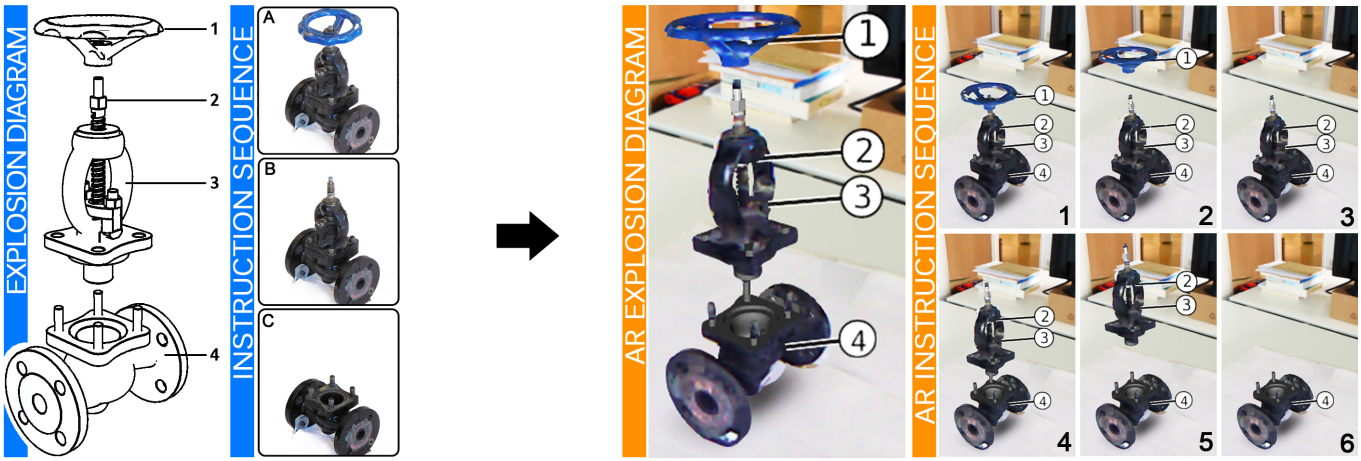
Retargeting Technical Documentation to Augmented Reality

Peter Mohr, Bernhard Kerbl, Michael Donoser, Dieter Schmalstieg, and Denis Kalkofen
Graz University of Technology
{ mohr | kerbl | donoser | schmalstieg | kalkofen }@icg.tugraz.at



(a) (c) (d)

Figure 1. Retargeted 2D documentation. (a) The input documentation consists of an annotated explosion diagram and a sequence of images presenting disassembly instructions. (b) From analyzing the 2D explosion diagram, our system is able to generate a 3D explosion diagram presented in AR. Moreover, our system generates 3D annotations in AR based on the input 2D documentation. (c) In addition, our system is able to analyze image sequences in order to create 3D animations from it. This allows to present animated 3D documentations in AR. Here we show six key-frames from the resulting AR animation.

ABSTRACT

We present a system which automatically transfers printed technical documentation, such as handbooks, to three-dimensional Augmented Reality. Our system identifies the most frequent forms of instructions found in printed documentation, such as image sequences, explosion diagrams, textual annotations and arrows indicating motion. The analysis of the printed documentation works automatically, with minimal user input. The system only requires the documentation itself and a CAD model or 3D scan of the object described in the documentation. The output is a fully interactive Augmented Reality application, presenting the information from the printed documentation in 3D, registered to the real object.

Author Keywords

Augmented reality; virtual reality; retargeting

ACM Classification Keywords

H.5.1 Information Interfaces and Presentation: Multimedia Information Systems-Artificial, augmented and virtual realities

CHI 2015, April 18 - 23, 2015, Seoul, Republic of Korea

ACM 978-1-4503-3145-6/15/04$15.00
http://dx.doi.org/10.1145/2702123.2702490

INTRODUCTION

Printed technical documentation is an essential requirement for understanding and using technical products and many other artifacts found in our daily life. We use handbooks to understand how things work and manuals to learn how to assemble or maintain them. Most documentation still exists on paper and relies on graphical illustrations and accompanying textual explanations to convey the relevant information to the reader. However, the use of printed manuals arguably introduces a cognitive seam, since users have to match the images in the manual with the physical object. In addition, inferring actions from a sequence of 2D images can be a mentally demanding task.

Augmented Reality (AR) overcomes this seam by presenting the documentation directly registered to the object in the 3D space surrounding the user. It has been shown that this can actually reduce the cognitive load when using technical documentation [8]. However, authoring AR documentation is a complex and time-consuming process. It requires skills with 3D modeling and animation tools and additional expertise with AR requirements such as registration and tracking. Meanwhile, a large amount of traditional documentation exists on paper (or in two-dimensional digital form), but remains unused for AR applications. To close this gap, we propose a system capable of automatically transferring traditional printed documentation to AR. We demonstrate our approach on several graphical elements commonly found in traditional documentations. Specifically, we present the transfer

of annotations labeling parts of the object, arrows indicating motions, explosion diagrams revealing an object's internal structure, and structural diagrams conveying the assembly or disassembly, translation and rotation of parts. Together, these illustrative elements cover the most frequent documentation styles.

We demonstrate that the conventions kept by illustrators allow to automatically interpret the images with only minimal help from the user. Thus, it becomes feasible to produce AR experiences from existing documentation at only a fraction of the effort it would take with conventional 3D modeling and animation software.

RELATED WORK

Documentations exhibit a variety of graphical elements, but usually follow established conventions. We identified the most frequent elements – in the following called diagrams – from books [18], online databases of popular products[1][2] and scientific publications [1, 7, 12].

A number of different strategies has been proposed to author Augmented Reality manuals. The two most common methods define behavior either directly by using commercially available modeling tools, such as Maya or 3DS MAX, or by scripting all actions and transitions between actions using specialized script languages [11, 3]. In both approaches, the user must define the manual from scratch. However, this requires excessive knowledge of both the modeling tool or the scripting language and the functionality of the object for which the manual was generated. In addition, manual authoring, either by programming or by using animation tools, is usually a very time consuming task.

An alternative approach is to build AR manuals from recorded videos [20]. This allows to author complicated manuals without additional knowledge about any programming language or animation tool. However, if the video is viewed from a different point of view than the one it was recorded from, it has to be warped to the new point of view, which is prone to rendering artifacts.

Zauner et al. [27] as well as Gupta et al. [6] proposed to generate instructions from 3D recordings of actions. However, both 2D and 3D recordings require the user to record all steps of an instruction, ignoring any existing documentation material. In contrast, our approach makes use of existing documentation material without any adjustment required. We achieve this by computing animations from 2D documentation material.

