CODA: A Design Assistant to Facilitate Specifying Constraints and Parametric Behavior in CAD Models

Category: Research

Abstract

We present CODA, an interactive software tool, implemented on top of Autodesk Fusion 360, that helps novice modelers with designing well-constrained parametric 2D and 3D CAD models. We do this by extracting relations that are present in the design but not yet enforced by constraints. CODA presents these relations as suggestions for constraints and clearly communicates the implications of every constraint by rendering animated visualizations in the CAD model. CODA also suggest dynamic alternatives for static constraints present in the model. These suggestions assist novices in CAD to work towards well-constrained models that are easy to adapt. Such well-constrained models are convenient to modify and simplify the process to make design alternatives to accommodate changing needs or specific requirements of machinery for fabricating the design.

Index Terms: Human-centered computing—Interactive systems and tools Human-centered computing—Visualization—Visualization techniques;

1 Introduction

The maker movement is largely driven by a community of DIY enthusiasts building on each other's work by sharing digital versions of artefacts through online platforms, such as Thingiverse[1] or Youmagine[2] [8, 33]. Makers frequently do this by making rough edits to mesh files or by starting from scratch while using concepts from existing designs [9]. A more convenient way to adapt an existing model is to adjust parameters in a well-constrained parametric model. Well-constrained parametric models allow for dimensional adjustments and personal and aesthetic refinements. Additionally, such changes can also make models compatible with machines and materials available in other labs. For example by compensating for shrinkage of ABS 3D printing filament or the kerf of a laser cutter. González-Lluch and Plumed [10] show however, that even for engineering students it is challenging to verify whether 3D models are fully constrained and thus behave as desired when changes are made. Especially under time pressure trained CAD modelers produce 3D models that are hard to modify because of errors or missing constraints [35]. For makers this can become even more challenging as many do not have a formal training in CAD modeling [22].

To empower and encourage makers to design and share parametric models, Thingiverse launched a Customizer platform [29] that allows for making adjustments to parametric 3D models using simple GUI controls, such as sliders and drop-down menus. However, Thingiverse Customizer requires making the entire 3D model in a CSG scripting language [34], which is significantly different from the feature-based modeling approach supported by popular 3D CAD environments [15], such as Autodesk Fusion and SolidWorks. An analysis of Thingiverse in 2015 shows that only a small fraction of designs (1.0%) are compatible with the Customizer [1]. To help users in specifying parametric models, recent versions of 3D CAD environments automatically add basic constraints to 2D sketches. AutoCAD's auto-constraint feature [19] and BricsCAD's Auto Parameterize functionality [12] go even further and automatically inject

[1] https://www.thingiverse.com/
[2] https://www.youmagine.com/

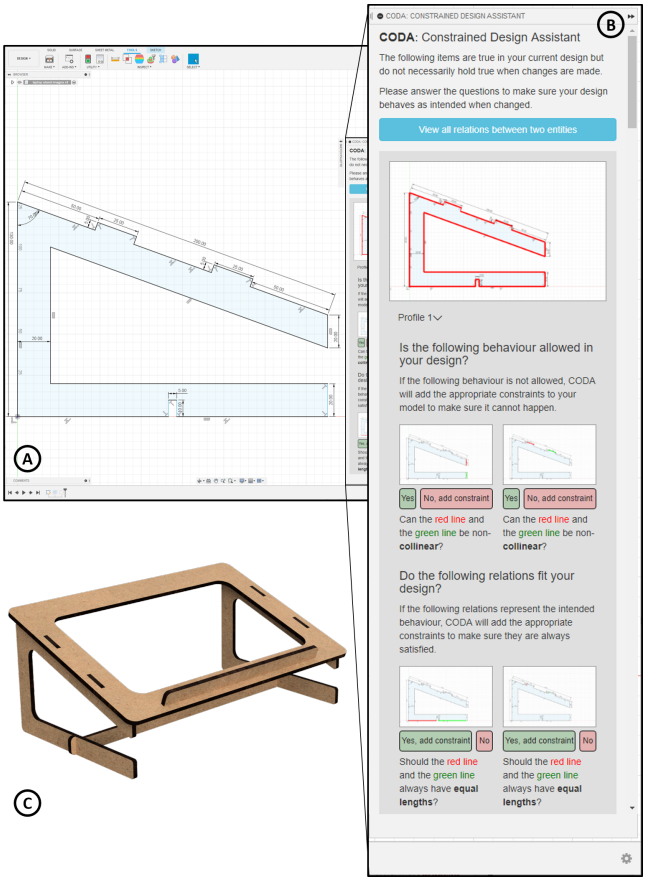

Figure 1: Using CODA to correctly specify parametric behavior in a 3D CAD model of a laptop stand. (a) A 2D sketch of the cross-sectional profile for the laptop stand designed by the user. (b) CODA lists relations that are present in the design but not yet enforced by constraints. (c) Accepting CODA's suggestions allows novice modelers to quickly transition to a well-constraint model ready for fabrication.

constraints models. As there are multiple valid alternatives to constrain models, fully-automatically introducing constraints does not always lead to the desired behavior of the model. For example, the hole for the charging cable of a phone holder (Figure 3) is positioned in the center of the design. It is up to the designers preferences whether to constrain this hole at a fixed distance or a ratio from the top or the bottom of the phone holder. Therefore, the system presented in this paper takes a different approach and suggests constraints and explains their differences and implications on the model.

In this paper, we present CODA, a **Co**nstrained **D**esign **A**ssistant. CODA is an interactive software tool that helps novice CAD modelers with designing well-constrained 2D and 3D CAD models. Such well-constrained models are convenient to modify which simplifies the process to make design alternatives or when making adjustments

for fabricating models with different machinery. First, our system makes users aware of relations that are present in the current design but are not yet enforced by constraints, such as edges being parallel without a parallel constraint. Second, CODA reconsiders static constraints added by the user and suggests more dynamic alternatives to make the design flexible to changes. CODA also helps in communicating the meaning and implications of all suggested constraints by animating the model and demonstrating its implications (Figure 2).

The core contribution of this paper is *CODA*, an interactive software assistant to aid novice modelers in making well-constrained CAD models that are robust to changes. More specifically, we contribute:

1. A computational approach for extracting relations in a model which are not yet enforced by constraints.
2. A set of novel interactive animations to clearly communicate the impact of constraints to novice users.

