Stair Design Using Quantified Smoothness

Abel Groenewolt
abel@groenewolt.nl

Abstract. This paper introduces metrics to evaluate stair geometry and shows how these metrics can be used to develop versatile computational stair design tools for the design of smooth stairs. The proposed stair smoothness metrics are based on the angles between tread lines, the angles between the walk line and tread lines, and the dimensions of tread sides. Using these metrics in combination with evolutionary algorithms results in computational methods that are highly flexible: as opposed to common software tools that generate particular classes of stairs (such as helical stairs or u-shaped stairs), this approach could be used for any stair design. The proposed methods produce results that match or surpass the smoothness of manually designed stairs and enable the implementation of features that are not available in other design tools, such as obstacle avoidance. Applications of the proposed method are shown for both freestanding stairs and stairs with a predefined footprint.

Keywords: Stairs, Stair Design, Evolutionary Algorithms, Computational Design

1 Introduction

1.1 Stair Balancing

A major design consideration in stair design is how to deal with turns in direction. For walking comfort and safety as well as for aesthetic reasons, gradual transitions are typically desired [1], [2]. Stairs in which treads are locally adjusted to create smoother transitions are called balanced stairs or dancing stairs [3].

The process of balancing stairs involves rotating treads horizontally around the point where they intersect the walk line. Various geometric methods to execute this process manually exist [2], [4-6], of which the unrolled projection method is the most widely applicable, as it can deal with any angle between straight series of treads. This method (illustrated in Fig. 1) has been known for over 200 years [7] and can be found in various sources [6], including recent publications [4], [8].
Fig. 1. Conventional balancing method using unrolled projection. From left to right: unbalanced stair; unrolled projection of inner side of stair, with construction lines needed to create an arc; balanced unrolled projection, created by shifting risers to the arc; balanced stair. The modified tread lines in the balanced stair are constructed by first transferring the tread dimensions on the inner side of the stair from the projection, then connecting these points to the original treads’ intersection points with the walk line.

1.2 Stair Design Tools

In current software implementations, tools for the creation of various stair types (such as straight stairs, helical stairs and winding stairs) are typically implemented. However, although stair design software has been around for over 30 years [9], the choice of balancing methods tends to be limited (if present at all). Therefore, using manual methods may still lead to better results than what can be achieved using software tools. This is remarkable, as due to the high constraints that are placed on their geometry, stair design should be well suited to a computational approach [10-12].

2 Proposition

Balancing improves stairs in terms of appearance, walking comfort and safety, by making sure no sudden changes occur between consecutive treads. A balanced stair has more gradual changes between treads and exhibits a smoother appearance, as illustrated in Fig. 2.

Stair smoothness can be evaluated visually, but as it is based on geometry, it should be possible to evaluate smoothness numerically. Stair geometry can be generated programmatically, so as soon as design alternatives can be evaluated quantitatively, it becomes possible to develop evolutionary methods for stair design. As such methods do not depend on pre-existing conceptions about particular stair categories, they should be more flexible and more generally applicable than existing design methods. Because the result of such an evolutionary process will be a stair with high smoothness, the outcome should be a stair that is aesthetically pleasing, as well as safe and comfortable to walk on.
Fig. 1. Conventional balancing method using unrolled projection. From left to right: unbalanced stair; unrolled projection of inner side of stair, with construction lines needed to create an arc; balanced unrolled projection, created by shifting risers to the arc; balanced stair. The modified tread lines in the balanced stair are constructed by first transferring the tread dimensions on the inner side of the stair from the projection, then connecting these points to the original treads' intersection points with the walk line.

1.2 Stair Design Tools

In current software implementations, tools for the creation of various stair types (such as straight stairs, helical stairs and winding stairs) are typically implemented. However, although stair design software has been around for over 30 years [9], the choice of balancing methods tend to be limited (if present at all). Therefore, using manual methods may still lead to better results than what can be achieved using software tools. This is remarkable, as due to the high constraints that are placed on their geometry, stair design should be well suited to a computational approach [10-12].

2 Proposition

Balancing improves stairs in terms of appearance, walking comfort and safety, by making sure no sudden changes occur between consecutive treads. A balanced stair has more gradual changes between treads and exhibits a smoother appearance, as illustrated in Fig. 2.