Our approach is inspired by the work of Li et al. [13], who generate an animated 2D explosion diagram from a single image. Their system requires the user to manually define all parts and all animations in the diagram. This is not practical for complex structures or multiple images in a sequence.

Recently, Shao et al. [24] presented a system which derives simple animations from interpreting 2D sketches. This reduces the effort required to generate animations between two images. However, this approach only works for simple structures. Since the user has to generate proxy geometry for every single part of the object, this system can handle only objects which consist of a small number of parts. Furthermore, the user has to re-create the 3D model for every image. Therefore, this approach does not scale easily to image sequences and large product databases.

Bergig et al. [2] present a system which is able to automatically generate 3D reconstructions of simple 2D sketches. Even though this system does not require any interaction to derive 3D geometry, it is limited by the capabilities of the sketch reconstruction approach. Their approach can handle only a few simple shapes. In contrast, we aim at the transfer of 2D documentations of complex structures, which may even be presented in multiple different configurations.

OVERVIEW

Our system generates interactive AR presentations from 2D documentation, given as a collection of images, and a 3D CAD model of the target object. The CAD model must be structured such that individually movable parts of the target object can be distinguished. If no CAD model exists, we use an RGB-D sensor (Microsoft Kinect) to obtain a 3D scan which can only be used for annotation transfer. However, oftentimes a 3D CAD model will already be available from the manufacturer of the target object, and the RGB-D sensor is only required for registering the CAD model with the real-world target object.

In the following, we outline our approach to analyze every major type of diagram. The result of our analysis is a 3D scene description including 3D animations representing instructions. Our results can be presented on a desktop PC using a Virtual Reality (VR) viewer or in the user's real world environment through an Augmented Reality display. The modules required to implement our approach are illustrated in Figure 3.

Annotated diagrams

A typical illustration is an annotated diagram, which allows to identify parts of an object by external labels, connected with leader lines to the referred parts (Figure 2(a)). The labels are often cross-referenced with more extensive textual descriptions.

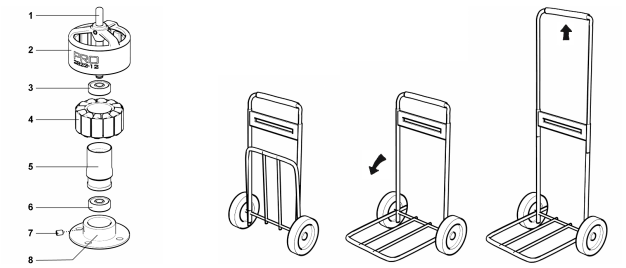
The first step for interpreting annotations (and any other diagram considered in this paper) is to determine camera parameters that were used to create the image, with respect to the coordinate systems used in the corresponding CAD model. Rendering the CAD model with the obtained parameters allows us to determine which part of the target object is covered by a given pixel in the image. Around the target object, we detect leader lines, identify the part of the target object from which the leader line originates and decode the text labels attached to the leader line.

Action diagrams

Action diagrams use auxiliary diagrammatic elements to present instructions within a single image. A common form uses arrows to encode the transformations which have to be applied to a part to perform the presented action (Figure 2(b)).

[1] http://service.lego.com/en-us/buildinginstructions

[2] http://www.ikea.com/ms/en_US/customer_service/assembly_instructions.html



(a) (b)

Figure 2. Diagrams used in traditional 2D documentations. (a) Explosion diagrams are commonly used to present the structure of an object. In addition, annotations identify parts. (b) Sequences of images are often used to represent an action. The images show the object in key poses. Occasionally, arrows are presented to demonstrate the necessary transformations of parts to perform the action. Images adapted from [18].

Like annotations, arrows are complementary graphical elements that cannot be derived from a CAD model. However, their special shape allows detecting them in the image and interpreting the intended motion and direction. The intention of the arrow can be interpreted by comparing its pointing direction to valid displacement directions of parts nearby.

Explosion diagrams

Another popular form of technical reference is the explosion diagram, which reveals the internal structure of an object in a single image. Explosions reposition each part of the object along one or more explosion axes. The position of parts is determined based on the structure of the assembly, i. e., parts are arranged on each explosion axis according to the order in which they can be removed (Figure 2(a)). The resulting diagram avoids occlusions and simultaneously encodes blocking relationships between the individual parts.

Detecting which parts are shown in which displaced position is an essential problem considered in this paper. To retarget explosion diagrams, we use disassembly planning [26] to determine the order in which parts may be removed and generate valid displacements for candidate parts. The candidate parts are then rendered with these displacements, and the resulting image is compared to the input image using robust image matching techniques.

Structural diagrams

Complex instructions are most commonly represented by image sequences, where each image represents one step of the procedure [7]. The most basic form is the structural diagram, where each consecutive image adds, removes or reconfigures one or multiple parts. Since each image portrays the object at a single point in time only, the procedure has to be interpreted by comparing one image to the next and identifying their differences. Note that structural diagrams change the structure of the object from one image to next, while explosion diagrams offset parts to present the structure of an object.

To retarget structural diagrams, we must thus determine which parts are added, removed, translated or rotated in each consecutive image. By using motion planning, we obtain a set of candidate configurations of the CAD model, which we can use to search for the one depicted in the diagram. To reduce the search space, we can use robust image differencing to determine the area of change. For each image pair, we use the area of change to discard motions occurring outside of these regions.

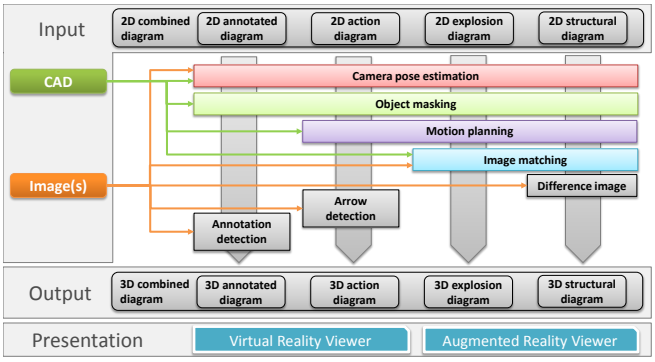


Figure 3. System Overview. Given an input image of a 2D diagram and a 3D CAD model of the object of interest, our system is able to generate interactive 3D diagrams, which it can display in an Augmented Reality environment. Our system consists of several modules. This overview shows which of these modules are used to retarget a specific type of input diagram. Once a 3D diagram is created, we present it either in Virtual Reality or within the user's real world environment by using an AR display.

Combined Diagrams

Documentation often combines multiple types of diagrams. For example, labels may be used in an image sequence to name the relevant parts of the presented instruction (Figure 2(a)). Also, partially exploded objects or sequences of action diagrams (Figure 2(b)) are commonly used in image sequences to present both an action and the object configuration after applying it to the object. Our system is able to analyze and retarget all these combinations.

