ModelCraft: Capturing Freehand Annotations and Edits on Physical 3D Models

Hyunyoung Song , François Guimbretière, Chang Hu
Human-Computer Interaction Lab
Department of Computer Science,
University of Maryland,
College Park, MD 20742, U.S.A
{hsong, francois, changhu}@cs.umd.edu

Hod Lipson
216 Upson Hall
Cornell University
Ithaca, NY 14852-7501, USA
hod.lipson@cornell.edu

ABSTRACT
With the availability of affordable new desktop fabrication techniques such as 3D printing and laser cutting, physical models are used increasingly often during the architectural and industrial design cycle. Models can easily be annotated to capture comments, edits and other forms of feedback. Unfortunately, these annotations remain in the physical world and cannot be easily transferred back to the digital world. Here we present a simple solution to this problem based on a tracking pattern printed on the surface of each model. Our solution is inexpensive, requires no tracking infrastructure or per object calibration, and can be used in the field without a computer nearby. It lets users not only capture annotations, but also edit the model using a simple yet versatile command system. Once captured, annotations and edits are merged into the original CAD models. There they can be easily edited or further refined. We present the design of a SolidWorks plug-in implementing this concept, and report initial feedback from potential users using our prototype. We also present how this prototype could be extended seamlessly to a fully functional system using current 3D printing technology.

ACM CLASSIFICATION: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

GENERAL TERMS: Design, Human Factors

Keywords: Pen based interactions, Tangible interactions, Rapid prototyping.

INTRODUCTION
In the process of designing artifacts, today's designers alternate between tangible, non-digital media such as paper or physical 3D models and intangible, digital media such as CAD models. An architect might start the design of a new building with sketches on paper, then, when her ideas solidify, create a rough model using cardboard, before finally creating the corresponding digital model. Once this model is finalized, it might be fabricated as a 3D object (either through rapid prototyping techniques or a modeling studio) so that her clients may have a better grasp of her vision. While a fully digital design process has long been advocated, it still seems a distant goal because tangible, non-digital media models present unique affordances often difficult to reproduce in digital media. Architectural models for example offer a unique presence that is difficult to reproduce on a screen. As a result, even projects that rely heavily on computer assisted design techniques (such as the recent Hearst building designed by Sir Foster) still employ

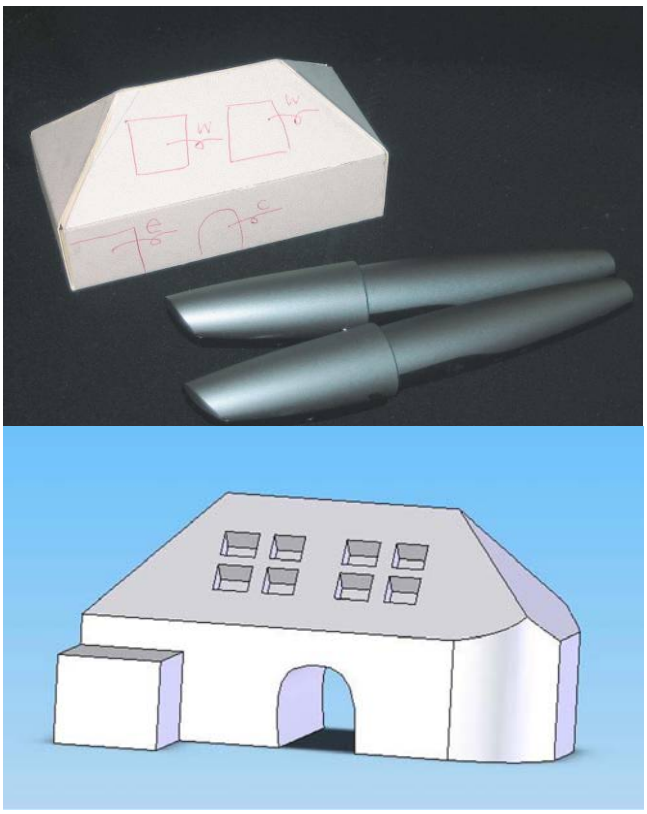

Figure 1: Our system in action. **Top**: paper model of a house with edits (a cut extrusion below the right front corner is not visible). **Bottom**: the same model in our rendering application showing the edits performed.

UIST'06, October 15-18, 2006, Montreux, Switzerland.

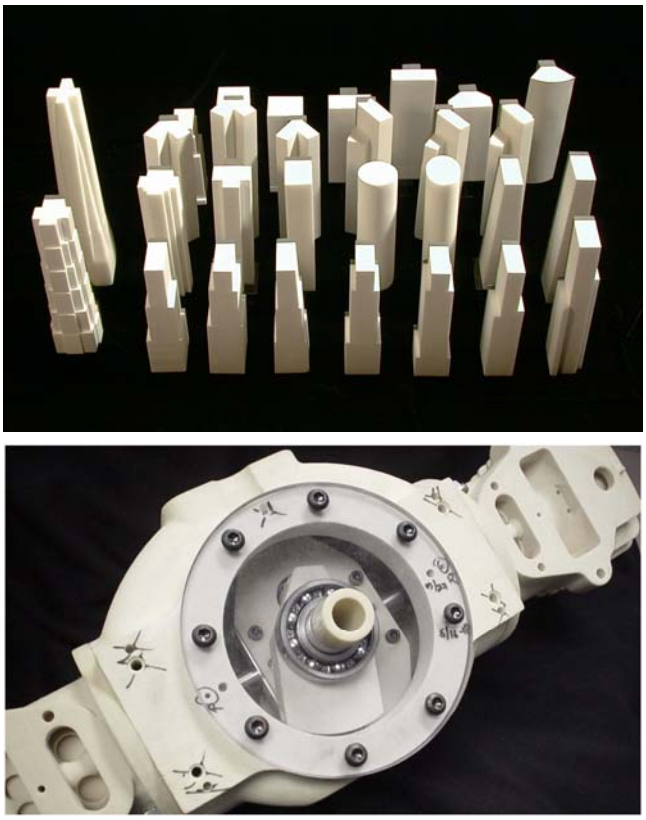

Figure 2 : 3D models are used extensively in design. **Top**: a structural model used during the design of the Hearst building (from [10]). **Bottom**: Annotations on a 3D model from a ZCorp printer (from [35]). Used with permission.

tangible models both for aesthetic and structural tasks [9, 10] (Figure 2, Top).

