Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®
Artificial Intelligence 151 (2003) 91–154
Artificial Intelligence
www.elsevier.com/locate/artint

Reconstructing force-dynamic models from video sequences

Jeffrey Mark Siskind [1]

Purdue University, School of Electrical and Computer Engineering, 465 Northwestern Ave., West Lafayette, IN 47907-1285, USA

Received 14 February 2002

Abstract

This paper presents a method for recovering the support, contact, and attachment (i.e., force dynamic) relations between objects depicted in video sequences. It first presents a stability-analysis procedure that determines whether a configuration of observed objects is stable under a given interpretation using a reduction to linear programming. It then presents a model-reconstruction procedure for searching the space of admissible interpretations to find the simplest stable interpretations or models. These stability-analysis and model-reconstruction procedures have been implemented as part of a system that recovers force-dynamic interpretations from video sequences and uses those interpretations to classify the events that occur in those sequences. This paper presents the details of the stability-analysis and model-reconstruction procedures and illustrates their operation on sample video sequences.
© 2003 Elsevier B.V. All rights reserved.

Keywords: Machine vision; Grounded lexical semantics; Event perception; Stability analysis; Perceiver framework; Circumscription; Knowledge representation; Physical reasoning; Perceptual inference; Knowledge-based perception

1. Introduction

People can make judgments about the support, contact, and attachment relations between visually observed objects. For example, if one were to look at the image in Fig. 1(a), one could say something like *The green block* ***supports*** *the red block*. Similarly,

E-mail address: qobi@purdue.edu (J.M. Siskind).
URL: http://www.ece.purdue.edu/~qobi.
[1] Part of this work was done while the author was at NEC Research Institute, Inc.

0004-3702/$ – see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S0004-3702(03)00112-7

Fig. 1. Force-dynamic and stability judgments. (a) A green block supporting a red block. (b) A hand attached to and supporting a red block. (c) A stable scene. (d) An unstable scene.

if one were to look at the image in Fig. 1(b), one could say something like *The hand is **attached to** and **supports** the red block*. People can also make judgments about the stability of visually observed scenes. For example, one would say that the scene depicted in Fig. 1(c) is stable, because the green block supports the red block and prevents it from falling, while one would say that the scene depicted in Fig. 1(d) is unstable, because the red block is unsupported and can fall. This paper describes an implemented system, called LEONARD, that makes similar judgments from video input.

I collectively refer to support, contact, and attachment relations as *force-dynamic* relations. Talmy [104] originally used the term force dynamics to describe the schematized relationships between participants in events described by simple utterances: one participant preventing, allowing, or constraining an action by another participant. In this paper, I use the term force dynamics to mean something somewhat different than what Talmy intended, namely the support, contact, and attachment relations between event participants. Nonetheless, the underlying motivation for use of the term force dynamics comes from the fact that support can be viewed as preventing an object from falling while attachment can be viewed as constraining the relative motion of two objects.

LEONARD does not operate directly on images. Instead, it operates on two-dimensional polygonal scenes extracted from images by a segmentation and tracking procedure.[2] For example, LEONARD will determine that the scene in Fig. 2(a) is stable, because block A cannot fall without penetrating the Table, in contrast to the scene in Fig. 2(b), which is unstable, because A can fall without penetrating the Table. The fact that solid objects cannot interpenetrate is known as the *substantiality* constraint and is central to determining support relations and scene stability. Spelke [98–100], Baillargeon, Spelke, and Wasserman [9], and Baillargeon [7,8] have shown that young infants have knowledge of the substantiality constraint. Thus this capacity is either innate or acquired very early and, accordingly, it seems reasonable that it can form the basis of the physical reasoning underlying perception.

One might hypothesize a simple definition of stability and support: a scene is stable if all objects are supported and object A supports object B if A is supported and B is above and in contact with A. Fig. 3 shows that this simple definition is inadequate. Block A is above

[2] A real-time color- and motion-based segmentation algorithm places a convex polygon around each colored and moving object in each frame. A tracking algorithm then forms a correspondence between the polygons in each frame and those in temporally adjacent frames. The tracker guarantees that each scene contains the same number of polygons and that they are ordered so that the ith polygon in each scene corresponds to the same object. The segmentation and tracking algorithms are extensions of the algorithms presented in Siskind and Morris [97] and Siskind [94], modified to place convex polygons around the participant objects instead of ellipses.

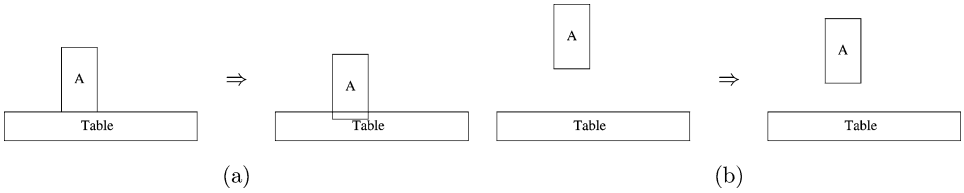


Fig. 2. The substantiality constraint is a factor in support judgments. (a) This scene is stable because block *A* cannot fall without penetrating the Table. (b) This scene is unstable because *A* can fall without penetrating the Table.

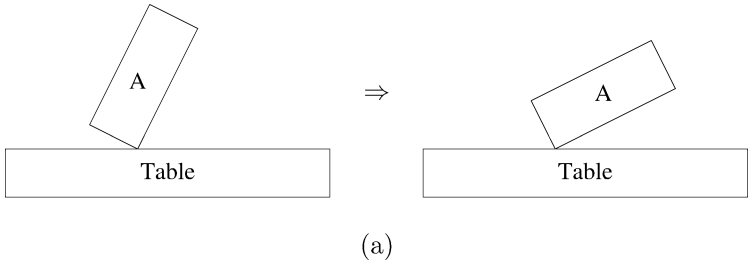


Fig. 3. This scene is not stable, despite the fact that block *A* is above and in contact with the Table, because *A* can fall over.

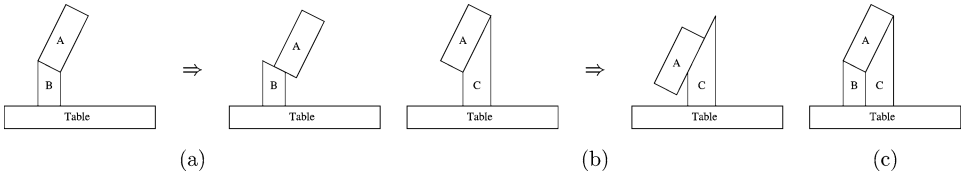


Fig. 4. Multiple sources of support. (a) An unstable scene. Block *B* alone does not support block *A*. (b) An unstable scene. Block *C* alone does not support *A*. (c) A stable scene. *B* and *C* together support *A*.

and in contact with the Table yet the scene is not stable and the Table does not support *A* because it can fall over.[3] Thus a definition of stability and support must take rotational motion and center of mass into account.

One might also hypothesize that an object is supported if one other object prevents it from falling. Under such a hypothesis, stability analysis would be a local procedure determining whether each object in a scene was supported by a chain of pairwise support relations. Fig. 4 shows that this, too, is inadequate. Neither block *B* nor block *C* alone prevent block *A* from falling yet *A* is supported by the combination of *B* and *C*. Thus stability analysis is not local. It cannot be based on simple pairwise support relations. Rather, it requires global analysis of the scene to determine if the objects can move under the influence of gravity in a fashion that upholds the substantiality constraint.

[3] This ignores potential metastable situations. I will discuss such metastable situations in Section 2.6.

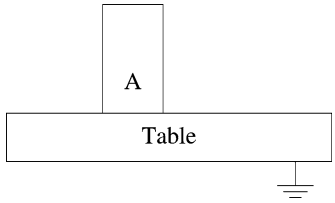


Fig. 5. Grounding the Table allows this scene to be stable.

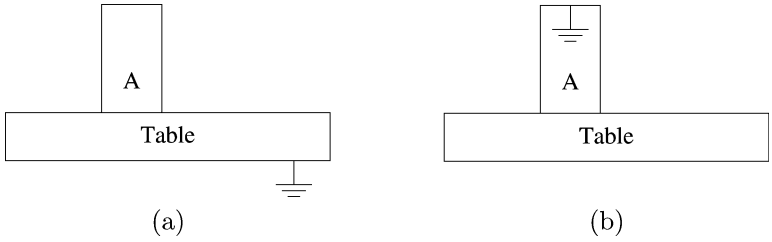


Fig. 6. Different interpretations of the same scene can lead to different stability judgments. Interpretation (a) is stable because the Table is grounded and supports block *A*. Interpretation (b) is unstable because the Table is unsupported.

Earlier, I oversimplified when I said that the scene depicted in Fig. 2(a) is stable. Although the Table supports block *A* and prevents it from falling, what supports the Table? The Table is supported by its legs which are in turn supported by the floor which is in turn supported by the building and so on. Since support is not 'Turtles all the way down', we must hypothesize an ultimate unobserved or unexplained source of support. I say that an object supported by such an unobserved source of support is *grounded* and indicate this fact by attaching a ground symbol '⏚' to a polygon as shown in Fig. 5. More precisely, a grounded polygon is constrained to have a fixed position and orientation in the 2D image plane. Grounding the Table allows the scene depicted in Fig. 5 to be stable.

Groundedness is not an observable property of an object. Rather, it is part of an *interpretation* one imparts to an observed scene. One can construct different interpretations of the same scene where different collections of objects are grounded. Such different interpretations can lead to different stability judgments. For example, if the Table is grounded, as depicted in the interpretation in Fig. 6(a), the scene depicted in Fig. 6 is stable, because the Table supports *A*. If, instead, block *A* is grounded, as depicted in the interpretation in Fig. 6(b), the scene is not stable, because the Table is unsupported.

Earlier, I oversimplified when I said that the scene depicted in Fig. 4(c) is stable, even when grounding the Table, because blocks *B* and *C* might slide sideways allowing block *A* to fall, as depicted in Fig. 7(a). To account for the stability of the scene depicted in Fig. 4(c), we could model friction between the Table and blocks *B* and *C*. We will discuss one way to model friction in Section 2.6. For now, we instead hypothesize that *B* and *C* are rigidly *attached* to the Table. I indicate such rigid attachments, or *joints*, by small solid circles, as depicted in Fig. 7(b). Rigid joints constrain the relative translation and rotation of two

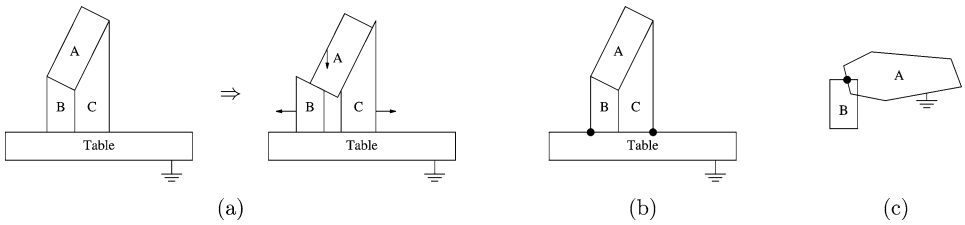


Fig. 7. Rigid attachment. (a) An unstable interpretation. Without attaching blocks *B* and *C* to the Table, they can slide sideways to allow block *A* to fall. (b) A stable interpretation. Attaching blocks *B* and *C* to the Table prevents them from sliding and allows this scene to be stable. (c) Attachment can model a hand *A* grasping a block *B*.

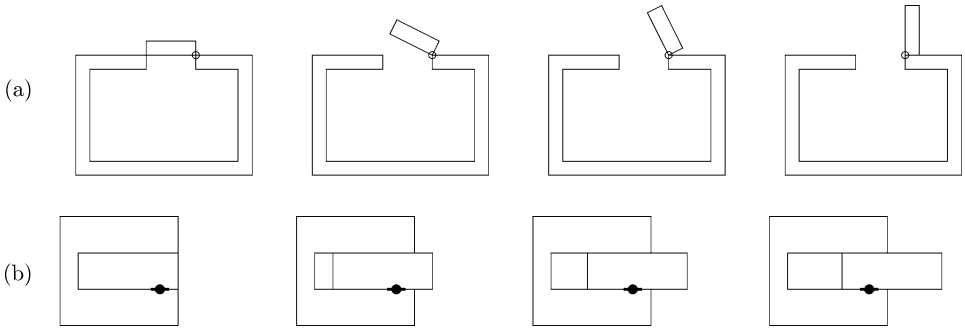


Fig. 8. Revolute and prismatic joints. (a) A revolute joint modeling a door hinge. (b) A prismatic joint modeling a sliding drawer.

attached objects at the point of attachment. Rigid joints can be used to model a hand *A* grasping a block *B*, as depicted in Fig. 7(c).

In addition to allowing rigid joints, we also allow *revolute* and *prismatic* joints. Revolute joints constrain the relative translation—but not the relative rotation—of two attached objects at the point of attachment. In contrast, prismatic joints constrain the relative rotation but not the relative translation. I indicate revolute joints by small hollow circles and prismatic joints by small solid circles with short thick lines along the *prismatic axis*, the axis of allowed translation. As depicted in Fig. 8, revolute joints can model door hinges while prismatic joints can model sliding drawers.

Just as is the case for groundedness, attachment relations are not directly visible. One doesn't see the actual attachment between a door knob and a door. One hypothesizes that attachment to explain the fact that the door knob doesn't fall. Thus, like groundedness, attachment relations are part of an interpretation that one constructs for an observation. Furthermore, as is the case for groundedness, there can be different interpretations of the same scene, with different attachment relations, that lead to different stability judgments. For example, Fig. 9 depicts four different interpretations that lead to different stability judgments. Interpretations (a) and (d) are unstable, because the joint allows block *B* to rotate, in the case of (a), or slide, in the case of (d), relative to block *A*, while interpretations (b) and (c) are stable, because, in the case of (b), *B* cannot rotate clockwise about the joint without penetrating *A* and, in the case of (c), *A* and *B* are rigidly attached.

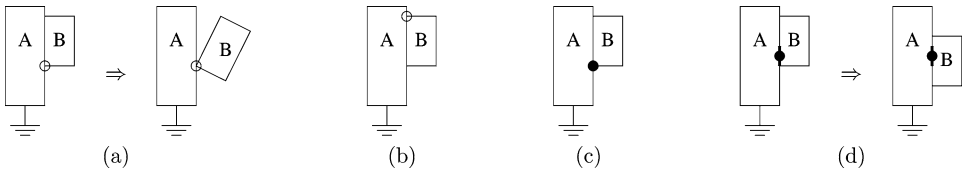


Fig. 9. Different interpretations of the same scene, with different attachment relations, can lead to different stability judgments. Interpretations (a) and (d) are unstable, because the joint allows block *B* to rotate, in the case of (a), or slide, in the case of (d), relative to block *A*, while interpretations (b) and (c) are stable, because, in the case of (b), *B* cannot rotate clockwise about the joint without penetrating *A* and, in the case of (c), *A* and *B* are rigidly attached.

Earlier, I oversimplified when I said that the scene depicted in Fig. 6(a) is stable. Two-dimensional polygonal scenes are derived from two-dimensional images and two-dimensional images are derived via projection from a 3D world. Different worlds, with objects at different depths, or distances from the observer, can yield the same image when projected on the image plane. While block *A* appears to touch the Table in the scene depicted in Fig. 6(a), it may, in fact, have been in front of or behind the Table in the world. LEONARD models depth qualitatively. Objects are taken to reside on *layers* and two objects are taken to reside on either the same layer or different layers. The polygons for two objects on the same layer must not overlap[4] or else a substantiality violation will occur. Conversely, for two objects to be in contact, their polygons must touch and they must be on the same layer. Such contact is necessary for a support relation to hold between two objects. LEONARD uses an impoverished notion of depth. Each object is taken to reside on a single layer and qualitative depth is treated as an equivalence relation with no notion of order or adjacency between layers.

While there are visual cues to depth, such as stereo, structure from motion, lighting, and the like, such cues can be unreliable. Thus we are going to ignore such cues and treat qualitative depth as part of an interpretation in the same fashion as groundedness and attachment relations. An interpretation will specify whether two objects are on the same layer or different layers. I indicate a same-layer relation by placing layer indices, small nonnegative integers, next to polygons in scenes. Polygons with the same layer index are taken to be on the same layer while polygons with different layer indices are taken to be on different layers. The integral value of the index does not carry any meaning other than

[4] Throughout this paper, I say that two line segments *overlap* if their intersection consists of more than one point and that two line segments *touch* when their intersection consists of a single point that is coincident with an endpoint of one or both of the line segments. Furthermore, I say that two polygons *overlap* when their intersection has nonzero area and that two polygons *touch* when they intersect but their intersection has zero area. Note that two line segments can intersect without touching or overlapping and cannot simultaneously touch and overlap. Also note that two intersecting polygons must touch or overlap but cannot simultaneously touch and overlap. We allow joints between intersecting polygons irrespective of whether they overlap or touch. Furthermore, we allow joints between polygons irrespective of whether they are on the same or different layers. However, since polygons that overlap must be on different layers and since joints occur more frequently between overlapping polygons, such joints occur more frequently between polygons on different layers. One can view (revolute) joints between overlapping or touching polygons on different layers as rivets and those between touching polygons on the same layer as hinges.

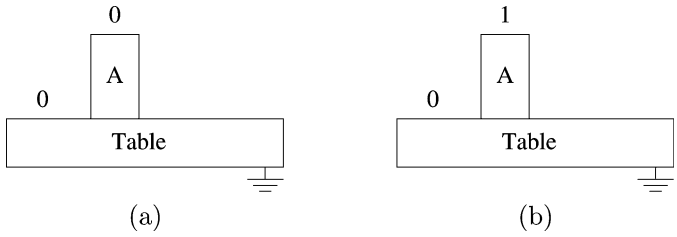


Fig. 10. Different interpretations of the same scene, with different same-layer relations, can lead to different stability judgments. Interpretation (a) is stable because block A is on the same layer as, and thus in contact with, the Table while interpretation (b) is unstable because A is on a different layer than, and thus not in contact with, the Table.

equivalence to other indices in a particular frame. Just as is the case for groundedness and attachment relations, there can be different interpretations of the same scene with different same-layer relations. These different interpretations can lead to different stability judgments. For example, Fig. 10(a) depicts a stable interpretation, because block A is on the same layer as, and thus in contact with, the Table, while Fig. 10(b) depicts an unstable one, because A is on a different layer than, and thus not in contact with, the Table.

Throughout this paper, I use q to denote points, l to denote line segments, and p to denote polygons. Let $x(q)$ and $y(q)$ denote the image coordinates of q, $q(x, y)$ denote the point whose image coordinates are x and y, $q_1(l)$ and $q_2(l)$ denote the endpoints of l, and $l(q_1, q_2)$ denote the line whose endpoints are q_1 and q_2. Furthermore, let $\theta(q)$ be the orientation of a vector from the origin to q, $q(\theta)$ be the point at unit distance from the origin such that the orientation of a vector from the origin to that point is θ, and $\bar{q}$ be the point that is the same distance as q from the origin but where $\theta(\bar{q})$ is $\theta(q)$ rotated clockwise 90°.[5]

$$\theta(q) \triangleq \tan^{-1} \frac{-y(q)}{x(q)}$$

$$q(\theta) \triangleq q(\cos\theta, -\sin\theta)$$

$$\bar{q} \triangleq q\big(-y(q), x(q)\big)$$

Joints can occur only at points in the intersection region of two polygons. I refer to such points as *contacts*. I indicate contacts by smaller solid circles than those used to indicate rigid and prismatic joints. LEONARD only considers contacts at the intersection points of polygon edges, not at other points in the intersection region of two polygons. Thus LEONARD only considers points q_1 and q_2 in Fig. 11(a) as contacts and does not consider point q_3. Ignoring overlapping edges, any two convex polygons have at most two edge-intersection points. When presented with polygons with overlapping edges, as depicted in Fig. 11(b), LEONARD only considers the two endpoints q_1 and q_2 of the overlapping-edge region as contacts and does not consider point q_3. Thus any two convex polygons have at most two points that are considered as contacts.

[5] With image coordinates, x increases in the rightward direction and y increases in the downward direction.

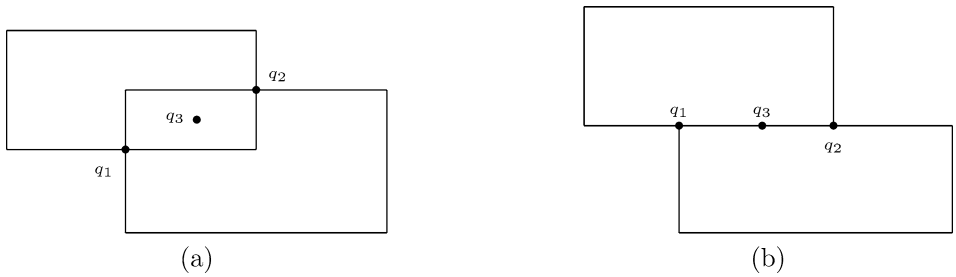


Fig. 11. LEONARD only considers contacts at the intersection points of polygon edges, not at other points in the interior of the intersecting polygons. Thus, in (a), points q_1 and q_2 are contacts while q_3 is not. When presented with polygons with overlapping edges, as in (b), only the endpoints q_1 and q_2 are considered as contacts, not interior points such as q_3.