2 Related work

This work draws from, and builds upon prior work on facilitating CAD modeling and work related to sharing models for fabrication.

2.1 Facilitating CAD Modeling

3D CAD modeling environments offer hundreds of features. How these features are used and combined determine the flexibility, adaptability, and ultimately the reusability of 3D models [5, 11]. Research shows, however, that even models designed by students with a formal 3D modeling training are often hard to reuse and adapt, especially when designed under time pressure [35]. In line with these observations, González-Lluch and Plumed [10] show that engineering students have a hard time reasoning whether profiles are over- or under-constrained. When considering modeling within the maker community, an emerging group of people learn 3D modeling practices by themselves through online resources [22]. However, these novice modelers could significantly benefit from high-quality models that are easy to adapt as they frequently make new artefacts by starting from existing designs [9].

To lower the barrier to get started with 2D and 3D modeling, various tools have been developed that specifically target novices, such as Autodesk Tinkercad [3] and BlocksCAD [4]. However, several studies with casual makers [16, 30], children [17], and students in special education schools [4] show that 3D modeling is still challenging. To facilitate further, the Chateau [18] system helps with CAD modeling by suggesting modeling operations based on simple sketch gestures by the user. Rod et al. [37] presents various novel interaction techniques to further facilitate 3D modeling on touch-screen devices.

Instead of adapting CAD environments and making custom modeling operations for novices, researchers also explored how to facilitate the process for novices to learn a new CAD environment. GamiCAD [27] and CADament [28] gradually introduce sketching and modeling operations using gamification techniques [27, 28] to lower the barrier and keep novices motivated to continue learning new aspects. Alternatively, Blocks-to-CAD [24] shows how to gradually introduce 3D modeling operations in sandbox games, such as Minecraft, to get newcomers introduced to the basics of CAD modeling. Additionally, recent research results show how modeling strategies from experts can be modeled, analyzed, and compared to provide guidance for other users during modeling sessions [6].

Instead of embedding expert knowledge in software systems, software systems can help in bringing novices in contact with experts while facing issues with 3D modeling. MicroMentor [20], for example, makes it possible for novices to request one-on-one help for specific issues. In contrast, the Maestro [7] system makes educators in workshops aware of student's progress and common challenges as they occur. Although experts typically provide more nuanced answers to the various challenges novices face, experts often need an incentive to help other users and first need to get familiar with the specific problem the user is facing [20].

Several techniques have been developed that specifically aim to improve the adaptability and re-usability of models by facilitating specifying parametric behaviour. Commercial feature-based CAD modeling tools support, for example, snapping interaction techniques to ease and improve precision in 2D sketches, such as centering a point in the middle of a line or sketching two perpendicular lines. Leveraging this snapping functionality oftentimes automatically fixes the relation by injecting the associated constraints in the 2D sketch. AutoConstraint [19] takes a different approach and adds constraints to a completed sketch until it is fully constrained. Closest to our work is the Auto Parameterize [12] feature of BricsCAD [5] which automatically converts all static dimensions of a 3D model to algebraic equations to facilitate scaling and adapting the model. However, there are always multiple valid alternatives to constrain models and fully-automatically introducing constraints does not always lead to the desired behavior of the model. We therefore take a different approach and present various geometry constraints and algebraic relations that could be applied to the model. CODA communicates the implications of all suggestions using in-context animations to allow novice users to make informed decisions.

Also related to our work are computational approaches to reverse engineering CAD models from mesh models by extracting modeling features [36, 41]. Several systems also present algorithms to detect and extract geometry constraints in mesh models using numerical methods for constrained fitting [2] and techniques for detecting repeating patterns [25] and symmetries [26].

2.2 Sharing Models for Fabrication

Over the past decade, digital fabrication has become accessible mainly via public maker labs and affordable digital fabrication equipment [32]. Shewbridge et al. [39] report that households are interested in replacing, modifying, customizing, repairing, or replicating household objects using digital fabrication machinery. However, starters frequently need help from more experienced users translate ideas into 3D models. This is often done via drawings, photographs, and spoken language [39]. While platforms, such as Upwork [6] and Cad Crowd [7] are available to outsource 3D modeling work, they require additional expenses.

Instead of designing CAD models from scratch, makers often adapt or combine existing 3D models, found on public repositories, such as Thingiverse [4, 9, 16]. This process can be challenging as many users only share triangular mesh file-formats (STL) [1]. While users can request changes for models through the comments section, studies show that only 32% of such requests are granted [1]. To empower novice CAD modelers to adapt models themselves, Thingiverse introduced the Customizer feature [29], a plugin that exposes GUI controls to adjust the parameters of models designed with the OpenSCAD [34] scripting language. While the Thingiverse Customizer is highly popular [8, 33], only a small portion (3.7%) of 3D models available on the platform are modeled in OpenSCAD, and only 1% are compatible with the Customizer [1]. Hudson et al. [16] observe that modeling in OpenSCAD is challenging and significantly different from feature-based parametric modeling environments traditionally used by CAD modelers.

In contrast to these efforts, CODA guides and stimulates novice modelers in making well-constrained models in popular feature-based CAD modeling environments. Well-constrained parametric

[3] https://www.tinkercad.com/
[4] https://www.blockscad3d.com/
[5] https://www.bricsys.com/
[6] https://www.upwork.com/
[7] https://www.cadcrowd.com/