Interacting with models is an intrinsic part of the design process for architects who see construction (and sometimes deconstruction) as a fundamental part of the idea forming process. For example, during the early phase of the design process called "massing" - a brain storming practice that iterates between incremental modifications and rebuilding of models for conceptualization purposes - inexpensive, easy-to-build models are used to better understand the shape requirement of a building. As rapid prototyping technology has become more commonplace, models are now employed in other areas of design as well. Mechanical designers use models to check form and functional compatibility with the context of an object's use. Models can also be extensively annotated (Figure 2 Bottom). Unfortunately, information captured on such models is difficult to integrate back into the digital world.

While this problem could be addressed by a conventional tracking system (either magnetic or optical) as proposed by Agrawala et al. [2], that approach is limited to a relatively small working volume. Magnetic or optical tracking systems require infrastructure and calibration on a per-model basis, and are somewhat expensive. They are also difficult to deploy in the field where models are frequently tested. This limits widespread adoption by architects or designers.

Noting that most annotations take place on the surface of the model, we present a system which uses the inexpensive, off-the-shelf Logitech io_2™ digital pen [20]. This pen is equipped with a built-in camera which captures position information by observing a digital pattern [4] printed on each model (Figure 1). Our system can capture not only annotations but also editing commands that are subsequently applied to the original digital models. Using our command system, and auxiliary tool such as a ruler, users can alter and adjust the shape of a model, such as modifying dimensions, filleting corners, creating holes, or extruding portions of a model based on requirements in the field. The information is naturally captured in the frame of reference of the model, without the need to worry about scale or orientation. Our approach does not have a predefined working volume, and can easily scale in terms of the number of objects tracked, number of pens used, and locations of usage. Furthermore, it does not require a per-model calibration. Because our approach advocates cohabitation of tangible and digital models, it integrates seamlessly with the current usage patterns among architects and mechanical designers for whom interacting with 3D models is a fundamental part of their creative process. By capturing annotations and edits on physical 3D models, our system streamlines the design process and simplifies documentation of the design history of a given project.

In this paper, we present the first prototype of such a system developed as a plug-in for SolidWorks [31], a commercial CAD application. Starting from a model inside SolidWorks, users can print and build simple, paper-based 3D models (or create water transfers to be applied on an existing 3D model). They can then use a digital pen to annotate them or draw gestures that will be executed upon pen synchronization. After describing the system architecture as well as our editing system, we report on our experiences while designing this system. We also report initial feedback gathered from potential users, professional architects and teachers. Finally, we explore in detail possible paths for the implementation of such a system using current 3D prototyping technology.

PREVIOUS WORK

Several systems allow users to draw (or paint) on digital models. Hanrahan and Haeberli [8] described a WYSIWYG system to paint on 3D models using a standard workstation. This approach has also been adapted to annotate CAD drawings [17, 30]. While drawing on a virtual object has many advantages, such as the ability to work at any scale, we believe that physical models will always play an important role in the design process because of their appeal to designers (Figure 2). In that respect, our approach is closely related to Agrawala et al.'s [2] 3D painting system. Our approach extends this work in several ways: By using a tracking system based on an optical pattern printed on the surfaces of the object, we offer a very short setup time requiring no calibration on a per-object basis. Our tracking

approach also provides greater flexibility for users as annotations can be captured at any location. Finally, our approach is inherently scalable, both in terms of number of models and in terms of annotating devices – a property difficult to achieve by either optical or magnetic tracking techniques. Using a different approach, Grasset et al. [6] proposed to use augmented reality techniques to annotate objects directly. On the one hand, by relying on passive props, our system is less powerful than such systems as it does not offer direct feedback. On the other hand, the simplicity of our system makes its cost of use very low (no need to wear or set up any equipment) – a key aspect for acceptance by designers and architects.

We believe that future systems should allow users to interact directly with the representation of a given object that is most convenient for the task at hand — be it a digital model on a screen or a 3D printout of that model, or a combination of the field sketching proposed here with augmented reality feedback. In that respect, our work is similar in spirit to the work by Guimbretiere [7] on digital annotations of document printouts.

Our work is also related to the large body of work on 3D sketching in systems like Sketch [37], Teddy [14], SketchUp [1] and the 3D Journal project [23]. Our system complements these systems by addressing the need to capture modifications sketched directly on the models at later stages of the design process. In particular, our system makes it easy for users to capture real world geometric information. Our command system is also quite different. While the systems mentioned above focus on a gesture-based interface, we adopt a syntax-based approach inspired by recent work on Tablet-PC-based interfaces such as Scriboli [13], Fluid Inking [36] and paper-based interfaces such PapierCraft [18]. We believe that this approach allows for a more flexible and extensive command set while retaining a sketching-like style.

Our work is also closely related to tangible interfaces [12, 16, 32-34] which let users interact with digital information through the use of tangible artifacts. All these systems leverage users' familiarity with spatial interactions to allow them to perform complex interactions with ease. Our system extends and complements these systems by offering a tighter correspondence between the tangible proxy and its digital representation. In doing so, we offer users the opportunity to modify the digital representation in the real world. In that respect, our system is also closely related to the Illuminating Clay system [26] and Liu's work on editing digital models using physical material [19], as they allow users to see modifications made in the real world applied to the equivalent 3D model. Sheng's thesis on modeling shapes using fingers and physical props [29] is also related to our approach, but Sheng's work focuses on free form shapes such as shaping clay. All these system require the use of somewhat complex tracking equipment only available in a lab setting, while our approach is very light-weight.

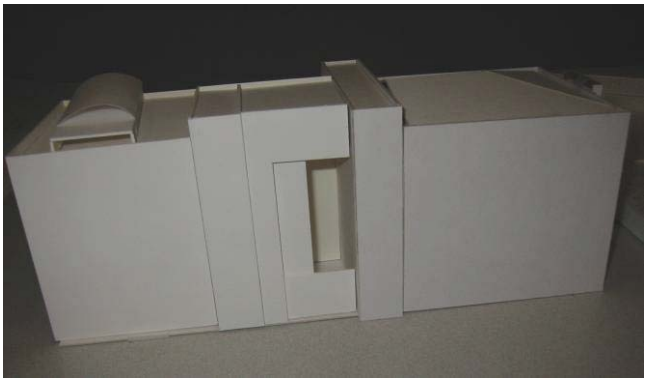

Figure 3 : A typical paper-based massing model used to refine the shape of a building (this example was provided to us during our interview with a professional architect)

MODELCRAFT IN ACTION

While our vision is to have a traceable pattern generated automatically while 3D-printing a model, our current prototype uses simple paper models instead. While building a paper model seems arduous, interviews with architects confirmed that they often build model out of paper (Figure 3), sometimes starting from a printout of an unfolded CAD model. Hence our approach augments current practice.