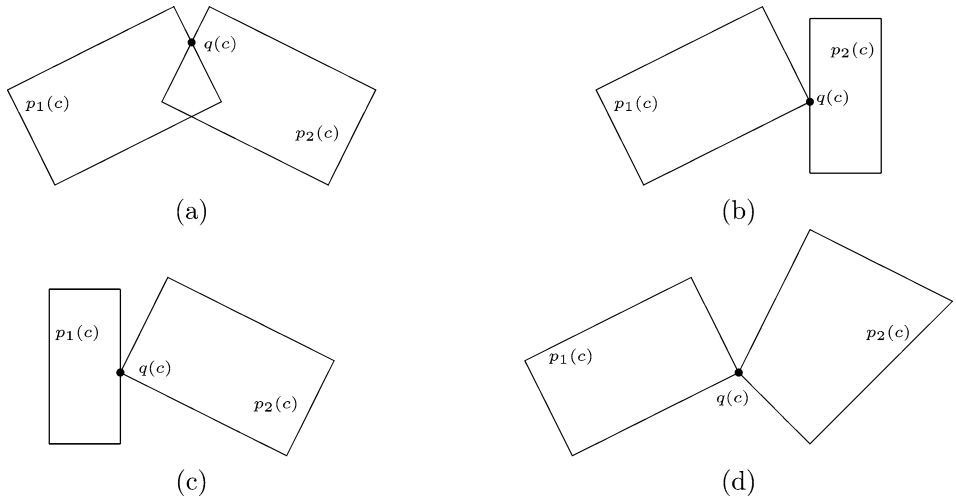


Fig. 12. Four kinds of contacts: (a) An edge-to-edge contact. (b) A corner-to-edge contact. (c) An edge-to-corner contact. (d) A corner-to-corner contact.

Throughout this paper, I use c to denote contacts. Let $q(c)$ denote the location of a contact c, $p_1(c)$ and $p_2(c)$ denote the two polygons that are potentially joined at $q(c)$, and $l_1(c)$ and $l_2(c)$ denote the edges of those polygons respectively that intersect at $q(c)$. Let P denote a scene, represented as a set of convex polygons, and let $C(P)$ denote the set of contacts in P. Contacts come in pairs, c_1 and c_2, where $p_1(c_1) = p_2(c_2)$, $p_2(c_1) = p_1(c_2)$, $l_1(c_1) = l_2(c_2)$, and $l_2(c_1) = l_1(c_2)$. $C(P)$ is constructed to contain only a single contact from each such pair.

Fig. 12 illustrates four kinds of contacts: (a) *edge-to-edge*, (b) *corner-to-edge*, (c) *edge-to-corner*, and (d) *corner-to-corner*. Two predicates are used to distinguish between the kinds of contacts. $\text{CORNER}_1(c)$ indicates that an endpoint of $l_1(c)$ lies on $l_2(c)$ while $\text{CORNER}_2(c)$ indicates that an endpoint of $l_2(c)$ lies on $l_1(c)$. A contact c is edge-to-edge if neither $\text{CORNER}_1(c)$ nor $\text{CORNER}_2(c)$ is true. It is corner-to-edge if $\text{CORNER}_1(c)$ is true

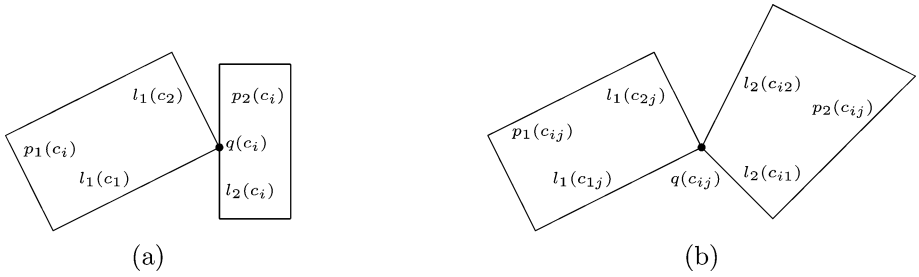


Fig. 13. (a) There are two coincident corner-to-edge contacts $c_i, i \in \{1,2\}$. $l_2(c_1) = l_2(c_2)$ but $l_1(c_1) \neq l_1(c_2)$. (b) There are four coincident corner-to-corner contacts $c_{ij}, i \in \{1,2\}, j \in \{1,2\}$.

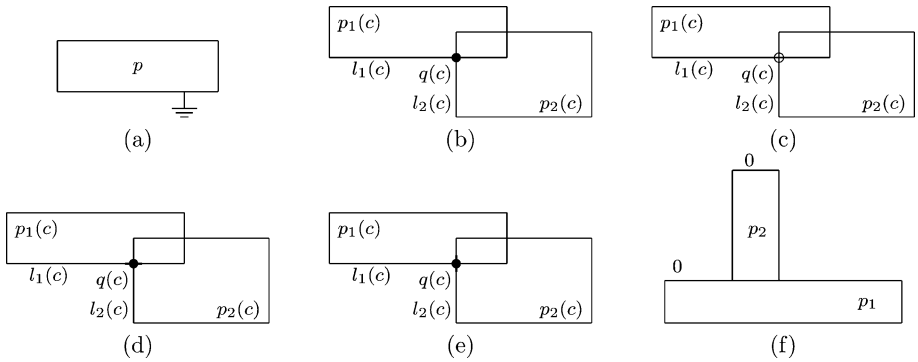


Fig. 14. Graphical depiction of interpretations. (a) GROUNDED(p). (b) RIGID(c). (c) REVOLUTE(c). (d) PRISMATIC$_1$(c). (e) PRISMATIC$_2$(c). (f) SAMELAYER(p_1, p_2).

but CORNER$_2$(c) is false. It is edge-to-corner if CORNER$_2$(c) is true but CORNER$_1$(c) is false. It is corner-to-corner if both CORNER$_1$(c) and CORNER$_2$(c) are true.

Corner-to-edge contacts, as depicted in Fig. 13(a), come in pairs, c_1 and c_2, where $l_1(c_1)$ and $l_1(c_2)$ are the two edges of $p_1(c)$ that intersect to form the corner. Likewise, edge-to-corner contacts also come in pairs. Corner-to-corner contacts, as depicted in Fig. 13(b), come in quadruples, c_{11}, c_{12}, c_{21}, and c_{22}, where $l_1(c_{11}) = l_1(c_{12})$ and $l_1(c_{21}) = l_1(c_{22})$ are the two edges of $p_1(c)$ that intersect to form one corner and $l_2(c_{11}) = l_2(c_{21})$ and $l_2(c_{12}) = l_2(c_{22})$ are the two edges of $p_2(c)$ that intersect to form the other corner.

The graphical depiction of interpretations corresponds straightforwardly to a more conventional textual one. A groundedness assertion, as depicted in Fig. 14(a), is specified as GROUNDED(p). This specifies that p is grounded. A rigid joint, as depicted in Fig. 14(b), is specified as RIGID(c). This specifies that $p_1(c)$ and $p_2(c)$ are attached by a rigid joint at $q(c)$. Similarly, a revolute joint, as depicted in Fig. 14(c), is specified as REVOLUTE(c). Specifying prismatic joints is a little more complex. It is necessary to specify the prismatic axis. LEONARD allows only two prismatic axes, one along $l_1(c)$ and the other along $l_2(c)$. A prismatic joint with the prismatic axis along $l_1(c)$, as depicted in Fig. 14(d), is specified as PRISMATIC$_1$(c) while a prismatic joint with the prismatic axis along $l_2(c)$, as depicted in Fig. 14(e), is specified as PRISMATIC$_2$(c). LEONARD allows at most one joint at each contact. Thus at most one of RIGID(c), REVOLUTE(c),

PRISMATIC$_1(c)$, and PRISMATIC$_2(c)$ will be true for any contact c. Finally, a same-layer assertion, as depicted in Fig. 14(f), is specified as SAMELAYER(p_1, p_2). This specifies that p_1 and p_2 are on the same layer. With this, an interpretation is a 6-tuple:

$$\langle \text{GROUNDED}, \text{RIGID}, \text{REVOLUTE}, \text{PRISMATIC}_1, \text{PRISMATIC}_2, \text{SAMELAYER} \rangle.$$

Throughout this paper, I use I to denote interpretations.

The objective of *stability analysis* is to determine whether a given scene P is stable under a given interpretation I. Stability analysis is expressed as a procedure STABLE(P, I) and is somewhat more involved than what is described in this overview. Section 2 presents the stability-analysis procedure used by LEONARD in greater detail.

Stability analysis can determine whether a given scene is stable under a given interpretation. However, the world does not come labeled with interpretations. An observer must *infer* an interpretation that is consistent with an observed scene. I call this process *model reconstruction* and give an overview of this process below.

While the stability-analysis procedure to be described in Section 2 can handle arbitrary interpretations as formulated above, the model-reconstruction procedure that will be described momentarily must search the space of possible interpretations. Since this space is too large to be explored using the techniques described in this paper, a smaller space of interpretations will be considered where PRISMATIC$_1(c)$ and PRISMATIC$_2(c)$ are always false for all c. In other words, the current model-reconstruction procedure will not consider interpretations with prismatic joints. I adopt an abbreviated notation where a 4-tuple:

$$\langle \text{GROUNDED}, \text{RIGID}, \text{REVOLUTE}, \text{SAMELAYER} \rangle$$

denotes the corresponding 6-tuple with PRISMATIC$_1(c)$ and PRISMATIC$_2(c)$ false for all c.

Figs. 15(a) and 15(b) show two different interpretations of the same scene. (This scene was derived from an actual image of a hand picking one block up off of another block.) The interpretation shown in Fig. 15(a) is stable, because the hand and lower block are grounded and the upper block is rigidly attached to the hand. The interpretation shown in Fig. 15(b) differs from the one shown in Fig. 15(a) by the removal of the groundedness assertion for the hand. This renders this interpretation unstable because the hand-plus-upper-block complex can fall over to the right. I refer to stable interpretations, such as the one shown in Fig. 15(a), as *models* of the scene, much as consistent interpretations of an axiomatic theory are considered models of that theory.

Fig. 15. Different interpretations of the same scene. Interpretation (a) is stable, because the hand and lower block are grounded and the upper block is rigidly attached to the hand. Interpretation (b) differs from (a) by the removal of the groundedness assertion for the hand. This renders (b) unstable because the hand-plus-upper-block complex can fall over to the right.

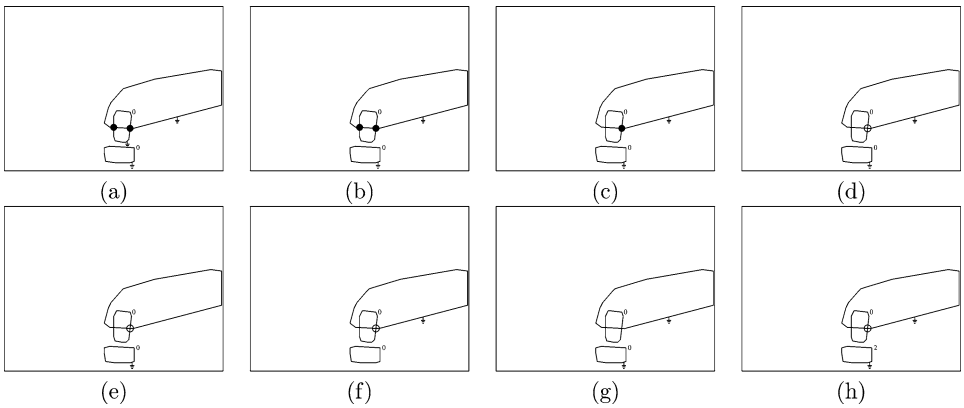


Fig. 16. The process of constructing a preferred model for the scene from Fig. 15. (a) shows the maximal interpretation, which is stable. (b) shows that removing the groundedness assertion from the upper block yields a stable interpretation. (c) shows that removing one of the rigid joints also yields a stable interpretation. (d) shows that changing the remaining rigid joint into a revolute joint also yields a stable interpretation. (e) shows that removing the groundedness assertion from the hand yields an unstable interpretation. (f) shows that removing the groundedness assertion from the lower block also yields an unstable interpretation. (g) shows that removing the revolute joint also yields an unstable interpretation. (h) shows that removing the same-layer assertion between the upper and lower blocks also yields an unstable interpretation. Therefore, (d) is a minimal, and thus preferred, model of this scene.

Fig. 16(a) shows another model of the same scene from Fig. 15. This interpretation is, in some sense, maximal: every object is grounded, there is a rigid joint at every contact, and as many polygons as possible are on the same layer. (Note that the hand and upper block must be on different layers to uphold the substantiality constraint.) In some sense, many of these groundedness assertions, joints, and same-layer assertions are redundant. It is possible to come up with a simpler interpretation—one that has fewer groundedness assertions, joints, and same-layer assertions—that is still stable. One can remove the groundedness assertion from the upper block, as shown in Fig. 16(b), and still have a stable interpretation. One can also remove one of the rigid joints, as shown in Fig. 16(c), and still have a stable interpretation. Furthermore, one can change the remaining rigid joint into a revolute joint, as shown in Fig. 16(d), and still have a stable interpretation, because even though the upper block can rotate about the revolute joint, the fact that it is on the same layer as the lower block prevents it from rotating. (Note that the lower block prevents the upper block from rotating, even though the two blocks don't touch, because of a nonzero tolerance added to the stability analysis. Such tolerances will be discussed in Section 2.8.)

The model shown in Fig. 16(d) is, in some sense, minimal. It is not possible to remove any groundedness assertions, remove or weaken any joints, or remove any same-layer assertions and still have a stable interpretation. For example, if the groundedness assertion is removed from the hand, as shown in Fig. 16(e), the resulting interpretation is unstable, because the hand can rotate clockwise. Furthermore, if the groundedness assertion is removed from the lower block, as shown in Fig. 16(f), the resulting interpretation is also unstable, because the lower block can fall. Still further, if the revolute joint is removed, as shown in Fig. 16(g), the resulting interpretation is also unstable, because the upper

block can slide rightward off of the lower block. Finally, if the same-layer assertion between the upper and lower blocks is removed, as shown in Fig. 16(h), the resulting interpretation is also unstable, because the upper block can now rotate counterclockwise without penetrating the lower block. I refer to minimal models, like the one in Fig. 16(d), as *preferred models*. This is in line with Occam's razor, the preference for the simplest explanation of an observed phenomenon.

The above model-reconstruction process can be viewed as *prioritized circumscription* [67]. At the first priority level, LEONARD selects models with a minimal set of groundedness assertions. Then, at the second priority level, LEONARD selects models with a minimal set of joints. Then, at the third priority level, LEONARD selects models with a minimal set of rigid joints. Finally, at the fourth priority level, LEONARD selects models with a minimal set of same-layer assertions. At each priority level, set inclusion is used as the ordering relation between models for the minimality criterion. Because this ordering relation is partial, there can be multiple minimal models. Fig. 17 shows all four minimal, or preferred, models of the scene from Fig. 15.

Because prioritized circumscription with set inclusion as the ordering relation can yield multiple preferred models, LEONARD prunes the space of models even further with an ordering relation based on cardinality instead of set inclusion. In this pruning process, which I call *cardinality circumscription*, the cost of a model is defined as a 4-tuple: the number of groundedness assertions in the model, the number of rigid joints in the model, the number of revolute joints in the model, and the number of same-layer assertions in the model. LEONARD finds the minimal-cost models where costs are lexicographically ordered. I refer to the minimal models that remain after cardinality circumscription as *more-preferred models*. Note that there may be multiple more-preferred models because two different models may have the same cost. For example, consider the preferred models in Fig. 17. Models (a) and (b) both have two groundedness assertions, no rigid joints, one revolute joint, and one same-layer assertion. Models (c) and (d) both have two groundedness assertions, one rigid joint, no revolute joints, and no same-layer assertions. Models (c) and (d) have higher cost than (a) and (b). Thus cardinality circumscription eliminates (c) and (d) leaving (a) and (b) as more-preferred models. Cardinality circumscription yields two more-preferred models for this scene because models (a) and (b) have the same cost.

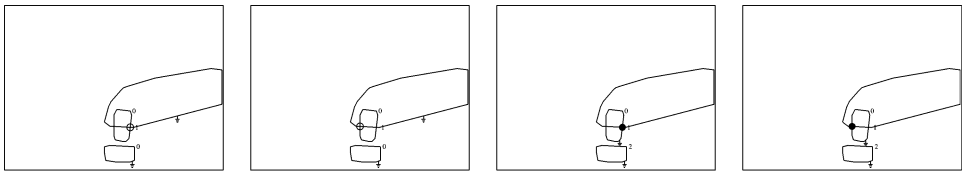


Fig. 17. All four minimal, or preferred, models of the scene from Fig. 15. Models (a) and (b) both have two groundedness assertions, no rigid joints, one revolute joint, and one same-layer assertion. Models (c) and (d) both have two groundedness assertions, one rigid joint, no revolute joints, and no same-layer assertions. Models (c) and (d) have higher cost than (a) and (b). Thus cardinality circumscription eliminates (c) and (d) leaving (a) and (b) as more-preferred models. Cardinality circumscription yields two more-preferred models for this scene because models (a) and (b) have the same cost.

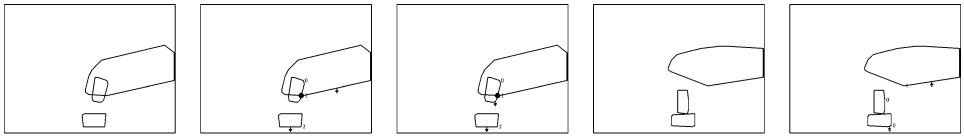


Fig. 18. Temporal circumscription. (a) A scene depicting a hand holding a block. Both (b) and (c) are more-preferred models of (a) as produced by prioritized and cardinality circumscription. In (b), the hand supports the block while, in (c), the block supports the hand. From this scene alone, there is no way to prefer (b) over (c) without knowing that the polygons depict hands and blocks and without knowledge of the affordances of such hands and blocks. However, analyzing (a) as following (d), which has a single more-preferred model (e), and choosing the interpretation for (a) that minimizes state changes in the groundedness property between scenes, induces a preference for (b) over (c) without knowledge of object class and the affordances of those classes.

Because cardinality circumscription can yield multiple more-preferred models, LEONARD prunes the space of models even further with a third ordering relation. The first two pruning processes, prioritized and cardinality circumscription, operate on each frame in isolation. No information is propagated across time between adjacent frames. In contrast, this third pruning process, which I call *temporal circumscription*, integrates information over time. The intuition behind this pruning process is illustrated by the following example. Cardinality circumscription yields two more-preferred models for the scene in Fig. 18(a), namely models (b) and (c). In (b), the hand is grounded and supports the upper block, which is not grounded, while in (c), the upper block is grounded and supports the hand, which is not grounded. Considering this scene in isolation, there is no way to prefer one model over the other, without knowing that the polygons depict hands and blocks and without knowledge of the affordances of such hands and blocks. Nonetheless, the scene in Fig. 18(a) is part of a movie which contains an earlier frame depicted in Fig. 18(d). Prioritized and cardinality circumscription yield a single more-preferred model for that scene. As shown in Fig. 18(e), in that model, the hand is grounded but the upper block is not. Taking model (c) as the follow-on to model (e) would require that the groundedness assertion move from the upper block to the hand. Intuitively, we wish to prefer model (b) over (c) as the follow-on to model (e) as this requires fewer state changes in the groundedness property. Essentially, temporal circumscription searches the space of model sequences, instead of individual models, and selects model sequences that minimize the number of such state changes. I refer to the model sequences that are produced by temporal circumscription as *most-preferred model sequences* and the models in those sequences as *most-preferred models*.

Note that there may be multiple most-preferred model sequences after temporal circumscription, because two different model sequences may have the same number of state changes in the groundedness property. For example, applying temporal circumscription to the movie shown in Fig. 19 yields two most-preferred model sequences, model sequence (a), where the hand is grounded and supports the block, which is not grounded, and model sequence (b), where the block is grounded and supports the hand, which is not grounded. Without knowledge of object class and the affordances of those classes, it is not possible to determine which of these two model sequences better corresponds to re-

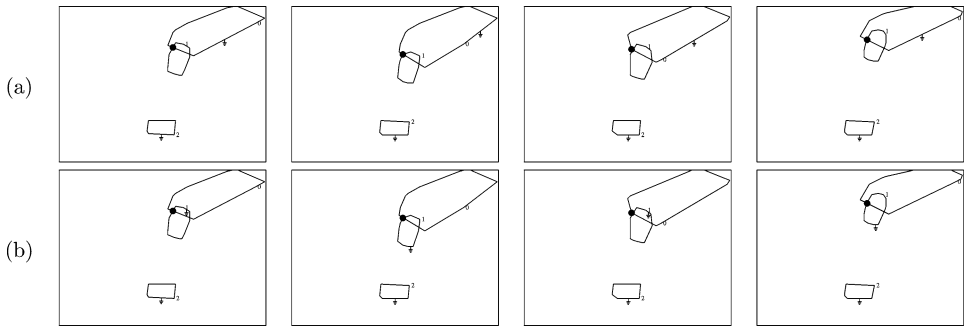


Fig. 19. Temporal circumscription cannot always resolve ambiguity. (a) and (b) both depict most-preferred model sequences for the same scene sequence. In (a), the hand supports the block while, in (b), the block supports the hand. Unlike the sequence in Fig. 18, this sequence does not contain a scene that disambiguates the interpretation. Thus temporal circumscription produces both (a) and (b) as most-preferred model sequences for this scene sequence.

ality. When faced with this remaining ambiguity, LEONARD chooses one model sequence arbitrarily as the most-preferred model sequence.