Augmented Reality interface

The analyzed diagrams are transformed into 3D overlays and presented in AR, registered to the physical object. The user can interactively step through the instruction sequence and explore the individual elements of the documentation.

ANNOTATED DIAGRAMS

We transfer 2D image data to 3D space by projecting a 3D model of the object of interest to 2D image space. If this rendering of the object fits to its input image, we can relate image elements to 3D structure. We segment all labels and their corresponding leader lines in image space. Subsequently, we generate 3D annotations relative to the 3D model by using the relations we found between image elements and 3D structures. This process is illustrated in Figure 4.

Estimating camera parameters

The 3D model needs to be rendered using the same camera parameters as the input image. Since the original camera is unknown, we need to approximate the intrinsic and extrinsic parameters. In principle, these parameters can be found automatically [28]. In practice, automatic extraction of reliable 2D-to-3D point correspondences from a single, highly stylized image is not robust in many cases. Thus, we opted for a

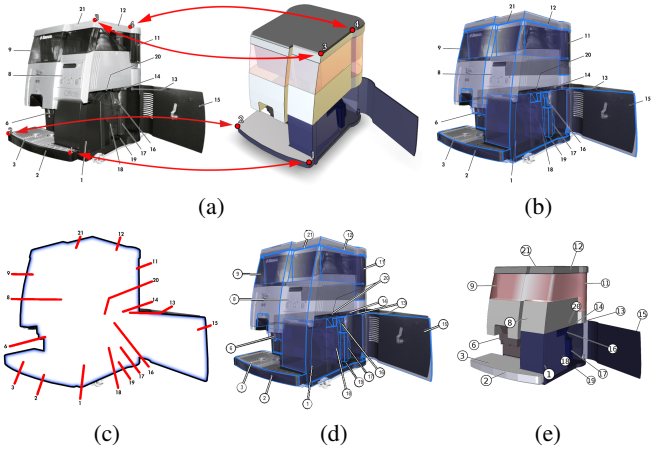


Figure 4. Retargeting annotations. (a) To estimate the camera pose, the user interactively defines corresponding points between the 3D model and the input image. (b) With the estimated camera pose, the rendering of the 3D model fits the 2D input image. This is illustrated by adding the rendering as semi-transparent blue layer on top of the input image. (c) Searching for MSER with an eccentricity close to 1 allows to identify leader lines. The mask defined by the rendering enables to detect their starting and ending points. (d) Running an OCR tool around the end point decodes text labels (e), which allows to add 3D annotations to the 3D model.

manual definition of point correspondences and field of view by providing a basic user interface (UI). The UI is a side-by-side arrangement of a 2D image viewer and a 3D viewer, wherein the user selects four or more points both in the input image and on the freely rotatable 3D model (Figure 4a). The camera pose is then determined using POSIT [5]. This procedure needs to be done only once for a set of instruction images that use the same camera parameters, which covers the majority of examples we have analyzed.

Annotation detection

By computing maximally stable extremal regions (MSER) [17], we search the 2D documentation for leader lines and labels. MSER provides an efficient means for segmenting connected components in an image. For every connected component, we calculate its eccentricity. If the eccentricity value is close to 1, we assume that the region contains a leader line. We then determine the end points of the line by fitting an ellipse to the region. Rendering the CAD model using the camera parameters estimated in the previous step yields a mask for looking up whether a pixel is occupied by the model. The pixel coincident with the end point inside the mask is used to look up the annotated part and defines the 3D anchor point, where the leader line is attached to the 3D object. The region around the other end point is automatically scanned with a standard optical character recognition (OCR) tool to decode a text label.

ACTION DIAGRAMS

Action diagrams encode instructions within a single image, mostly by using arrows. Therefore, to retarget action diagrams, we need to identify and interpret arrows and the parts of the target object they are referring to. We start by detecting

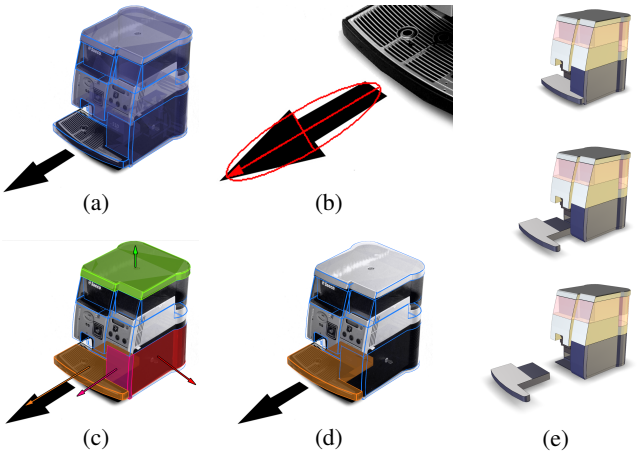


Figure 5. Retargeting action diagrams. (a) The input image with a semi-transparent overlay of the 3D CAD model. (b) We detect the arrow and its direction in 2D image space. (c) We analyze the 3D CAD model to find all removable parts and their directions. (d) By comparing the assembly directions to the direction of the arrow, we reduce the set of possible part motions. The part which is closest to the shaft of the arrow is selected. (e) The animation is defined by the path which was derived during motion planing.

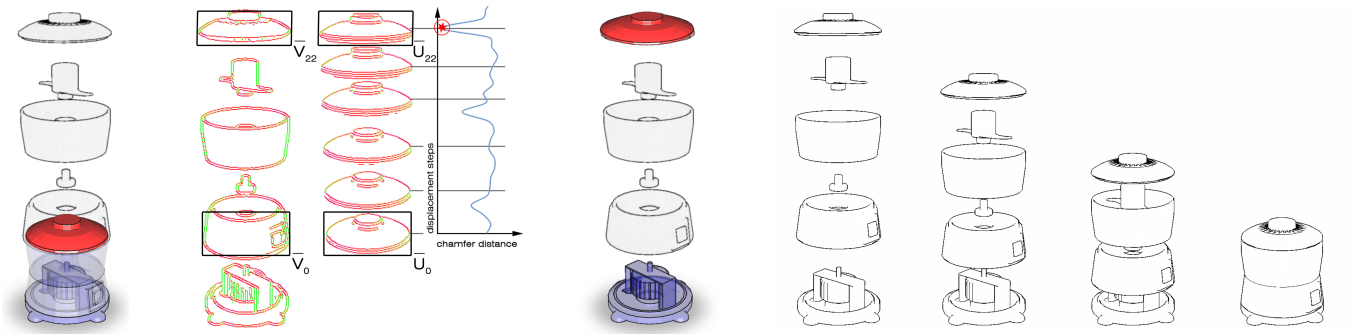
and analyzing each arrow in 2D space (Figure 5(b)). Subsequently, we use motion planning to identify the set of geometrically feasible motions of each part (Figure 5(c)). We then identify a set of parts the arrow might refer to by selecting those which can be moved similarly to the motion encoded in the arrow. To present the action in 3D, we animate the selected part using the corresponding movement, as suggested by the motion planner (Figure 5(e)).