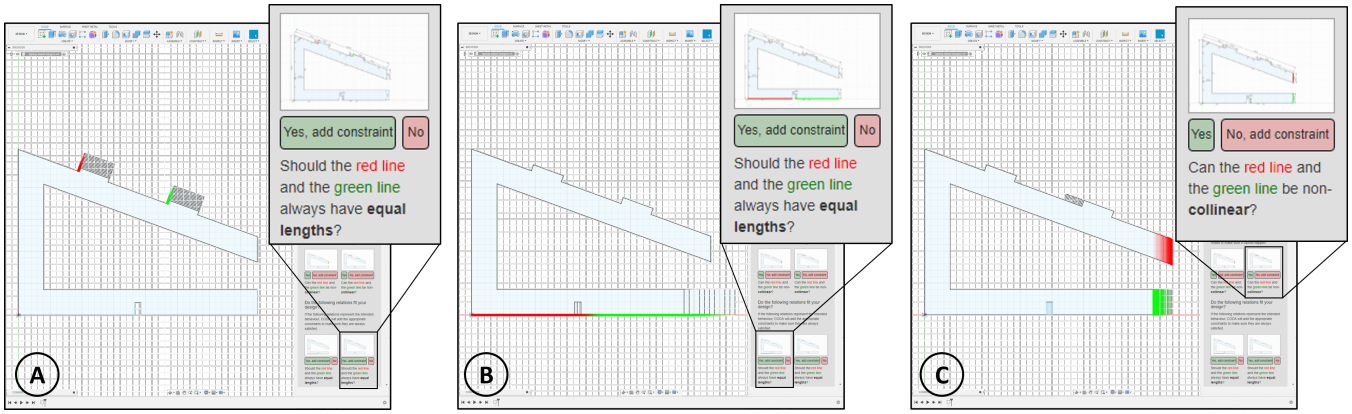

Figure 2: CODA's animations show the implications of suggested constraints. (a) A suggestion for constraining the two tabs to have the same height. (b) A suggestion for overwriting a constraint that ensures the slot at the bottom is always centered. (c) A suggestion for making the front edges collineair. (All three figures are edited to visualize the animation)

models are convenient to adapt as they represent a family of alternative models [3].

3 System Overview

This section gives an overview of CODA's core features. We start with a short walkthrough demonstrating how our system can be used in a real modeling workflow. Afterwards, we discuss CODA's features in more detail.

3.1 Walkthrough

This walkthrough demonstrates the design process Emily, a novice modeler in Autodesk Fusion 360, follows to design a laptop stand that can be laser cut (Figure 1c). During this process, CODA offers support to make a laptop stand that is well-constrained, and easy to adapt and scale to other laptops or devices (Figure 1b).

As shown in Figure 1a, Emily starts with sketching the 2D cross-sectional profile for the laptop stand. She adds dimensions to the sketch to fit the size of her laptop. While sketching, CODA informs Emily that the slot and tabs are 5mm in size and asks whether these features should always have the same dimension. When hovering this suggestion, CODA animates the model by resizing these features at the same time to demonstrate the effect of the suggested constraint (Figure 2a). Emily accepts the suggested constraint and CODA replaces the static dimension constraint with dimensions that share the same value (variable).

CODA also notices that the slot at the bottom is currently positioned in the center but not constrained as such. Therefore the system suggests to replace the dimension that offsets the slot from the left edge with a geometry constraint that ensures that the bottom edges on both sides of the slot are always equal. Again, Emily accepts the suggestion after inspecting the animation to understand the implications of the constraint (Figure 2b).

Next, Emily notices a suggestion for making the two vertical edges at the right of the profile collinear. When hovering the suggestion, the animation informs her that when the size of the slanted edge of the stand would change, the bottom of the laptop stand does not yet adjust accordingly (Figure 2c). Emily accepts the suggestion to make the two edges collinear as she prefers a laptop stand that is well aligned. CODA offers more relevant suggestions to improve the constraints in this cross-sectional profile which Emily can accept as desired. Examples include, making all three sides equally wide (uniform thickness), relating the width of the two tabs, and suggestions related to the positioning of the two tabs.

Further in the design process, when extruding the profiles 5mm, CODA also notices this extrusion depth equals the size of the tabs and slots and suggests creating a constraint. When the final laptop stand is finished, Emily can easily adjust the stand to fit other laptops or adjust the material thickness to fabricate it with different material. She also decides to make the model available on Thingiverse as it is versatile and robust to changes.

3.2 Extracting Relations

In order to suggest constraints, CODA continuously extracts the following four types of relations in CAD models [38]:

1. Ground relations: relations with respect to the reference coordinate system, such as a line being horizontal or vertical in a 2D sketch (Figure 3a).
2. Geometric relations: relations that define known geometric alignments, such as tangency, collinearity, parallelism, perpendicularity, and coincidence of points (Figure 3b).
3. Dimensional relations: relations between sizes or offsets between elements, such as edges with an equal length or a point in the middle of an edge (Figure 3c).
4. Algebraic relations: restrictions on the model in the form of mathematical expressions. For example, a edge being twice as long as another edge (Figure 3d).

While extracting relations, CODA considers parameters of modeling operations as well as attributes of all entities in a sketch (i.e. sketch entities). Sketch entities in Autodesk Fusion 360 include points, lines, circles, ellipses and arcs. Rectangles, for example, are not sketch entities but profiles as they consist of multiple lines. While CODA always extracts relations between exactly two attributes, atributes of various types can be related by CODA. Here we can distinguish the following combinations:

- Relations between sketch entities within a single sketch. Within a sketch, all four types of relations are applicable. For example, a line being tangent to a circle or a rectangle having twice the width of the diameter of a circle.
- Relations between sketch entities across different sketches. For these relations, dimensional and algebraic relations are applicable, such as a slot in two different sketches being equal in size.
- Relations between parameters of modeling operations. For these relations, dimensional and algebraic relations are applicable, such as two fillet operations with the same radius or the

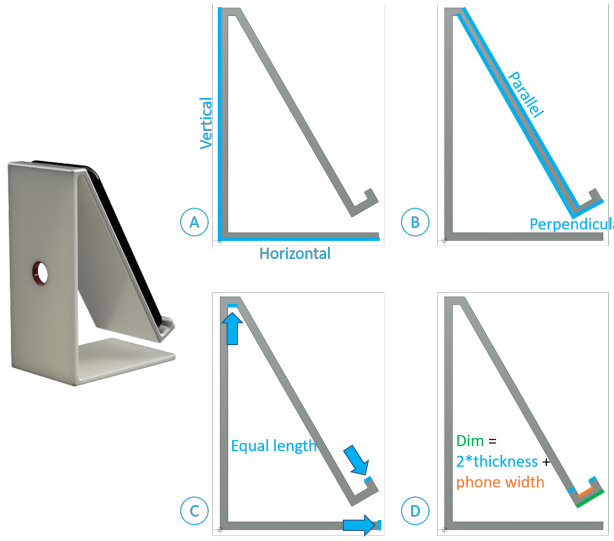

Figure 3: Cross section profile of a phone holder with four types of relations annotated: (a) Ground relations. (b) Geometric relations. (c) Dimensional relations. (d) Algebraic relations.