Stair smoothness can be evaluated visually, but as it is based on geometry, it should be possible to evaluate smoothness numerically. Stair geometry can be generated programmatically, so as soon as design alternatives can be evaluated quantitatively, it becomes possible to develop evolutionarily methods for stair design. As such methods do not depend on pre-existing conceptions about particular stair categories, they should be more flexible and more generally applicable than existing design methods. Because the result of such an evolutionary process will be a stair with high smoothness, the outcome should be a stair that is aesthetically pleasing, as well as safe and comfortable to walk on.

Fig. 2. Comparison of a stair with abrupt changes from straight segments to winding segments (left) and a balanced stair created with a method described by Reitmayer [5] and Wattjes [6]. Note how the tread angles in the tread angle diagram for the balanced stair (right) describe a smooth curve, as do the tread depth values on the inside of the treads.

3 Principles

The proposed stair generation approach is based on the following premises:

1. The rise height of all treads of a stair should be constant.
2. The going distance on the walk line should be constant.
3. Consecutive treads should not exhibit large tread angle differences.
4. Consecutive treads should have similar side depths on at least one of their sides.
5. The lines defining the front of treads should be as close as possible to being perpendicular to the walk line.

The tread angle and side depth (as mentioned in premises 3 and 4) are illustrated in Fig. 3, with $\alpha$ denoting the tread angle and $d_1$ and $d_2$ denoting the tread width. These properties are geometrically related and for freestanding stairs, they could be used interchangeably. In winding stairs, side depth is the property that most directly affects the visual smoothness of the hand rail and the side of a stair. However, for walking comfort, the tread angle is the more directly relevant property, as it defines the change in tread orientation in relation to the walking direction.

Fig. 3. A tread in a freestanding stair (left) and a tread in a balanced stair (right). In both cases, the angle $\alpha$ is defined by the lines that define the front and the back of the tread. The tread depth we use in our algorithms is illustrated here, but depending on the applicable building code, different definitions may need to be used. We will refer to $d_1$ and $d_2$ as tread sides.
As the word suggests, balancing is a trade-off of properties: balancing methods typically decrease the differences in tread side depth at the treads' shortest side, but as a side effect, the tread lines are no longer perpendicular to the walk line. This trade-off needs to be reflected in the definition of a smoothness metric.

4 Abstraction

In the proposed computational model, a tread is principally represented by the angle between the lines that define the front and the back of a tread (excluding the tread nose). The position of a tread is either defined by a point on the walk line, or in case of a free-standing stair, the position simply follows from the position and geometry of the previous tread. The sides of a tread are either defined by the stair’s side lines, or in case of a free-standing stair, by arc segments. Although series of arc segments with non-constant radii have only G1 continuity, the resulting curves are virtually indistinguishable from curves with G2 continuity, as can be seen in Fig. 4.

Fig. 4. Three consecutive treads, constructed by defining point \( c_1 \) and radii \( r_1, r_2 \) and \( r_3 \). Point \( c_2 \) is constructed on line \( c_1-a \), using given distance \( r_2 \). As distances \( d_2 \) and \( d_3 \) are on the walk line, they should equal distance \( d_1 \) and thus angle \( a-c_2-b \) can be defined. Note that center points for two consecutive treads are always on the extension of the shared line between two treads; therefore, arc segments that form the sides of treads are tangent.

Stair objects are defined by a series of constant values (stair width, start point, start direction, rise height, going, and optionally a walk line and side lines) and a list of tread objects. Tread angles are the only variable that changes during the generative process. Thus, the only data to operate on is a single list of numbers.

Any number of constraints can be linked to a stair object, for example the desired end position and end direction, particular fixed points on the stair, or obstacles that need to be avoided.

5 Smoothness Metric

As discussed in Section 3, various properties are relevant for a good stair design. We will only be looking at stairs with constant riser height and constant tread depth on the walk line, so premise 1 and premise 2 will always be fulfilled. This leaves three premises; for each of these, we describe a matching metric that a stair could be
As the word suggests, balancing is a trade-off of properties: balancing methods typically decrease the differences in tread side depth at the treads’ shortest side, but as a side effect, the tread lines are no longer perpendicular to the walk line. This trade-off needs to be reflected in the definition of a smoothness metric.

### 4. Abstraction

In the proposed computational model, a tread is principally represented by the angle between the lines that define the front and the back of a tread (excluding the tread nose). The position of a tread is either defined by a point on the walk line, or in case of a free-standing stair, the position simply follows from the position and geometry of the previous tread. The sides of a tread are either defined by the stair’s side lines, or in case of a free-standing stair, by arc segments. Although series of arc segments with non-constant radii have only G1 continuity, the resulting curves are virtually indistinguishable from curves with G2 continuity, as can be seen in Fig. 4.

