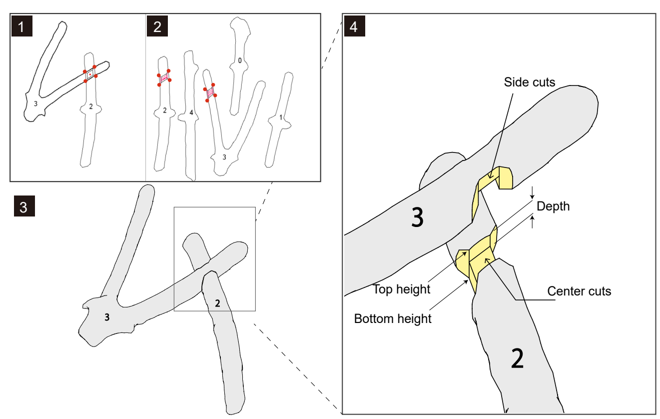

Figure 4: *1, 2: an interface of G-Code Generator. 1. a layout defined by a user. 2. the original orientations of branches with generated milling paths with red color. 3. an assembled pair of branches with branch 3 is flipped. 4. milled branches with center cuts and side cuts.*

4 BranchConnect: The Game

The online game is accessible by laptops and mobile touch devices, and many users can play at the same time. The objective of the game is to collect valid layouts of branches which are fabricatable with 3 axis CNC milling machines. By analyzing the connectivity of branches and target points, the game informs the feasibility of a given layout. Similar to our game, the work *guidance system during furniture design* inspected connectivity, durability, and stability [Umetani et al. 2012]. Unlike their work, our game puts emphasis on *fabricatability*, as well as *geometric connectivity*, and does not calculate structural performance of each joint. Instead we use simple geometric analysis to compute validity. We also assume that every fabricated joint works as a rigid joint, thus single connection is counted as stable to hold a pair of branches.

The overall user experience is as follows. In the selection interface (Figure 5 left), each frame comes with a set of predefined target points to be connected. The distribution of these target points is predefined by the system, and end users can not modify them. After a user selects a target frame and a set of branches on a plate, the user is guided to the game interface (Figure 5 right), consisting of the frame with the target points, and the set of available branches at the bottom. The user picks a branch from the available set on the bottom, and drag&drop it to the inside of the target frame. By selecting and dragging these dropped branches, the user searches a good 2D pose through basic direct geometric manipulations such as move, rotate, and horizontal flip (or mirror). While the manipulation, the user receives simple feedback with colors and score. Within the limited number of available branches, the user needs to bridge all the target points by connecting all the dropped branches in one group. The game is completed when all the target points are connected. To achieve higher score, the user can keep modifying the design, and save it to the database.

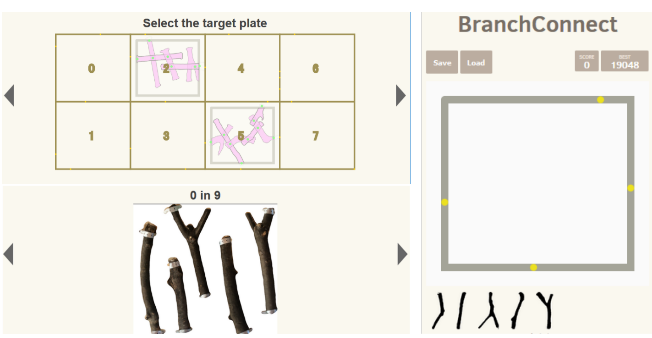

Figure 5: *Left: the selection interface for target frames (top) and branch panels (bottom). Right: the start interface of the game.*

4.1 The Game System

There are many physics simulation libraries for game, however, our game needs to detect intersected branch pairs, thus collision detection with physics engines is overkill for our browser game. Also, branches have free-form concave shapes, thus further geometric preparation such as convex decomposition is necessary for using these libraries. For fast and robust intersection detection, our game extensively uses down-sampled skeletons of branches.

Hubbard and Philip developed collision detection by representing an object with hierarchical 3D spheres aligned on a skeleton [Hubbard 1996]. Our game takes similar approach but limited in 2D, but more focused on searching fabricatable joints. In the game, down-sampled skeletons are used to find the pair of closest skeleton points between two branches. When a branch is selected, the system searches the closest skeleton point of the selected branch with other skeletons of available branches. More precise joint calculation with high-resolution contours is further described in Section 3.2.