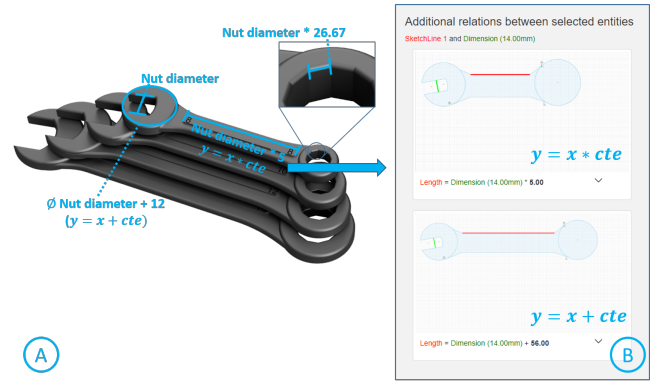

Figure 4: (a) This wrench requires custom constraints to ensure it scales appropriately with respect to the nut diameter. (b) CODA assist users in making such custom relations.

depth of an extrusion being equal or half the size of the radius of a fillet operation.

- Relations between a parameter of a modeling operation and a sketch entity. For these relations, dimensional and algebraic relations are applicable, such as the width of a slot in a sketch being equal to the depth of an extrusion.

To not overwhelm users with suggestions and to offer suggestions in context, we present relations within a single sketch only when the user is editing the respective sketch. All other relations are presented outside the 2D sketching mode in 3D modeling mode.

As the number of algebraic relations is possibly very large, especially when considering all pairs of sketch entities and parameters of modeling features, CODA first extracts algebraic relationships that are frequently present in CAD models. Studies of Mills et al. [31] and Langbein et al. [25] show that the most common relations in CAD models include equal radii and lengths of edges, congruent faces, and radii and edges being half, one third, and one fourth in length. CODA thus presents algebraic relations between all pairs of sketch entities and parameters of modeling operations and vice versa that are equal, half, one third, and one fourth in value. To help users create well-constrained models with less common relations, the next section covers CODA's features to facilitate specifying custom algebraic relations between specific pairs of entities.

3.3 Custom Algebraic Constraints

To help users identify and specify custom algebraic constraints (Figure 4a), CODA supports an in-depth search between two entities. When starting this feature, the user selects two entities. These can be sketch entities (e.g. points, lines, circles, ellipses, arcs), dimensions, or modeling features (e.g. extrusion, fillet operations). CODA now calculates mathematical expressions between all pairs of attributes of the two entities, independent of whether the relation is a common ratio. As shown in Figure 4b, CODA suggests a relation for every pair of attributes both as ratios ($y = x * cte$) and sums ($y = x + cte$).

3.4 Communicating and Explaining Constraints

To clearly communicate the meaning and implications of suggested constraints, CODA creates animated visualizations for all suggestions. Hovering a suggested constraint animates the model and communicates the meaning of the constraint in the context of the model. Explaining a constraint using animations on top of the model, in contrast to generic visualizations, textual explanations, or symbols, makes it convenient for users to understand its implications in the model. We developed two classes of animations to visualize the behavior of the four supported types of relations:

- For ground and geometric relations, we animate the variability currently present in the model when the suggested ground or geometric constraint would not be added. For example, hovering the suggested collinear constraint in Figure 2c, repeatedly moves the two edges that are collinear but not yet constrained as such. The animation makes the user aware that these two edges can still move with respect to each other. Using the "yes" and "no" buttons, the user specifies whether the demonstrated movement of these lines is allowed. "no" adds the collinear constraint while "yes" discards the suggestion without further action. Figure 5a gives an overview of how CODA animates all geometric and ground relations.
- For dimensional and algebraic relations, we animate how two entities would behave when their position or sizes would be constrained to each other. For example, hovering the suggested dimensional constraint in Figure 2a and Figure 2b shows how two lines would scale when they are constrained to have equal dimensions. For these relations CODA asks whether the suggested relations fit the design. "yes" adds the respective dimensional or algebraic constraint, while "no" discards the suggestion. Figure 5b gives an overview of how CODA animates all dimensional and algebraic relations.

3.5 Invalidated Relations

It is important to note that CODA only suggests constraints to enforce relations that are present in the current version of the model. Oftentimes while testing the robustness of a model, the model breaks because of constraints that are still missing. Figure 6a shows how the symmetry in the laptop stand breaks when changing the width because of a missing constraint. These missing constraints will not be suggested by CODA as the relations are not present anymore in the broken model. To solve this inconvenience, CODA continuously presents a message communicating how many relations are invalidated by the last modeling operation (Figure 6b). As shown in Figure 6c, the user then gets the option to revert the last action that broke the model and see the list of invalidated relations that could be added to further constrain the model. This is an iterative and powerful workflow that allows users to break relations in a model and show which constraints can be added to enforce relations.

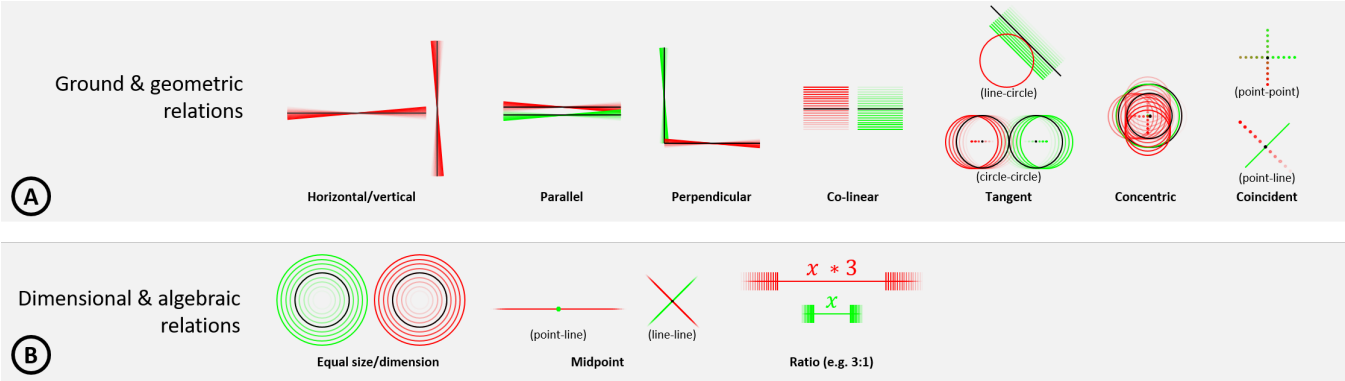

Figure 5: CODA animates the meaning and implications of suggested constraints. (a) shows an abstraction of animations for demonstrating ground and geometric relations. (b) shows an abstraction of animations for demonstrating dimensional and algebraic relations.