Motion planning

We assume that input images show only geometrically feasible configurations, which have been generated by moving parts without penetrating other parts of the object. To find such valid motions for each part in the CAD model, we need to test whether or not a motion causes collisions. Using Minkowski differences of polyhedra [16], we detect for a given direction of motion how far each part can be displaced. By default, we test for the principal axes in local and global coordinates of the part.

Arrow interpretation

In order to identify an arrow, we begin by computing MSER regions outside the target object. Among all candidates with connected components, we select the ones with exactly two concavities along the boundary, and we determine the main axis by ellipse fitting (Figure 5(b)). The tip of the arrow is the extremal point on this axis closer to the concavities. To identify candidate actions, we search for parts which could be removed in the direction indicated by the arrow. Therefore, we project the motion vectors that have been computed during motion planning to image space using the camera parameters we have estimated before (Figure 5(c)). We compare the projected directions to the direction of the arrow. We select part motions as candidates if the angle between their projected motion vector and the arrow vector is below a given threshold. If there is more than one candidate, the part which is closest to the shaft of the arrow is chosen (Figure 5(d)).



(a) (b) (c) (d)

Figure 6. Retargeting explosion diagrams. (a) The 2D input image and the object rendered semi-transparently using the estimated camera parameters. The motion planner generates candidate places for the lid (red part) first. (b) Our matching compares renderings of the lid within its bounding rectangle $\bar{U}$ at geometrically possible locations $\bar{V}$. The orientation of the local gradient, as used during image matching, has been color coded from red (horizontal) to green (vertical). The chamfer distance between $\bar{V}_i$ and $\bar{U}_i$ has been plotted in the line chart. The red star in the line chart highlights the minimal chamfer distance $\bar{U}_{22} \rightarrow \bar{V}_{22}$. (c) The lid is placed corresponding to the analysis. (d) To replicate the entire 2D explosion diagram, we search for the optimal placement of each part. We generate animations by consecutively applying the motions that were identified by our candidate evaluation algorithm. This example shows output key frames of a generated 2D animation for collapsing the input explosion diagram. Note that occlusions are correctly resolved with the 3D model.

EXPLOSION DIAGRAMS

In order to retarget explosion diagrams, we need to find a sequence of unblocked motions of parts such that the resulting 3D scene setup visually matches the input 2D explosion diagram. The problem of detecting such sequences has been considered in assembly planning [1, 26, 21]. Thus, we incorporate our motion planning system into a sequencing approach similar to Srinivasan et al. [25]. This generates a set of feasible candidate explosions in 3D space. We find the candidate that matches the input image best by comparing the it with rendered images of each candidate setup (Figure 6).

By using only motions which we identified as geometrically feasible, we generate candidate displacements of each part in discrete locations. The CAD model is rendered to a texture with its parts displaced according to feasible motions. The resulting image U is compared to the input image V using a variation of directional chamfer matching (DCM), as given in Equation 1 [15]. To emphasize relevant edge features of the CAD model for DCM, we assign a random color to each part and add contour lines to the scene before rendering. We then apply a Canny edge detector [4] and store the normalized vector of the locally detected 2D gradient in $\hat{U}(\mathbf{u})$ and $\hat{V}(\mathbf{v})$, respectively. Our notation employs the zero norm $||\cdot||_0$ to count the non-zero pixel values in an image. With a small constant ϵ, we obtain a continuous distance function d_{DCM} for comparing images U and V:

$$d_{DCM}(U,V) = \frac{1}{||\hat{U}||_0} \sum_{\mathbf{u_i} \in U} \underset{\mathbf{v_j} \in V}{min} \frac{|\mathbf{v_j} - \mathbf{u_i}|}{|(\hat{U}(\mathbf{u_i}) \cdot \hat{V}(\mathbf{v_j}))| + \epsilon} \quad (1)$$

Reducing the size of the test area

Calculating the DCM over the entire image for each candidate explosion is computationally demanding. However, if a part is visible in the explosion diagram, we search specifically for this part at valid candidate locations. We generate a new image template $\bar{U} \subset U$ for each candidate motion by projecting the displaced 3D part into 2D image space and calculating its bounding rectangle. The search space in V is restricted to the matching bounding coordinates $\bar{V} \subset V$. We refer to the procedure of finding the place of the template $\bar{U}$ in V as local template fitting (see Figure 6(b)).

Reducing number of tests

While restricting the image matching to the bounding rectangles of displaced parts helps to increase system performance, the amount of possible combinations of part displacement easily becomes very large. Even with the restriction to geometrically feasible motions, the search space grows quickly with the number of parts and candidate displacements. Thus, we accelerate the system with a greedy search approach. Instead of generating all possible configurations in advance, we combine candidate generation and evaluation to progressively find the best fit for a given input image. In each iteration, each movable part is offset according to its geometrically feasible candidate motions without modifying the others. The displacement that produced the smallest local distance value according to $d_{DCM}(\bar{U}, \bar{V})$ is applied to the corresponding part. The search continues as long as at least one tested part displacement yields a distance that is significantly smaller than the currently stored minimum. If multiple choices are available, the motion that introduces the largest improvement is chosen and applied. If no such motion exists, the algorithm terminates. Note that exploding a part may unblock others, which results in additional feasible motions of the formerly blocked parts. Thus, blocking information of the remaining parts needs to be continuously updated.

STRUCTURAL DIAGRAMS

Structural diagrams appear in 2D documentations as sequences of at least two images. Every two consecutive input images differ with regard to changes in configuration that are applied to one or multiple parts. Given a pair of input images, we can restrict candidate motions to those which fall in the image region where visual changes are observed. This allows us to reject candidate motion based on where parts have

been moved, rather than on their matching score with the target image. Early rejection of candidate motion based on the region of change is essential to increase system performance. Therefore, we must ensure reliable identification of these regions.

Difference images

In order to evaluate the relevance of a candidate motion with regard to the visible region of change, we need to compute the difference image $\texttt{diff}(U_1, U_2)$ of the rendering of the object before and after applying the candidate motion, and the difference between the two consecutive input images, $\texttt{diff}(V_1, V_2)$. Computing the percentage of overlap of the two regions (Equation 2) allows us to discard motions that are not related to the observed differences in the input images, i.e., where the coverage falls below a given threshold.