Here, we introduce two important entities in the game: joint and group. A joint is created when an intersecting pair is detected, and the pair forms a group. The group is used for evaluating connections between target points. The conditions of joint and group are indicated with simple color-code. Once the user finishes geometric manipulation, score is updated with weighted sum of score parameters. Together with the color-code, the score update guides the user to form a feasible design.

4.1.1 Joint Condition

Joint is the essential entity not only in the game but also in the fabrication process of customized lapped joints. Importantly, each pair of branches must have one flipped branch for fabrication constraint (Section 3.2). Figure 6 illustrates valid and invalid joint conditions. Our joint takes only crossed pair because they are structurally stable, relatively simple to fabricate, and creates diverse designs (Figure 6.1). Due to fabrication process with CNC milling, we do not take conditions such as terminal connection, joint at metal fixture, and T-shaped connection (Figure 6.3, 6.4, and 6.5 respectively) . A valid joint's angle stays within a fixed range (Figure 6.1 and 2). Valid and invalid joints are displayed with green and red respectively.

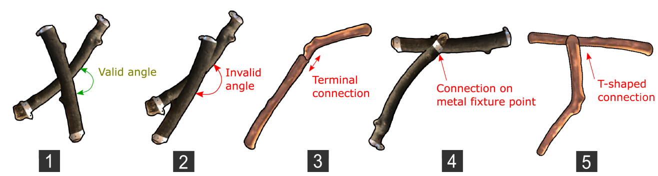

Figure 6: *Joint conditions. 1. valid joint. 2. invalid for violating the angle. 3. invalid terminal connection. 4. invalid for connecting on a metal fixture point. 5. invalid for T-shaped connection.*

To describe the joint update process, let each branch b_i be a member of a set of the branches $\mathcal{B}$ dropped inside of the target frame by a user. The user freely choose $\mathcal{B} \in \mathcal{B}_{\text{plate}}$, where $\mathcal{B}_{\text{plate}}$ is the branches fixed on a selected plate, denoted as $\mathcal{B}_{\text{plate}}$. Note that we accept one joint with a pair of branches, but a branch can have multiple joints with other branches. The process starts from the selected branch b_i and updates joint conditions of the selected branch with paired branches. When an intersected pair is detected, it stores j-th joint $j_{i,j}$ in b_i with joint conditions (Figure 6). After evaluation, as in Figure 6, we have joints labeled as valid or invalid. When a branch b_i is connected to one of target points $t_j \in \mathcal{T}$, the target point $t_{i,j}$ is stored in b_i. Note that we also take one target point per branch. The branch connected to the target point is trimmed at the target point. The trimmed length $l_{\text{trim},i}$ is stored in b_i and use as a penalty for score calculation. In this way, the user tries to position the branch as inside of the frame as possible. This process is iteratively executed while a user is positioning a branch with mouse-drag.

4.1.2 Group Condition

After a user finishes positioning a branch, the system evaluates group conditions. Firstly it updates groups, and then detects invalid group conditions, as well as connections to target points. If a branch is connected to a target point, colors of the branch and its belonging group are updated, guiding users the validity of their layouts. When a group bridges a pair of target points, a special score is added and displayed with an animation.

The group update process is described as follows. Each time the process is called, firstly it initializes a set of groups denoted as $\mathcal{G}$.

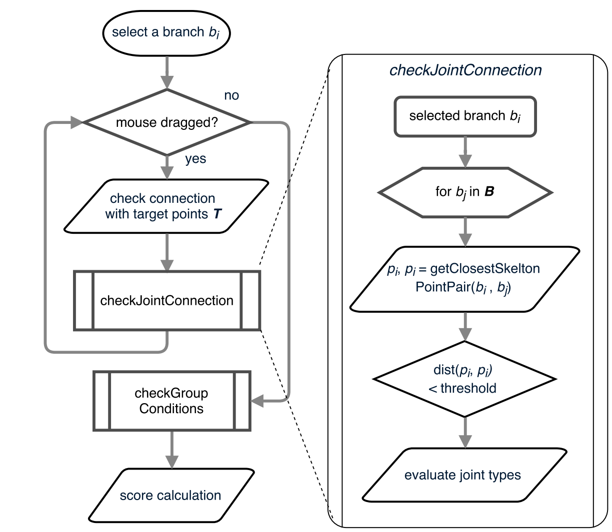

Figure 7: *Left: an overview of the game system with 1. joint update 2. group update and 3. score calculation. This process is iteratively executed while a user is exploring layout by dragging a branch. The joint update process is further illustrated in the right, and group condition update is described in Algorithm 1.*