Each SolidWorks model is printed as an unfolded paper cutout on a page of paper that has been pre-printed with a unique Anoto pattern [4]. This pattern provides a very large space of uniquely identifiable pages (in excess of 2^{48} letter sized pages). Importantly, this makes it possible to interact with different objects or different printouts of the same object at once. Practical paper models are usually quite simple since early designs often rely on a vocabulary of basic shapes (cube, cylinder, pyramid, cone, sphere) as proposed by D. K. Ching [5]. We show a typical example in Figure 3. Moreover, complex shapes are currently supported by printing the object with a 3D printer and printing a slide transfer to be applied to each face.

All interactions are carried out with the Logitech io_2™ pen [20], a commercial implementation of the Anoto system. As each Anoto digital pen has a unique ID, it is also possible to distinguish several different pens interacting on one object. Our system also lets us designate special objects as tools. For example, we instrumented one of our rulers by taping a strip of Anoto pattern onto it (Figure 6). Making marks on this ruler is interpreted by the system as making measurements during command operation.

Annotations

Annotating a model is straightforward: Simply pick up the model and annotate directly on any surface (Figure 4a). Upon pen synchronization, the marks will be merged onto the corresponding surface of SolidWorks model. Users can use several pens for different colors. Marks created by annotation pens are not interpreted by the system.

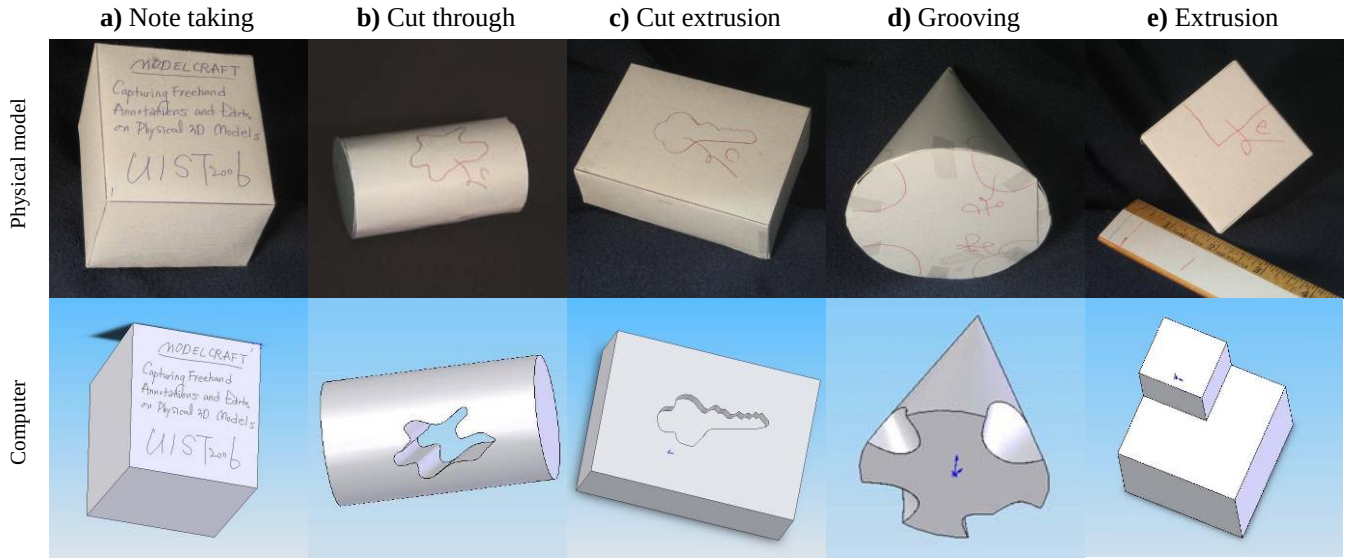

Figure 4 : Current command set. Annotations are done with a black pen, edits are done with a red pen.

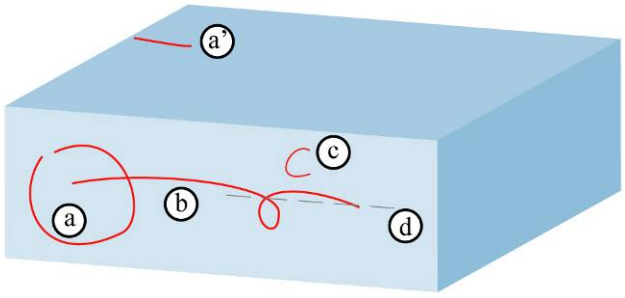

Figure 5 : Command syntax. **a**) main parameter; **a'**) secondary parameter; **b**) pigtail delimiter; **c**) command name; **d**) reference line for character recognition.

Form Editing

Our objective here is not to suggest that our interface can replace the standard (and far more accurate) CAD construction process, but instead to address two different needs. First, in the early stages of design, an approximate solution is often good enough to allow the designer to move to the next stage. For instance, if after 3D printing it is found that a piece conflicts with another element in the design, simply marking the conflicting area and cutting it away may be all that is needed. Second, we found that when a large number of marks are made on the prototype (during massing work, for example), it is somewhat difficult upon synchronization to understand how the marks relate to each other. In that context, providing a tentative execution of the operations helps the users understand the structure of the marks. Furthermore, since all annotations and command parameters are created as first class objects inside the SolidWorks features tree, each annotation and command parameter can be easily modified inside SolidWorks. A simple update of the model will automatically reflect these changes.

Command Syntax. All commands are performed with a command pen which lays ink in a different color (red in our system). We choose a "command" pen approach as it fits well with the current practice of using color coded annotations. Other solutions such as having a command button on the pen are also possible.