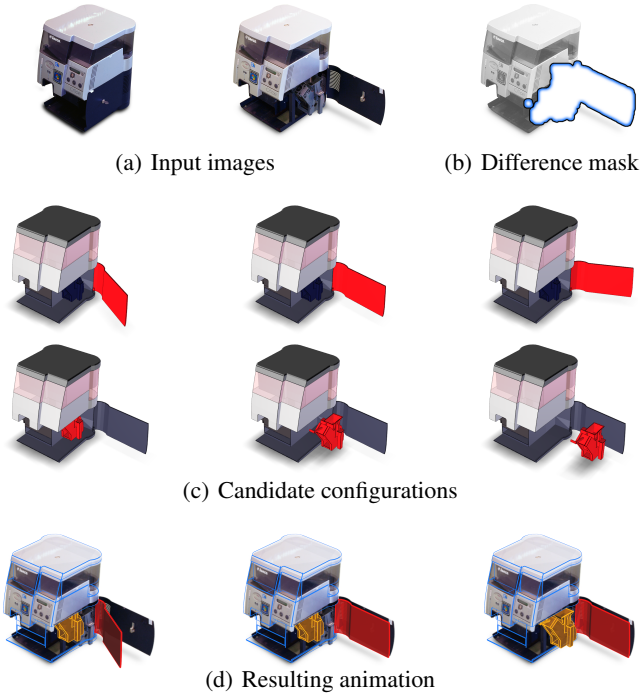
$$coverage = \frac{||\texttt{diff}(V_1, V_2) \cap \texttt{diff}(U_1, U_2)||_0}{||\texttt{diff}(V_1, V_2)||_0} \quad (2)$$

We generate $\texttt{diff}(U_1, U_2)$ by rendering all parts in the object except for the current candidate to the depth buffer in the first pass. In a second pass, we enable the color buffer and render the part before and after applying candidate motion. The resulting pixels in the color buffer correspond to the region of change $\texttt{diff}(U_1, U_2)$ (Figure 7(b)). The method used for obtaining $\texttt{diff}(V_1, V_2)$ differs based on the type of input images. Difference images from input with clear distinct colors, such as LEGO assembly manuals, can be directly computed by subtraction and thresholding. However, a reasonable amount of assembly instructions are simple black-and-white closed-line drawings, as known from do-it-yourself furniture [3]. For such line drawings, we apply a silhouette check to identify the background area in both images and reject motions that exhibit a low foreground coverage. A flood-fill from the four image corners (which are assumed to be empty) creates enhanced versions of the input images, where background and foreground pixels have a unique color. By conjoining the foreground regions of V_1 and V_2, we receive a generous enclosure for the area of visible change. Change detection in photographic material is most challenging. Simple image differencing is defeated by lighting, reflection, shadows and surface texture. For robust determination of the region of change, we employ SIFT flow [14], which densely relates pixels from the source image to the most similar pixel in the destination image.

Global bi-directional chamfer distance matching

Local template fitting works well for detecting detached and isolated parts, as they appear in explosion diagrams. However, this approach performs poorly when trying to verify subtle structural changes, which include reconfiguring, removing or attaching parts. The possibility to compare between images in a sequence obviates the necessity to depict each action individually. Involved parts may be partially obscured or simply disappear from one image to the next, thus making local fitting inapplicable. Consequently, a more thorough

[3] www.ikea.com



(a) Input images (b) Difference mask

(c) Candidate configurations

(d) Resulting animation

Figure 7. Retargeting structural diagrams. (a) Input images. (b) We generate a difference image, which reveals changes between the images. Our system creates a set candidate configurations by moving and rotating all parts of the machine which are identified by the difference image. (d) From all candidate configurations, the one which fits the target image best is selected. The motion which had to be applied to transform the object in the selected configuration is used to create the 3D animation. In this case, a rotation is generated to open the service door, and the brewing unit is removed.

analysis of the entire image domain is required. For a full analysis and comparison of two images U and V, we compute the DCM over the entire image area in both direction $U \rightarrow V$ and $V \rightarrow U$. We refer to this global image matching of U and V as the bi-directional chamfer distance (BCM):

$$d_{BCM}(U, V) = d_{DCM}(U, V) + d_{DCM}(V, U) \quad (3)$$

Finding a candidate rendering U that minimizes $d_{BCM}(U, V)$ ensures a strong visual similarity between U and target image V. For a scene with n parts and up to c feasible configurations per part, evaluating all possible setups requires $\mathcal{O}(c^n)$ comparisons. Since this quickly becomes infeasible even for moderately complex objects, we use the greedy algorithm mentioned before to progressively select the best structural improvement based on individual part motions. However, greedy global matching may at early stages favor part configurations that approximately resemble multiple unmatched parts in the target image. To suppress this effect, we put an additional emphasis on local consistency to penalize discrepancies in the subimage regions $\bar{U}$ and $\bar{V}$ enclosing the projection of the currently tested part. Thus, the distance function uses a weighted bidirectional chamfer matching (WBC):

$$d_{WBC}(U, V) = \underbrace{d_{BCM}(U, V)}_{\text{global matching}} + \underbrace{\lambda\, d_{DCM}(\bar{U}, \bar{V})}_{\text{local fitting}} \quad (4)$$

The additional weighting of local consistency for $\bar{U}$ and $\bar{V}$ is controlled by the parameter λ. In practice, we found that choosing $\lambda \approx 0.25$ yields satisfactory results for all examined sequences.

Candidate generation

Similar to explosion diagrams, we generate candidate configurations by iteratively moving and testing parts based on geometrically feasible motions. In addition, we allow to move parts outside the visible area in order to handle instructions which require to append or to remove parts. The instructions shown in Figure 7(a) present multiple consecutive actions. To access the brewing unit from the coffee machine, the hatch has to be opened before the unit can be removed. Since we iteratively change candidate configurations, our system succeeds in correctly recreating the depicted motions. This example demonstrates that our algorithm is robust to hidden or incomplete input data. Although the brewing unit is completely obscured by the closed hatch, its removal was successfully detected by our system (Figure 7(d)).

Since structural diagrams may also encode actions requiring to rotate a part, we additionally generate candidates that involve part rotations. Candidate rotation axes are generated by combining principal axes with the origin of either the local coordinate system or the center of gravity of a part. The testing of rotations starts with the part in default configuration, and proceeds by increasing the angle of rotation in discreet steps. Testing rotations around an axis terminates after reaching 360° or if a collision with another part occurs. For example, to analyze the input images in Figure 7, our system generates and evaluates candidate configurations by rotating the service door of the coffee machine around its hinge.