Iterating $b_i \in \mathcal{B}$, the first group $g_0 \in \mathcal{G}$ is created and g_0 stores b_0. In the iteration, the process checks connections of a branch b_i with all the existing groups $g_j \in \mathcal{G}$. If b_i is connected to g_j, g_j is stored in a list of connected groups, denoted as C_i. After collecting all the connected groups with b_i, we evaluates C_i. If C_i does not have any stored group, then a new group g storing b_i is created and added to $\mathcal{G}$. If C_i is not empty, then it also creates a new group g, but puts all the groups $g_j \in C_i$ to g, then adds g to $\mathcal{G}$. Finally groups $g_j \in C_i$ are removed from $\mathcal{G}$. Iterating all $b_i \in \mathcal{B}$, we acquire the updated group condition.

After updating the group, the process evaluates the connections with target points. If $b_i \in g_k \in \mathcal{G}$ has connection with a target point, the target point is stored in the group g_k. If multiple groups with target points for each exist, the entire structure is feasible, however, these groups should be connected together (Figure 8 middle). If the group does not have any target point, the group is labeled as *Islanded*, which is structurally infeasible thus counted as a penalty in the cost calculation (Figure 8 right). If the group has multiple target points, these target points are bridged. The game completes when the number of $\mathcal{G}$ is one, and all the target points are connected with the group without invalid joints. The group update process is described in Algorithm 1. The entire flowchart of the joint update process as well as the group update process are illustrated in Figure 7.

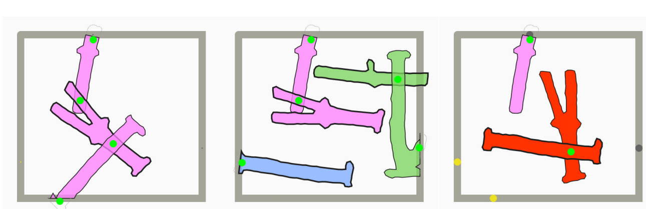

Figure 8: *Left: valid group with two target points connected. Middle: valid but three groups. Right: invalid due to the Islanded situation.*

Algorithm 1 Group Condition Update Algorithm

function UPDATEGROUPS($\mathcal{B}$)
  Reset all the groups $\mathcal{G}$
  Create new group g_0
  g_0.adds (b_0)
  if b_0.contains$(t_i \in \mathcal{T})$ **then**
    g_0.adds(t_i)
  $\mathcal{G}$.adds(g_0)
  for each branch $b_i \in \mathcal{B}$ except b_0 **do**
    initiate a list of connected group C_i
    for each group $g_j \in \mathcal{G}$ **do**
      if g_j.contains(b_i) **then**
        G'_i.add(g_j)
    if C_i.isEmpty **then**
      Create new group g
      g.adds(b_i)
      $\mathcal{G}$.adds(g)
    else
      Create new group g'
      g'.adds(b_i)
      for each group $g_j \in G'_i$ **do**
        for each branch $b_k \in g_j$ **do**
          if g'.contains$(b_k) = false$ **then**
            g'.adds(b_k)
      for each group $g_j \in C_i$ **do**
        $\mathcal{B}$.removes(g_j)
      $\mathcal{G}$.adds(g)

4.1.3 Score Calculation

We calculate the score with weighted sum of following entities: the numbers of valid and invalid joints on each branch, the number of groups as $N(\mathcal{G})$, the number of islanded groups as $N(g_{\text{islanded}} \in \mathcal{G})$, the number of bridged target points as $N(t_{\text{bridged},i}) \in \mathcal{T})$. The trimmed lengths of branches which are connected with target points are denoted as $trimmed(t_j, b_i)$. The score is weighted sum of these joint and group conditions, denoted in Equation (1). The weights $w_1 \dots w_5$ are non-negative weight coefficients pre-adjusted in advance by authors.