Model reconstruction is somewhat more involved than what is described in this overview. Section 3 presents the model-reconstruction procedure used by LEONARD in greater detail. Model reconstruction, as performed by LEONARD, falls into the *perceiver framework* [52]. In the perceiver framework, one must specify four things. First, one must specify a set of *observables*, which quantities can be discerned by direct observation. In LEONARD, the observables are the 2D positions of polygon vertices in each frame. Second, one must specify an *ontology*, a vocabulary of quantities, properties, and relations that are not directly observable but are used to formulate interpretations. In LEONARD, the ontology consists of the groundedness assertions, the various kinds of joints, and the same-layer assertions. Third, one must specify a *theory* that distinguishes between consistent interpretations (i.e., models) and inconsistent ones. In LEONARD, the theory is kinematic stability analysis. Finally, one must specify a *preference ordering* between models. In LEONARD, the preference ordering is specified by way of a sequence of circumscription operations.

2. Stability analysis

Stability analysis is embodied in a predicate STABLE(P, I). STABLE(P, I) takes a scene P and an interpretation I as input and returns true, if the scene is stable under the specified interpretation, and false otherwise. Each polygon $p \in P$ consists of an ordered set of vertices from a clockwise traversal of its perimeter, each vertex $q \in p$ being a point. Let $q_1(p)$ denote the first vertex in p. Each pair of adjacent vertices of a polygon constitutes an edge of the polygon, each edge being a line segment. For each polygon p, let $q(p)$ denote the centroid of p and let $\theta(p)$ denote the orientation of a vector from $q(p)$ to $q_1(p)$.

$$q(p) \triangleq \frac{1}{||p||} \sum_{q \in p} q$$

$$\theta(p) \triangleq \theta\big(q_1(p) - q(p)\big)$$

The interpretation I that is given as input to STABLE is formulated in slightly different terms than what was described in Section 1. I will describe the relationship between the two momentarily. An interpretation I is a quintuple $\langle$GROUNDED, RIGID$_1$, RIGID$_2$, RIGID$_\theta$, SAMELAYER$\rangle$ where GROUNDED is a property of polygons, RIGID$_1$, RIGID$_2$, and RIGID$_\theta$, are properties of contacts, and SAMELAYER is a relation between pairs of polygons. GROUNDED(p) indicates that p is grounded. Collectively, RIGID$_1(c)$, RIGID$_2(c)$, and RIGID$_\theta(c)$ indicate the kind of joint that holds between $p_1(c)$ and $p_2(c)$ at $q(c)$. Any two polygons have three degrees of freedom in their relative motion. The properties RIGID$_1$, RIGID$_2$, and RIGID$_\theta$ independently offer constraint or lack of constraint along each of these degrees of freedom. The property RIGID$_1(c)$ indicates that the position of $q(c)$ is fixed along $l_1(c)$. The property RIGID$_2(c)$ indicates that the position of $q(c)$ is fixed along $l_2(c)$. The property RIGID$_\theta(c)$ indicates that the relative orientation of $\theta(p_1(c))$ and $\theta(p_2(c))$ is fixed. Finally, SAMELAYER(p_1, p_2) indicates that p_1 and p_2 are on the same layer.

There is a simple correspondence between the format of interpretations given in Section 1 and the format of interpretations described above. The GROUNDED property and SAMELAYER relation in both formulations are identical. PRISMATIC$_1(c)$ corresponds to $\neg$RIGID$_1(c) \wedge$ RIGID$_2(c) \wedge$ RIGID$_\theta(c)$. PRISMATIC$_2(c)$ corresponds to RIGID$_1(c) \wedge \neg$RIGID$_2(c) \wedge$ RIGID$_\theta(c)$. REVOLUTE(c) corresponds to RIGID$_1(c) \wedge$ RIGID$_2(c) \wedge \neg$RIGID$_\theta(c)$. RIGID(c) corresponds to RIGID$_1(c) \wedge$ RIGID$_2(c) \wedge$ RIGID$_\theta(c)$. And the lack of a joint between $p_1(c)$ and $p_2(c)$ at $q(c)$ corresponds to $\neg$RIGID$_1(c) \wedge \neg$RIGID$_2(c) \wedge \neg$RIGID$_\theta(c)$. Formulating interpretations in terms of RIGID$_1$, RIGID$_2$, and RIGID$_\theta$ allows three additional joint types, namely joints that allow combinations of two prismatic and/or revolute motions. While the stability-analysis procedure can handle such joints, no use is made of this capacity in the (current) model-reconstruction procedure.

The stability-analysis procedure can be described intuitively as follows:

> A scene is stable under an interpretation if the polygons in the scene cannot move in a fashion that is consistent with the interpretation so that the potential energy of the scene decreases.

To flesh out this intuitive description, it is necessary to (a) represent the potential motion of polygons, (b) formulate the constraints that an interpretation imparts on such motion, and (c) specify a method for computing the potential energy of a scene.

To represent the potential motion of a polygon, I postulate that each polygon has an instantaneous linear and angular velocity. These are represented as $\dot{q}(p)$ and $\dot{\theta}(p)$ respectively. Since $q(p)$ has x and y components, a scene with n polygons has $3n$ free variables. We now need to formulate the constraints that GROUNDED, RIGID$_1$, RIGID$_2$, RIGID$_\theta$, and SAMELAYER impart to the collective $\dot{q}(p)$ and $\dot{\theta}(p)$ values.

2.1. Groundedness assertions and revolute joints

It is easy to formulate the constraint that GROUNDED imparts on the motion of a polygon. A grounded polygon p, one for which GROUNDED(p) holds, is constrained so that $\dot{q}(p) = 0$ and $\dot{\theta}(p) = 0$. It is also easy to formulate the constraint that RIGID$_\theta$ imparts on the relative motion of a pair of polygons. For each c for which RIGID$_\theta(c)$ holds, $p_1(c)$ and $p_2(c)$ are constrained so that $\dot{\theta}(p_1(c)) = \dot{\theta}(p_2(c))$. Note that all of these constraints are linear in the collective $\dot{q}(p)$ and $\dot{\theta}(p)$ values. The constraints that are imparted by RIGID$_1$, RIGID$_2$, and SAMELAYER are somewhat more complex but still turn out to be linear.

2.2. Prismatic joints

To formulate these more complex constraints, we first need to formulate how contact points and edges move as the polygons that form those contacts move. First, let us say that two line segments l_1 and l_2 are *collinear* if they intersect and $[q_2(l_1) - q_1(l_1)] \cdot \overline{[q_2(l_2) - q_1(l_2)]} = 0$. A contact c is *collinear* if $l_1(c)$ and $l_2(c)$ are collinear. Now, let us express a point q in terms of the position and orientation of a polygon p. The distance from the center of p to q is $||q - q(p)||$ and the angular difference between the orientation of a vector from the center of p to q and the orientation of p is $\theta(q - q(p)) - \theta(p)$. Let us treat these both as constants. Given these constants, let $\gamma(q, p)$ express q in terms of the position and orientation of p:

$$\gamma(q, p) \triangleq q(p) + \boxed{\|q - q(p)\|}\; q\big(\theta(p) + \boxed{\theta(q - q(p)) - \theta(p)}\big).$$

I enclose the constants in the above formula with boxes to indicate that they will be treated as constants when we later differentiate $\gamma(q, p)$. Now, let us express the contact edges $l_1(c)$ and $l_2(c)$ in terms of the positions and orientations of the polygons $p_1(c)$ and $p_2(c)$ that intersect to form the contact c. Just as $\gamma(q, p)$ expresses q relative to the position and orientation of p, let $\gamma_1(c)$ and $\gamma_2(c)$ express $l_1(c)$ and $l_2(c)$ in terms of the positions and orientations of $p_1(c)$ and $p_2(c)$ respectively.

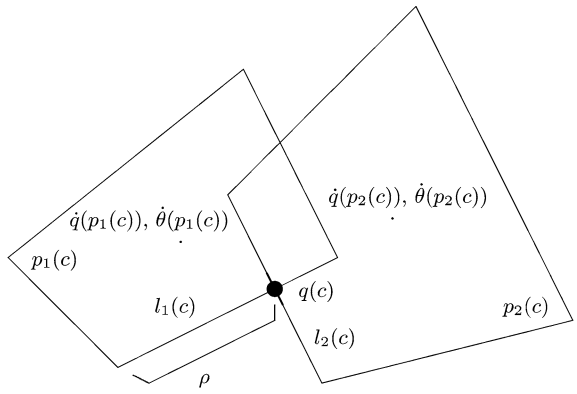
$$\gamma_1(c) \triangleq l\big(\gamma\big(q_1(l_1(c)), p_1(c)\big), \gamma\big(q_2(l_1(c)), p_1(c)\big)\big)$$
$$\gamma_2(c) \triangleq l\big(\gamma\big(q_1(l_2(c)), p_2(c)\big), \gamma\big(q_2(l_2(c)), p_2(c)\big)\big)$$

Now, let us express the contact point $q(c)$ in terms of the positions and orientations of the polygons $p_1(c)$ and $p_2(c)$ that intersect to form the contact c. When it exists and is unique, let $\mathcal{I}(l_1, l_2)$ denote the point of intersection of the two line segments l_1 and l_2. If we let

$$A(l_1, l_2) \triangleq \begin{pmatrix} y(q_1(l_1)) - y(q_2(l_1)) & x(q_2(l_1)) - x(q_1(l_1)) \\ y(q_1(l_2)) - y(q_2(l_2)) & x(q_2(l_2)) - x(q_1(l_2)) \end{pmatrix},$$

$$b(l_1, l_2) \triangleq \begin{pmatrix} y(q_1(l_1))(x(q_2(l_1)) - x(q_1(l_1))) + x(q_1(l_1))(y(q_1(l_1)) - y(q_2(l_1))) \\ y(q_1(l_2))(x(q_2(l_2)) - x(q_1(l_2))) + x(q_1(l_2))(y(q_1(l_2)) - y(q_2(l_2))) \end{pmatrix},$$

then $\mathcal{I}(l_1, l_2) \triangleq A(l_1, l_2)^{-1} b(l_1, l_2)$. Given this, $q(c)$ can be expressed in terms of $\gamma_1(c)$ and $\gamma_2(c)$ as $\mathcal{I}(\gamma_1(c), \gamma_2(c))$, when c is not collinear. When c is collinear, we select one of the endpoints of $l_1(c)$ and $l_2(c)$ that lie on both $l_1(c)$ and $l_2(c)$ as $q(c)$.



$$\textsc{Rigid}_1(c) \rightarrow \dot{\rho}(q(c), \gamma_1(c)) = 0$$

c	the joined contact
$p_1(c)$	one polygon joined by c
$p_2(c)$	the other polygon joined by c
$\dot{q}(p_1(c))$	the linear velocity of $p_1(c)$
$\dot{\theta}(p_1(c))$	the angular velocity of $p_1(c)$
$\dot{q}(p_2(c))$	the linear velocity of $p_2(c)$
$\dot{\theta}(p_2(c))$	the angular velocity of $p_2(c)$
$l_1(c)$	the edge of $p_1(c)$ joined by c
$l_2(c)$	the edge of $p_2(c)$ joined by c
$\gamma_1(c)$	$l_1(c)$ expressed in terms of $q(p_1(c))$ and $\theta(p_1(c))$
$\gamma_2(c)$	$l_2(c)$ expressed in terms of $q(p_2(c))$ and $\theta(p_2(c))$
$q(c)$	the location of c expressed in terms of $q(p_1(c))$, $\theta(p_1(c))$, $q(p_2(c))$, and $\theta(p_2(c))$
$\rho(r,l)$	the fraction of the distance a point r on line segment l is between the endpoints of l

Fig. 20. An illustration of the constraint that $\textsc{Rigid}_1(c)$ imparts on the relative motion of $p_1(c)$ and $p_2(c)$.

To formulate the constraints that $\textsc{Rigid}_1$ and $\textsc{Rigid}_2$ impart on the relative motion of a pair of polygons, first let $\rho(q,l)$ denote the fraction of the distance that a point q on a line segment l is between its endpoints $q_1(l)$ and $q_2(l)$.

$$\rho(q,l) \triangleq \frac{||q - q_1(l)||}{||q_2(l) - q_1(l)||}$$

Given this, the constraint that $\textsc{Rigid}_1$ imparts on the relative motion of a pair of polygons is simply $\dot{\rho}(q(c), \gamma_1(c)) = 0$, i.e., the constraint that the contact point $q(c)$ not move along $l_1(c)$. This is illustrated in Fig. 20. Similarly, the constraint that $\textsc{Rigid}_2$ imparts on the relative motion of a pair of polygons is simply $\dot{\rho}(q(c), \gamma_2(c)) = 0$, i.e., the constraint that the contact point $q(c)$ not move along $l_2(c)$.

These constraints can be fleshed out as follows. Let $\alpha(c)$ be the vector

$$\big[x(q(p_1(c))), y(q(p_1(c))), \theta(p_1(c)), x(q(p_2(c))), y(q(p_2(c))), \theta(p_2(c))\big].$$

Given this, $\dot{\alpha}(c)$ is simply

$$\big[x(\dot{q}(p_1(c))), y(\dot{q}(p_1(c))), \dot{\theta}(p_1(c)), x(\dot{q}(p_2(c))), y(\dot{q}(p_2(c))), \dot{\theta}(p_2(c))\big],$$

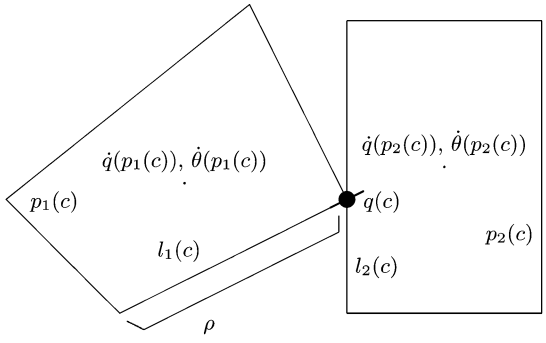
namely the six variables representing the instantaneous motion of $p_1(c)$ and $p_2(c)$, the two polygons joined by c. By the chain rule:

$$\dot{\rho}\big(q(c),\gamma_1(c)\big) = \frac{\partial\rho(q(c),\gamma_1(c))}{\partial\alpha(c)}\cdot\dot{\alpha}(c),$$
$$\dot{\rho}\big(q(c),\gamma_2(c)\big) = \frac{\partial\rho(q(c),\gamma_2(c))}{\partial\alpha(c)}\cdot\dot{\alpha}(c).$$

Both $\partial\rho(q(c),\gamma_1(c))/\partial\alpha(c)$ and $\partial\rho(q(c),\gamma_2(c))/\partial\alpha(c)$ are constant vectors whose values can be computed solely from the observed positions of the vertices of $p_1(c)$ and $p_2(c)$. It is straightforward to derive analytic expressions for these constant vectors using a computer-algebra system such as MAPLE. The actual expressions are not shown here as they are large. Note, however, that since these coefficient vectors are constant, RIGID_1 and RIGID_2 impart linear constraints on the instantaneous motion of a pair of polygons.

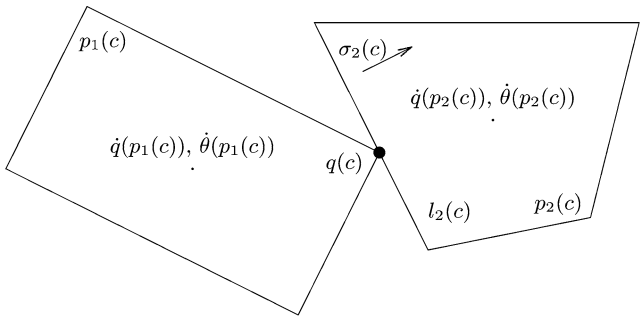
Note that the above requires computing the derivative of $q(c)$. This is not possible when c is collinear. Thus we only allow joints at noncollinear contacts. This does not restrict the expressive power of the framework because we only consider convex polygons. Without loss of generality, adjacent collinear edges of a polygon can be merged. Thus collinear contacts are all corner-to-edge, edge-to-corner, or corner-to-corner contacts. As shown in Fig. 13, all such contacts come in pairs or quadruples. Each collinear contact is paired with a noncollinear contact with the same affordances. Let $C'(P)$ denote the noncollinear contacts in $C(P)$. We retain collinear contacts in $C(P)$ because, as will become apparent later, we need collinear contacts to enforce the substantiality constraint.

A special situation arises with prismatic joints at corner-to-edge, edge-to-corner, and corner-to-corner contacts. If $\text{CORNER}_1(c)$, as illustrated in Fig. 21, even if c is prismatic along $l_1(c)$, i.e., $\neg\text{RIGID}_1(c),\ q(c)$ cannot move off of the end of $l_1(c)$. This manifests itself as the constraint that $\dot{\rho}(q(c),\gamma_1(c)) \geqslant 0$ when $\rho(q(c),l_1(c)) = 0$ and $\dot{\rho}(q(c),\gamma_1(c)) \leqslant 0$ when $\rho(q(c),l_1(c)) = 1$. Similarly, if $\text{CORNER}_2(c)$, this manifests



$$(\{[\text{RIGID}_2(c) \vee \text{RIGID}_\theta(c)] \wedge \rho(q(c),l_1(c)) = 0\} \rightarrow \dot{\rho}(q(c),\gamma_1(c)) \geq 0) \wedge$$
$$(\{[\text{RIGID}_2(c) \vee \text{RIGID}_\theta(c)] \wedge \rho(q(c),l_1(c)) = 1\} \rightarrow \dot{\rho}(q(c),\gamma_1(c)) \leq 0)$$

Fig. 21. A prismatic joint at a corner-to-edge contact. Even though c is prismatic along $l_1(c)$, because $\neg\text{RIGID}_1(c),\ q(c)$ cannot move off of the end of $l_1(c)$.



$$\textsc{SameLayer}(p_1(c), p_2(c)) \rightarrow \textsc{Outside}_2(c) \rightarrow \dot{q}_1(c) \cdot \sigma_2(c) \leq \dot{q}_2(c) \cdot \sigma_2(c)$$

$p_1(c)$	one polygon in contact
$p_2(c)$	the other polygon in contact
$l_2(c)$	the edge of $p_2(c)$ in contact
$q(c)$	the contact point
$q_1(c)$	the contact point $q(c)$ expressed in terms of the position and orientation of $p_1(c)$
$q_2(c)$	the contact point $q(c)$ expressed in terms of the position and orientation of $p_2(c)$
$\dot{q}_1(c)$	the velocity of $q_1(c)$ resulting from the motion of $p_1(c)$
$\dot{q}_2(c)$	the velocity of $q_2(c)$ resulting from the motion of $p_2(c)$
$\sigma_2(c)$	a vector normal to $l_2(c)$ toward the center of $p_2(c)$

Fig. 22. An illustration of the constraint that $\textsc{SameLayer}(p_1(c), p_2(c))$ imparts on the relative motion of $p_1(c)$ and $p_2(c)$.

itself as the additional constraint that $\dot{\rho}(q(c), \gamma_2(c)) \geqslant 0$ when $\rho(q(c), l_2(c)) = 0$ and $\dot{\rho}(q(c), \gamma_2(c)) \leqslant 0$ when $\rho(q(c), l_2(c)) = 1$. Note that such constraints are also linear in the instantaneous motion of the constrained polygons.[6]

2.3. The substantiality constraint

The substantiality constraint prevents two polygons that are on the same layer from interpenetrating. From an instantaneous perspective, two polygons can interpenetrate only when they touch. Two convex polygons touch only at corner-to-edge, edge-to-corner, or corner-to-corner contacts. Let us first consider corner-to-edge contacts. This case is illustrated in Fig. 22. Let $q_1(c)$ and $q_2(c)$ be the contact point $q(c)$ expressed in terms of the position and orientation of $p_1(c)$ and $p_2(c)$ respectively.

$$q_1(c) \stackrel{\triangle}{=} \gamma\big(q(c), p_1(c)\big)$$
$$q_2(c) \stackrel{\triangle}{=} \gamma\big(q(c), p_2(c)\big)$$

[6] LEONARD does not currently implement these constraints.

With this, $\dot{q}_1(c)$ is the velocity of the corner of $p_1(c)$ that forms the contact while $\dot{q}_2(c)$ is the velocity of the contact point on the edge $l_2(c)$ of $p_2(c)$ that forms the contact. Let $\sigma_2(c)$ be a vector that is perpendicular to $l_2(c)$ in the direction from $q(c)$ toward $q(p_2(c))$.

$$\sigma(l) \triangleq \overline{q_2(l) - q_1(l)}$$
$$\sigma_2(c) \triangleq \big\{\sigma(\gamma_2(c)) \cdot \big[q(p_2(c)) - q(c)\big]\big\}\sigma\big(\gamma_2(c)\big)$$

Note that $\sigma_2(c)$ is a constant vector that can be computed solely from the observed positions of the vertices of $p_1(c)$ and $p_2(c)$. If the projection of $\dot{q}_1(c)$ along $\sigma_2(c)$ is greater than the projection of $\dot{q}_2(c)$ along $\sigma_2(c)$, a penetration will occur. Further note that a penetration will occur only when $p_1(c)$ is outside $p_2(c)$.

$$\begin{aligned}\textsc{Outside}_2(c) \triangleq\ & \big\{\big[q_1(l_1(c)) - q_1(l_2(c))\big] \cdot \sigma_2(c) \leqslant 0\big\} \\ & \wedge \big\{\big[q_2(l_1(c)) - q_1(l_2(c))\big] \cdot \sigma_2(c) \leqslant 0\big\}\end{aligned}$$

In other words, at corner-to-edge contacts c, where $\textsc{Outside}_2(c)$, $\textsc{SameLayer}(p_1(c), p_2(c))$ imparts the constraint that $\dot{q}_1(c) \cdot \sigma_2(c) \leqslant \dot{q}_2(c) \cdot \sigma_2(c)$.