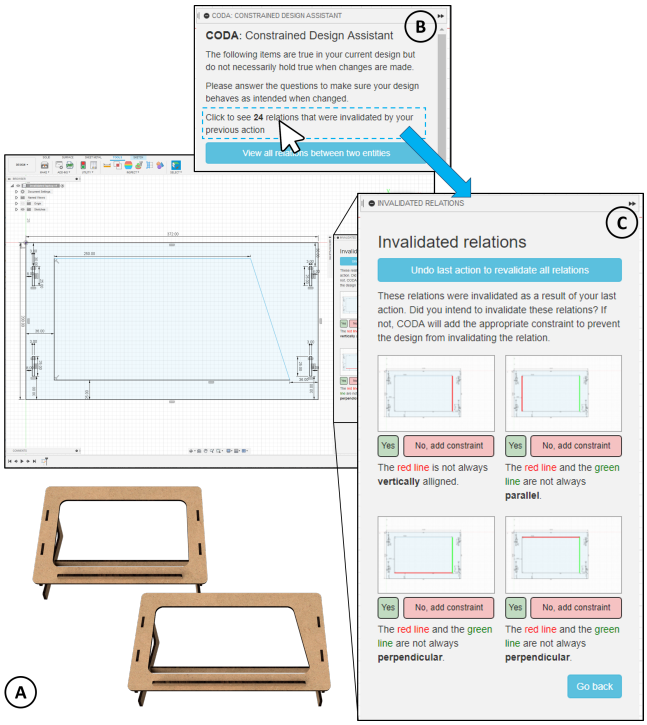

Figure 6: CODA only suggests relations to enforce options that are currently present in the model. (a) Changing parameters can break relations because of missing constraints. (b-c) CODA resolves this by showing a list of constraints that were invalidated by the last modeling operation.

4 Implementation

CODA is implemented as a Python plugin for Autodesk Fusion 360 [8]. The concepts and features presented in CODA, however, are not specific to the Fusion 360 environment and could be implemented in other feature-based parametric CAD environments, such as SOLIDWORKS, Rhinoceros 3D, or Autodesk Inventor.

[8] https://www.autodesk.com/products/fusion-360/overview

4.1 Extracting unconstrained relations

To suggest geometry constraints, CODA continuously checks for the presence of such relations between all pairs of sketch entities within a single sketch and across different sketches. Figure 7 gives an overview of how CODA checks if a geometric relation is present between two sketch entities.

Perpendicular	a, b	$\bar{a} \cdot \bar{b} = 0$
Parallel	a, b	$\frac{ax}{ay} = \frac{bx}{by}$
Collinear	a, b	$\bar{a} = n\bar{b}$ for a real n
Tangent (line-circle/ellipse)	$y = mx + c$, l_1, l_2, (x_1, y_1) ($l_1 = l_1$ for circle)	$\frac{(x-x_1)^2}{{l_2}^2} + \frac{(mx+c-y_1)^2}{{l_1}^2} = 1$ has one unique real root
Tangent (circle/ellipse-circle/ellipse)	l_{11}, l_{12}, l_{21}, l_{22}, (x_1, y_1), (x_2, y_2) ($l_{11} = l_{12}, l_{21} = l_{22}$ for circle)	$\begin{cases} \frac{(x-x_1)^2}{{l_{12}}^2} + \frac{(y-y_1)^2}{{l_{11}}^2} = 1 \\ \frac{(x-x_2)^2}{{l_{22}}^2} + \frac{(y-y_2)^2}{{l_{21}}^2} = 1 \end{cases}$ has one unique real root
Concentric	a, b	$C(a) = C(b)$ with C = center point
Coincident (point-point)	a,b	$a = b$
Coincident (point-line)	$y = mx + c$, b	$b_y = mb_x + c$

Figure 7: CODA uses basic linear algebra methods to check for geometric relations between entities in a sketch.

Sketch entity	Attributes
Points	(x,y) position
Lines	(x,y) position of start, mid, and end point, length of line
Circles	(x,y) position of center, diameter of circle
Ellipses	(x,y) position of center, size along minor and major axis
Arcs	(x,y) position of start, mid, and end point, radius of arc

Table 1: Attributes of sketch entities considered by CODA to extract dimensional and algebraic relations.

When extracting dimensional and algebraic relations, CODA finds pairs of attributes in a 3D CAD model that are equal, half, one third, and one fourth in value. These are frequently used relations in CAD models according to studies by Mills et al. [31] and Langbein et al. [25]. Note that for these relations, parameter values of modeling operations, such as extrusions and fillet, as well as the position and sizes of sketch entities are considered. Table 1 gives an overview of the attributes CODA considers for extracting dimensional and algebraic relations per sketch entity.

For CODA to only suggest relations that are not yet enforced by constraints, our algorithm needs to take into account constraints already present in CAD models. The Fusion 360 API, however, only exposes constraints that are explicitly present in the CAD model. Sketch entities, however, can also be implicitly constrained by other constraints. The next subsection discusses how we extract those implicit constraints to ensure CODA does not offer these suggestions.

4.2 Extracting implicit constraints present in the model

Fusion 360's API does not provide access to the constraint graph and only exposes constraints that are explicitly added by the user or through the API. Besides these constraints, however, other implicit constraints can be present in a sketch. For example, two lines with a parallel constraint to a third line are always parallel to each other without that parallel constraint being present. CODA needs to be aware of these implicit constraints to avoid suggesting constraints that are already present in the model (Section 4.1) as well as to prevent over-constraining models (Section 4.3).