### 5. Smoothness Metric

As discussed in Section 3, various properties are relevant for a good stair design. We will only be looking at stairs with constant riser height and constant tread depth on the walk line, so premise 1 and premise 2 will always be fulfilled. This leaves three premises; for each of these, we describe a matching metric that a stair could be evaluated on. Depending on the type of stair, the spatial requirements and design intent, one or more of these metrics may be used.

#### 5.1 Tread Angle Metric

According to premise 3, consecutive tread angles (as illustrated in Fig. 3) should be similar and changes should be gradual. Two consecutive small changes in tread angle are preferable over a single large one. Therefore, we sum the squared differences between pairs of tread angles:

\[
A = \sum_{t=2}^{n} |\alpha_t - \alpha_{t-1}|^2
\]

(1)

*A* represents the tread angle metric, *n* the number of treads, *t* the tread number and *α* the tread angle for a particular tread.

#### 5.2 Tread Side Metric

According to premise 4, the dimensions of tread sides (as illustrated in Fig. 3) of consecutive treads should only change gradually. For the curvature of a hand rail, relative changes in tread dimensions appear to be more closely linked to visual smoothness than absolute changes. Therefore we look at the proportion between treads, rather than absolute lengths:

\[
D_s = \sum_{t=2}^{n} \max \left( \frac{d_{s,t}}{d_{s,t-1}}, \frac{d_{s,t-1}}{d_{s,t}} \right)^2
\]

(2)

*D*_s represents the tread side metric, *n* the number of treads, *t* the tread number and *d* the dimension of the tread side for a tread with number *t* or *t-1* on either the left or the right side (*s*) of the tread. The values are squared as it is preferable to have many small changes instead of a few larger ones.

#### 5.3 Tread Direction Metric

According to premise 5, the front line of a tread should ideally be perpendicular to the walk line. For this metric, we square the angles between the front line and ideal perpendicular line for each tread:

\[
W = \sum_{t=1}^{n} \left( \theta_{tl,wl} - 90^\circ \right)^2
\]

(3)

*W* represents the tread direction metric, *n* the number of treads, *tl* the front line of tread *t*, *wl* the walk line direction at tread *t* and *θ* the angle between a tread’s front line and the walk line.
5.4 Combined Metric

To be able to numerically evaluate stair designs, relevant metrics need to be combined in a single smoothness metric. In order to control the influence of each metric, weighting factors are introduced. These can be set to 0 in case a particular metric is not to be used.

\[ S = \left( a \cdot A + b \cdot W + c \cdot D_l + d \cdot D_r \right)^{-1} \]  

(4)

Here, \( S \) is the smoothness, \( a, b, c \) and \( d \) are weighting factors and \( A, W \) and \( D \) are the metrics described above. For \( D \), separate weighting factors can be set for the left and right side of the stair.

6 Algorithm

The algorithm we propose is an evolutionary algorithm. The process starts by generating a stair object with the required number of treads. If a walk line is provided, the tread positions will be defined on the walk line. The initial tread angles can be set to any value. As at this point there is only one stair object, it is marked as the best candidate. Geometry output for this stair object can be created using the stair object’s tread positions, angles and tread side lines, or if these are not given, by constructing treads one by one, as illustrated in Fig. 4.

![Fig. 4](image)

**Fig. 4.** Scheme of the proposed evolutionary process for free-standing stairs. In the evaluation steps, the tick marks mean that a goal has been met, while the plus sign indicates that even though the goal has not been met, the result is an improvement over the previously best candidate. A minus sign indicates that a candidate does less well than the best candidate and is to be discarded. The constraints evaluation step is optional and for winding stairs, the distance evaluation can be omitted.

Variations of the best candidate are generated by duplicating the object representing the best candidate and slightly modifying one or more angles in the new candidate’s tread list. As illustrated in Fig. 5, goals for the algorithms are compared sequentially. In case of free-standing stairs, initially only the distance to the desired end position is evaluated and if a candidate’s distance to the target point is smaller than the best
To be able to numerically evaluate stair designs, relevant metrics need to be combined in a single smoothness metric. In order to control the influence of each metric, weighting factors are introduced. These can be set to 0 in case a particular metric is not to be used.

Here, $S$ is the smoothness, $a$, $b$, $c$ and $d$ are weighting factors and $A$, $W$ and $D$ are the metrics described above. For $D$, separate weighting factors can be set for the left and the right side of the stair.