Enforcing the substantiality constraint at edge-to-corner contacts is similar. Let $\sigma_1(c)$ be a vector that is perpendicular to $l_1(c)$ in the direction from $q(c)$ toward $q(p_1(c))$.

$$\sigma_1(c) \triangleq \big\{\sigma(\gamma_1(c)) \cdot \big[q(p_1(c)) - q(c)\big]\big\}\sigma(\gamma_1(c))$$
$$\begin{aligned}\textsc{Outside}_1(c) \triangleq\ & \big\{\big[q_1(l_2(c)) - q_1(l_1(c))\big] \cdot \sigma_1(c) \leqslant 0\big\} \\ & \wedge \big\{\big[q_2(l_2(c)) - q_1(l_1(c))\big] \cdot \sigma_1(c) \leqslant 0\big\}\end{aligned}$$

Note that $\sigma_1(c)$ is a constant vector that can be computed solely from the observed positions of the vertices of $p_1(c)$ and $p_2(c)$. If the projection of $\dot{q}_2(c)$ along $\sigma_1(c)$ is greater than the projection of $\dot{q}_1(c)$ along $\sigma_1(c)$, a penetration will occur. In other words, at edge-to-corner contacts c, where $\textsc{Outside}_1(c)$, $\textsc{SameLayer}(p_1(c), p_2(c))$ imparts the constraint that $\dot{q}_2(c) \cdot \sigma_1(c) \leqslant \dot{q}_1(c) \cdot \sigma_1(c)$.

These constraints can be fleshed out as follows. Let $\alpha_1(c)$ and $\alpha_2(c)$ be the vectors

$$\big[x\big(q(p_1(c))\big), y\big(q(p_1(c))\big), \theta(p_1(c))\big] \quad \text{and}$$
$$\big[x\big(q(p_2(c))\big), y\big(q(p_2(c))\big), \theta(p_2(c))\big]$$

respectively. Given this, $\dot{\alpha}_1(c)$ and $\dot{\alpha}_2(c)$ are simply

$$\big[x\big(\dot{q}(p_1(c))\big), y\big(\dot{q}(p_1(c))\big), \dot{\theta}(p_1(c))\big] \quad \text{and}$$
$$\big[x\big(\dot{q}(p_2(c))\big), y\big(\dot{q}(p_2(c))\big), \dot{\theta}(p_2(c))\big]$$

respectively, namely the six variables representing the instantaneous motion of $p_1(c)$ and $p_2(c)$. By the chain rule:

$$\dot{q}_1(c) = \frac{\partial q_1(c)}{\partial \alpha_1(c)} \cdot \dot{\alpha}_1(c),$$
$$\dot{q}_2(c) = \frac{\partial q_2(c)}{\partial \alpha_2(c)} \cdot \dot{\alpha}_2(c).$$

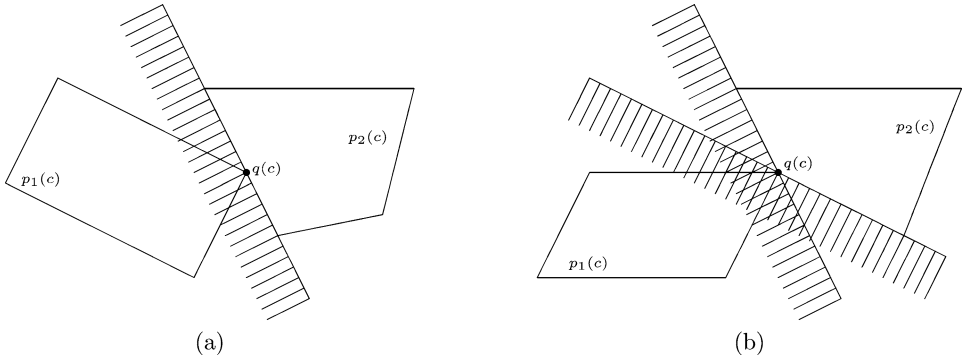


Fig. 23. (a) A substantiality constraint at a corner-to-edge contact c constrains the corner of $p_1(c)$ at the contact to move in the shaded half plane when $p_2(c)$ is fixed. (b) A substantiality constraint at a corner-to-corner contact c constrains the corner of $p_1(c)$ at the contact to move in the union of the two shaded half planes when $p_2(c)$ is fixed.

Like before, both $\partial q_1(c)/\partial \alpha_1(c)$ and $\partial q_2(c)/\partial \alpha_2(c)$ are constant vectors whose values can be computed solely from the observed positions of the vertices of $p_1(c)$ and $p_2(c)$. And again, it is straightforward to derive analytic expressions for these constant vectors. Note, however, that since $\sigma_1(c)$, $\sigma_2(c)$, and these coefficient vectors are constant, SAMELAYER imparts linear constraint on the instantaneous motion of a pair of polygons.

A substantiality constraint at a corner-to-edge or edge-to-corner contact constrains the relative motion of the corner to a half plane, as illustrated in Fig. 23(a). A substantiality constraint at a corner-to-corner contact is more complex. It constrains the relative motion of the corners to the union of two half planes, as illustrated in Fig. 23(b). Thus, at corner-to-corner contacts, SAMELAYER$(p_1(c), p_2(c))$ imparts the constraint that

$$\big[\dot{q}_2(c) \cdot \sigma_1(c) \leqslant \dot{q}_1(c) \cdot \sigma_1(c)\big] \vee \big[\dot{q}_1(c) \cdot \sigma_2(c) \leqslant \dot{q}_2(c) \cdot \sigma_2(c)\big].$$

While the individual constraints above are linear in the instantaneous motion of the constrained polygons, the disjunction does not correspond to a set of linear constraints. Thus substantiality constraints at corner-to-corner contacts cannot be handled by the linear-programming techniques that will be described momentarily. Depending on the situation, LEONARD may instantiate just one of the disjuncts or may instantiate the conjunction of the two disjuncts. This can result in unsound stability judgments, unsound judgments of instability in the former and unsound judgments of stability in the latter.

There is one common case, however, where corner-to-corner contacts c lead to motion constrained by a single half plane instead of by the union of two half planes. This case occurs when an edge of $p_1(c)$ overlaps an edge of $p_2(c)$ and is illustrated in Fig. 24(a). Correct handling of such cases, however, requires some additional machinery. The constraint $\dot{q}_1(c) \cdot \sigma_2(c) \leqslant \dot{q}_2(c) \cdot \sigma_2(c)$ correctly prevents $p_1(c)$ from moving in the direction of $\sigma_2(c)$ to penetrate $p_2(c)$. But the constraint $\dot{q}_2(c) \cdot \sigma_1(c) \leqslant \dot{q}_1(c) \cdot \sigma_1(c)$ would incorrectly prevent $p_1(c)$ from sliding along the surface of $p_2(c)$ in the direction of $\sigma_1(c)$. Thus we need to instantiate the former but not the latter. The way this is handled is illustrated in Figs. 24(b)–(d). We want to prevent motion of $l_1(c)$ along $\sigma_2(c)$ in Fig. 24(b) but not in Figs. 24(c) and 24(d). We say that a corner-to-corner contact c is *adjacent*

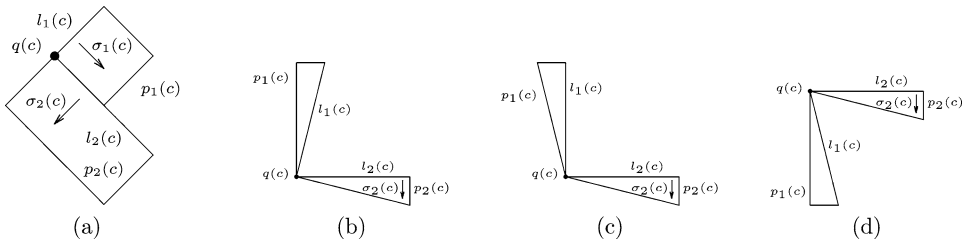


Fig. 24. (a) A corner-to-corner contact where motion is constrained by a single half plane instead of by the union of two half planes. Polygon $p_1(c)$ can slide along $p_2(c)$ in the direction of $\sigma_1(c)$ but cannot penetrate $p_2(c)$ by moving in the direction of $\sigma_2(c)$. (b) A situation where contact c is adjacent along $\sigma_2(c)$ so motion of $p_1(c)$ along $\sigma_2(c)$ would lead to penetration. (c), (d) Situations where contact c is not adjacent along $\sigma_2(c)$ so motion of $p_1(c)$ along $\sigma_2(c)$ would not lead to penetration.

along σ_2, denoted $\text{ADJACENT}_2(c)$, when motion of $p_1(c)$ in the direction of $\sigma_2(c)$ would penetrate $p_2(c)$ when $p_2(c)$ is stationary. The notion of adjacency along σ_1 can be defined similarly. The following definitions make these notions precise:

$$\text{ADJACENT}_2(c) \triangleq$$
$$\left[\begin{array}{l}
\left(q_1(l_1(c)) = q_1(l_2(c)) \rightarrow \left\{\begin{array}{l}[(q_2(l_1(c)) - q_1(l_1(c))) \cdot (q_2(l_2(c)) - q_1(l_2(c))) > 0] \wedge \\ [(q_2(l_1(c)) - q_1(l_1(c))) \cdot \sigma_2(c) \leqslant 0]\end{array}\right)\right) \wedge \\
\left(q_2(l_1(c)) = q_1(l_2(c)) \rightarrow \left\{\begin{array}{l}[(q_1(l_1(c)) - q_2(l_1(c))) \cdot (q_2(l_2(c)) - q_1(l_2(c))) > 0] \wedge \\ [(q_1(l_1(c)) - q_2(l_1(c))) \cdot \sigma_2(c) \leqslant 0]\end{array}\right)\right) \wedge \\
\left(q_1(l_1(c)) = q_2(l_2(c)) \rightarrow \left\{\begin{array}{l}[(q_2(l_1(c)) - q_1(l_1(c))) \cdot (q_1(l_2(c)) - q_2(l_2(c))) > 0] \wedge \\ [(q_2(l_1(c)) - q_1(l_1(c))) \cdot \sigma_2(c) \leqslant 0]\end{array}\right)\right) \wedge \\
\left(q_2(l_1(c)) = q_2(l_2(c)) \rightarrow \left\{\begin{array}{l}[(q_1(l_1(c)) - q_2(l_1(c))) \cdot (q_1(l_2(c)) - q_2(l_2(c))) > 0] \wedge \\ [(q_1(l_1(c)) - q_2(l_1(c))) \cdot \sigma_2(c) \leqslant 0]\end{array}\right)\right)
\end{array}\right],$$
$$\text{ADJACENT}_1(c) \triangleq$$
$$\left[\begin{array}{l}
\left(q_1(l_2(c)) = q_1(l_1(c)) \rightarrow \left\{\begin{array}{l}[(q_2(l_2(c)) - q_1(l_2(c))) \cdot (q_2(l_1(c)) - q_1(l_1(c))) > 0] \wedge \\ [(q_2(l_2(c)) - q_1(l_2(c))) \cdot \sigma_1(c) \leqslant 0]\end{array}\right)\right) \wedge \\
\left(q_2(l_2(c)) = q_1(l_1(c)) \rightarrow \left\{\begin{array}{l}[(q_1(l_2(c)) - q_2(l_2(c))) \cdot (q_2(l_1(c)) - q_1(l_1(c))) > 0] \wedge \\ [(q_1(l_2(c)) - q_2(l_2(c))) \cdot \sigma_1(c) \leqslant 0]\end{array}\right)\right) \wedge \\
\left(q_1(l_2(c)) = q_2(l_1(c)) \rightarrow \left\{\begin{array}{l}[(q_2(l_2(c)) - q_1(l_2(c))) \cdot (q_1(l_1(c)) - q_2(l_1(c))) > 0] \wedge \\ [(q_2(l_2(c)) - q_1(l_2(c))) \cdot \sigma_1(c) \leqslant 0]\end{array}\right)\right) \wedge \\
\left(q_2(l_2(c)) = q_2(l_1(c)) \rightarrow \left\{\begin{array}{l}[(q_1(l_2(c)) - q_2(l_2(c))) \cdot (q_1(l_1(c)) - q_2(l_1(c))) > 0] \wedge \\ [(q_1(l_2(c)) - q_2(l_2(c))) \cdot \sigma_1(c) \leqslant 0]\end{array}\right)\right)
\end{array}\right].$$

We only instantiate $\dot{q}_1(c) \cdot \sigma_2(c) \leqslant \dot{q}_2(c) \cdot \sigma_2(c)$ at a corner-to-corner contact c when c is adjacent along σ_2. Likewise, we only instantiate $\dot{q}_2(c) \cdot \sigma_1(c) \leqslant \dot{q}_1(c) \cdot \sigma_1(c)$ at a corner-to-corner contact c when c is adjacent along σ_1.

A further complication arises as illustrated in Fig. 25(a). In order to prevent p_1 from penetrating p_2, we need to prevent both q_1 and q_2 from moving in the direction of σ. To get σ, the contacts must include l_4. To create such a contact at q_1, the contact can include either l_1 or l_3. Unfortunately, however, a contact between l_1 and l_4 would not be adjacent along σ. Thus there must be a contact between l_3 and l_4. This is why we need collinear contacts to enforce substantiality and why a distinction must be made between $C(P)$ and $C'(P)$. This is not sufficient, however. A contact between l_3 and l_4 might be located at either q_1 or q_2. We need to prefer q_1 over q_2 as the location of the contact between l_3 and l_4 because q_2 is already the location of the contact between l_2 and l_4. Thus, for a

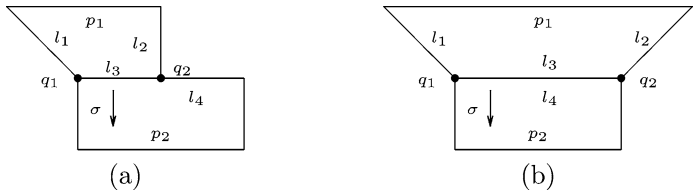


Fig. 25. (a) To prevent p_1 from penetrating p_2, contacts are needed at both q_1 and q_2 that include l_4 to get the direction σ. A contact at q_1 that includes l_4 and that is adjacent to σ must include l_3. This is why collinear contacts are needed for substantiality. Furthermore, q_1 must be preferred over q_2 as the contact location for the contact between l_3 and l_4 because q_2 is already the location of the contact between l_2 and l_4. (b) To prevent p_1 from penetrating p_2, contacts are needed at both q_1 and q_2 that include l_4 to get the direction σ. A contact at q_1 that includes l_4 and that is adjacent to σ must include l_3. A contact at q_2 that includes l_4 and that is adjacent to σ must also include l_3. Thus the only contact that can prevent penetration is the one between l_3 and l_4. But this contact can exist either at q_1 or q_2 but not both. And we need a contact at both q_1 and q_2. One is assigned to $q(c)$ and the other is assigned to $q'(c)$.

collinear contact c, we need to prefer a coincident endpoint of $l_1(c)$ and $l_2(c)$, if one exists, as $q(c)$.

An even further complication arises with adjacent corner-to-corner contacts as illustrated in Fig. 25(b). Like before, in order to prevent p_1 from penetrating p_2 we need to prevent both q_1 and q_2 from moving in the direction of σ. To get σ, the contacts must include l_4. A contact at q_1 that includes l_4 and that is adjacent along σ must include l_3. A contact at q_2 that includes l_4 and that is adjacent along σ must also include l_3. Thus the only contact that can prevent penetration is the one between l_3 and l_4. But this contact can exist either at q_1 or q_2 but not both. And we need a contact at both q_1 and q_2.

To address this issue, we define a contact c as *adjacent*, denoted $\text{ADJACENT}(c)$, when either $q_1(l_1(c)) = q_1(l_2(c))$ and $q_2(l_1(c)) = q_2(l_2(c))$ or $q_1(l_1(c)) = q_2(l_2(c))$ and $q_2(l_1(c)) = q_1(l_2(c))$. For adjacent contacts c, we let $q(c) \stackrel{\triangle}{=} q_1(l_1(c))$ and $q'(c) \stackrel{\triangle}{=} q_2(l_1(c))$. Then we define

$$q'_1(c) \stackrel{\triangle}{=} \gamma\big(q'(c), p_1(c)\big), \qquad q'_2(c) \stackrel{\triangle}{=} \gamma\big(q'(c), p_2(c)\big).$$

By the chain rule:

$$\dot{q}'_1(c) = \frac{\partial q'_1(c)}{\partial \alpha_1(c)} \cdot \dot{\alpha}_1(c), \qquad \dot{q}'_2(c) = \frac{\partial q'_2(c)}{\partial \alpha_2(c)} \cdot \dot{\alpha}_2(c).$$

Then we instantiate the additional constraints $\dot{q}'_2(c) \cdot \sigma_1(c) \leqslant \dot{q}'_1(c) \cdot \sigma_1(c)$ and $\dot{q}'_1(c) \cdot \sigma_2(c) \leqslant \dot{q}'_2(c) \cdot \sigma_2(c)$ when $\text{SAMELAYER}(p_1(c), p_2(c))$.

2.4. The potential energy of a scene

Now, let us compute the potential energy of a scene. The potential energy of a scene is the sum of the potential energies of the polygons in the scene. The potential energy of a polygon p is proportional to the product of its mass $m(p)$ times its vertical position.

$$E(P) \propto \sum_{p \in P} m(p) y\big(q(p)\big) \tag{1}$$

Let us assume that the mass of a polygon is uniformly distributed and that all polygons have the same density. Thus the mass of a polygon is proportional to its area and is a constant that can be computed solely from the observed positions of its vertices. Differentiating (1) gives:

$$\dot{E}(P) = \sum_{p \in P} m(p) y\big(\dot{q}(p)\big).$$

Note that the derivative of the potential energy of a scene is linear in the collective instantaneous motion of the polygons in the scene.

2.5. Reduction to linear programming

For a scene to be stable, it must be impossible to assign instantaneous velocities to the polygons in the scene that uphold the constraints imparted by the interpretation and that leads to a decrease in the potential energy. In other words:

$$\begin{aligned}
&\textsc{Stable}(P = \{p_1, \ldots, p_n\}, \langle \textsc{Grounded}, \textsc{Rigid}_1, \textsc{Rigid}_2, \textsc{Rigid}_\theta, \textsc{SameLayer} \rangle) \triangleq \\
&\quad \neg(\exists \dot{q}(p_1))(\exists \dot{\theta}(p_1)) \cdots (\exists \dot{q}(p_n))(\exists \dot{\theta}(p_n)) \\
&\quad \left[\begin{array}{l}
\big((\forall p \in P)\{\textsc{Grounded}(p) \rightarrow [\dot{q}(p) = 0 \wedge \dot{\theta}(p) = 0]\}\big) \wedge \\
\left(\begin{array}{l}
(\forall c \in C'(P)) \left\{ \begin{array}{l}
[\textsc{Rigid}_1(c) \rightarrow \dot{\rho}(q(c), \gamma_1(c)) = 0] \wedge \\
\left(\begin{array}{l}
[\textsc{Rigid}_2(c) \vee \textsc{Rigid}_\theta(c)] \rightarrow \\
\left\{ \begin{array}{l}
[\rho(q(c), l_1(c)) = 0 \rightarrow \dot{\rho}(q(c), \gamma_1(c)) \geqslant 0] \wedge \\
[\rho(q(c), l_1(c)) = 1 \rightarrow \dot{\rho}(q(c), \gamma_1(c)) \leqslant 0]
\end{array} \right.
\end{array} \right) \wedge \\
[\textsc{Rigid}_2(c) \rightarrow \dot{\rho}(q(c), \gamma_2(c)) = 0] \wedge \\
\left(\begin{array}{l}
[\textsc{Rigid}_1(c) \vee \textsc{Rigid}_\theta(c)] \rightarrow \\
\left\{ \begin{array}{l}
[\rho(q(c), l_2(c)) = 0 \rightarrow \dot{\rho}(q(c), \gamma_2(c)) \geqslant 0] \wedge \\
[\rho(q(c), l_2(c)) = 1 \rightarrow \dot{\rho}(q(c), \gamma_2(c)) \leqslant 0]
\end{array} \right.
\end{array} \right) \wedge \\
[\textsc{Rigid}_\theta(c) \rightarrow \dot{\theta}(p_1(c)) = \dot{\theta}(p_2(c))]
\end{array} \right\}
\end{array} \right) \wedge \\
\left(\begin{array}{l}
(\forall c \in C(P)) \left\{ \begin{array}{l}
\textsc{SameLayer}(p_1(c), p_2(c)) \rightarrow \\
\left[\begin{array}{l}
\left(\begin{array}{l}
\textsc{Outside}_2(c) \rightarrow \\
\left\{ \begin{array}{l}
\left[\left(\left\{ \begin{array}{l} \textsc{Corner}_1(c) \wedge \\ \left[\begin{array}{l} \textsc{Corner}_2(c) \rightarrow \\ \textsc{Adjacent}_2(c) \end{array} \right] \end{array} \right\} \vee \textsc{Adjacent}(c) \right) \rightarrow \right] \wedge \\
\dot{q}_1(c) \cdot \sigma_2(c) \leqslant \dot{q}_2(c) \cdot \sigma_2(c) \\
[\textsc{Adjacent}(c) \rightarrow \dot{q}'_1(c) \cdot \sigma_2(c) \leqslant \dot{q}'_2(c) \cdot \sigma_2(c)]
\end{array} \right.
\end{array} \right) \wedge \\
\left(\begin{array}{l}
\textsc{Outside}_1(c) \rightarrow \\
\left\{ \begin{array}{l}
\left[\left(\left\{ \begin{array}{l} \textsc{Corner}_2(c) \wedge \\ \left[\begin{array}{l} \textsc{Corner}_1(c) \rightarrow \\ \textsc{Adjacent}_1(c) \end{array} \right] \end{array} \right\} \vee \textsc{Adjacent}(c) \right) \rightarrow \right] \wedge \\
\dot{q}_2(c) \cdot \sigma_1(c) \leqslant \dot{q}_1(c) \cdot \sigma_1(c) \\
[\textsc{Adjacent}(c) \rightarrow \dot{q}'_2(c) \cdot \sigma_1(c) \leqslant \dot{q}'_1(c) \cdot \sigma_1(c)]
\end{array} \right.
\end{array} \right)
\end{array} \right]
\end{array} \right\}
\end{array} \right) \\
\dot{E}(P) < 0
\end{array} \right].
\end{aligned}$$

Let $\mathbf{z} = [x(\dot{q}(p_1)), y(\dot{q}(p_1)), \dot{\theta}(p_1), \ldots, x(\dot{q}(p_n)), y(\dot{q}(p_n)), \dot{\theta}(p_n)]$, the collective motion of the polygons in the scene.[7] Because all of the above constraints are linear in this

[7] In the following, I treat vectors as single-row or single-column matrices as appropriate.

motion, the above set of constraints can be formulated as the following system of linear equations and inequalities:

$$\begin{aligned} A\mathbf{z} &= 0 \\ B\mathbf{z} &\leqslant 0 \\ \mathbf{c}\cdot\mathbf{z} &< 0. \end{aligned}$$

A scene is unstable if and only if this system has a solution.