All commands follow a uniform syntax (Figure 5) inspired by Scriboli [13] and PapierCraft [18]. First, users draw on objects or on tools (like our ruler) a set of strokes that represent the parameters of the action to be performed (Figure 5a). Then they draw a pigtail gesture which is used as a separator between the parameter strokes and the command name, (Figure 5b). Next, they write the name of the command they wish to execute (a simple letter in our current implementation) on top of the pigtail (Figure 5c). During pen synchronization, the command is then executed using the area on which the pigtail started as the primary command parameter. For example, to create a hole though an object (Figure 5), the user would draw the shape of the hole onto the object surface, then draw a pigtail starting inside the shape, and then write a C (for cut) on top of the pigtail. Note that starting the pigtail outside of the shape would have created a pillar instead.

The use of the pigtail proved to be very reliable for pen-based interaction [13] and is well-adapted to our case as it does not require any feedback besides the ink laid on the surface [18]. For our system, the pigtail has two advantages. First, it serves as a natural callout mark when one needs to execute a command on a small area (like cutting a hole for a screw). Under such conditions, it would be difficult to write the name of the command directly on the area of interest because the area is too small or too close to the surface border. Second, the pigtail provides a natural orientation for the surface. While up and down are well understood in a Tablet-PC context, this is not the case on 3D objects which people may place in arbitrary orientations to facilitate the

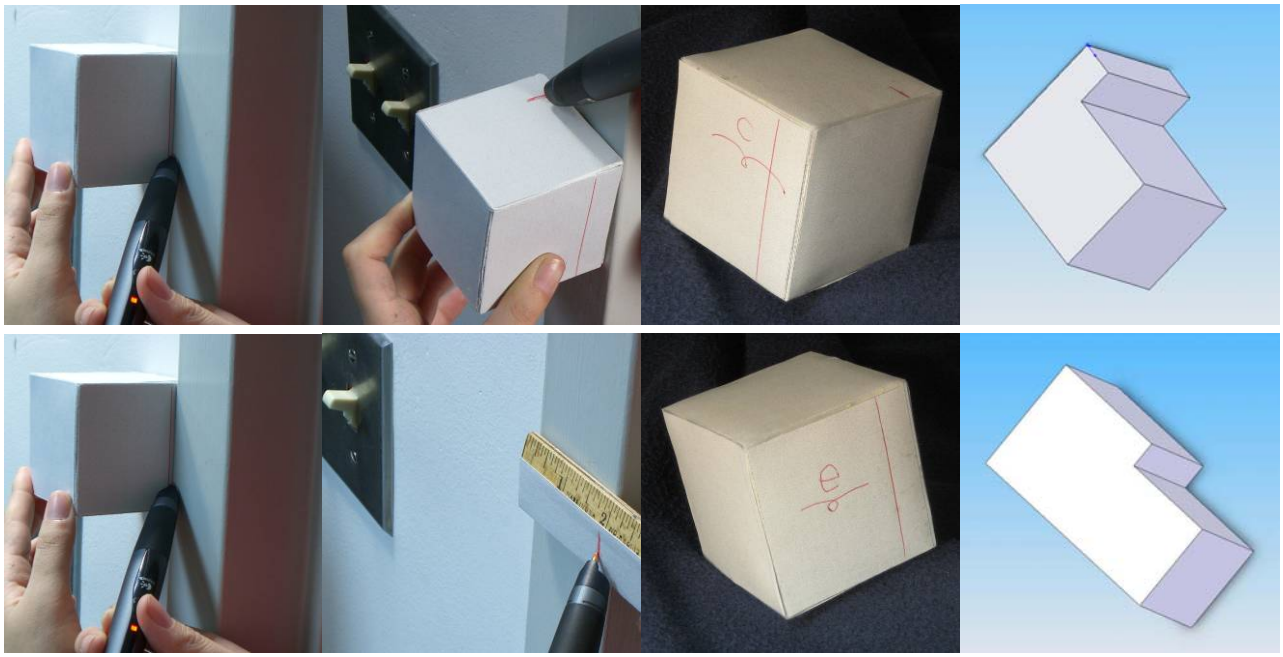

Figure 6 : Using external references to perform a command. **Top**: a cube is cut extruded to fit a door frame. First we mark the thickness of the frame, and then the width before executing. **Bottom:** a side of a cube is extruded to cover the same door frame. First we mark the thickness of the frame, and then use our ruler to mark the width of the frame before executing.

annotation process. Accordingly, when interpreting a command, we consider the pigtail as the baseline for the command name (Figure 5d).

As shown in Figure 4, some operations may require several sets of strokes. For example, to create a groove on an object (Figure 4d) one first draws the profile of the groove on one surface, then the extent of the groove on an adjacent surface, then one uses a pigtail to indicate the inner region, and finally one writes a *G* (for "groove") on top of the pigtail. Another example is the creation of a cut of a given depth. To do so, the user first creates the shape of the cut, then marks the depth of the cut on another face, and then uses a pigtail to issue the cut command. As shown in Figure 6, top, this syntax makes it very easy to use real world objects as references without the need for further measurements.

So far, we have considered cut operations, as they are easily specifiable by drawing on the available sides of the models. Our system also provides a way to create extrusions through the use of our "digital" ruler. For example, to extrude a shape from a surface, one simply draws the shape on the surface, then draws a mark on the ruler to indicate the extrusion length, and finally, using the pigtail, issues the extrude command (E) on the area to extrude. Figure 4e provides an example of such an operation. As in the case of cutouts, this command facilitates the process of using real-world objects as references (Figure 6 bottom).

Together the Cut, Groove and Extrude operations represent the fundamental modification tools (removing material and adding material) defined by D. K. Ching [5], these are also the interactions most frequently described by architects. But of course our system is easily extendable to more complex commands as we will describe later.

Dealing with Errors in Batch Processing. In our system, the annotations and commands are captured in batch mode. There are several reasons for this choice. First, as explained previously, it is important for the intended pattern of use of our system that interactions can take place away from a computer. Second, by delaying execution, a batch approach might help keeping users in the "flow" of their task by avoiding unnecessary interruptions.

Of course, errors will need to be corrected eventually. Our interface offers two main mechanisms to deal with errors. For marking errors in annotations and commands we use a simple scratch out gesture to indicate that the underlying gestures should be removed, or that the underlying command should not be performed. For execution errors, it is important to remember that while our system might misrecognize gestures and command names, it accurately captures the parameters of the commands on the correct faces. Since this information is directly transferred to SolidWorks, it becomes a trivial matter to make corrections because all the relevant command parameters are already in place which saves transcription time.