### 6.1 Constraints

There may be additional design constraints that are not included in the smoothness metric. For example, one might want to specify a minimum tread side dimension, or for freestanding stairs, a particular angle at which the target point should be approached.

For such constraints, we add an additional step in the evolutionary process: before evaluating the smoothness at all, we first evolve the stair until it fulfils the additional constraints. From that point on, the current stair will be replaced by smoother candidates as long as they still fulfil all additional constraints.

On top of smoothness criteria and additional constraints, the designer may want to control a particular tread angle, or may want to exclude certain treads from being modified. This can easily be accommodated by excluding these treads from the perturbation process of the stair candidates.

### 6.2 Implementation

We implemented the algorithm described above in C# as Grasshopper components for Rhino [13]. However, the methods could be implemented in practically any language and environment. The Grasshopper components and the source code of our implementation have been released under the name Ammonite [14].

### 7 Application A: Winding Stair

In this example, we apply our method to the design of a winding stair with two winding sections. The contours of the stair in the floor plan are given, so we can draw a walk line and divide it evenly. Treads are then created at the resulting location and initially drawn perpendicularly to the walk line. After this, the algorithm described in section 6 will run, optimizing for some of the metrics described in section 5.
Fig. 6. A winding stair, optimized using metric $W$ (left), metric $D$ (right) and a combination of both metrics (middle).

When only using metric $W$, the tread sizes on both sides of the stair show a lot of variation and at some treads, the front and back line of the tread come together in a single point. This solution is identical to an unbalanced stair.

When only using metric $D$, the transition of tread dimensions on the sides is smoother. However, due to the geometry of this stair, many of the treads are skewed and far from perpendicular to the walk line.

By optimizing for a combination of metrics $W$ and $D$, a good solution can be found. The weighting of the two parameters is done by the designer. As can be seen in Fig. 7, the balancing process not only influences the walking comfort, but also has a significant influence on the shape of the handrail, with balanced solutions exhibiting fewer kinks.

Fig. 7. Isometric views of the stair designs shown in Fig. 6. Large changes in tread sizes in the unbalanced stair (left) result in kinks in the handrail segments.
8 Application B: Free-Standing Stair

In this example, we want to create a design for a free-standing stair with a fixed start point and start direction, as well a particular end point and end direction. The geometry of the stair will be generated algorithmically, by optimizing for smoothness metric A (as described in section 5).

**Fig. 8.** Evolutionary process: in the first step, the position of the last tread is incorrect, but the direction of the last tread is already perfect. After a few thousand iterations, the distance to the target point is within desired range. From that point on, smoothness gradually increases.

The initial stair object is created in such a way that the sum of the tread angles equals the angle between the start direction and the desired end direction of the stair. Variations are created by first changing the tread angle of \( n \) treads by random amounts, and then subtracting the sum of these amounts from the tread angle of an unaffected tread. This way, all stair variations will approach the end point in the desired direction, which leaves us with only the end position as a constraint. Applying the method described in section 6 and Fig. 5 results in the process illustrated in Fig. 8. Results for various target points and target directions are illustrated in Fig. 9.

**Fig. 9.** Examples of stairs that are generated using smoothness metric A.

8.1 Obstacle Avoidance

Obstacle avoidance can be added as an additional constraint. Instead of just checking for distance to the target and smoothness, the evaluation of candidates is now a three
stage process: first, the amount of overlap with an obstacle is compared with the current stair; if the amount of overlap decreases, the tread list is replaced. Once there is no overlap at all, the distance to the target point is used to compare solutions. Finally, when there is no overlap and the distance to the target point is within the desired range, smoothness will be improved. Examples of the outcome of this process are shown in Fig. 10.

![Fig. 10. Result of automatic obstacle avoidance, using a column (middle) and a wall (right) as obstacles. As a reference, the result without obstacle is shown on the left.](image)

9 Results

9.1 Design Quality

Objectively comparing the results of our method with known good examples is difficult, as using the evaluation methods introduced in this paper would be an unfair advantage. Furthermore, for any given situation, there are multiple manual balancing methods that could be used, and there are also multiple combinations of metrics and weighting factors that can be selected when using our method.

When evaluating the results of our methods visually in plan, in unrolled section and in 3d, in our opinion the results are often very good. However, as can be seen in Fig. 5, there are situations where particular optimization metrics lead to less desirable results and the designer will need to experiment with various optimization metric settings.