$$
\begin{aligned}
Score = {} & w_1 \sum_{1}^{N(\mathcal{B})} \sum_{1}^{N(\mathcal{J}_{\text{valid},i})} j_{\text{valid},j,i} + w_2 \sum_{1}^{N(\mathcal{B})} \sum_{1}^{N(\mathcal{J}_{\text{invalid},i})} j_{\text{invalid},j,i} \\
& + w_3 \sum_{1}^{N(\mathcal{G})} g_{\text{islanded}} + w_4 \sum_{1}^{N(\mathcal{T})} t_{\text{bridged}} + w_5 \sum_{1}^{N(\mathcal{T})} trimmed(t_j, b_i) \\
& \text{s.t. } w_j \geq 0 \; \forall j \in 1, \dots, 5
\end{aligned} \tag{1}
$$

5 Case Study

A design and fabrication workshop was organized to examine the feasibility of our system with a specific design target and a location. We selected a public community house where people in the community share the space and regularly use the facility. Participants of the workshop were selected among them, who were four children (aged 4, 7, 9, and 10) and two parents (Figure 9). We specifically selected children with this range of age as non-experts without experiences in computational design or digital fabrication, also for observing the clarity and attractiveness of the game.

The entire workshop was filmed and summarized in the supplementary video material. Prior to the workshop, we have adjusted the weights in Equation 1 manually to ensure that the feasibility of layouts is correlating to the scores (Figure 11). Through adjusting the game setting, we have built six target frames.

The goal for the participants was to contribute to an ongoing design and fabrication process of screen wall (2000 $mm \times$ 900 mm) consisting of eight rectangles (500 $mm \times$ 450 mm). Six frames were already designed and built, thus the rest two frames were prioritized for them to design and fabricate in this workshop.

Participants were informed about the goal of the workshop, and each process was introduced by experienced tutors. The processes were from collecting and fixing the branches on a plate, scanning the plate, complete designs by playing the game, and assembly after CNC milling.

Figure 9: *An overview of the workshop. 1. the overview of the space. 2. collect branches. 3. cut in certain lengths 4. attach on a plate 5. scan the plate 6. play the game 7. CNC milling.*

5.1 Preparations and User Experiences

System and Hardware

We used two iPad minis with iSense depth cameras attached for scanning branches, and a 3 axis CNC milling router with a 6 mm diameter milling bit. We used a laptop PC for running *Branch Importer* and *G-Code generator*, as well as operating the milling machine. The scan area of iSense camera is 500 $mm \times$ 500 mm, and the milling machine's stroke length along z-axis is 70 mm, which provide geometric constraints for available branch sizes. *BranchConnect* was hosted at *Heroku* cloud server [2], and we used *MongoDB* [3] as a cloud database.

Preparations

The participants were asked to collect branches with 20 - 100 mm in diameter. The lower bound was for the milling bit size, and the upper bound was for the limited length of z-stroke of the CNC router. The collected branches were cut in arbitrary lengths, not longer than 500 mm due to the limit of scanning area. As our game system and fabrication process take 3D branch shapes as 2D contours (with limited use of point cloud), these constraints were benefitial for the system by filtering out branches with large 3D twists. The diameter and length constraints worked as guidelines for participants rather than restricting finding and cutting arbitrary branches. The number of available branches per plate was different depending on branch sizes. Within the feasible diameter of branches and the plate size, the number of available branches was most up to six (Figure 10).

After cutting branches in certain lengths, participants fixed branches on plates by thin metal fixtures with screw holes. After an instruction, participants successfully fixed branches by themselves. They built two plates with three and five branches fixed on each plate.

After fixing branches on plates, we asked participants to scan them prepare feasible mesh model on iSence application running on iPad mini. Thanks to the intuitive interface of iSense, participants practiced several scans and successfully scanned models without problem. After obtaining mesh models, tutors imported models from iPads to a laptop and uploaded them to the database by *Branch Importer*.

User Experiences

As for more general user experiences with the game, Section 4 as well as the video material. In this section, we describe more specific user experiences and feedback from participants.

All participants used iPads for navigating pages and playing the game. They had difficulties with mobile touch interface, such as rotation and flipping operation by gestures. We had several requests from participants regarding the game interface but also related to the workshop organization. Several participants requested to allow multiple branch plates for designing a frame, or even remove the target frame and let them freely design with branches. Also a participant who gave up the game with iPad requested additional buttons for mobile touch interface, such as to keep an active branch selected. The participant with four years old failed to complete the game. He insisted on accepting his design to be fabricated Similarly, a branch plate made by a participant had only three branches, which was not enough to fulfill bridging target points, however, the participant insisted on accepting it in the selection.