In our system it is also possible to issue several "alternative" commands by simply drawing the new command over the last command, a common pattern in practice. Each command will be recognized as a different operator (or "feature" in SolidWorks terminology) and appear in the feature tree managed by SolidWorks. Once the strokes have been transferred to SolidWorks, the user can compare the results of different commands, pick the best of them, and delete

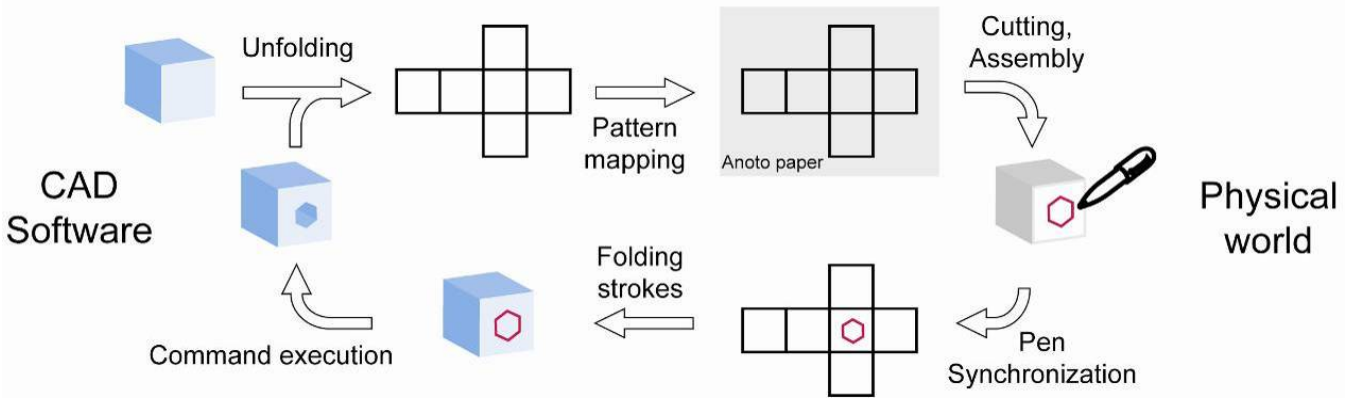

Figure 7 : Life cycle of a model using our system. Here we present the cycle for paper-based model construction, but a similar cycle would be used for applying water slide transfers onto existing 3D models.

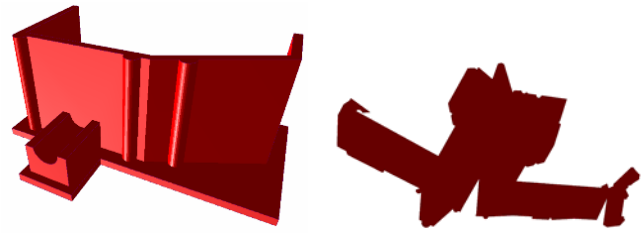

Figure 8 : Example of our unfolding algorithm.

alternative executions. Also, to document the design process, each of the alternative commands could be applied in a different configuration using the configuration management provided by SolidWorks.

IMPLEMENTATION

As shown in Figure 7, the life cycle of a model in our prototype can be broken down into 3 phases: 1) print the 3D model as a paper prototype with a unique pattern on each side; 2) capture the strokes made on the paper prototype and map the strokes onto the correct virtual 3D model's face; 3) execute the commands themselves. We are now considering each phase in turn.

Printing the Model

To print the paper model or water slide paper patch, we start from a SolidWorks file and unfold it as described below. The unfolded model is then printed on paper pre-printed with the Anoto pattern. During printing, we use the PADD infrastructure [7] to maintain the relationship between a given model and the unique page ID on which it has been printed, and to record the calibration data and the geometric transformation used during printing. This information is used during the synchronization process, to identify on which digital model a stroke has been made. Note that we are using the Anoto system as it is the only commercially available tracking system at this time, but other systems such Data Glyphs [11] could have been used as well.

Unfolding the Model. While many unfolding approaches are possible, such as [24] (see [27] for a review), we focused on limiting the number of discontinuities because they interfere with the tracking system. We use a heuristic approach which starts at the triangle with the longest perimeter, and transforms it onto a plane. Then it recursively visits all neighboring triangles and attempt to repeat the transformation, while maintaining adjacency of neighboring triangles and disallowing triangle overlap in the plane. Since the level of discontinuity is proportional to the length of seams, the recursion order follows a greedy heuristic of taking the next untransformed triangle sharing the longest boundary with transformed triangles that can be transformed without creating an overlap in the plane. When no such triangle is found, a new pattern is initiated and the process repeats. Once a set of patterns is generated, the patterns are collected and nested in a page by recursively packing pattern bounding boxes.

Importing Back Captured Strokes

During pen synchronization, our application receives all the strokes captured by a pen. Strokes are recorded with a time stamp and the page ID on which they were made. We use this ID to recover the model that the strokes were drawn on as well as the calibration and geometric transformation stored during the printing process. Once the user is ready to process the strokes, they are imported from our application onto the unfolded model; each stroke point is mapped from page coordinates back into 3D coordinates by applying the inverse of the transformation that was originally used to move the local triangle from 3D onto the plane.

Executing Commands

As pointed out above, all command strokes have been made by a special "command" pen, so it is easy for our system to distinguish them. Our first step in processing these strokes is to segment the stream of command strokes into individual commands. To do so, we first detect strokes that might look like a valid pigtail using a set of simple heuristics such as looking for gestures with a relatively small loop and large outside tails. Once these are detected, we observe if there is a stroke recognizable as a character that has been drawn above the candidate pigtail within a pre-set time out. If this is the case, the stroke is recognized as a valid pigtail, and the strokes drawn since the last command are used as parame-

ters for the command execution. We also check for natural command separators (such as creating an annotation) and check that the parameter set matches the command. For example, the face IDs associated with shape and pigtail delimiter and command character should all be the same. In practice, this approach worked well for our prototype. Once command syntax has been validated, the command is executed. Commands that use input from tools as parameters are processed in a similar way, but the strokes that were performed on the tool are processed in a tool-dependent way. If a command is not recognized, it is skipped, but its strokes are still presented on the surface of the model.