Because of scale invariance, if it is possible to decrease the potential energy by a collective motion, it is also possible to decrease the potential energy by speeding up or slowing down that motion so long as the motion doesn't change direction. Thus if $\hat{\mathbf{z}}$ satisfies the above system, $k\hat{\mathbf{z}}$ also satisfies that system for any $k > 0$. We could try to determine the stability of a scene by asking whether

$$\begin{aligned} A\mathbf{z} &= 0 \\ B\mathbf{z} &\leqslant 0 \\ \mathbf{c}\cdot\mathbf{z} &= -k \end{aligned}$$

has a solution for some fixed positive k but this leads to a stiff satisfiability problem. Instead, we will reformulate the satisfiability problem as the following optimization problem:

$$\begin{aligned} \min\quad & \mathbf{c}\cdot\mathbf{z} \\ & A\mathbf{z} = 0 \\ & B\mathbf{z} \leqslant 0. \end{aligned}$$

This linear-programming problem has a particular characteristic: $\mathbf{z} = 0$ is always a feasible solution that leads to $\mathbf{c}\cdot\mathbf{z} = 0$. In other words, if there is no motion, a consistent interpretation remains consistent and there is no change in potential energy. Furthermore, because of scale invariance, if it is possible to reduce the potential energy at a given rate with a given motion, it is possible to increase the rate of potential-energy reduction arbitrarily by increasing the motion. Thus linear-programming problems of the above form have only two potential minima: 0 or $-\infty$.

The above linear-programming problem is not in canonical form because it does not constrain $\mathbf{z} \geqslant 0$. One can address this issue by replacing each $\mathbf{z}_i$ with two variables ${\mathbf{z}_i}^+$ and ${\mathbf{z}_i}^-$ such that $\mathbf{z}_i = {\mathbf{z}_i}^+ - {\mathbf{z}_i}^-$. Let $(A \mid B)$ denote the horizontal concatenation of the matrices A and B, let $(\mathbf{x} \mid \mathbf{y})$ denote the concatenation of the vectors $\mathbf{x}$ and $\mathbf{y}$, and let $\mathbf{z}' = (\mathbf{z}^+ \mid \mathbf{z}^-)$, $A' = (A \mid -A)$, $B' = (B \mid -B)$, and $\mathbf{c}' = (\mathbf{c} \mid -\mathbf{c})$. With this, the original linear-programming problem has the same solution as the following linear-programming problem:

$$\begin{aligned} \min\quad & \mathbf{c}'\cdot\mathbf{z}' \\ & A'\mathbf{z}' = 0 \\ & B'\mathbf{z}' \leqslant 0 \\ & \mathbf{z}' \geqslant 0. \end{aligned}$$

The above linear-programming problem is still not in canonical form because it contains the equational constraints $A'\mathbf{z}' = 0$. One might be tempted reformulate these constraints

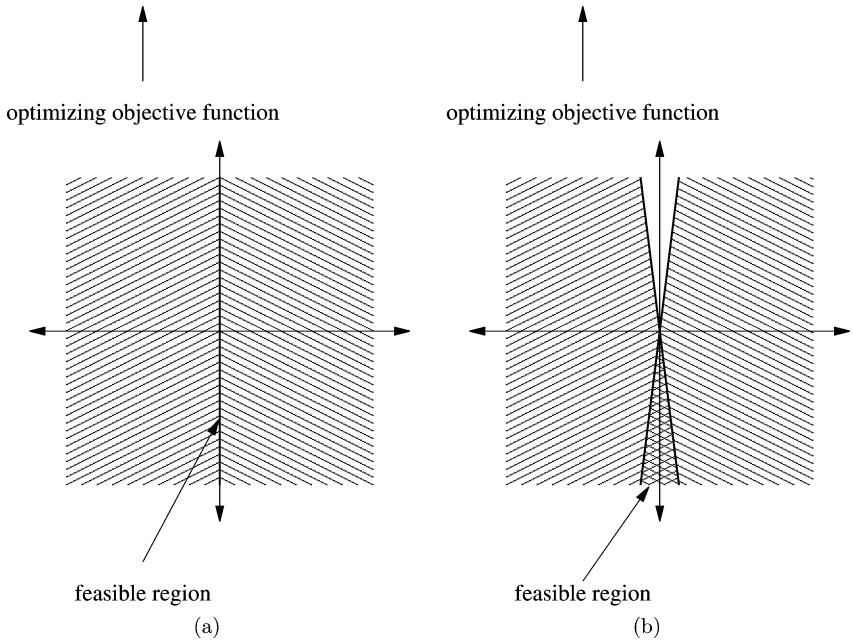


Fig. 26. An illustration of the numerical instability that results from treating equational constraints as pairs of inequalities. (a) shows two coincident hyperplanes with opposite half spaces. The feasible region consists of the hyperplane boundary between these two half spaces. The objective function is unbounded in the direction of optimization. (b) shows a slight pivoting of the two hyperplanes that can result from roundoff error. Now, the feasible region consists of the indicated triangular region and the optimal value of the objective function is at the origin. Thus roundoff error can affect stability judgments.

as the additional inequalities $A'\mathbf{z}' \leqslant 0$ and $-A'\mathbf{z}' \leqslant 0$. However, this leads to numerical instability for reasons illustrated in Fig. 26. While the linear-programming packages that I have tried, Numerical Recipes in C [80] and `lp_solve` [10], both nominally support equational constraints, they both exhibit numerical instability when given equational constraints. One can address this issue by eliminating equational constraints with a presolver. Ignoring issues of pivoting and matrices A' whose rank is less than the number of rows, a presolver finds a D such that $DA' = (I \mid E)$ for some E. Then let $\mathbf{z}' = (\mathbf{z}_1 \mid \mathbf{z}_2)$. Multiplying both sides of the constraint $A'\mathbf{z}' = 0$ with D gives $(I \mid E)(\mathbf{z}_1 \mid \mathbf{z}_2) = 0$. Thus $\mathbf{z}_1 = -E\mathbf{z}_2$. The constraint $B'\mathbf{z}' \leqslant 0$ becomes $(B_1 \mid B_2)(-E\mathbf{z}_2 \mid \mathbf{z}_2) \leqslant 0$ and the objective function $\mathbf{c}' \cdot \mathbf{z}'$ becomes $(\mathbf{c}_1 \mid \mathbf{c}_2) \cdot (-E\mathbf{z}_2 \mid \mathbf{z}_2)$. This leads to the following linear-programming problem:

$$\begin{aligned} \min \quad & \mathbf{c}'' \cdot \mathbf{z}_2 \\ & B''\mathbf{z}_2 \leqslant 0 \\ & \mathbf{z}_2 \geqslant 0 \end{aligned}$$

where $B'' = B_2 - B_1E$ and $\mathbf{c}'' = \mathbf{c}_2 - \mathbf{c}_1E$. This linear-programming problem is in canonical form.

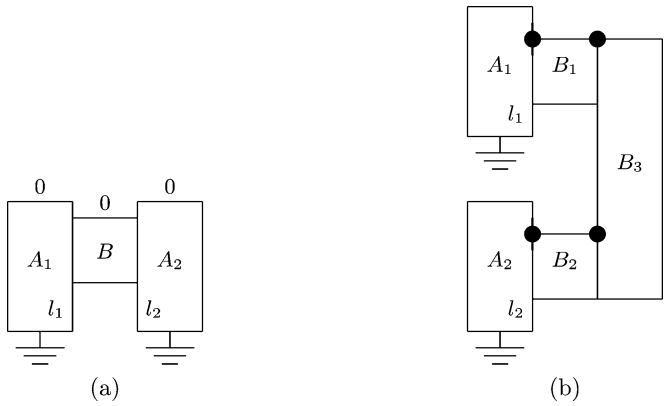


Fig. 27. An illustration of the numerical instability that results from stiff mechanisms. In (a), the substantiality constraint between B and edges l_1 of A_1 and l_2 of A_2 yields two coincident hyperplanes with opposite half spaces. B can fall only if edges l_1 and l_2 are precisely parallel. In (b), the two prismatic joints yield two coincident hyperplanes that result from two equational constraints. Again, the assembly consisting of B_1, B_2, and B_3 can fall only if edges l_1 and l_2 are precisely parallel.

The numerical instability described above is unavoidable with stiff mechanisms such as those illustrated in Fig. 27. Stiff mechanisms lead to linear-programming problems with infinitesimally narrow feasible regions. In Fig. 27(a), the substantiality constraint between B and edges l_1 of A_1 and l_2 of A_2 yields two coincident hyperplanes with opposite half spaces. B can fall only if edges l_1 and l_2 are precisely parallel. In Fig. 27(b), the two prismatic joints yield two coincident hyperplanes that result from two equational constraints. Again, the assembly consisting of B_1, B_2, and B_3 can fall only if edges l_1 and l_2 are precisely parallel. It is possible to address this numerical instability by solving the linear-programming problems using arbitrary-precision rational arithmetic. This is not done, however, in the current version of LEONARD.

2.6. Friction

Stiff mechanisms lead to linear-programming problems with infinitesimally narrow regions that lead to judgments of instability. Metastable situations, such as the one illustrated in Fig. 28, lead to linear-programming problems with infinitesimally narrow regions that lead to judgments of stability. The interpretation in Fig. 28(a) is stable only if the Table is perfectly horizontal. If the Table is tilted infinitesimally clockwise, as in Fig. 28(b), or counterclockwise, as in Fig. 28(c), block A will slide off to the right or left respectively. Situations such as these are pervasive. They occur whenever there are horizontal surfaces. Numerical instability and sensor noise imply that such situations would lead to incorrect judgments of instability.

One way to address this issue would be to model friction between surfaces in contact. Without friction, support by the substantiality constraint would be impossible. It would be necessary to attach objects to the horizontal surfaces that they are resting on. This is undesirable, since it would render the notion of substantiality useless. Thus we need some notion of friction. For reasons that will be discussed in Section 5, we choose not

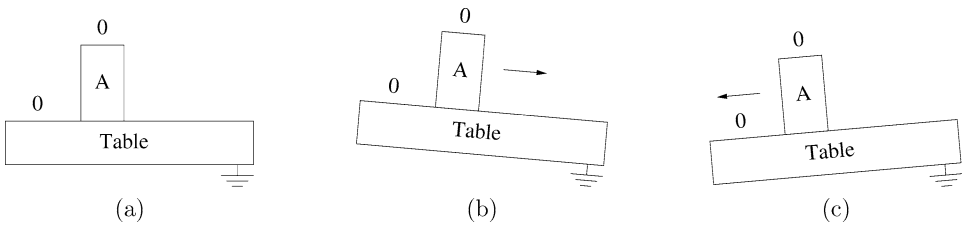
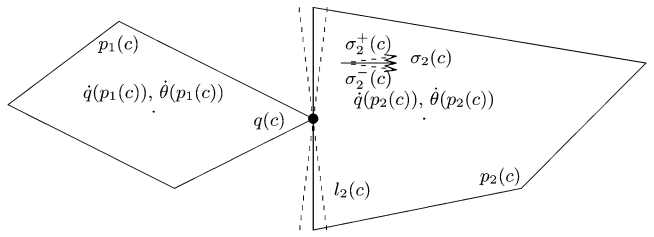


Fig. 28. An illustration of the numerical instability that results from a metastable situation. Interpretation (a) is stable only if the Table is perfectly horizontal. If the Table is tilted infinitesimally clockwise (b) or counterclockwise (c), block *A* will slide off to the right or left respectively.



$$\textsc{SameLayer}(p_1(c), p_2(c)) \rightarrow \textsc{Outside}_2(c) \rightarrow [(\dot{q}_1(c) \cdot \sigma_2^-(c) \leq \dot{q}_2(c) \cdot \sigma_2^-(c)) \wedge (\dot{q}_1(c) \cdot \sigma_2^+(c) \leq \dot{q}_2(c) \cdot \sigma_2^+(c))]$$

Fig. 29. An illustration of how the substantiality constraint is modified to model friction. Instead of instantiating the single inequality as described earlier, we instantiate two inequalities: one corresponding to a slight clockwise rotation of $l_2(c)$ about the point of contact and one corresponding to a slight counterclockwise rotation. This is done by instantiating the earlier inequality for two different $\sigma_2(c)$, namely $\sigma_2^-(c)$ and $\sigma_2^+(c)$, that correspond to a clockwise and counterclockwise rotation of $\sigma_2(c)$ by a tolerance θ respectively.

to model friction directly as a force. Instead, we model friction in the following way. Every time we instantiate a substantiality constraint at a contact c between a vertex $q(c)$ of polygon $p_1(c)$ and an edge $l_2(c)$ of polygon $p_2(c)$, instead of instantiating the single inequality as described earlier, we instantiate two inequalities: one corresponding to a slight clockwise rotation of $l_2(c)$ about the point of contact and one corresponding to a slight counterclockwise rotation. This is done, as illustrated in Fig. 29, by instantiating the two inequalities $\dot{q}_1(c) \cdot \sigma_2^-(c) \leqslant \dot{q}_2(c) \cdot \sigma_2^-(c)$ and $\dot{q}_1(c) \cdot \sigma_2^+(c) \leqslant \dot{q}_2(c) \cdot \sigma_2^+(c)$ where $\sigma_2^-(c)$ and $\sigma_2^+(c)$ denote $\sigma_2(c)$ rotated clockwise and counterclockwise by a tolerance θ respectively. Similarly, every time we instantiate a substantiality constraint at a contact c between a vertex $q(c)$ of polygon $p_2(c)$ and an edge $l_1(c)$ of polygon $p_1(c)$, instead of instantiating the single inequality as described earlier, we instantiate two inequalities: one corresponding to a slight clockwise rotation of $l_1(c)$ about the point of contact and one corresponding to a slight counterclockwise rotation. This is done by instantiating the two inequalities $\dot{q}_2(c) \cdot \sigma_1^-(c) \leqslant \dot{q}_1(c) \cdot \sigma_1^-(c)$ and $\dot{q}_2(c) \cdot \sigma_1^+(c) \leqslant \dot{q}_1(c) \cdot \sigma_1^+(c)$ where $\sigma_1^-(c)$ and $\sigma_1^+(c)$ denote $\sigma_1(c)$ rotated clockwise and counterclockwise by the same tolerance θ respectively.

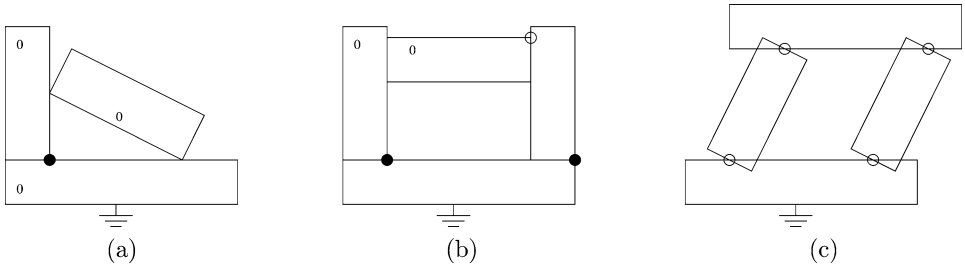


Fig. 30. Scenes and interpretations that contain closed-loop kinematic chains. LEONARD will claim that these interpretations are stable when they are in fact unstable.

2.7. Closed-loop kinematic chains

The stability-analysis techniques described in this section cannot handle closed-loop kinematic chains. A closed-loop kinematic chain arises when there is a sequence $p_1, \ldots, p_n$ of polygons where each pair $(p_i, p_{i+1 \bmod n})$ of polygons is constrained by a joint or substantiality constraint. Fig. 30 illustrates some scenes along with interpretations that contain closed-loop kinematic chains. Closed-loop kinematic chains can constrain motion in a nonlinear fashion. This can lead to unsound stability judgments when using the techniques described in this section. All such unsound judgments, however, err in the direction of claiming that an interpretation is stable when it is in fact unstable.

2.8. Tolerance

Often, due to sensor noise, roundoff error, and inaccuracies during segmentation and tracking, an image where two objects touch can be rendered into a scene where two polygons almost touch, as illustrated in Fig. 31(a), or slightly overlap, as illustrated in Fig. 31(b). Such situations will prevent the desired interpretation where the upper block is supported by the lower block by way of a substantiality constraint. In the case of almost-touching polygons, as illustrated in Fig. 31(a), this is because no contact is discovered between the almost-touching polygons because they do not actually intersect and thus no substantiality constraint is instantiated. In the case of slightly overlapping polygons, as illustrated in Fig. 31(b), this is because a same-layer constraint is disallowed because the polygons overlap. To deal with the latter, we allow two polygons to be on the same layer even when they slightly overlap. More precisely, two polygons are taken to overlap only when the area of their intersection is greater than some nonnegative tolerance ε_1. The formalization of the substantiality constraint continues to enforce the pretheoretic intention even when instantiated on polygons that slightly overlap. To deal with the former, we consider contacts between two almost-intersecting polygons. More precisely, we consider contacts c between $l_1(c)$ of $p_1(c)$ and $l_2(c)$ of $p_2(c)$ when $l_1(c)$ and $l_2(c)$ intersect according to the following. First, a point is taken to lie on a line segment when the distance from it to the line segment is less than some nonnegative tolerance ε_2. Second, two line segments are taken to intersect either when an endpoint of one lies on the other or when the lines that extend the line segments intersect and the point of intersection lies on both

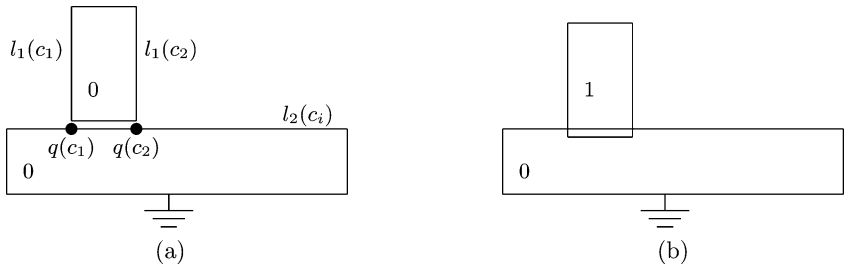


Fig. 31. Due to sensor noise, roundoff error, and inaccuracies during segmentation and tracking, an image where two objects touch can be rendered into a scene where two polygons almost touch, as illustrated in (a), or slightly overlap, as illustrated in (b). In (a), we entertain contacts c_1 and c_2 because the distance between their locations $q(c_1)$ and $q(c_2)$ and $l_1(c_1)$ and $l_2(c_2)$ respectively is less than some nonnegative tolerance ε_2. In (b), we entertain a same-layer assertion because the intersection area is less than some nonnegative tolerance ε_1.

line segments. For example, in the case of Fig. 31(a), we consider the contacts c_1, with the indicated edges $l_1(c_1)$ and $l_2(c_1)$, and c_2, with the indicated edges $l_1(c_2)$ and $l_2(c_2)$. Even though the edges that form the contact do not intersect, the lines on which the edges lie do intersect and yield a virtual location $q(c)$ for the contact. The definitions of $\text{CORNER}_1(c)$ and $\text{CORNER}_2(c)$ must be suitably modified according to the above definition of points lying on line segments. Similarly, the definitions of $\text{ADJACENT}_1(c)$, $\text{ADJACENT}_2(c)$, and $\text{ADJACENT}(c)$ must also be suitably modified to consider two points to be coincident when the distance between them is less than the same nonnegative tolerance ε_2. Again, the formalization of the joint and substantiality constraints continues to enforce the pretheoretic intention even when instantiated on such virtual contact locations.