Global Design Consensus and Fabrication

As the target frame selection page could display all layout designs, we could get an overview of design options. The layout designs were displayed as score descending order with limited numbers (three top highest scores for each target frame), we could find feasible layout designs easily with mostly all the target points were bridged. As participants were excited by seeing their branches and designs, we took two invalid layout designs and one plate which did not have enough branches for the global design. After selecting layouts, an experienced tutor operates *G-Code Generator* as well as the CNC router. Participants were asked to assembly branches after joineries were milled.

5.2 Results

The entire workshop took 4.6 hours to complete the whole process, including introduction, moving, and pauses. Table below shows durations of each task.

Task	Duration (hour)	Fraction (%)
Introduction	0.3	6.5
Collecting branches	0.6	13.0
Preparing plates	0.8	17.4
Preparing models	0.3	6.5
Uploading models	0.2	4.3
Designing by the game	0.5	10.8
Inspecting models	0.2	4.3
CNC milling	0.5	10.8
Assembling	0.2	4.3
Moving, pauses	1.0	21.7
In total	4.6	100

[2]Heroku is a platform as a service (PaaS) that enables developers to build, run, and operate applications entirely in the cloud. https://www.heroku.com/

[3]MongoDB is a free and open-source cross-platform document-oriented database program. https://www.mongodb.com/

Model Acquisition

Each scanning and re-touching took 2-3 minutes, and 30 seconds for generating data by *Branch Importer*. Including the prepared panels previously, we scanned 15 plates in total, 75 branches, and 35.3m of total length including sub-branches. We got 59 branches with a single skeleton, 16 branches with multiple skeletons for grafting. The result is shown in Figure 10.

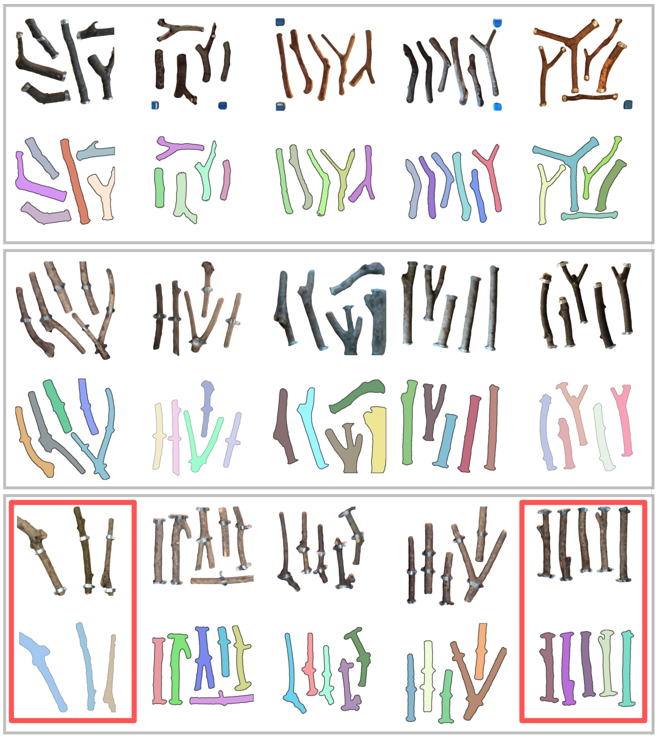

Figure 10: *An overview of all the 15 scanned plates for the construction of the fence. Top raw of each set shows ortho-top views of scanned mesh models, and the bottom raw is the recognized branches with randomly assigned colors. The red-lined rectangles indicate the plates built by participants in the workshop.*

Design with the Game

Figure 11 shows 32 example layouts with scores. We set the weights in Equation 1 as shown in Table below.

weigths	w_1	w_2	w_3	w_4	w_5
our setting	100	-100	-1000	500	-5

One participant switched to play by a PC for more precise control due to the problem. Interestingly, all the participants chose to develop their own designs from the scratch, although they had instruction about the ”continue existing designs”.