To extract these implicit constraints, we re-implemented and extended the technique of Juan-Arinyo and Soto [21]. This technique requires all constraints to be expressed as either constrained distance (CD) sets, constrained angle (CA) sets, or constrained perpendicular distance (CH) sets. While CD sets includes points between which all distances are constrained, CA sets consist of line segments between which all angles are constrained, and CH sets consist of a point for which the perpendicular distance to a line segment is constrained. We convert all length/size constraints in Fusion 360 to constrained distance (CD) sets, angle constraints to constrained angles (CA) sets, and offset constraints to constrained perpendicular distance (CH) sets. For ground and geometry constraints in Fusion 360 we use the following conversion:

- Horizontal/vertical: We add a constrained angle (CA) set representing an angle of 0° between the line segment and the x- or y-axis.
- Perpendicular: We add a constrained angle (CA) set representing an angle of 90° between the two line segments.
- Parallel: We add a constrained angle (CA) set representing an angle of 0° between the two line segments.
- Collinear: For all pairs of end-points of the two line segments, we add a constrained perpendicular distance (CH) set with a distance of 0.
- Concentric: We add a constrained distance (CD) set, representing a distance of 0 between the mid-points of the two concentric circles.
- Coincident (point-point): We add a constrained distance (CD) set, representing a distance of 0 between the coincident points.
- Coincident (point-line): We add a constrained perpendicular distance (CH) set, representing a distance of 0 between the point and the line.
- Tangent (circle/ellipse-line): We add a constrained angle (CA) set, representing an angle of 90° between the line segment and the radius of the circle at the tangency point.
- Tangent (circle/ellipse-circle/ellipse): We add a constrained angle (CA) set, representing an angle of 0° between the radii of the two circles at the tangency point.

Once all Fusion constraints are converted to CD, CA, and CH sets, we compute the transitive closure of the constrained angles (CA) sets and use the 20 rules presented by Juan-Arinyo and Soto [21] to merge constraints. Now we get CD, CA, and CH sets that reflect the implicit constraints. When two sketch entities have a relation that is not yet enforced by a explicit constraint, CODA can now check whether these entities are implicitly constraint using the following rules:

- Horizontal/vertical: Implicitly constrained if a constrained angle (CA) set exist between the line segment and a line segment representing the x- or y-axis.
- Perpendicular/parallel: Implicitly constrained if a constrained angle (CA) set exist representing both line segments or if a constrained distance (CD) set exists representing the four points of the two line segments.
- Collinear: Implicitly constrained if a constrained perpendicular distance (CH) set exist representing both line segments or if a constrained distance (CD) set exists representing the four points of the two line segments.
- Concentric: Implicitly constrained if a constrained distance (CD) set exists representing the center points of the two circles.
- Coincident point-point: Implicitly constrained if a constrained distance (CD) set exists representing the two points.
- Coincident point-line: Implicitly constrained if a constrained perpendicular distance (CH) set exists representing the point and the line or if a constrained distance (CD) set exist representing the point and both points of the line segment.
- Tangent circle/ellipse-line: Implicitly constrained if a constrained angle (CA) set exists representing the line segment and a radius (line segment) passing through the tangency point. Alternatively, if a constrained distance (CD) set exists including the two points of the line segments and the two points of the radius passing through the tangency point.
- Tangent circle/ellipse-circle/ellipse: Implicitly constrained if a constrained angle (CA) set exists representing the radii of both circles passing through the tangency point. Alternatively, if a constrained distance (CD) set exist including the point of the four line segments passing through the tangency point.

4.3 Removing dimensions to avoid over-constraining models

Accepting suggested constraints oftentimes requires removing existing constraints present in the model. In the sketch in Figure 8a-left, for example, CODA suggests to constrain the length of the lines so that they are always half the length of each other. To accept this constraint, CODA replaces the static dimensional constraint with the dynamic constraint shown in Figure 8a-right. However, when both dimensions are implicitly constrained as shown in Figure 8b-left, CODA needs to remove one of the other constraints before adding the suggested constraint shown in Figure 8b-right. For CODA to know the explicit constraints that are responsible for every implicit

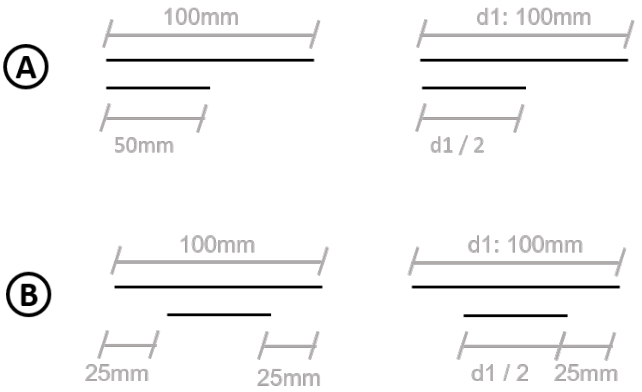

Figure 8: When accepting suggested constraints CODA oftentimes (a) overwrites existing constraints or (b) removes existing constraints.

constraint, we keep track of all explicit constraints while merging CD, CA, and CH sets in the algorithm explained in Section 4.2. When multiple explicit constraints are responsible for an implicit constraint (Figure 8b), CODA shows multiple suggestions and communicates their differences through the animated visualizations.

4.4 Rendering animations in CAD models

When hovering suggested constraints, CODA previews the implications of a constraint by animating features in the CAD model (Figure 5). When the animation requires lines to tilt, we continuously rotate the line between -5° and +5°. When changes in size are required, we apply a scaling that transitions between half and double the size of the entity. Finally, when animating a parameter of a modeling operation, such as an extrusion, we alternate between half and double of the original value. Animations are updated every 100ms and are repeated as long as the user hovers the suggested constraint.

As CODA directly manipulates parameters in the original design, we make a copy of all the attributes to be able to restore them afterwards. For some sketch entities, additional temporary construction lines are required to realize the animation. For example, to vary the angle between two parallel lines, Fusion 360 does not allow to temporarily add an angular constraint between the two lines as they are exactly parallel. CODA therefore first adds a temporary construction line perpendicular to both lines and varies the angle between the perpendicular line and both parallel lines during the animation (Figure 5a).