DISCUSSION

As we were developing our prototype we conducted several formative studies about the potential use of our approach. So far, we have conducted six semi-structured interviews, including a demonstration and a hands-on test. Our participant population covered a wide range of architectural backgrounds and included a student in an architecture school working as a drafter, several young architects, a senior architect, a senior partner and a faculty member at a school of architecture. Despite the current shortcomings of our prototype (such as the requirement that each contour parameter uses only one stroke, and the use of handwriting recognition without training) seasoned architects' reaction to the system was very positive. Several architects pointed out that our system would be perfect for massing a building. During massing, new models are built based on marks or shapes that were suggested in the previous iterative cycle. This type of practice is well suited for the ModelCraft interactions.

The professor remarked that our system would allow students to explore prototyping and develop 3D thinking skills. For example, visualizing the 3D results of subtractive operations drawn on a face of a cube is a common task in architecture training. ModelCraft might also create a natural bridge between the traditional approach to architecture (based mostly on paper-based sketching) and the use of modern applications such as SketchUp [1]. Architects further pointed out that annotations on paper models could be useful for capturing feedback from some of their clients who might be intimidated by digital models. The response to the system was more muted for younger participants (one a student drafter and one a CAD modeler), since their work did not require extensive use of tangible 3D models. Yet, the architecture student pointed out that the system would be very useful for teaching and would support current practice taught at school. The CAD modeler, while skilled in building models, was not using them at work. This participant also pointed out that she often "deconstructed" her models in order to reconfigure them, so annotations did not seem as useful for her. We are considering ways to support this type of approach with our system.

Several users were concerned about the limitations of the digital pen (mainly that one has to remember to aim the pen correctly). One user suggested that this problem could be alleviated by slightly modifying the design of the pen. Overall, our interviews confirmed our hypothesis that a system bridging the gap between the digital and physical worlds would be useful for practitioners and teachers alike.

Editing the Models

The design of our command language followed a different path than that of Teddy and Sketch. While those systems adopted a gesture-based approach well-suited for sketching, we used a structured approach based on a simple extendable command structure and a pigtail as a separator between parameter strokes and command selection [13, 18]. One of the strengths of our approach is that, while keeping an informal feel, it can be easily extended to more complex commands and a wider set of commands by using longer command names. We implemented a Window (W) command to create windows of a certain depth in buildings (Figure 1). Two additional commands supported by SolidWorks were implemented as part of our system: first, (S)hell, a command that creates a shell given a volume, and second, a (F)illet command used to round out a selected edge. Using techniques described in the PapierCraft system [18], we could also transfer a shape captured on transfer paper onto a given surface and extrude it. It would also be easy to extend the system to accept post command parameters like numerical arguments.

Another important difference to other systems is that in our system, there is not always a plane on which to draw. This limitation is not merely the result of our tracking technology. Even if more complex tracking systems were used, it would still be difficult for people to draw in free space. This makes several techniques (such as free form extrusion) that were used by Teddy more difficult to implement in the present system. However, as discussed above, we were able to address this problem through the use of simple tools such as rulers. We are also exploring how users could use sketches drawn on a drawing board to create new geometry.

Tracking Performance and Limitations

One of our goals during this project was to better understand the limitations of a tracking method based on a pattern printed on the model surface. We now discuss our observations derived from working with our prototype.

Printing Models. Our current prototype used only printed, paper-based models. For simple models, this approach worked extremely well. Cutting and scoring (to simplify folding) the models by hand proved to be easy and accurate. Using a laser cutter would greatly simplify and increase the accuracy of this process. As shown Figure 4 this approach works well with basic shapes such as a cube, cone, cylinder, pyramid, and tetrahedron. Using a more advanced unfolding algorithm like the one proposed by Mitani and Suzuki [24] would allow for more complex shapes. Yet, it is clear that the paper-based approach seriously limits the complexity of objects. One simple alternative is to add the pattern to existing models once they have been built. For example, using our system, one can print the unfolded surface on a water slide transfer paper, and apply the transfer onto the model. Our tests showed that this technique is a viable option for models printed with a ZCorp printer. This approach allows

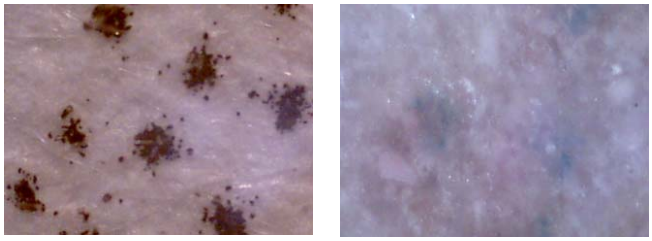

Figure 9 : Printing the Anoto pattern (all prints from a 600pdi rendering). **Left**: Anoto pattern printed using a 2400 dpi laser printer in black and white mode; **Right:** A pattern printed using a ZCorp Z510 printer (600 dpi). All pictures were taken at about x200 magnification.

for more complex shapes to be built rapidly and only adds a small time to the production process. Yet, for the system to stay accurate, one needs to be careful while applying the transfer.

Of course the preferred solution would be to have the 3D printer print the pattern at the same time as the 3D object itself. Some 3D printers (ZCorp Z510) can already print at a resolution up to 600 dpi in the plane of the printing bed and 540 dpi vertically [35]. This is in the same range as for laser printers able to reproduce the Anoto pattern. Unfortunately, our tests showed that a pattern printed with the ZCorp Z510 printer was not recognized by the digital pen. To understand why, we show in Figure 9 segments of patterns printed on a laser printer and on a Z510. As can be seen on Figure 9, left, the dots produced by our laser printer are of somewhat irregular shape but use black ink to provide a highly contrasted image. The dots printed on the Z510 (Figure 9, right) are diffuse and do not use true black ink but a combination of C, M, and Y inks to simulate black. As a result, they are likely invisible to the infrared pen sensor. We believe that this problem can be readily addressed by introducing a truly CMYK printing process and using finer grained printing material. Another solution would be to use another tracking system like the Data Glyph designed for 300 dpi printing on par with the minimum layer thickness of .089 mm (286 layers per inch) of the ZCorp process, or a more robust encoding scheme.