One further modification must be made to deal with situations of almost-touching and slightly overlapping polygons. The antecedents $\rho(q(c), l_1(c)) = 0$, $\rho(q(c), l_1(c)) = 1$, $\rho(q(c), l_2(c)) = 0$, and $\rho(q(c), l_2(c)) = 1$ that enforce the constraints that prevent prismatic joints from separating must be replaced with $|\rho(q(c), l_1(c))| < \varepsilon$, $|\rho(q(c), l_1(c)) - 1| < \varepsilon$, $|\rho(q(c), l_2(c))| < \varepsilon$, and $|\rho(q(c), l_2(c)) - 1| < \varepsilon$ respectively, for some nonnegative tolerance ε.

3. Model reconstruction

Fig. 32 shows an approximation of the architecture of the model-reconstruction component of LEONARD. Nominally, this can be viewed as a generate-and-test architecture. The input consists of a sequence of scenes, one scene per video frame. Each scene consists of a set P of convex polygons. For each scene, the set of all possible interpretations is constructed. This set is filtered to remove inadmissible interpretations, leaving a set of admissible interpretations for each scene. This set of admissible interpretations for each frame is then filtered to remove unstable interpretations, leaving a set of stable admissible interpretations, or models, for each scene. This set of models for each scene is then filtered by the prioritized-circumscription process, leaving a set of minimal, or preferred, models for each scene. This set of preferred models for each scene is further filtered by the cardinality-circumscription process, leaving a set of more-preferred models for each

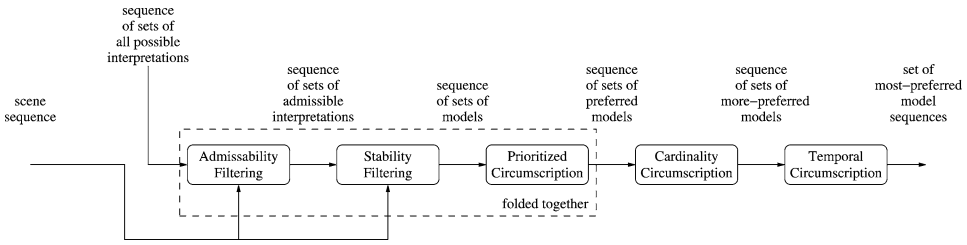


Fig. 32. The architecture of the model-reconstruction component in LEONARD.

scene. Finally, the sequence of sets of more-preferred models is filtered by the temporal-circumscription process to yield a set of most-preferred model sequences for each scene. If there is more than one most-preferred model sequence, one is chosen arbitrarily and the remainder are discarded. The actual architecture differs from this approximation in that the admissibility and stability filters are folded into prioritized circumscription in a way that will be described momentarily.

The cardinality-circumscription process is the same as described in Section 1 and needs no further elaboration. The admissibility criteria and the prioritized- and temporal-circumscription processes are somewhat more complex than what was described in Section 1. These are elaborated further below.

3.1. Admissibility

An interpretation is *admissible*, if it meets the following three criteria:

(1) There is no contact c for which RIGID(c) and REVOLUTE(c) are both true.
(2) Any two polygons that overlap must not be on the same layer. This follows from the substantiality constraint.
(3) The SAMELAYER relation must be an equivalence relation, i.e., it must be reflexive, symmetric, and transitive.

I denote admissibility by the predicate ADMISSIBLE(P, I). Note that the ADMISSIBLE predicate takes the scene P as an argument, in addition to the interpretation I, since the scene is necessary to determine whether two polygons overlap and this, in turn, is necessary to enforce the second criterion. In the discussion below, I use ADMISSIBLE(P) to denote partial application of the ADMISSIBLE predicate, i.e., ADMISSIBLE(P) $\stackrel{\triangle}{=}$ λI.ADMISSIBLE(P, I). Also note that admissibility of the SAMELAYER relation, i.e., the second and third criteria, can be checked independently of the RIGID and REVOLUTE properties. Accordingly, I overload the ADMISSIBLE predicate and apply it to SAMELAYER relations as well as interpretations.

3.2. Prioritized circumscription

Before presenting the prioritized-circumscription component, I will define some notation. GROUNDED is a one-argument predicate that ranges over polygons, elements

of P. RIGID and REVOLUTE are one-argument predicates that range over contacts, elements of $C'(P)$. SAMELAYER is a two-argument predicate that ranges over pairs of polygons, i.e., elements of $P \times P$. I will let r range over these predicates. Furthermore, I treat all such predicates in a set-theoretic fashion as relations, i.e., the set of all arguments, in the case of one-argument predicates, or pairs of arguments, in the case of two-argument predicates, that satisfy the predicate. I use $\bot_s$ to denote the empty relation, one that is false on all arguments from s, and $\top_s$ to denote the universal relation, one that is true on all arguments from s. More precisely, $\bot_P$ and $\top_P$ will qualify as empty and universal GROUNDED properties respectively, $\bot_{C'(P)}$ and $\top_{C'(P)}$ will qualify as empty and universal RIGID and REVOLUTE properties respectively, and $\bot_{P\times P}$ and $\top_{P\times P}$ will qualify as empty and universal SAMELAYER relations respectively. Furthermore, I use $r_1 \setminus r_2$, $r_1 \subset r_2$, and $r_1 \supset r_2$ with the standard set-theoretic meaning. Let the *parents* of a relation r be the smallest proper supersets of r and let the *children* of r be the largest proper subsets of r:

$$\textsc{Parents}(r) \stackrel{\triangle}{=} \left\{r' \mid (r \subset r') \wedge \left[\neg(\exists r'')(r \subset r'' \subset r')\right]\right\},$$

$$\textsc{Children}(r) \stackrel{\triangle}{=} \left\{r' \mid (r' \subset r) \wedge \left[\neg(\exists r'')(r' \subset r'' \subset r)\right]\right\}.$$

As will be seen later, some parents or children of an admissible relation might not be admissible. In this case, we will want to compute the smallest admissible proper supersets or the largest admissible proper subsets of r. More generally, if $\mathcal{P}$ is a predicate over relations, we will want to compute the smallest proper supersets or the largest proper subsets of r that satisfy $\mathcal{P}$:

$$\textsc{Parents}(r, \mathcal{P}) \stackrel{\triangle}{=} \left\{r' \mid (r \subset r') \wedge \mathcal{P}(r') \wedge \left[\neg(\exists r'')\mathcal{P}(r'') \wedge (r \subset r'' \subset r')\right]\right\},$$

$$\textsc{Children}(r, \mathcal{P}) \stackrel{\triangle}{=} \left\{r' \mid (r' \subset r) \wedge \mathcal{P}(r') \wedge \left[\neg(\exists r'')\mathcal{P}(r'') \wedge (r' \subset r'' \subset r)\right]\right\}.$$

Also, as will be seen later, $\bot_s$ or $\top_s$ might not be admissible. To deal with this, we define $\textsc{Ancestors}(\top_s, \mathcal{P})$ to be $\{\top_s\}$, if $\top_s$ satisfies $\mathcal{P}$, and $\textsc{Children}(\top_s, \mathcal{P})$, if not. Similarly, we define $\textsc{Descendents}(\bot_s, \mathcal{P})$ to be $\{\bot_s\}$, if $\bot_s$ satisfies $\mathcal{P}$, and $\textsc{Parents}(\bot_s, \mathcal{P})$, if not. Finally, let $\textsc{Circumscribe}(\mathcal{P})$ denote the smallest relations that satisfy $\mathcal{P}$:

$$\textsc{Circumscribe}(\mathcal{P}) \stackrel{\triangle}{=} \left\{r \mid \mathcal{P}(r) \wedge \left[\neg(\exists r')(r' \subset r) \wedge \mathcal{P}(r')\right]\right\}.$$

The prioritized-circumscription component finds all stable admissible interpretations that meet the criteria given in Fig. 33. The prioritized-circumscription component contains four circumscription levels. The first circumscribes over the GROUNDED property, the second over the RIGID property, the third over the REVOLUTE property, and the fourth

1. Find all GROUNDED such that:
   - the scene is stable for some RIGID and admissible SAMELAYER and
   - the scene is not stable for any proper subset of GROUNDED.
2. For each such GROUNDED, find all RIGID such that:
   - the scene is stable for some admissible SAMELAYER and
   - the scene is not stable for any proper subset of RIGID.
3. For each such RIGID, find all subsets REVOLUTE of RIGID such that:
   - the scene is stable for some admissible SAMELAYER when the joints in REVOLUTE are made revolute and
   - the scene is not stable when the joints in some proper superset of REVOLUTE are made revolute.
4. For each such REVOLUTE, find all admissible SAMELAYER such that:
   - the scene is stable and
   - the scene is not stable for any admissible proper subset of SAMELAYER.

Fig. 33. The prioritized-circumscription criteria used by LEONARD.

over the SAMELAYER relation. The circumscription processes at each level are similar. Let the predicates $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$, and $\mathcal{P}_4$ be defined as follows:

$$\begin{aligned}
&\mathcal{P}_1(\text{GROUNDED}) \triangleq \\
&\quad (\exists \text{RIGID})(\exists \text{REVOLUTE})(\exists \text{SAMELAYER}) \\
&\quad \text{ADMISSIBLE}(P, \langle \text{GROUNDED}, \text{RIGID}, \text{REVOLUTE}, \text{SAMELAYER}\rangle) \wedge \\
&\quad \text{STABLE}(P, \langle \text{GROUNDED}, \text{RIGID}, \text{REVOLUTE}, \text{SAMELAYER}\rangle), \\
&\mathcal{P}_2(\text{GROUNDED}, \text{RIGID}) \triangleq \\
&\quad (\exists \text{REVOLUTE})(\exists \text{SAMELAYER}) \\
&\quad \text{ADMISSIBLE}(P, \langle \text{GROUNDED}, \text{RIGID}, \text{REVOLUTE}, \text{SAMELAYER}\rangle) \wedge \\
&\quad \text{STABLE}(P, \langle \text{GROUNDED}, \text{RIGID}, \text{REVOLUTE}, \text{SAMELAYER}\rangle), \\
&\mathcal{P}_3(\text{GROUNDED}, \text{RIGID}, \text{REVOLUTE}) \triangleq \\
&\quad (\exists \text{SAMELAYER}) \\
&\quad \text{ADMISSIBLE}(P, \langle \text{GROUNDED}, \text{RIGID} \setminus \text{REVOLUTE}, \text{REVOLUTE}, \text{SAMELAYER}\rangle) \wedge \\
&\quad \text{STABLE}(P, \langle \text{GROUNDED}, \text{RIGID} \setminus \text{REVOLUTE}, \text{REVOLUTE}, \text{SAMELAYER}\rangle), \\
&\mathcal{P}_4(\text{GROUNDED}, \text{RIGID}, \text{REVOLUTE}, \text{SAMELAYER}) \triangleq \\
&\quad \text{ADMISSIBLE}(P, \langle \text{GROUNDED}, \text{RIGID} \setminus \text{REVOLUTE}, \text{REVOLUTE}, \text{SAMELAYER}\rangle) \wedge \\
&\quad \text{STABLE}(P, \langle \text{GROUNDED}, \text{RIGID} \setminus \text{REVOLUTE}, \text{REVOLUTE}, \text{SAMELAYER}\rangle).
\end{aligned}$$

Now, let $\mathcal{P}_2(\text{GROUNDED})$, $\mathcal{P}_3(\text{GROUNDED}, \text{RIGID})$, and $\mathcal{P}_4(\text{GROUNDED}, \text{RIGID}, \text{REVOLUTE})$ denote partial application of $\mathcal{P}_2$, $\mathcal{P}_3$, and $\mathcal{P}_4$ respectively. Given this, the prioritized-circumscription component computes the following:

$$\left\{ \left\langle \begin{array}{l} \text{GROUNDED}, \\ \text{RIGID}, \\ \text{REVOLUTE}, \\ \text{SAMELAYER} \end{array} \right\rangle \middle| \begin{array}{l} \text{GROUNDED} \in \text{CIRCUMSCRIBE}(\mathcal{P}_1) \wedge \\ \text{RIGID} \in \text{CIRCUMSCRIBE}(\mathcal{P}_2(\text{GROUNDED})) \wedge \\ \text{REVOLUTE} \in \text{CIRCUMSCRIBE}(\mathcal{P}_3(\text{GROUNDED}, \text{RIGID})) \wedge \\ \text{SAMELAYER} \in \text{CIRCUMSCRIBE}(\mathcal{P}_4(\text{GROUNDED}, \text{RIGID}, \text{REVOLUTE})) \end{array} \right\}.$$

Note that STABLE is monotonic in the GROUNDED, RIGID, REVOLUTE, and SAMELAYER components of I. For fixed RIGID, REVOLUTE, and SAMELAYER components, if a scene is stable for some GROUNDED, it is stable for all supersets GROUNDED′ of

GROUNDED, because grounding more objects cannot make a stable scene unstable. Similarly, for fixed GROUNDED and SAMELAYER components, if a scene is stable for some RIGID and REVOLUTE, it is stable when some nonattached contact is attached by a rigid or revolute joint or when some revolute joint is made rigid, because adding or strengthening joints cannot make a stable scene unstable. Finally, for fixed GROUNDED, RIGID, and REVOLUTE components, if a scene is stable for some SAMELAYER, it is stable for all supersets SAMELAYER′ of SAMELAYER, because adding a substantiality constraint cannot make a stable scene unstable.

The monotonicity of STABLE implies that $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$, and $\mathcal{P}_4$ can be computed with less quantification as follows:

$$\mathcal{P}_1(\text{GROUNDED}) \stackrel{\triangle}{=} \tag{2}$$
$$(\exists \text{SAMELAYER} \in \text{ANCESTORS}(\top_{P\times P}, \text{ADMISSIBLE}(P)))$$
$$\text{STABLE}(P, \langle \text{GROUNDED}, \top_{C'(P)}, \bot_{C'(P)}, \text{SAMELAYER}\rangle),$$
$$\mathcal{P}_2(\text{GROUNDED}, \text{RIGID}) \stackrel{\triangle}{=} \tag{3}$$
$$(\exists \text{SAMELAYER} \in \text{ANCESTORS}(\top_{P\times P}, \text{ADMISSIBLE}(P)))$$
$$\text{STABLE}(P, \langle \text{GROUNDED}, \text{RIGID}, \top_{C'(P)} \setminus \text{RIGID}, \text{SAMELAYER}\rangle),$$
$$\mathcal{P}_3(\text{GROUNDED}, \text{RIGID}, \text{REVOLUTE}) \stackrel{\triangle}{=} \tag{4}$$
$$(\exists \text{SAMELAYER} \in \text{ANCESTORS}(\top_{P\times P}, \text{ADMISSIBLE}(P)))$$
$$\text{STABLE}(P, \langle \text{GROUNDED}, \text{RIGID} \setminus \text{REVOLUTE}, \text{REVOLUTE}, \text{SAMELAYER}\rangle),$$
$$\mathcal{P}_4(\text{GROUNDED}, \text{RIGID}, \text{REVOLUTE}, \text{SAMELAYER}) \stackrel{\triangle}{=} \tag{5}$$
$$\text{STABLE}(P, \langle \text{GROUNDED}, \text{RIGID} \setminus \text{REVOLUTE}, \text{REVOLUTE}, \text{SAMELAYER}\rangle).$$

There are four differences between the above formulation and the earlier formulation. First, the admissibility check has been removed. This is sound because the prioritized-circumscription search process, combined with the above formulation, generates only admissible interpretations. Second, the monotonicity of STABLE allows replacing the quantification over RIGID and REVOLUTE in $\mathcal{P}_1$ with a single stability check using $\top_{C'(P)}$ for RIGID and $\bot_{C'(P)}$ for REVOLUTE. This corresponds to checking for stability when all contacts are rigidly joined. If the scene is not stable under such an interpretation, it cannot be stable with fewer or weaker joints. Third, the monotonicity of STABLE allows replacing the quantification over REVOLUTE in $\mathcal{P}_2$ with a single stability check using $\top_{C'(P)} \setminus$ RIGID. This corresponds to checking for stability when all contacts that were not attached rigidly are attached with revolute joints. If the scene is not stable under such an interpretation, it cannot be stable with fewer revolute joints. Finally, the monotonicity of STABLE allows quantifying over fewer SAMELAYER relations. Nominally, one would only need a single stability check using $\top_{P\times P}$. However, $\top_{P\times P}$ might not be admissible. Thus it is necessary to quantify over ANCESTORS($\top_{P\times P}$, ADMISSIBLE(P)).

The potential GROUNDED properties form a lattice with $\bot_P$ as the bottom element and $\top_P$ as the top element. Note that $\mathcal{P}_1(\bot_P)$ must be false, because a scene needs some grounded objects to be stable. And note that $\mathcal{P}_1(\top_P)$ must be true, because a scene where

all objects are grounded must be stable. Further note that $\mathcal{P}_1$ is monotonic. If $\mathcal{P}_1(r)$ is false, $\mathcal{P}_1(r')$ must be false for all $r' \subset r$. Likewise, if $\mathcal{P}_1(r)$ is true, $\mathcal{P}_1(r')$ must be true for all $r' \supset r$. Thus the lattice of potential GROUNDED properties is divided by a cut, where $\mathcal{P}_1$ is true for all elements above the cut and false for all elements below the cut. CIRCUMSCRIBE($\mathcal{P}_1$) finds those elements just above the cut.

Similarly, the potential RIGID properties form a lattice with $\bot_{C'(P)}$ as the bottom element and $\top_{C'(P)}$ as the top element. And for a given GROUNDED that satisfies $\mathcal{P}_1$, $\mathcal{P}_2(\text{GROUNDED}, \top_{C'(P)})$ must be true. Though $\mathcal{P}_2(\text{GROUNDED}, \bot_{C'(P)})$ might not be false. Furthermore, $\mathcal{P}_2(\text{GROUNDED})$ is monotonic. Thus either all potential RIGID properties satisfy $\mathcal{P}_2(\text{GROUNDED})$, in which case CIRCUMSCRIBE($\mathcal{P}_2(\text{GROUNDED})$) yields just $\bot_{C'(P)}$, or the lattice of potential RIGID properties is also divided by a cut, where $\mathcal{P}_2(\text{GROUNDED})$ is true for all elements above the cut and false for all elements below the cut, in which case CIRCUMSCRIBE($\mathcal{P}_2(\text{GROUNDED})$) finds those elements just above the cut. The same holds for the potential REVOLUTE properties and $\mathcal{P}_3(\text{GROUNDED}, \text{RIGID})$.

The fact that the GROUNDED, RIGID, and REVOLUTE properties form lattices with cuts allows circumscription over each of the levels $\mathcal{P}_1$, $\mathcal{P}_2(\text{GROUNDED})$, and $\mathcal{P}_3(\text{GROUNDED}, \text{RIGID})$, to be performed using either top-down or bottom-up search. Top-down search starts with the top of the lattice. At each stage, the current element is checked for stability. If it is unstable, the empty set is returned. If it is stable, form the union of the sets returned by recursively searching the children of the current element. Return this set, if it is nonempty. If this set is empty, return the singleton set containing the current element.

```
(define (top-down-monotonic-circumscription P s)
 (let LOOP ((r ⊤_s))
  (if P(r)
      (let ((R ⋃_{r'∈CHILDREN(r)} LOOP(r')))
       (if R = {}
           {r}
           R))
      {})))
```

Bottom-up search starts with the bottom of the lattice. At each stage, the current element is checked for stability. If it is unstable, return the union of the sets returned by recursively searching the parents of the current element. If it is stable and some child is also stable, return the empty set. If it is stable and no child is stable, return the singleton set containing the current element.

```
(define (bottom-up-monotonic-circumscription P s)
 (let LOOP ((r ⊥_s))
  (if P(r)
      (if (∃r' ∈ CHILDREN(r))P(r')
          {}
          {r})
      ⋃_{r'∈PARENTS(r)} LOOP(r'))))
```

With the above, CIRCUMSCRIBE($\mathcal{P}_1$) can be computed using either

(top-down-monotonic-circumscription $\mathcal{P}_1$ P)

or

(bottom-up-monotonic-circumscription $\mathcal{P}_1$ P).

CIRCUMSCRIBE($\mathcal{P}_2$(GROUNDED)) can be computed using either

(top-down-monotonic-circumscription $\mathcal{P}_2$(GROUNDED) $C'(P)$)

or

(bottom-up-monotonic-circumscription $\mathcal{P}_2$(GROUNDED) $C'(P)$).

And CIRCUMSCRIBE($\mathcal{P}_3$(GROUNDED, RIGID)) can be computed using either

(top-down-monotonic-circumscription
$\mathcal{P}_3$(GROUNDED, RIGID)$C'(P)$)

or

(bottom-up-monotonic-circumscription
$\mathcal{P}_3$(GROUNDED, RIGID)$C'(P)$).

The choice of which to use is motivated purely by efficiency. If one expects the cut to be closer to the top, top-down search is more likely to be the most efficient. If one expects the cut to be closer to the bottom, bottom-up search is more likely to be the most efficient.