5 Limitations and Future Work

Although CODA offers many novel opportunities to facilitate making well-constrained CAD models, our work also has several limitations which reveal many exciting directions for future research.

First, future research could study how novices in CAD modeling use CODA during their design workflow. While CODA offers suggestions in real-time while modeling, our tool could also be used after a CAD model is finished or at a later time to make an existing model more flexible and adaptable. We believe this could be a major asset as it allows novices to further improve CAD models shared via platforms, such as Thingiverse, and thus distribute the workload across the community. Furthermore, it would be interesting to investigate whether CODA also provides value to expert modelers and for students that learn CAD modeling. For example, the suggested constraints can make students aware of available CAD features and how they are composed.

Second, while CODA offers suggestions for the most common constraints in CAD, more advanced suggestions can be supported in the future. For geometry relations this includes identifying patterns, such as symmetry or repetition in models and offering suggestions to convert these patterns into more adaptable features. For algebraic constraints, the current version of CODA supports frequently used ratios in CAD models according to Mills et al. [31] and Langbein et al. [25]. Other common ratios used in design could be supported in the future, such as the Golden ratio or the Lichtenberg ratio. Future versions could also offer suggestions for relations that are nearly present in the model, such as lines that are almost perpendicular or are almost equal in length. Similar approaches have been explored for improving mesh models [23].

Third, to further facilitate adapting models, future versions of CODA could offer suggestions for other types of constraints, such as limiting the range in which parameters and dimensions can change without breaking the model. While computing valid ranges of features has been investigated for 2D sketches [13, 14, 40], more research is needed to compute valid ranges for all features in 3D to ensure the integrity of the model when changing parameters.

6 Conclusion

In this paper we presented CODA, an interactive software tool that helps novice modelers to design well-constrained parametric 2D and 3D CAD models. In order to do so, CODA contributes a computational approach for extracting and suggesting relations in a model that are not yet enforced by constraints. CODA also clearly communicates the meaning and implications of suggested constraints using novel animated visualizations rendered in the CAD model. By facilitating the creation of well-constrained parametric designs we hope to further democratize CAD and encourage users to upload high quality parametric models to public sharing repositories, such as Thingiverse.

References

[1] C. Alcock, N. Hudson, and P. K. Chilana. Barriers to using, customizing, and printing 3d designs on thingiverse. In *Proceedings of the 19th International Conference on Supporting Group Work*, GROUP '16, p. 195–199. Association for Computing Machinery, New York, NY, USA, 2016. doi: 10.1145/2957276.2957301

[2] P. Benko. Constrained fitting in reverse engineering. *Computer Aided Geometric Design*, 19(3):173–205, Mar. 2002. doi: 10.1016/s0167-8396(01)00085-1

[3] B. Bettig and J. Shah. Derivation of a standard set of geometric constraints for parametric modeling and data exchange. *Computer-Aided Design*, 33(1):17–33, Jan. 2001. doi: 10.1016/s0010-4485(00)00058-0

[4] E. Buehler, S. K. Kane, and A. Hurst. Abc and 3d: Opportunities and obstacles to 3d printing in special education environments. In *Proceedings of the 16th International ACM SIGACCESS Conference on Computers & Accessibility*, ASSETS '14, p. 107–114. Association for Computing Machinery, New York, NY, USA, 2014. doi: 10.1145/2661334.2661365

[5] J. D. Camba, M. Contero, and P. Company. Parametric CAD modeling: An analysis of strategies for design reusability. *Computer-Aided Design*, 74:18–31, May 2016. doi: 10.1016/j.cad.2016.01.003

[6] M. Chang, B. Lafreniere, J. Kim, G. Fitzmaurice, and T. Grossman. Workflow graphs: A computational model of collective task strategies for 3d design software. In *Proceedings of Graphics Interface 2020*, GI 2020, pp. 114 – 124. Canadian Human-Computer Communications Society / Société canadienne du dialogue humain-machine, 2020. doi: 10.20380/GI2020.13

[7] V. Dziubak, B. Lafreniere, T. Grossman, A. Bunt, and G. Fitzmaurice. Maestro: Designing a system for real-time orchestration of 3d modeling workshops. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, p. 287–298. Association for Computing Machinery, New York, NY, USA, 2018. doi: 10.1145/3242587.3242606

[8] C. M. Flath, S. Friesike, M. Wirth, and F. Thiesse. Copy, transform, combine: Exploring the remix as a form of innovation. *Journal of Information Technology*, 32(4):306–325, Dec. 2017. doi: 10.1057/s41265-017-0043-9

[9] S. Friesike, C. M. Flath, M. Wirth, and F. Thiesse. Creativity and productivity in product design for additive manufacturing: Mechanisms and platform outcomes of remixing. *Journal of Operations Management*, 65(8):735–752, Apr. 2019. doi: 10.1016/j.jom.2018.10.004

[10] C. González-Lluch and R. Plumed. Are we training our novices towards quality 2d profiles for 3d models? In *Advances on Mechanics, Design Engineering and Manufacturing II*, pp. 714–721. Springer International Publishing, 2019. doi: 10.1007/978-3-030-12346-8_69

[11] N. W. Hartman. Defining expertise in the use of constraint-based cad tools by examining practicing professionals. *The Engineering Design Graphics Journal*, 69(1), 2005.

[12] H. Hewitt. New in bricscad® v19: Parametrize. `https://blog.bricsys.com/new-bricscad-v19-parametrize/`, 2018. Accessed: 2020-11-02.