COMBINED DIAGRAMS

Traditional 2D documentation often consists of diagrams which combine aspects of annotated, structural, explosion and action diagrams. A typical example is to use annotations in any of the other diagrams (Figure 2(a)). Another very common combination is to show one or multiple parts in displaced positions (explosion diagram), before showing their final position on the object in the next image of a sequence (structural diagram). Displaced parts may further be highlighted with diagrammatic elements such as arrows or leader lines. Consider the example in Figure 8(a), which shows two images in a sequence of a LEGO assembly manual. Instructions to add bricks are given by showing the brick and leader line connecting the brick to the intended location on the object (as used in action diagrams). In addition, some bricks just appear in the second image.

The system we present is able to handle all kinds of combinations. Interpreting the mixed sequence displayed in Figure 8(a) can be achieved by employing the approach that is used for analyzing structural diagrams. Since WBC considers both global and local features, it is applicable to partial explosions as well. The additional visual information that is conveyed by arrows in sequences can be processed by the established methods for arrow diagrams and may be used to complement candidate selection. However, we found that for all assessed sequences, our algorithm for retargeting structural diagrams was able to reproduce the portrayed instructions without interpreting the arrows. Moreover, since we approximate any displayed object configurations by accumulating several part motions, our system is able to retarget image pairs which convey multiple instructions at once. This includes manipulations of multiple parts between two images as well as multiple consecutive actions. For example, Figure 8(a) shows two images taken from a LEGO instruction which adds six bricks from one image to the next. Since our system correctly identifies the configurations in both images it is able to generate the correct motion to animation.

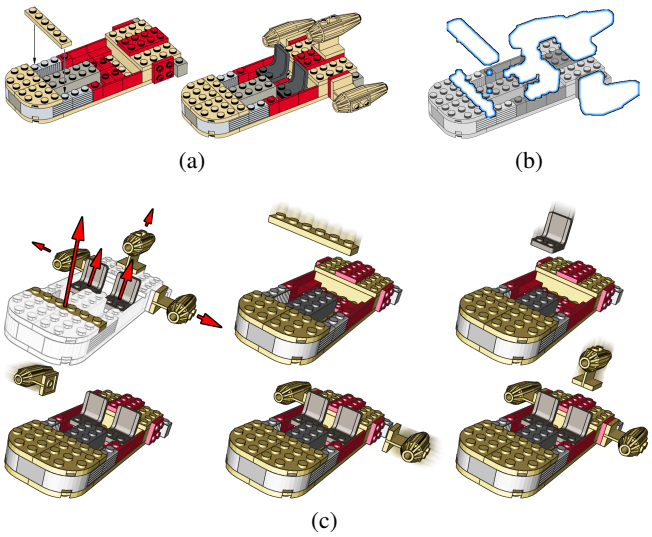
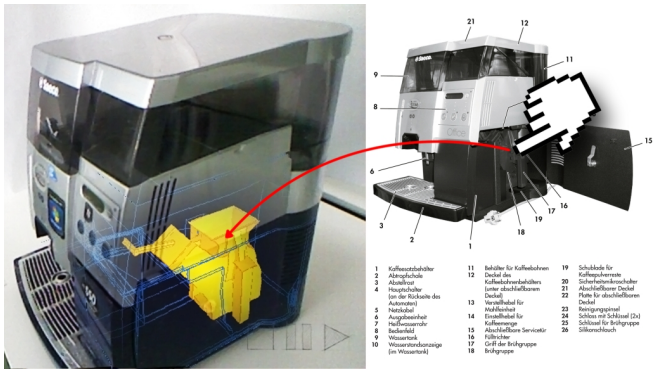


Figure 8. Retargeting combined diagrams. (a) Two input images illustrate multiple assembly steps. (b) We derive a difference image from the inputs, which indicates where changes occurred. (c) Our analysis yields a set of motions and corresponding parts, in this example indicated by red arrows. The key frames show the animation generated from the computed motions. ©2014 The LEGO Group, used with permission.

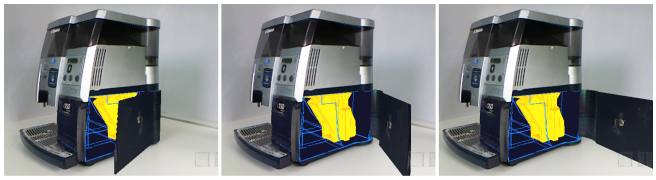
Our approach to generate 3D annotations is independent of other elements present in the diagram. Therefore, we can handle annotations, which appear in any of the other diagrams. Only the current configuration of the parts depicted in the image is required to render a mask which matches the input image. Since DCM is robust against image noise and smaller artifacts, additional graphical elements do not influence the candidate matching. Figure 1 shows an example of an annotated explosion diagram, which we successfully analyzed before retargeting to AR.

AUGMENTED REALITY INTERFACE

Our analysis of 2D documentations reveals any combination of an annotated 3D model and a set of consecutive 3D motions. Our final goal is to easily relate the information to real world objects using an AR interface. Therefore, we register the 3D model to its real world counterpart and track the user's display to make the augmentation visible. Since we present the documentation in a dynamic and often visually complex environment, we also provide interactive visualization tools as part of the AR interface.



(a)



(b)

(c)

Figure 9. Interaction. (a) The relation between 2D image elements and 3D structure allows us to manipulate the 3D visualization with 2D interactions. Selection of the brewing unit triggers highlighting of the corresponding 3D object. If the selected object in the real world is occluded, we provide the user with a ghosted view x-ray visualization. (b) We allow to replicate an optimized object overview configuration in a 2D image by computing the steps for transforming the real world object into the one depicted in the 2D documentation (e.g., opening the service door). (c) The optimized object configuration is most effective from the point of view used in the 2D documentation. We extract the point of view from which the 2D image was generated and present it to the user by showing an avatar. This example shows the user navigating to the extracted point of view.

Registration and tracking

The registration of the 3D model to its physical counterpart uses an RGB-D sensor and PCL-Kinfu, which is an implementation of KinectFusion [19] using the Point Cloud Library (PCL) [23]. Our system registers the 3D model automatically to the point cloud received from the RGB-D sensor using an implementation of Sample Consensus Initial Alignment (SAC-IA) [22]. Camera tracking is realized using PCL-Kinfu and initialized with SAC-IA.

Visualization

The registered 3D model allows to present the 2D documentation within the real world environment of the user. However, in AR, we usually have to cope with a visually complex background and dynamic camera motion. Therefore, AR documentations demand special techniques to provide effective visualizations. To reduce the visual complexity of our AR visualizations, we allow animations to be presented as video phantoms [9]. This resembles a real-time photomontage (Figure 1, Figure 9(b)) and thus reduces the visual complexity by decreasing the amount of virtual objects in AR.