The above search techniques work for levels one, two, and three because each level has unique admissible top and bottom elements and the parent and child functions preserve admissibility. This does not hold when circumscribing over the SAMELAYER relation. For this relation, the search techniques must be modified slightly as shown below:

$$\text{(define (top-down-monotonic-circumscription } \mathcal{P}\ \mathcal{Q}\ s\text{)}$$
$$\bigcup_{s' \in \text{ANCESTORS}(\top_s, \mathcal{Q})} \left(\begin{array}{l} \text{(let LOOP}((r\ s'))\ \text{(if } \mathcal{P}(r) \\ \quad \text{(let } ((R \bigcup_{r' \in \text{CHILDREN}(r,\mathcal{Q})} \text{LOOP}(r'))) \\ \quad\quad \text{(if } R = \{\} \\ \quad\quad\quad \{r\} \\ \quad\quad\quad R\text{))} \\ \quad \{\}\text{))} \end{array} \right),$$

$$\text{(define (bottom-up-monotonic-circumscription } \mathcal{P}\ \mathcal{Q}\ s\text{)}$$
$$\bigcup_{s' \in \text{DESCENDANTS}(\bot_s, \mathcal{Q})} \left(\begin{array}{l} \text{(let LOOP } ((r\ s'))\ \text{(if } \mathcal{P}(r) \\ \quad \text{(if } (\exists r' \in \text{CHILDREN}(r, \mathcal{Q}) \mathcal{P}(r') \\ \quad\quad \{\} \\ \quad\quad \{r\}\text{)} \\ \quad \bigcup_{r' \in \text{PARENTS}(r,\mathcal{Q})} \text{LOOP}(r')\text{))} \end{array} \right).$$

These contain three changes from the earlier formulation. First, they take an additional argument $\mathcal{Q}$ which is an admissibility predicate. Second, they union over searches commencing from all ancestors or descendants instead of searching from a single top or bottom element. Third, they use the two-argument versions of PARENTS and CHILDREN to preserve admissibility at each step in the search. The latter formulation generalizes the earlier formulation. The latter formulation reduces to the earlier formulation when $\mathcal{Q}$ is taken to be the universally-true predicate. With the latter formulation, CIRCUMSCRIBE($\mathcal{P}_4$(GROUNDED, RIGID, REVOLUTE)) can be computed using either

$$\begin{array}{l}\texttt{(top-down-monotonic-circumscription}\\ \quad \mathcal{P}_4(\text{GROUNDED}, \text{RIGID}, \text{REVOLUTE})\ \text{ADMISSIBLE}(P)\ P \times P\texttt{)}\end{array} \tag{6}$$

or

$$\begin{array}{l}\texttt{(bottom-up-monotonic-circumscription}\\ \quad \mathcal{P}_4(\text{GROUNDED}, \text{RIGID}, \text{REVOLUTE})\ \text{ADMISSIBLE}(P)\ P \times P\texttt{)}.\end{array} \tag{7}$$

The above algorithm considers all admissible SAMELAYER relations. It is easy to see, however, that whether or not two polygons are on the same layer can affect stability judgments only when those two polygons touch adjacently. Thus preferred models will never have two polygons on the same layer when they do not touch adjacently. Two polygons p_1 and p_2 touch adjacently, denoted TOUCHADJACENTLY(p_1, p_2), if they do not overlap and a substantiality constraint would be instantiated for some contact between them if they were on the same layer. The latter happens when

$$(\exists c \in C(P)) \left[\begin{array}{l} \left(\begin{array}{l} \{[q_1(l_2(c)) - q_1(l_1(c))] \cdot \sigma_1(c) \leqslant 0\} \wedge \{[q_2(l_2(c)) - q_1(l_1(c))] \cdot \sigma_1(c) \leqslant 0\} \wedge \\ \{[\text{CORNER}_2(c) \wedge (\text{CORNER}_1(c) \rightarrow \text{ADJACENT}_1(c))] \vee \text{ADJACENT}(c)\} \end{array} \right) \vee \\ \left(\begin{array}{l} \{[q_1(l_1(c)) - q_1(l_2(c))] \cdot \sigma_2(c) \leqslant 0\} \wedge \{[q_2(l_1(c)) - q_1(l_2(c))] \cdot \sigma_2(c) \leqslant 0\} \wedge \\ \{[\text{CORNER}_1(c) \wedge (\text{CORNER}_2(c) \rightarrow \text{ADJACENT}_2(c))] \vee \text{ADJACENT}(c)\} \end{array} \right) \end{array} \right].$$

The prioritized-circumscription process can be made more efficient if it takes advantage of this constraint. One way of doing so is to incorporate it into the admissibility predicate: a SAMELAYER relation is admissible only if every pair of polygons on the same layer touch adjacently. A somewhat more efficient way is to replace $P \times P$ with $\{\langle p_1, p_2\rangle \in P \times P \mid \text{TOUCHADJACENTLY}(p_1, p_2)\}$ in (2), (3), (4), and (6). It is not necessary to do the same replacement for (7) as s is only used in (7) to compute $\perp_s$ which is the empty set irrespective of whether s is $P \times P$ or $\{\langle p_1, p_2\rangle \in P \times P \mid \text{TOUCHADJACENTLY}(p_1, p_2)\}$.

It is also easy to see that preferred models will never have more than one rigid joint between two polygons, two polygons joined by both a rigid joint and a revolute joint, or two polygons joined by more than two noncolocational revolute joints. Furthermore, an interpretation with two noncolocational revolute joints between the same pair of polygons yields the same stability judgment as the same interpretation with the revolute joints replaced with a rigid joint at one of the contacts. And an interpretation with two colocational revolute joints between the same polygon pair yields the same stability judgment as the same interpretation with one of those joints removed. Thus a preferred model will never have more than one joint per polygon pair. The prioritized-circumscription process can be

made more efficient if it takes advantage of this constraint. This constraint can also be incorporated into the admissibility predicate.

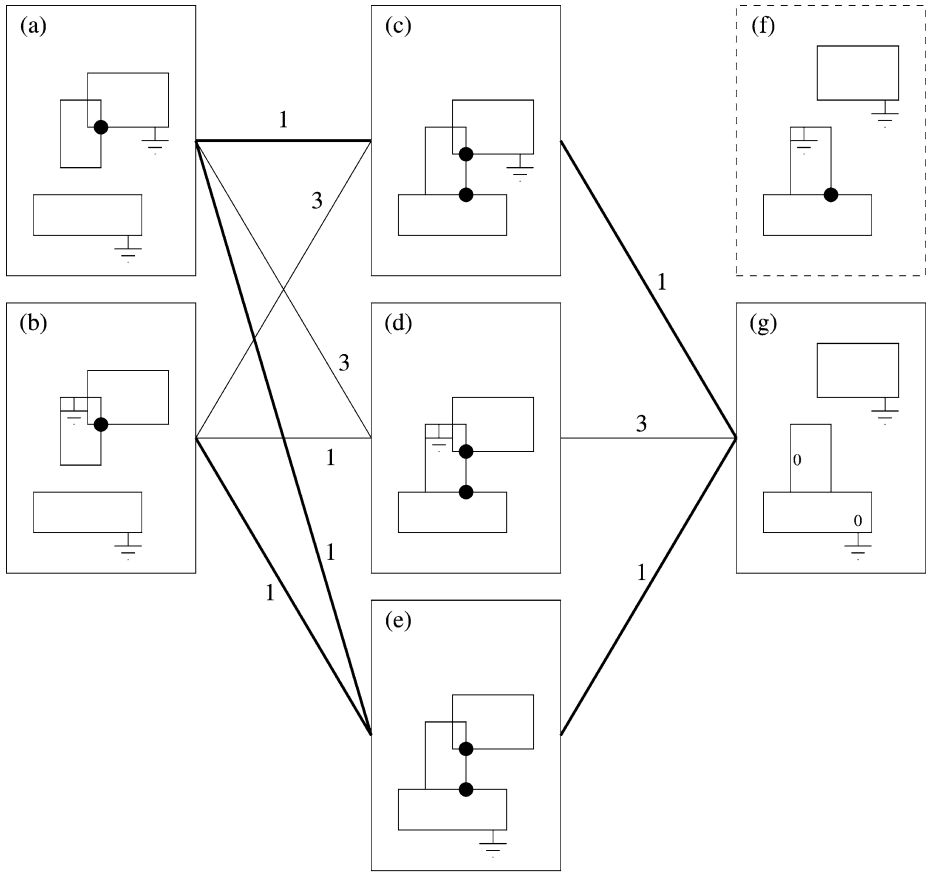
It is also easy to see that an interpretation with two rigidly attached polygons on the same layer yield the same stability judgment as the same interpretation with those polygons on different layers. Thus a preferred model will never have two rigidly attached polygons on the same layer. This constraint can also be incorporated into the admissibility predicate. To reduce the size of the search space at the expense of changing the prioritized-circumscription criteria, LEONARD extends this constraint so that two attached polygons, irrespective of whether they are attached by a rigid or a revolute joint, must not be on the same layer.

It is also easy to see that the stability judgment for an interpretation does not depend on which contact is used to rigidly join a polygon pair. Thus preferred models are isomorphic to ones which rigidly join a polygon pair at different contacts. Such isomorphic preferred models are redundant. The prioritized-circumscription process can be made more efficient if it avoids such redundancy by only considering rigid joints at a single contact per polygon pair. This constraint can also be incorporated into the admissibility predicate.

The stability judgment for an interpretation, however, does depend on which contact is used to attach a polygon pair by a revolute joint. While there can be at most two contacts per pair of convex polygons without tolerance, when nonzero tolerance is allowed, there can be more than two contacts per polygon pair. This can combinatorially increase the size of the search space that prioritized circumscription must consider. Because of this, LEONARD has an option `-fast` that disables consideration of revolute joints. When this option is enabled, $C'(P)$ need only contain one contact per polygon pair. This option also subsumes the earlier admissibility criterion of at most one joint per polygon pair.

3.3. Temporal circumscription

Section 1 suggested that temporal circumscription minimize state changes in the GROUNDED property as the criterion for preferring one model sequence over another. Fig. 34 illustrates why this does not always produce the desired outcome. This schema depicts a three-frame sequence where the hand holds a block in the first frame, places it on another block in the second frame, and releases it in the third frame. Except for variance in joint placement, (a) and (b) are the preferred models produced by prioritized circumscription for the first frame, (c), (d), and (e) are the preferred models of the second frame, and (f) and (g) are the preferred models of the third frame. Cardinality circumscription rules out (f) but leaves all of the remaining interpretations as more-preferred models for their corresponding frames. We desire a-c-g and a-e-g as the most-preferred model sequences. Using state changes in the GROUNDED property as the cost function for temporal circumscription, the a-c, a-e, b-d, b-e, c-g, and e-g transitions each involve a single state change. The remaining transitions each involve three state changes. Of the six possible model sequences, the highlighted ones, namely a-c-g, a-e-g, and b-e-g, all have the same minimal cost, namely two. Thus using state changes in the GROUNDED property as the minimization criterion for temporal circumscription is not able to prefer a-c-g or a-e-g over b-e-g.



a-c-g: $1+\frac{1}{2}+\frac{1}{3}+1+\frac{1}{2}=3\frac{1}{3}$
a-d-g: $1+\frac{1}{2}+1+1+\frac{1}{2}=4$
a-e-g: $1+\frac{1}{2}+1+\frac{1}{2}+\frac{1}{3}=3\frac{1}{3}$
b-c-g: $1+\frac{1}{2}+1+1+\frac{1}{2}=4$
b-d-g: $1+1+\frac{1}{2}+1+\frac{1}{2}=4$
b-e-g: $1+1+1+\frac{1}{2}+\frac{1}{3}=3\frac{5}{6}$

Fig. 34. An illustration of why state change in the GROUNDED property is undesirable as the cost function for temporal circumscription. Except for variance in joint placement, (a) and (b) are the preferred models produced by prioritized circumscription for the first frame of a three-frame sequence, (c), (d), and (e) are the preferred models of the second frame, and (f) and (g) are the preferred models of the third frame. Cardinality circumscription rules out (f) but leaves all of the remaining interpretations as more-preferred models for their corresponding frames. We desire a-c-g and a-e-g as the most-preferred model sequences. Using state changes in the GROUNDED property as the cost function for temporal circumscription, the a-c, a-e, b-d, b-e, c-g, and e-g transitions each involve a single state change. The remaining transitions each involve three state changes. Of the six possible model sequences, the highlighted ones, namely a-c-g, a-e-g, and b-e-g, all have the same minimal cost, namely two. Using state changes in the GROUNDED property is not able to prefer a-c-g or a-e-g over b-e-g. However, the alternate cost function yields the indicated metrics for the different model sequences. Thus the alternate cost function allows a preference of a-c-g and a-e-g over b-e-g as the most-preferred model sequences.

LEONARD, accordingly, adopts a different cost function for temporal circumscription.

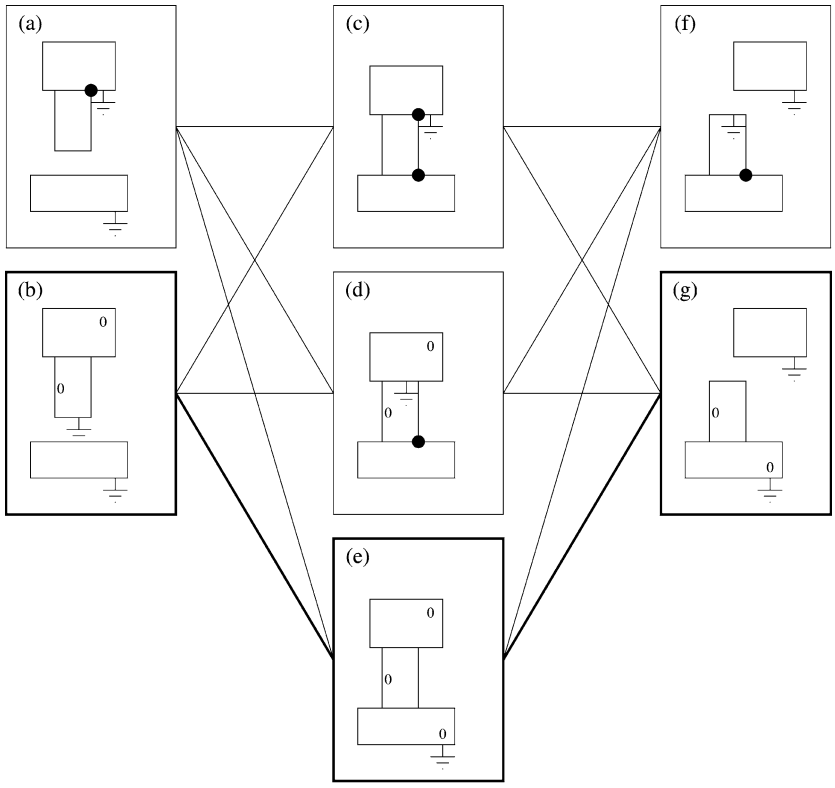
$$\sum_{i=1}^{\#\text{ objects}} \sum_{j=1}^{\#\text{frames where object } i \text{ is grounded}} \frac{1}{j}$$

This gives a cost of 1 for the first frame where an object is grounded, a cost of $1/2$ for the second frame, a cost of $1/3$ for the third frame, and so forth. This cost function prefers grounding an object that has already been grounded over one that has not and prefers grounding an object that has already been grounded more times over one that has already been grounded fewer times. Fig. 34 illustrates the computation of this alternate cost function for the different model sequences in that figure. With this alternate cost function, temporal circumscription gives the desired preference of a-c-g and a-e-g over b-e-g as the most-preferred model sequences.

Temporal circumscription searches the space of all model sequences. With multiple models per scene in the sequence, the number of model sequences is exponential in the length of the sequence. The optimal scene sequence, using either of the cost functions described above, can be found in polynomial time using the Viterbi algorithm [112].

Section 1 and Fig. 32 presented a circumscription pipeline where prioritized circumscription precedes cardinality circumscription which, in turn, precedes temporal circumscription. Each circumscription component further filters the set of models yielded by earlier components. An earlier version of LEONARD was implemented in this fashion. When I presented that system at AAAI-2000 [95], Reid Simmons asked whether it was possible with this strict pipeline for earlier circumscription components to remove models that would have allowed later circumscription components to produce better overall models. At the time, I had never observed this occur in practice and did not know if it could occur in theory. Subsequently, I discovered a situation where it could occur. Fig. 35 illustrates a schematized version of such a situation that actually occurred in video input. This schema depicts a three-frame sequence. This differs from the schema in Fig. 34 only in that the hand touches but does not overlap with the upper block. This allows models where the upper block supports the hand by the substantiality constraint without a joint. Except for variance in joint placement, (a) and (b) are the preferred models produced by prioritized circumscription for the first frame, (c), (d), and (e) are the preferred models of the second frame, and (f) and (g) are the preferred models of the third frame. We desire a-c-g and a-e-g as the most-preferred model sequences. Cardinality circumscription, however, rules out (a), (c), (d), and (f) as more-preferred models of their corresponding frames. Unlike the schema in Fig. 34, cardinality circumscription rules out (a) and (c) because the interpretations where the upper block supports the hand by substantiality require fewer joints than the interpretations where the hand is attached to the upper block. And (d) is ruled out because (e) has even fewer joints. Applying cardinality circumscription before temporal circumscription precludes adopting a-c-g and a-e-g as the most-preferred model sequences, because b-e-g is the only model sequence remaining after cardinality circumscription.

To deal with this situation, LEONARD now adopts a different circumscription pipeline. LEONARD first applies temporal circumscription to the sequence of sets of preferred models produced by prioritized circumscription to yield a single model sequence that minimizes the temporal-circumscription cost function. If there are multiple model



a-c-f:	$1+\frac{1}{2}+\frac{1}{3}+1+1=3\frac{5}{6}$	b-c-f:	$1+\frac{1}{2}+1+\frac{1}{2}+1=4$
a-c-g:	$1+\frac{1}{2}+\frac{1}{3}+1+\frac{1}{2}=3\frac{1}{3}$	b-c-g:	$1+\frac{1}{2}+1+1+\frac{1}{2}=4$
a-d-f:	$1+\frac{1}{2}+1+\frac{1}{2}+1=4$	b-d-f:	$1+1+\frac{1}{2}+\frac{1}{3}+1=3\frac{5}{6}$
a-d-g:	$1+\frac{1}{2}+1+1+\frac{1}{2}=4$	b-d-g:	$1+1+\frac{1}{2}+1+\frac{1}{2}=4$
a-e-f:	$1+\frac{1}{2}+1+1+\frac{1}{2}=4$	b-e-f:	$1+1+\frac{1}{2}+1+\frac{1}{2}=4$
a-e-g:	$1+\frac{1}{2}+1+\frac{1}{2}+\frac{1}{3}=3\frac{1}{3}$	b-e-g:	$1+1+1+\frac{1}{2}+\frac{1}{3}=3\frac{5}{6}$

Fig. 35. A situation where applying cardinality circumscription before temporal circumscription can produce undesired results. Except for variance in joint placement, (a) and (b) are the preferred models produced by prioritized circumscription for the first frame of a three-frame sequence, (c), (d), and (e) are the preferred models of the second frame, and (f) and (g) are the preferred models of the third frame. We desire a-c-g and a-e-g as the most-preferred model sequences. Cardinality circumscription, however, rules out (a), (c), (d), and (f) as more-preferred models of their corresponding frames. Applying cardinality circumscription before temporal circumscription precludes adopting a-c-g and a-e-g as the most-preferred model sequences, because b-e-g is the only model sequence remaining after cardinality circumscription. Applying temporal circumscription on the results of prioritized circumscription, before applying cardinality circumscription, allows selecting either a-c-g or a-e-g as the preferred model sequence.

sequences that minimize this cost function, one is selected arbitrarily. This yields a single model for each frame in the sequence. For each frame in the sequence, select the subset of the preferred models produced by prioritized circumscription on that frame that have the same GROUNDED property as the single model produced by temporal circumscription for that frame. Then filter this subset on a frame-by-frame basis using the cardinality-circumscription process described in Section 1. This will produce a set of more-preferred models for each frame. Finally, this sequence of sets of more-preferred models is fed into the temporal-circumscription process a second time to yield a set of most-preferred model sequences. For example, when applying this technique to the schema in Fig. 35, all of the preferred models (a)–(g) are fed into the first application of temporal circumscription. This arbitrarily selects either a-c-g or a-e-g as the best sequence. Suppose that it selects a-c-g. Then models (b), (d), (e), and (f) are discarded because they don't match the GROUNDED property of the corresponding frames in a-c-g. Cardinality circumscription and a second pass of temporal circumscription are then applied to the remaining models to yield the desired outcome of a-c-g. A similar result is obtained if the first pass of temporal circumscription arbitrarily selects the other alternative, a-e-g. In this simple example, the first pass of temporal circumscription removes all ambiguity, alleviating the need for cardinality circumscription and a second pass of temporal circumscription. In practice, however, situations arise where the first pass of temporal circumscription does not remove all ambiguity and subsequent application of cardinality circumscription and a second pass of temporal circumscription are necessary.

3.4. Implementation

The prioritized- and cardinality-circumscription components operate independently on each scene in the video sequence. Thus they can operate in parallel. In fact, the current implementation of LEONARD can run the prioritized-circumscription component in parallel when multiple (potentially distributed) processors are available. The cardinality-circumscription component has not been parallelized since it runs quickly. The temporal-circumscription component operates on the entire scene sequence and has also not been parallelized.