[13] M. Hidalgo and R. Joan-Arinyo. Computing parameter ranges in constructive geometric constraint solving: Implementation and correctness proof. *Computer-Aided Design*, 44(7):709–720, July 2012. doi: 10.1016/j.cad.2012.02.012

[14] C. Hoffmann and K.-J. Kim. Towards valid parametric CAD models. *Computer-Aided Design*, 33(1):81–90, Jan. 2001. doi: 10.1016/s0010-4485(00)00073-7

[15] C. M. Hoffmann and R. Joan-Arinyo. Parametric modeling. In *Handbook of Computer Aided Geometric Design*, pp. 519–541. Elsevier, 2002. doi: 10.1016/b978-044451104-1/50022-8

[16] N. Hudson, C. Alcock, and P. K. Chilana. Understanding newcomers to 3d printing: Motivations, workflows, and barriers of casual makers. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, p. 384–396. Association for Computing Machinery, New York, NY, USA, 2016. doi: 10.1145/2858036.2858266

[17] N. Hudson, B. Lafreniere, P. K. Chilana, and T. Grossman. Investigating how online help and learning resources support children's use of 3d design software. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, p. 1–14. Association for Computing Machinery, New York, NY, USA, 2018. doi: 10.1145/3173574.3173831

[18] T. Igarashi and J. F. Hughes. A suggestive interface for 3d drawing. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, UIST '01, p. 173–181. Association for Computing Machinery, New York, NY, USA, 2001. doi: 10.1145/502348.502379

[19] A. Inc. Autoconstrain (command). `https://knowledge.autodesk.com/support/autocad/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/AutoCAD-Core/files/GUID-2E53F0A6-640C-4B3A-A650-18F1A5F781E1-htm.html`. Accessed: 2020-11-02.

[20] N. Joshi, J. Matejka, F. Anderson, T. Grossman, and G. Fitzmaurice. Micromentor: Peer-to-peer software help sessions in three minutes or less. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, p. 1–13. Association for Computing Machinery, New York, NY, USA, 2020. doi: 10.1145/3313831.3376230

[21] R. Juan Arinyo and A. Soto Riera. A set of rules for a constructive geometric constraint solver. 1995.

[22] K. Kiani, G. Cui, A. Bunt, J. McGrenere, and P. K. Chilana. Beyond "one-size-fits-all": Understanding the diversity in how software newcomers discover and make use of help resources. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, p. 1–14. Association for Computing Machinery, New York, NY, USA, 2019. doi: 10.1145/3290605.3300570

[23] I. Kovács, T. Várady, and P. Salvi. Applying geometric constraints for perfecting CAD models in reverse engineering. *Graphical Models*, 82:44–57, Nov. 2015. doi: 10.1016/j.gmod.2015.06.002

[24] B. Lafreniere and T. Grossman. Blocks-to-cad: A cross-application bridge from minecraft to 3d modeling. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, p. 637–648. Association for Computing Machinery, New York, NY, USA, 2018. doi: 10.1145/3242587.3242602

[25] F. C. Langbein, B. I. Mills, A. D. Marshall, and R. R. Martin. Recognizing geometric patterns for beautification of reconstructed solid models. In *Proceedings International Conference on Shape Modeling and Applications*, pp. 10–19, 2001. doi: 10.1109/SMA.2001.923370

[26] M. Li, F. C. Langbein, and R. R. Martin. Detecting design intent in approximate CAD models using symmetry. *Computer-Aided Design*, 42(3):183–201, Mar. 2010. doi: 10.1016/j.cad.2009.10.001

[27] W. Li, T. Grossman, and G. Fitzmaurice. Gamicad: A gamified tutorial system for first time autocad users. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, p. 103–112. Association for Computing Machinery, New York, NY, USA, 2012. doi: 10.1145/2380116.2380131

[28] W. Li, T. Grossman, and G. Fitzmaurice. Cadament: A gamified multiplayer software tutorial system. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, p. 3369–3378. Association for Computing Machinery, New York, NY, USA, 2014. doi: 10.1145/2556288.2556954

[29] M. I. LLC. Thingiverse customizer. `https://www.thingiverse.com/customizer`, 2020. Accessed: 2020-02-19.

[30] C. Mahapatra, J. K. Jensen, M. McQuaid, and D. Ashbrook. Barriers to end-user designers of augmented fabrication. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19. Association for Computing Machinery, New York, NY, USA, 2019. doi: 10.1145/3290605.3300613

[31] B. Mills, F. Langbein, D. Marshall, and R. Martin. Estimate of frequencies of geometric regularities for use in reverse engineering of simple mechanical components, 03 2001. doi: 10.13140/RG.2.1.3683.2087

[32] C. Mota. The rise of personal fabrication. In *Proceedings of the 8th ACM Conference on Creativity and Cognition*, C&C '11, p. 279–288. Association for Computing Machinery, New York, NY, USA, 2011. doi: 10.1145/2069618.2069665

[33] L. Oehlberg, W. Willett, and W. E. Mackay. Patterns of physical design remixing in online maker communities. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, p. 639–648. Association for Computing Machinery, New York, NY, USA, 2015. doi: 10.1145/2702123.2702175

[34] OpenSCAD. Openscad. `https://www.openscad.org/`, 2020. Accessed: 2020-02-19.

[35] X. Peng, P. McGary, M. Johnson, B. Yalvac, and E. Ozturk. Assessing novice cad model creation and alteration. *Computer-Aided Design and Applications, PACE (2)*, pp. 9–19, 2012.

[36] R. Plumed, P. Varley, and P. Company. Features and design intent in engineering sketches. In *Studies in Computational Intelligence*, pp. 77–106. Springer Berlin Heidelberg, 2013. doi: 10.1007/978-3-642-31745-3_5

[37] J. Rod, C. Li, D. Zhang, and H. Lee. Designing a 3d modelling tool for novice users. In *Proceedings of the 28th Australian Conference on Computer-Human Interaction*, OzCHI '16, p. 140–144. Association for Computing Machinery, New York, NY, USA, 2016. doi: 10.1145/3010915.3010957

[38] J. Shah. *Parametric and feature-based CAD/CAM : concepts, techniques, and applications*. Wiley, New York, 1995.

[39] R. Shewbridge, A. Hurst, and S. K. Kane. Everyday making: Identifying future uses for 3d printing in the home. In *Proceedings of the 2014 Conference on Designing Interactive Systems*, DIS '14, p. 815–824. Association for Computing Machinery, New York, NY, USA, 2014. doi: 10.1145/2598510.2598544

[40] H. A. van der Meiden and W. F. Bronsvoort. A constructive approach to calculate parameter ranges for systems of geometric constraints. In *Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling*, SPM '05, p. 135–142. Association for Computing Machinery, New York, NY, USA, 2005. doi: 10.1145/1060244.1060260

[41] J. Vergeest, Y. Song, T. Langerak, et al. Design intent management for design reuse. 2006.