Accuracy. The Anoto tracking system reports points with 678 dpi accuracy, but, taking into account the errors introduced by pen orientation and the printing process, the system's maximum error is around 1 mm. Of course, the overall accuracy of the system also depends on the accuracy at which the paper is cut and folded (around 1 mm in our current manual process). Using a laser cutter would further improve accuracy.

Optical Tracking of Passive Patterns. Another problem inherent to optical tracking is that the system might lose tracking because the pen camera overhangs on a face or because users are trying to draw inside a groove or on an indented face. At overhangs, the pen looses track when the tip is about 3mm from the border, at which point it vibrates. As a result, the smallest square surface on which a command can be issued is 12mm wide. For indented faces the problem is exacerbated by the fact that the Anoto firmware is expecting a continuous pattern in the field of view. In our tests, the pen was able to track a pattern at the bottom of a 4.8 mm x 4.8 mm groove or mark a 6.4 mm diameter circle using a 1.6 mm thick template. Finally, because the pen was developed for tracking on flat surfaces, the system cannot track strokes on cylinders (or cones) whose radius of curvature is smaller than 12 mm. It is not clear how significant these limitations will be in practice and future work will be necessary to evaluate their impact. Other 3D encoding schemes may remediate this.

Another limitation of our tracking system is that it cannot track in free space. As demonstrated above, instrumentation of traditional tools used by wood workers (such as rulers, squares, tracing paper) may help to address this problem. For example, we used our instrumented ruler to indicate the height of an extrusion.

Finally, the current version of our digital pen does not provide orientation information for the object itself. So far, this limitation proved to be mainly relevant for handwriting recognition and our use of the pigtail as a reference mark addressed the problem successfully.

Character Recognition. Character recognition and pigtail recognition determine the total number of successfully recognized editing commands. Several problems might affect the recognition rate: first the pen provides samples at a relatively low temporal resolution which might influence the recognizer. To address this problem we add/subtract points so that the points are sampled not according to the time stamp but equidistantly. The orientation of the command might also have some effects. Our informal tests showed that using the pigtail as a baseline of character recognition was quite successful as the orientation of the characters seems to have little influence over the recognition rate. Finally, we observed that when users write on curved surfaces, letters are slightly deformed. This is due to users writing from a planar surface perspective. This problem can easily be addressed by projecting each letter on the plane normal to the surface at the centroid of the letter. Overall our tests show that our pigtail recognition rate is about 99% and, given our small dictionary of commands, we could reach a command recognition rate of about 92%. Further empirical evaluation will be needed to confirm these numbers.

FUTURE WORK

The system presented in this paper is built as an exploration tool allowing us to investigate the feasibility of our approach and provide us with a hands-on demonstration for potential users. In the near future we are planning to expand the system so that it can accommodate more complicated models and can be used during long term studies.

Dealing with Non-Developable Surfaces

Non-developable surface are problematic for our system because unfolding of such surfaces leads to multiple discontinuities in the pattern space (Figure 10) and creates gaps in tracking. Our tests suggest that the pen's field of view is

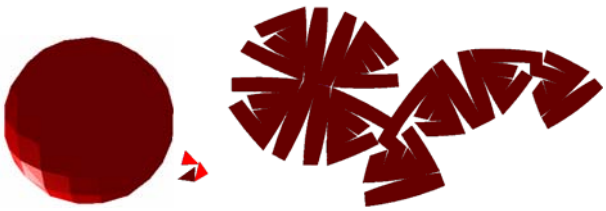

Figure 10 : An example of a non-developable surface creating many discontinuities in the pattern space.

about 5mm wide and that the current pen firmware only decodes the pattern correctly if there is only one continuous pattern in its field of view. A closer look at the design of the Anoto pattern [21] reveals that this is merely a limitation of the current implementation. In principle, one could uniquely resolve a position if any 2.4 mm x 2.4 mm patch is visible. We believe that if the firmware were modified to detect the edge of each continuous pattern region (maybe by recognizing printed edges) and each face of the model was wider than 2.4mm, the pen would be able to uniquely identify its position even around discontinuities in the pattern. Another solution to this problem would be to adopt a different approach to tracking altogether. Instead of mapping a 2D pattern onto our models, we could tile them with small (2-3mm) optical tags which can be tracked by the pen. For example, one could use the system proposed by Sekendur [28], or the Data Glyph system [25], or, of course, the Anoto position pattern itself. All of these provide the large number of unique identifiers that is necessary. In all cases, the requirement of the minimum patch size can be accomplished using subdivision-based techniques such as the one used in the Skin system [22] and extended by Igarashi and Hughes [15].

We would also like to examine in more detail how our system could be adapted to 3D printing systems. In particular, we would like explore the feasibility of a 3D version of the Anoto pattern. This would not only simplify the printing process and alleviate the pattern discontinuity problem but also allow for annotations on newly exposed, cut, or fractured surfaces of objects.

Extended Feature Set

Our interviews with architects pointed to several directions in which the current system could be extended. One of them is to provide a better support for “free space” sketching by using information sketched on free paper to be incorporated as parameters to commands. Another one is to provide an operation to “glue” objects together. This multiple object operation will be very useful in early design phases as it is often the case that architects create new designs by stacking or joining available building blocks. This will provide a functionality similar with Anderson et al.’s system [3].

Finally, while our current system focuses on batch processing, new Anoto pens can transmit the strokes they capture in near real-time. One of the appeals of our system is that it can be used without a nearby computer. Nevertheless, several applications might benefit from streaming capabilities (for example by combining our system with the Urp system [34]). It will be a simple matter to adapt our system to streaming-based interactions.

With these new functionalities in place we intend to conduct longer term usability studies to better understand how our system will be accepted and how it might change current design practices.

CONCLUSION

We presented a new system which lets users capture annotations and editing commands on physical 3D models and transfer them onto the corresponding digital models. Our system is inexpensive and easily scalable in term of objects, pens, and interaction volume. Our command system reflects current practices of model builders and integrates seamlessly with current practice. Our system allows users to bridge the gap between the digital and the physical worlds by allowing them to deploy resources of both media for the task at hand. We believe that our approach will provide an efficient tool for the early phases of design in both architecture and product design.