We set 30 minutes for playing the game, and eight layout designs were given by participants. Two frames were completed per participants and two participants completed the whole eight target frames. The average score was xxx, and average playing duration was xxx to complete each target frame. TODO: recheck the numbers by server

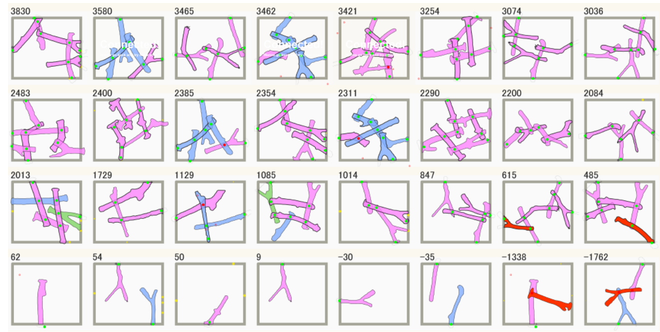

Figure 11: *Example layouts by authors in score-descending order. The highest score is top-left and the lowest score is bottom-right.*

Fabrication

We observed most of scanned models had occluded regions between plates and branches, which create interpolated faces during solidifying process, resulting in outwardly offsetted contours. TODO: check offsetted correct english After milling was finished and when branches were assembled, six pairs of branches were loosely connected because the calculated contours were 2-3 mm eroded than the actual sizes. We avoided this problem by trimming branches from 2-5 mm higher than the plate surface. After this operation, the rest of connections were tightly connected.

We also observed that many milling paths were 5-10 mm off from the center of planned joints. Multiple reasons could be considered as reasons such as,

- deformation of mechanical parts of the CNC router
- not dense resolution of acquired contours of branches
- misaligned orientation of the plate compared to the scanned model

To avoid the misalignments, we modified the *G-Code Generator* so that an operator can freely adjust the absolute origin of the generated milling paths. The origin was usually set with around the center of the plate. After this modification, the misalignment from joint center was reduced with 5 mm off at the maximum misalignment. Branches could absorb 3-5 mm misaligned joint positions due to the elasticity of branches, and solidifying the structure with residual stresses from misalignments.

6 Conclusion

In this paper, we presented a workflow to design and fabricate with branches with their native forms, which are not large enough for producing standardized building components. Our workflow was validated by the case study with lower-aged participants without design and fabrication experiences.

Our online platform with stored scanned branches is accessible and multiple users can submit design layouts and explore a global design. Our branch joint detection and group condition update algorithms are running on the browser game which can be accessed from laptops and mobile devices, contributing to the accessibility of the presented workflow. Together with the accessibility, the intuitive interface was simple for non-expert users, validated by the case study. We successfully built a network of branches with rigid joints generated by our joinery milling path generator. Each of joints has customized lapped-joint geometry, which extends design possibilities of branches or woods with their native forms.

Our workflow touches many developed research areas such as skeleton extraction, structural optimization, object detection/recognition, and data-driven design-fabrication. Focusing on the use of native forms, each step of our workflow has potential to contribute to each area with the use of native forms of natural materials. Also, our workflow was developed based on the participation of users, thus the entire process is not necessarily automated, however, some tasks could be improved to assist users.

The skeleton extraction could take incomplete point-set directly from original tree branches before they are cut in length. With data-driven approach, the system could distinguish trees and which part of tree the branch from. With morphological analysis, the system could suggest users where to cut branches to achieve user-defined target design. Structural and geometrical validity/invalidity of obtained materials could be analyzed. Our workflow requires branches to be fixed on a plate, which takes the longest duration in the workflow except for in-between tasks such as moving and pausing. Using a robotic manipulator with a gripper, the attaching process could be skipped.

Our game system is limited in 2D, whereas original branch forms have rich 3D geometry with textures. In our case, these information was used in limited ways such as in skeleton extractions and G-Code generation. Despite of successfully fabricated non-orthogonal joineries, we did not complete the attaching branches to the target frames, as we prioritized to validate branch-branch joineries. Our layout design process is fully dependent on users with limited feedback during design process. The game can provide suggestive feedback with structural analysis of each joint and entire structure.

Our joint and group detection algorithms are limited with materials with skeletons, and our joinery generator is limited to branches. Both steps use down-sampled or high-resolution point sets. It is valuable to validate the approach by comparing with other available methods such as collision detections or joint detection with down-sampled model by interpolation.

Finally, our game-based design could be applied to different purposes, not only for participatory layout design but also for collecting data of user behaviors during design. Also application to other kinds of materials could be investigated.