Interaction

Traditional diagrams have been optimized to present their information from a carefully chosen vantage point. However, interactive diagrams can become confusing, if many parts are animated, or the user chooses a poor point of view. Thus, we let the user interactively control the progress of an animation by influencing the displacement of all or individual parts at runtime. Furthermore, the relation between 2D image data and 3D objects allows us to connect interactions in image space to 3D visualizations in AR. For example, we allow to quickly find 3D parts in AR by interactively selecting 2D elements in image space. Consider the example in Figure 9(a). The 2D illustration is taken from a traditional handbook. It is showing the object of interest in a configuration, which was optimized for visibility. Note the open door, which reveals the parts inside the machine. While optimized 2D graphics support selection tasks, our system adds support to identify the selected elements within the real world environment. Thus, selecting an element in image space activates highlighting of the related 3D structure in AR. If the selected element is hidden in the real world, we provide the user with ghosted view visualization [10] (Figure 9(a)). Notice the transparency modulation, which creates the impression of seeing through the closed real world door.

While ghosted views enable effective exploration of hidden objects, they fail to create effective visualizations for visually complex structures. However, since our 2D input graphics provide optimized visualization, we furthermore provide the user with a replicate of the 2D configuration in 3D AR. We implement this by computing the object configuration shown in the optimized 2D image. To provide a smooth animation from the object configuration in reality to the one depicted in 2D, we furthermore derive the configuration of the real world object in front of the user. Therefore, we run our diagram analysis using a screenshot of the live video and the optimized 2D graphics. Figure 9(b) shows the resulting animation from using the images seen in Figure 9(a) as input. The selected brewing unit is clearly shown after replicating the object configuration used in the traditional documentation material.

Replicating the object configuration from an optimized 2D illustration works best from a point of view which is close to the one used in 2D. During camera estimation, we derive this point in space, and, thus, we can guide the user to these optimized points of view. As demonstrated in Figure 9(c) we provide this information to the user by adding an avatar looking at the object to the scene. The avatar indicates the point of view the image was taken from, and thus for which the configuration of the object is optimized.

PERFORMANCE

To evaluate the performance of our system, we choose a set of objects, which have been illustrated in different styles. The results are listed in Table 1. The table shows the number of parts the object consist of (#Parts), the number of images in the sequence (#Images) and the time used to retarget the input

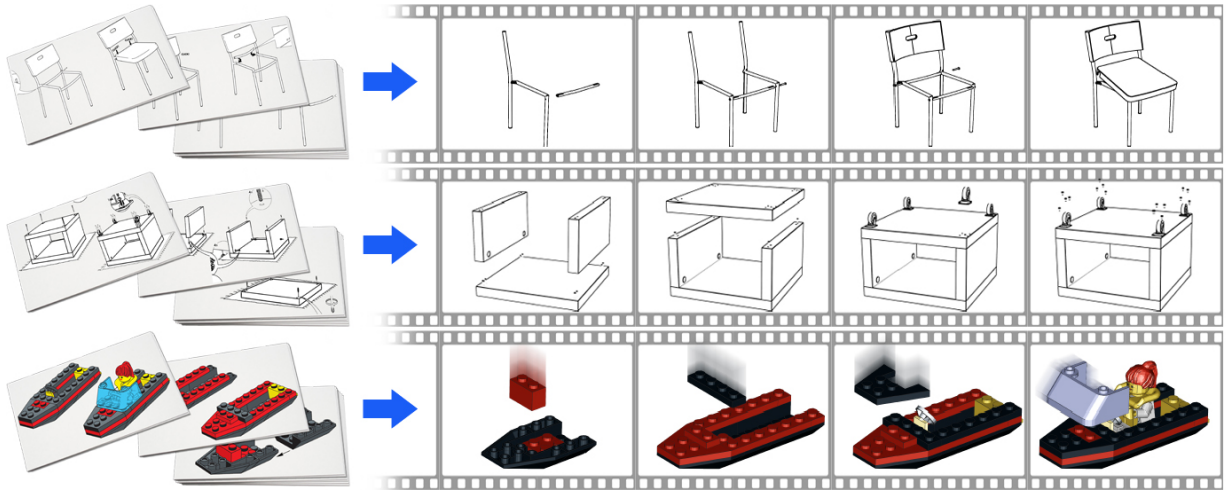


Figure 10. Various examples. ©IKEA Chair Herman and Table Lack, used with permission. LEGO Boat, ©2014 The LEGO Group, used with permission.

Table 1. Assessed test cases. We list the number of parts involved, as well as the length of the input documentation material and runtimes for retargeting.

Testcase	#Parts	#Images	Runtime
Valve (Fig. 1)	4	3	8s
Coffee Machine (Fig. 4)	6	4	9s
Mixer (Fig. 6)	6	*1	45s
LEGO Boat (Fig. 10)	18	8	1m 27s
LEGO Landspeeder (Fig. 8)	39	14	4m 10s
IKEA Chair Herman (Fig. 10)	10	8	5m 58s
IKEA Table Lack (Fig. 10)	42	5	16m 29s
LEGO Tower	42	7	2m 21s
LEGO House	88	13	10m 25s

* Explosion diagram, single image only

data to a 3D diagram (Runtime). The runtimes were recorded on a personal computer with an i7-4771 CPU @ 3.50 Ghz, 16 GB RAM and an NVidia GeForce GTX 780 Ti with 3 GB graphics memory. Input diagrams and resulting animations for test cases that are not shown elsewhere are depicted in Figure 10 and in the supplementary video. Although the LEGO Landspeeder and the IKEA Table Lack have a similar part count, the manual of the latter takes significantly longer to retarget. This results from the different illustration styles used and the number of parts involved in each step of the manuals. Photos and renderings allow us to apply image differencing to obtain tight areas of change for early rejection of unrelated candidate motions. However, this is not possible for line drawings. Furthermore, the manual of the IKEA Table depicts a larger number of instructions in each individual diagram. This is possible, because many parts of the object have no sequential dependency (e.g., wheels and screws can be added in parallel). Consequently, a high number of movable parts has to be considered at each point in time. The performance of our approach thus depends strongly on the employed illustrating style and the overall structure of the examined object.

LIMITATIONS

Our current implementation has several limitations. The pose estimation is the basis for the data extraction in our approach. Therefore, an erroneous estimation will propagate through the entire pipeline. If the error is exceeds a certain threshold, the system will fail to associate image regions with 3D objects. In addition, invisible elements in an image cannot be extracted with our current implementation. For example, if several screws have to be removed from an assembly within a single step often only a few representatives are shown in an image, while others remain occluded. Since our system can only detect visible elements it will fail to retarget such cases. A similar problem occurs when parts are small and those barely visible in the image. We provide visual examples for the described limitations in the accompanying video material.