9.2 Speed

The calculation speed varies significantly between scenarios. For stairs with 16 treads and a fixed walk line, metrics A and W can be calculated at a speed of 200,000 candidates per second using a single thread on a 2.7 GHz laptop processor. In practice, this means that the result can be displayed instantly. For metric D, line intersections need to be calculated and the speed drops to around 4,000 candidates per second. This is still fast enough to work interactively, but sometimes one has to wait for a few seconds before a good result has been reached.

For free-standing stairs, about 10,000 candidates can be generated and evaluated per second on the hardware mentioned above. However, convergence speed depends very much on the tolerance of the target point, because tight tolerances result in many candidates being rejected before smoothness is evaluated. An effective way to deal
with this fact is to start with a large tolerance (for example 1 cm) and tighten it once a good solution has been found.

When using obstacle avoidance, many geometric intersections have to be calculated and the speed drops significantly, down to about 200 evaluations per second. However, it may well be possible that significant speed can be gained by implementing more efficient intersection checks.

The evolutionary process we propose is suitable for multi-threaded computation, so further speed improvements are possible. However, even the current computation speed already enables an interactive workflow.

9.3 Workflow

The methods we propose take far less time and also less skill and knowledge of the designer than manual methods. However, as the weighting of metrics influences the outcome, the proposed method should be considered a design tool rather than a fully automated process.

The tool can be used to address design challenge that to our knowledge cannot be solved with commonly used methods. This includes balancing both the left and the right side of a stair in parallel, designing free-standing stairs based on constraints only, and designing free-standing stairs with obstacle avoidance. Furthermore, the computational approach can be used to develop further interactive design methods, for example allowing the designer to manually control a particular tread position or orientation while the evolutionary process is running.

9.4 Robustness and Repeatability

So far, the methods we developed resulted in a solution in the vast majority of cases. Some additional checks proved to be necessary to ensure that the process would not get stuck on wrong solutions; for example, very large tread angles could result in intersecting tread lines, which should be avoided.

As variations are created randomly, there will be slight differences between solutions when starting with a different random seed. However, these differences are typically too small to have any impact on the design process. In some situations, the results of different seeds can be grouped in two sets of solutions, in particular in symmetrical or near-symmetrical conditions.

9.5 Applicability

A precursor to the work presented in this paper has already been used to aid in the design of a stair in the Forum shopping center in Helsinki (see Fig. 11). In this particular project, there were tight geometric constraints that would have been very hard to resolve with other approaches.
Fig. 11. A stair in Helsinki designed by SARC Architects. Development of design tools for this stair initiated the research presented in this paper.

10 Conclusion

We introduced metrics to evaluate the smoothness of stairs and showed how this metric can be used to develop evolutionary tools for stair design. Rather than automating the design of various known stair types, our method is more general and should in principle be able to deal with any stair type.

The methods we propose are fast enough to be used in interactive design tools. The implementation of the proposed methods is straightforward and additional constraints can be easily incorporated.

We have shown how our methods can be used to design free-standing stairs as well as winding stairs. For winding stairs, the left and right sides of the stair can be balanced in parallel, which is not possible using conventional methods.

Free-standing stairs can be generated based on just a user-defined target position and target direction. As an example of additional constraints that can be included in the evolutionary process, integrated obstacle avoidance has been demonstrated.

Acknowledgements. Part of this research has been carried out as thesis project at the MAS Architecture and Information program led by prof. Hovestadt at ETH Zürich. The author thanks SARC Architects and Design for Production for providing inspiring environments to work on this topic.
Fig. 11. A stair in Helsinki designed by SARC Architects. Development of design tools for this stair initiated the research presented in this paper.

Conclusion

We introduced metrics to evaluate the smoothness of stairs and showed how this metric can be used to develop evolutionary tools for stair design. Rather than automating the design of various known stair types, our method is more general and should in principle be able to deal with any stair type. The methods we propose are fast enough to be used in interactive design tools. The implementation of the proposed methods is straightforward and additional constraints can be easily incorporated.

We have shown how our methods can be used to design free-standing stairs as well as winding stairs. For winding stairs, the left and right sides of the stair can be balanced in parallel, which is not possible using conventional methods. Free-standing stairs can be generated based on just a user-defined target position and target direction. As an example of additional constraints that can be included in the evolutionary process, integrated obstacle avoidance has been demonstrated.

Acknowledgements.

Part of this research has been carried out as thesis project at the MAS Architecture and Information program led by prof. Hovestadt at ETH Zürich. The author thanks SARC Architects and Design for Production for providing inspiring environments to work on this topic.

References

7. Rondelet, J.: Traité théorique et pratique de l'art de bâtir, Tome 4, Partie 1, Paris (1810)