ACKNOWLEDGEMENTS

This work was supported in by NSF Grant IIS-0447703 and Microsoft Research (as part of the Microsoft Center for Interaction Design and Visualization at the University of Maryland) and a graduate fellowship from the department of Computer Science at the University of Maryland. We would like to thank the architectural and interiors firm of BeeryRio for their support during the interview process (with special thanks to Rosana Keleher), Irena Savakova of DMJM H&N and all our participants. Corinna Löckenhoff and Adam Bender provided many useful comments to help improve this document. We would also like to thank Ben Bederson, Bobby Bhattacharjee and Bill Pugh for their support. Foster & Partners kindly provided us with the picture shown in Figure 2 top. ZCorp kindly provided us with the picture shown in Figure 2 bottom.

REFERENCES

1. @Last Software, SketchUp. 2005.
2. Agrawala, M., A.C. Beers, and M. Levoy. 3D painting on scanned surfaces. *Proceedings of I3D'95*, pp. 145 - 150.
3. Anderson, D., J.S. Yedidia, J.L. Frankel, J. Marks, A. Agarwala, P. Beardsley, J.H.D. Leigh, K. Ryall, and E. Sullivan. Tangible interaction + graphical interpretation: a new approach to 3D modeling. *Proceedings of SigGraph'00*, pp. 393 - 402.
4. Anoto, Development Guide for Service Enabled by Anoto Functionality. 2002, Anoto.
5. Ching, F.D.K., *Architecture: Form, Space, and Order*. 2nd ed. 1996: Wiley.
6. Grasset, R., L. Boissieux, J.D. Gascuel, and D. Schmalstieg. Interactive mediated reality. *Proceedings of Pro-*

ceedings of the Sixth Australasian conference on User interface (2005), pp. 21 - 29.

7. Guimbretiere, F. Paper Augmented Digital Documents. *Proceedings of UIST'03*, pp. 51 - 60.
8. Hanrahan, P. and P. Haeberli. Direct WYSIWYG painting and texturing on 3D shapes. *Proceedings of SigGraph'90*, pp. 215 - 223.
9. Hart, S., Building a State-of-the Art Home: Part II. Achitectural Record Innovation, 2005: p. 24 - 29.
10. Hart, S., An Icon is Completed After 80 Years: Part I. Achitectural Record Innovation, 2005: p. 20 - 23.
11. Hecht, D.L. Embedded Data Glyph Technology for Hardcopy Digital Documents. *Proceedings of SPIE Color Hard Copy and Graphic Arts III*, pp. 341 - 352.
12. Hinckley, K., R. Pausch, J.C. Goble, and N.F. Kassell. Passive real-world interface props for neurosurgical visualization. *Proceedings of CHI'94*, pp. 452 - 458.
13. Hinckley, K., P. Baudisch, G. Ramos, and F. Guimbretiere. Design and analysis of delimiters for selection-action pen gesture phrases in scriboli. *Proceedings of CHI'05*, pp. 451 - 460.
14. Igarashi, T., S. Matsuoka, and H. Tanaka. Teddy: a sketching interface for 3D freeform design. *Proceedings of SigGraph'99*, pp. 409 - 416.
15. Igarashi, T. and J.F. Hughes. Smooth meshes for sketch-based freeform modeling. *Proceedings of I3D'03*, pp. 139 - 142.
16. Ishii, H. and B. Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. *Proceedings of CHI'97*, pp. 234-241.
17. Jung, T., M.D. Gross, and E.Y.-L. Do. Sketching annotations in a 3D web environment. *Proceedings of CHI'02 Extended Abstracts*, pp. 618 - 619.
18. Liao, C., F. Guimbretière, and K. Hinckley. PapierCraft: a command system for interactive paper. *Proceedings of UIST'05*, pp. 241 - 244.
19. Liu, X., Editing Digital Models Using Physical Materails, PhD thesis, University of Toronto. 2004
20. Logitech, IO digital pen. (http://www.logitech.com). 2005.
21. Lynggard, S. and M.P. Pettersson, Devices Method and Computer Program For Position Determination, in *US Patent Office*. 2005, Anoto AB: USA.
22. Markosian, L., J.M. Cohen, T. Crulli, and J. Hughes. Skin: a constructive approach to modeling free-form shapes. *Proceedings of SigGraph'99*, pp. 393 - 400.
23. Masry, M., D. Kang, and H. Lipson, A Pen-Based Freehand Sketching Interface for Progressive Construction of 3D Objects. Computer & Graphics, 2005. **29**: p. 563 - 575.
24. Mitani, J. and H. Suzuki, Making papercraft toys from meshes using strip-based approximate unfolding. ACM Transactions on Graphics, 2004. **23**(3): p. 259 - 263.
25. Petrie, G.W. and D.L. Hecht, Parallel propagating embedded binary sequences for characterizing objects in N-dimensional address space, in *US Patent Office*. 1999, Xerox Corporation.
26. Piper, B., C. Ratti, and H. Ishii. Illuminating clay: a 3-D tangible interface for landscape analysis. *Proceedings of CHI'02*, pp. 355 - 362.
27. Polthier, K., Imaging maths - Unfolding polyhedra. Plus Magazine, 2003. **27**.
28. Sekendur, O.F., Absolute Optical Position Determination, in *US Patent office*. 1998: USA.
29. Sheng, J., A Gestural 3D modeling Interface using Fingers and a Physical Prop Traceked in 3D, PhD thesis, University of Toronto. 2005
30. Solid Concepts Inc, SolidView. 2004.
31. SolidWorks, SolidWorks. 2005.
32. Ullmer, B. and H. Ishii. The metaDESK: models and prototypes for tangible user interfaces. *Proceedings of UIST'97*, pp. 223 - 232.
33. Underkoffler, J. and H. Ishii. Illuminating light: an optical design tool with a luminous-tangible interface. *Proceedings of CHI'98*, pp. 542-9.
34. Underkoffler, J. and H. Ishii. Urp: a luminous-tangible workbench for urban planning and design. *Proceedings of CHI'99*, pp. 386 - 393.
35. ZCorp, ZCorp 3D printing system. 2005.
36. Zeleznik, R. and T. Miller. Fluid inking: augmenting the medium of free-form inking with gestures. *Proceedings of GI'06*, pp. 155 - 162.
37. Zeleznik, R.C., K.P. Herndon, and K.P. Herndon. SKETCH: an interface for sketching 3D scenes. *Proceedings of SigGraph'96*, pp. 163 - 170.