References

CIMERMAN, B. 2000. Participatory design in architecture: can computers help? In *PDC*, 40–48.

GREENBERG, B., HITTLER, G., AND PERRY, K., 2010. Smart scrap. Accessed: 2016-09-30.

HAUERT, S., LO, J. H., NACHUM, O., WARREN, A. D., AND BHATIA, S. N. Crowdsourcing swarm control of nanobots for cancer applications.

HUBBARD, P. M. 1996. Approximating polyhedra with spheres for time-critical collision detection. *ACM Trans. Graph. 15*, 3 (July), 179–210.

KHOSHNEVIS, B. 2004. Automated construction by contour craftingrelated robotics and information technologies. *Automation in construction 13*, 1, 5–19.

KNAACK, U., CHUNG-KLATTE, S., AND HASSELBACH, R. 2012. *Prefabricated systems: Principles of construction*. Walter de Gruyter.

LAFRENIERE, B., GROSSMAN, T., ANDERSON, F., MATEJKA, J., KERRICK, H., NAGY, D., VASEY, L., ATHERTON, E., BEIRNE, N., COELHO, M. H., ET AL. 2016. Crowdsourced fabrication. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, ACM, 15–28.

LIMPAECHER, A., FELTMAN, N., TREUILLE, A., AND COHEN, M. 2013. Real-time drawing assistance through crowdsourcing. *ACM Transactions on Graphics (TOG) 32*, 4, 54.

LUKKA, T. J., TOSSAVAINEN, T., KUJALA, J. V., AND RAIKO, T. 2014. Zenrobotics recycler–robotic sorting using machine learning. In *Proceedings of the International Conference on Sensor-Based Sorting (SBS)*.

MAIRS, J., 2016. AA design and make students use a robotic arm to build a woodland barn. Accessed: 2016-09-30.

MONIER, V., BIGNON, J. C., AND DUCHANOIS, G. 2013. Use of irregular wood components to design non-standard structures. In *Advanced Materials Research*, vol. 671, Trans Tech Publ, 2337–2343.

MUELLER, S., LOPES, P., AND BAUDISCH, P. 2012. Interactive construction: interactive fabrication of functional mechanical devices. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, ACM, New York, NY, USA, UIST '12, 599–606.

OLIVER, P. 1997. *Encyclopedia of vernacular architecture of the world*. Cambridge University Press.

PYE, D. 1968. *The nature and art of workmanship*. Cambridge UP.

SCHINDLER, C., TAMKE, M., TABATABAI, A., BEREUTER, M., AND YOSHIDA, H. 2014. Processing branches: Reactivating the performativity of natural wooden form with contemporary information technology. *International Journal of Architectural Computing 12*, 2, 101–115.

SUJAN, V., DUBOWSKY, S., OHKAMI, Y., ET AL. 2000. Design and implementation of a robot assisted crucible charging system. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2, IEEE, 1969–1975.

TALTON, J. O., GIBSON, D., YANG, L., HANRAHAN, P., AND KOLTUN, V. 2009. Exploratory modeling with collaborative design spaces. *ACM Transactions on Graphics-TOG 28*, 5, 167.

UMETANI, N., IGARASHI, T., AND MITRA, N. J. 2012. Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph. 31*, 4, 86–1.

WESTON, R. 2003. *Materials, form and architecture*. Yale University Press.

WILLIS, K. D., XU, C., WU, K.-J., LEVIN, G., AND GROSS, M. D. 2011. Interactive fabrication: new interfaces for digital fabrication. In *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction*, ACM, 69–72.

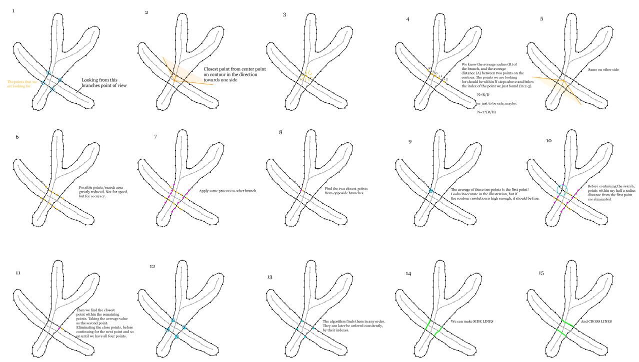

Figure 12: *An overview of the intersection search in G-Code generator.*