CONCLUSIONS AND FUTURE WORK

We presented a system to generate interactive 3D documentations from 2D image data. The resulting 3D scene can be presented on a regular desktop or tablet computer, or can be displayed directly within the user's real world environment. Thus, our system enables to reuse existing 2D documentation material to create 3D documentations. Therefore, we believe that our approach will become a key enabling technology to move from traditional 2D to modern interactive 3D documentation.

The central idea for transfer of 2D documentation to 3D is to replicate the 2D image by rendering a 3D model of the object of interest. This relates 2D image elements to 3D objects. We have demonstrated how this relation can be used to interact in 2D image space in order to control the corresponding 3D visualization. Thus, we believe that applications of our system will not substitute 2D documentations entirely. Instead, they will combine 2D and 3D data to provide the user with best of both presentation and interaction spaces.

While our system is able to handle images present in existing 2D documentation, it is not limited to documentation material. Our system can generate 3D animations from any image or photo sequence which shows an object in different configurations. Therefore, our system furthermore provides a new tool to control 3D animations by images.

Several direction for future work exist. For example, some documentations present instructions by visualizing both the cause and the effect of an action within a single image. This is usually done by adding commonly known glyphs to abstract the depicted effect. To retarget cause and effect visualizations, we need to extend our system to be able to identify such glyphs and to subsequently derive the class of possible motion from it. Another direction of future work is concerned with retargeting documentations which depict interactions with soft tissue or deformable objects. Generating candidate configurations of 3D deformable objects presents additional challenges such as a large amount of candidate configurations and complex models describing the deformations.

ACKNOWLEDGMENTS

This work was funded by a grant from the Competence Centers for Excellent Technologies (COMET) 843272 with support from AVL List GmbH and the Austrian Science Fund (FWF) under contract P-24021.

REFERENCES

1. Agrawala, M., Phan, D., Heiser, J., Haymaker, J., Klingner, J., Hanrahan, P., and Tversky, B. Designing effective step-by-step assembly instructions. ACM Trans. Graph. 22, 3 (2003), 828–837.
2. Bergig, O., Hagbi, N., El-Sana, J., and Billinghurst, M. In-place 3d sketching for authoring and augmenting mechanical systems. In Proc. of ISMAR (2009), 87–94.
3. Butz, A. Betty: Planning and generating animations for the visualization of movements and spatial relations. In Proc. of Advanced Visual Interfaces (1994), 53–58.
4. Canny, J. A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. 8, 6 (June 1986), 679–698.
5. Dementhon, D. F., and Davis, L. S. Model-based object pose in 25 lines of code. Int. J. Comput. Vision 15, 1-2 (June 1995), 123–141.
6. Gupta, A., Fox, D., Curless, B., and Cohen, M. F. Duplotrack a real-time system for authoring and guiding duplo block assembly. In UIST (2012), 389–402.
7. Heiser, J., Phan, D., Agrawala, M., Tversky, B., and Hanrahan, P. Identification and validation of cognitive design principles for automated generation of assembly instructions. In Proc. of AVI (2004), 311–319.
8. Henderson, S., and Feiner, S. Exploring the benefits of augmented reality documentation for maintenance and repair. IEEE TVCG 17, 10 (2011), 1355–1368.
9. Kalkofen, D., Tatzgern, M., and Schmalstieg, D. Explosion diagrams in augmented reality. In Proc of IEEE VR (2009), 71–78.
10. Kalkofen, D., Veas, E. E., Zollmann, S., Steinberger, M., and Schmalstieg, D. Adaptive ghosted views for augmented reality. In IEEE ISMAR (2013), 1–9.
11. Ledermann, F., and Schmalstieg, D. April: A high-level framework for creating augmented reality presentations. In Proc. of IEEE VR (2005), 187–194.
12. Li, W., Agrawala, M., Curless, B., and Salesin, D. Automated generation of interactive 3d exploded view diagrams. ACM Trans. Graph. 27, 3 (2008).
13. Li, W., Agrawala, M., and Salesin, D. Interactive image-based exploded view diagrams. In Prof of Graphics Interface (2004), 203–212.
14. Liu, C., Yuen, J., Torralba, A., Sivic, J., and Freeman, W. T. Sift flow: Dense correspondence across different scenes. In Proc. of ECCV (2008), 28–42.
15. Liu, M.-Y., Tuzel, O., Veeraraghavan, A., and Chellappa, R. Fast directional chamfer matching. In Proc. of CVPR (2010), 1696–1703.
16. Lozano-Perez, T. Spatial planning: A configuration space approach. IEEE Trans. Comput. 32, 2 (1983), 108–120.
17. Matas, J., Chum, O., Urban, M., and Pajdla, T. Robust wide baseline stereo from maximally stable extremal regions. In Proc. of BMVC (2002).
18. Mijksenaar, P., and Westendorp, P. Open Here: The Art of Instructional Design. Thames & Hudson, 1999.
19. Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A. W. Kinectfusion: Real-time dense surface mapping and tracking. In ISMAR, IEEE (2011), 127–136.
20. Petersen, N., and Stricker, D. Learning task structure from video examples for workflow tracking and authoring. In Proc of ISMAR (2012), 237–246.
21. Romney, B., Godard, C., Goldwasser, M., and Ramkumar, G. An efficient system for geometric assembly sequence generation and evaluation. In Proc. of Computers in Engineering (1995), 699–712.
22. Rusu, R. B., Blodow, N., and Beetz, M. Fast point feature histograms (fpfh) for 3d registration. In ICRA, IEEE (2009), 3212–3217.
23. Rusu, R. B., and Cousins, S. 3D is here: Point Cloud Library (PCL). Proc. of ICRA (2011).
24. Shao, T., Li, W., Zhou, K., Xu, W., Guo, B., and Mitra, N. J. Interpreting concept sketches. TOG 32, 4 (2013).
25. Srinivasan, H., and Gadh, R. Efficient geometric disassembly of multiple components from an assembly using wave propagation. Mech. Design 122 (2000), 179.
26. Wilson, R. H., and Latombe, J.-C. Geometric reasoning about mechanical assembly. In Proc. Algorithmic Foundations of Robotics (1995), 203–220.
27. Zauner, J., Haller, M., Brandl, A., and Hartmann, W. Authoring of a mixed reality assembly instructor for hierarchical structures. In ISMAR (2003), 237–246.
28. Zheng, Y., Kuang, Y., Sugimoto, S., strm, K., and Okutomi, M. Revisiting the pnp problem: A fast, general and optimal solution. In ICCV (2013), 2344–2351.