The current implementation of LEONARD does one further optimization when computing prioritized circumscription. Video sequences often have multiple-frame stretches where object positions change but the force-dynamic relations between the objects do not. Thus it is unnecessary to recompute the set of preferred models for a frame when it is the same as the one for a previous frame. To take advantage of this opportunity, LEONARD caches the sets of preferred models computed for earlier frames. Before running prioritized circumscription on a new frame, it first checks whether one of the cached sets is applicable to the new frame. A cached set is applicable if (a) the new frame has the same number of polygons as the cached frame, (b) the new frame has the same contact set as the cached frame, (c) each interpretation from the cached set of interpretations is admissible and stable in the new frame, and (d) no admissible child of any interpretation in the cached set is stable. If there is an applicable cached set, it is taken as the set of preferred models for the new frame. If not, prioritized circumscription is run to compute the set of preferred

PICKUP(x, y, z)	x picks y up off of z
PUTDOWN(x, y, z)	x puts y down on z
STACK(w, x, y, z)	w puts x down on y which is resting on z
UNSTACK(w, x, y, z)	w picks x up off of y which is resting on z
MOVE(w, x, y, z)	w picks x up off of y then puts it down on z
ASSEMBLE(w, x, y, z)	w puts y down on z then stacks x on y
DISASSEMBLE(w, x, y, z)	w unstacks x off of y then picks y up off of z

Fig. 36. Informal definition of the seven event types used to evaluate LEONARD.

models. Note however, that this optimization precludes parallel computation of prioritized circumscription.

4. Experimental results

To evaluate the methods described in this paper, seven event types were defined: *pick up*, *put down*, *stack*, *unstack*, *move*, *assemble*, and *disassemble*. An informal definition of these event types is given in Fig. 36. Thirty movies were filmed for each of the seven event types for a total of 210 movies. These movies were filmed using a Canon VC-C3 camera and a Matrox Meteor frame grabber at 320×240 resolution at 30 fps. A single subject (the author) performed all 210 event executions. For each event type, fifteen movies were filmed with the event being performed from the left and fifteen were filmed with the event being performed from the right. To simplify the analysis, an attempt was made to use the same colored blocks to fill the given roles of each event type, i.e., the events were all

PICKUP(**hand**, **red-block**, **green-block**),

PUTDOWN(**hand**, **red-block**, **green-block**),

STACK(**hand**, **red-block**, **green-block**, **blue-block**),

UNSTACK(**hand**, **red-block**, **green-block**, **blue-block**),

ASSEMBLE(**hand**, **red-block**, **blue-block**, **green-block**), or

DISASSEMBLE(**hand**, **red-block**, **blue-block**, **green-block**).

Due to experimenter error, this discipline was only partially followed for *move*. The move events from the left were all MOVE(**hand**, **red-block**, **green-block**, **blue-block**) while the move events from the right were all MOVE(**hand**, **red-block**, **blue-block**, **green-block**). The movies contain a total of 11946 frames.[8]

A real-time color- and motion-based segmentation algorithm was used to place a convex polygon around each colored and moving object in each frame. A tracking algorithm was then used to form a correspondence between the polygons in each frame and those in temporally adjacent frames. This tracker guarantees that each frame contains the same

[8] These movies, the source code for LEONARD, and scripts for reproducing this experiment are available from ftp://ftp.ecn.purdue.edu/qobi/leonard.tar.Z.

number of polygons and that they are ordered so that the ith polygon in each frame corresponds to the same object. The segmentation and tracking algorithms are extensions of the algorithms presented in Siskind and Morris [97] and Siskind [94], modified to place convex polygons around the participant objects instead of ellipses.

One movie (`assemble-left-qobi-04`, 100 frames) was discarded because the segmentation and tracking algorithms found only three participant objects and four are needed for an *assemble* event. The remaining scene sequences produced by the segmentation and tracking algorithms were then processed by the model-reconstruction procedure (prioritized, cardinality, and temporal circumscription) to obtain a most-preferred model sequence for each movie. For these runs, the `-fast` option and the tolerances $\varepsilon_1 = 1$, $\varepsilon_2 = 10$, and $\theta = 20^\circ$ were used.

LEONARD systematically produces models that differ from human intuition for a portion of all seven event types. This discrepancy is most easily illustrated for *pick up* events. *Pick up* events nominally consist of three phases as shown in Fig. 37. In the first phase, as shown in Fig. 37(a), the hand and lower block are grounded while the upper block is in contact with and on the same layer as the lower block. In the second phase, as shown in Fig. 37(b), the hand and lower block are grounded, and the upper and lower block are in contact. In the third phase, as shown in Fig. 37(c), the hand and lower block are grounded, the upper and lower block are not in contact, and the upper block

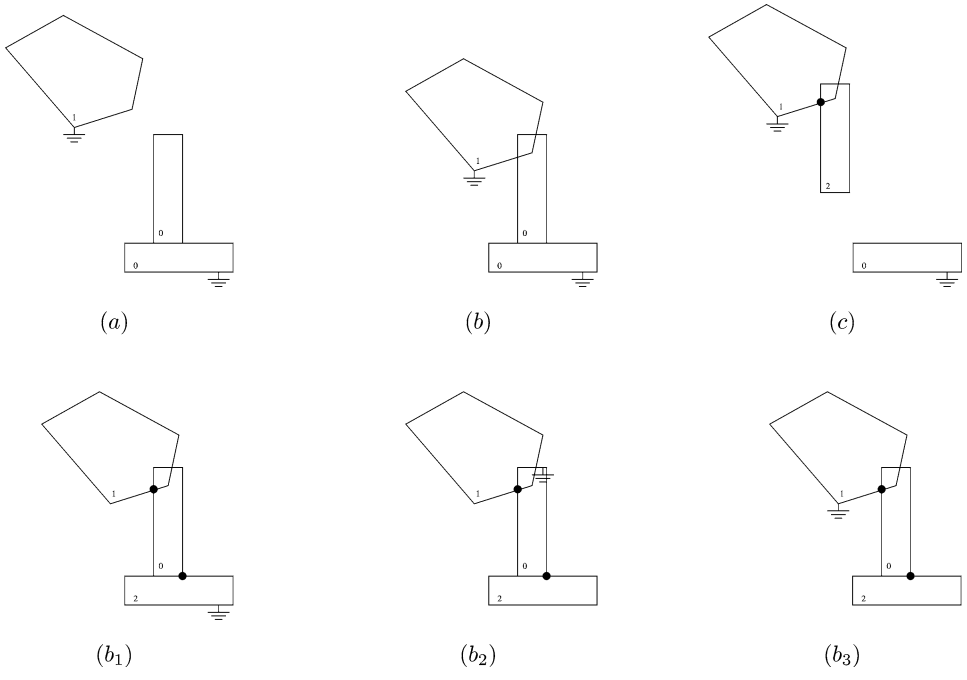


Fig. 37. An illustration of the 'interlude' problem for a *pick up* event. A *pick up* event consists of a sequence of models a, followed by a sequence of models b, followed by a sequence of models c. For the interlude, i.e., the b-portion, human intuition would suggest model b while the prioritized- and cardinality-circumscription components of LEONARD produce model b_1, b_2, or b_3.

is attached to the hand. For the 'interlude', i.e., the second phase, LEONARD produces one of the models in Figs. 37(b_1), 37(b_2), or 37(b_3) instead of the model in Fig. 37(b). In these models, the hand is attached to the upper block, the upper block is attached to the lower block, and either the hand, lower block, or upper block is grounded. Prioritized circumscription produces b, b_1, b_2, and b_3 as preferred models. However b_1, b_2, and b_3 each have lower cardinality-circumscription cost than b because b_1, b_2, and b_3 each have one groundedness assertion while b has two groundedness assertions. Thus cardinality circumscription prunes b leaving b_1, b_2, and b_3 as more-preferred models. (While temporal circumscription will eliminate b_2, both b_1 and b_3 remain as most-preferred models.) This same systematic error happens for all seven event types.

For each event type, the set of all distinct model types was collected from this output. There were 8, 8, 16, 16, 19, 38, and 36 distinct model classes for the event types *pick up*, *put down*, *stack*, *unstack*, *move*, *assemble*, and *disassemble* respectively. These were manually classified as to whether they are the intended or unintended models for the scenes that they represent. The spurious but systematic misinterpretation of interludes was considered intended. The event types *pick up*, *put down*, *stack*, *unstack*, *move*, *assemble*, and *disassemble* have 4, 4, 4, 4, 7, 9, and 9 intended models for different temporal portions of the event respectively. Fig. 38 depicts these intended models in the order in which they occur during an event occurrence. Note that there are multiple variants for each of the interludes: two variants of model b for *pick up*, *put down*, *stack*, *unstack*, *move*, and *disassemble*, two variants of model d for *move*, two variants of each of the models c and f for *assemble*, and two variants of model e for *disassemble*. These variants differ only in which one of a rigidly connected set of objects is grounded. Thus 4/8, 4/8, 4/16, 4/16, 7/19, 9/38, and 9/36 model classes were deemed intended for the event types *pick up*, *put down*, *stack*, *unstack*, *move*, *assemble*, and *disassemble* respectively. The model instances produced on a frame-by-frame basis for each of the movies were then labeled as either intended or unintended. 9875/11846 (83.4%) were labeled with the intended interpretation by this evaluation method. Fig. 38 illustrates the intended model classes and Figs. 39–40 illustrate the unintended model classes for each event type respectively. Figs. 41 and 42 give a breakdown of the number of frames of each intended and unintended model class for each event type respectively.

Eight error modes account for almost all of the unintended models, i.e., all except *move* x_2 and *assemble* x_7, x_8, and x_{23}. Seven of these error modes are illustrated in Fig. 43. Error mode VI, which is not illustrated, results from segmentation and tracking errors and not from model reconstruction. The error mode(s) associated with each unintended model class are illustrated in Fig. 42. Error mode I indicates an object being supported despite its center of mass being distant from the contact location. The object should fall over but does not. This is an unintended consequence of the method used to model friction discussed in Section 2.6. This problem can be ameliorated somewhat by using a smaller value for the tolerance θ. There is a limit, however, to how small a value for θ can be chosen. Too small a value will prevent an object from being supported on a slanted surface and require a joint instead of a substantiality constraint. Error mode II indicates an object being supported by a substantiality constraint from above instead of below. Error modes IIIa and IIIb indicate failure to determine a support relation. In the case of error mode IIIa, a joint replaces the

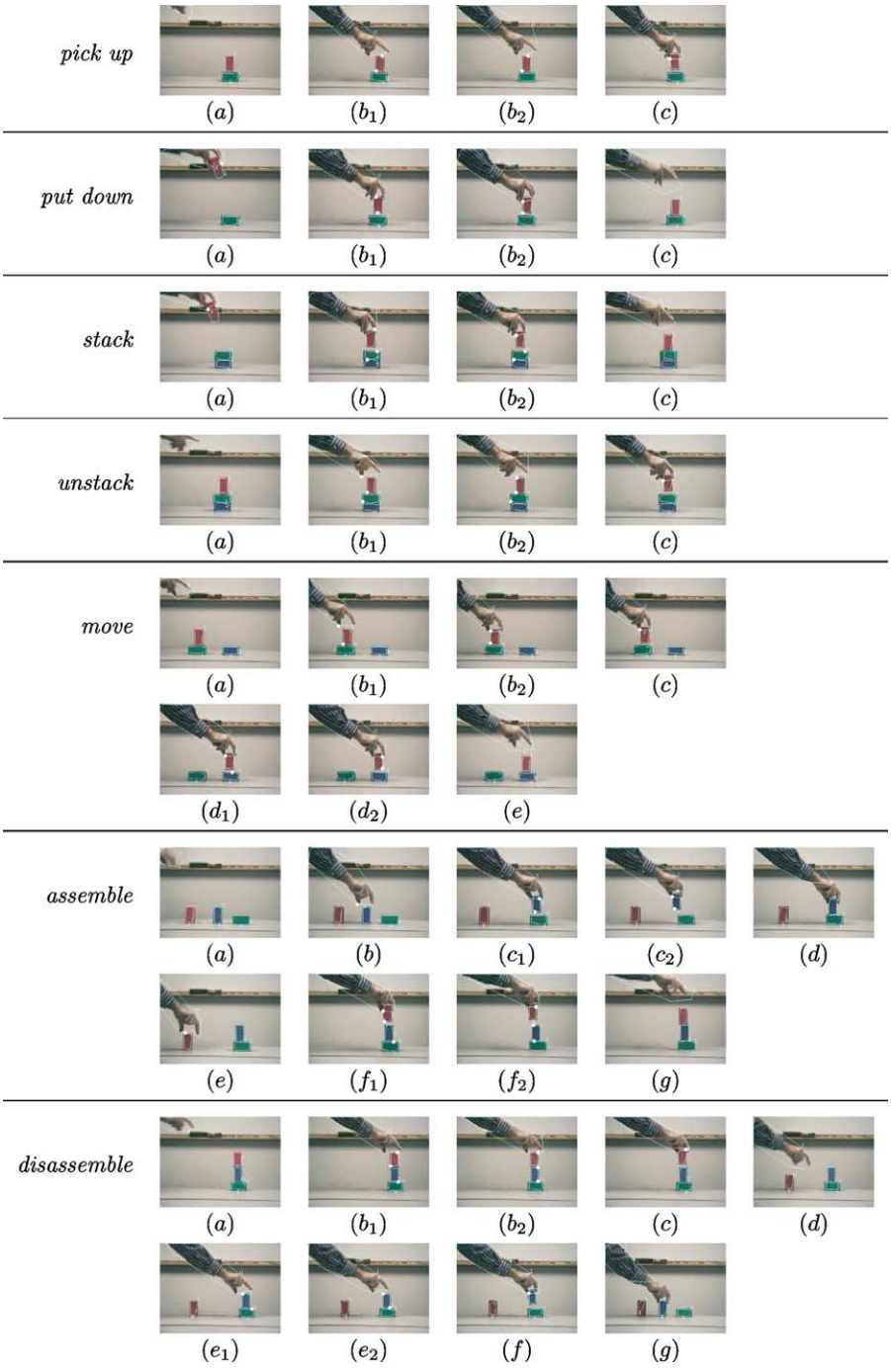


Fig. 38. The intended most-preferred model classes for each of the seven event types from Fig. 36.

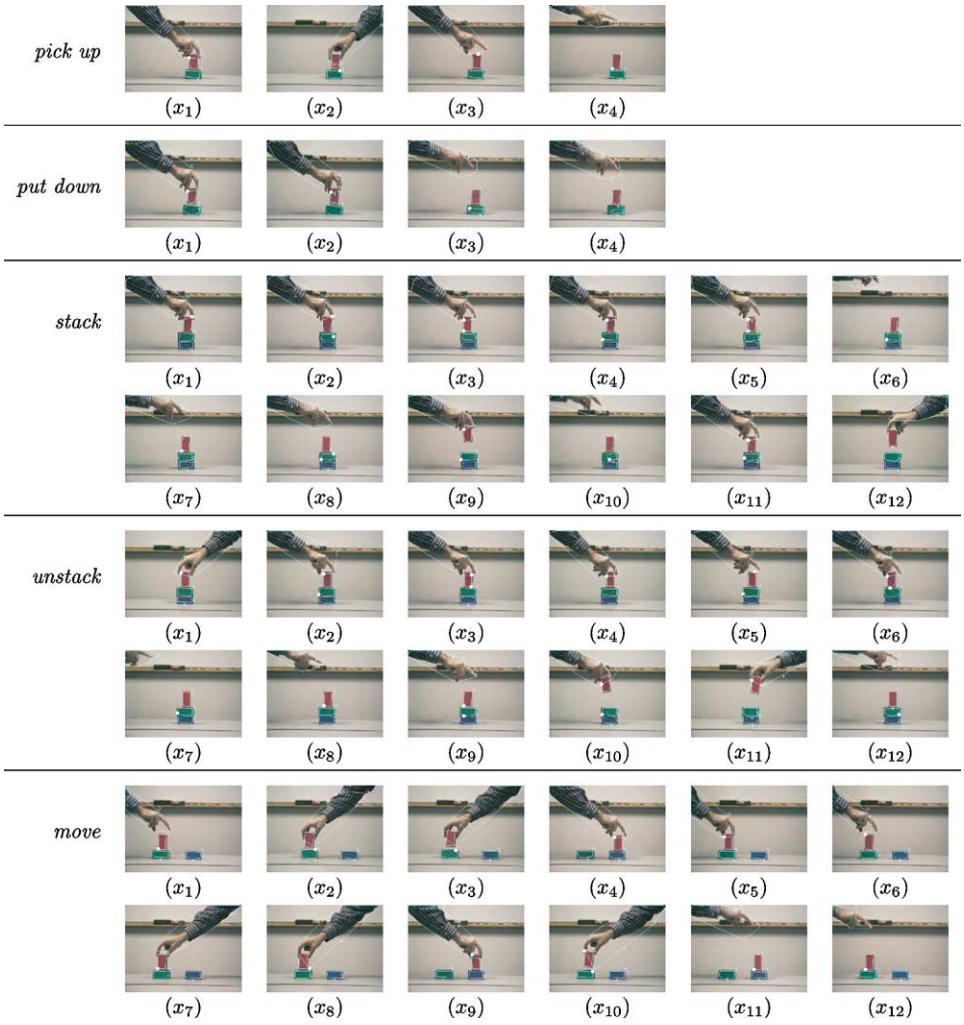


Fig. 39. The unintended most-preferred model classes for each of the first five event types from Fig. 36.

support relation. In the case of error mode IIIb, a groundedness assertion replaces the support relation. Error mode IV indicates a temporal-circumscription error. The wrong object is grounded. The causes of error modes II, IIIa, IIIb, and IV have not yet been determined. Error modes Va and Vb indicate unintended joints. Error mode Va indicates the hand being attached to an extra block. Error mode Vb indicates the hand being attached to the wrong block. Error modes Va and Vb illustrate shortcomings of the theory presented in this paper. While these models are legitimate according to this theory, they do not correspond to human intuition. Future work will attempt to determine and address the causes of these error modes.

Model reconstruction in LEONARD is not an end to itself. Rather, it is a means to the end of visual-event recognition. The overall goal of LEONARD is to take short video sequences

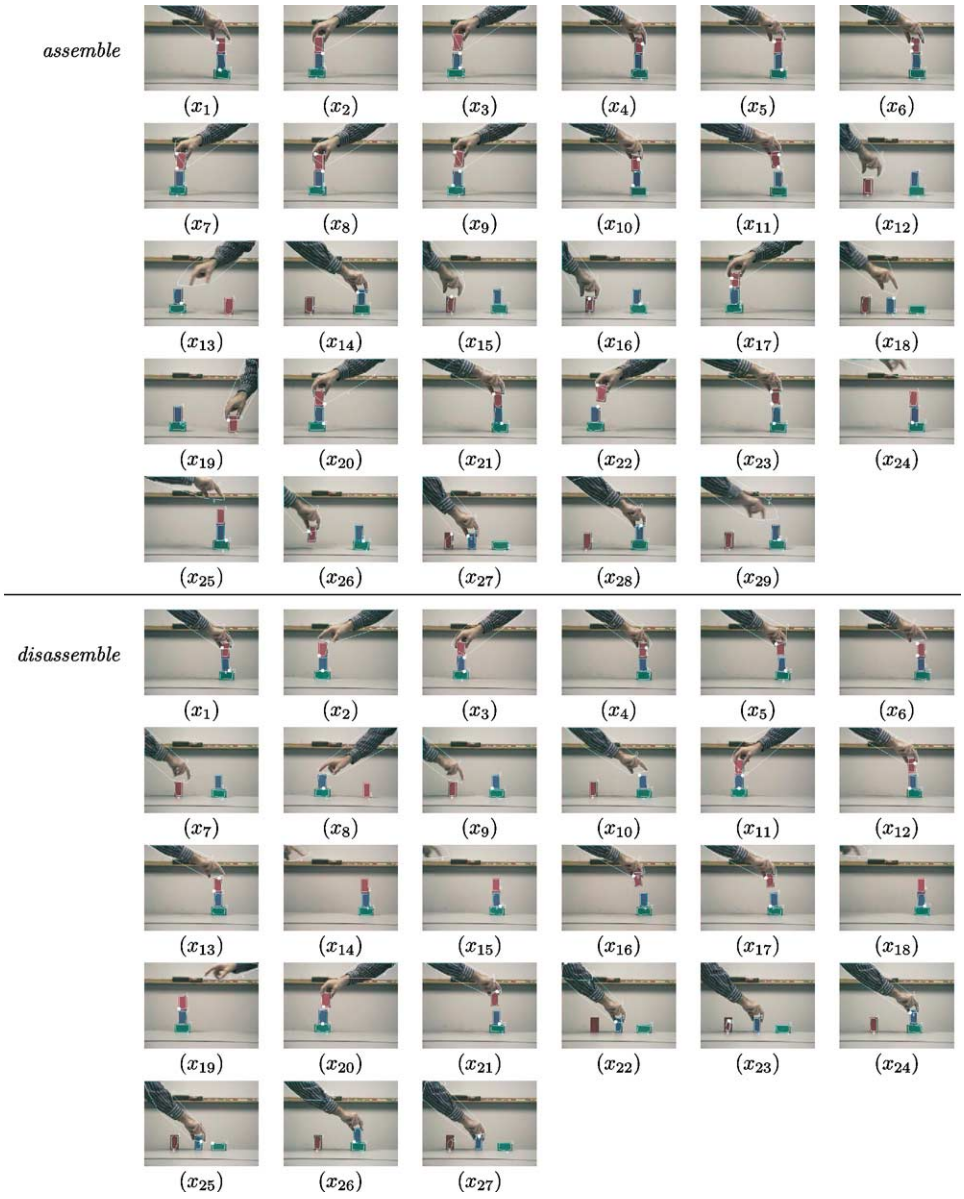


Fig. 40. The unintended most-preferred model classes for each of the last two event types from Fig. 36.

as input and recognize the events that take place in those sequences. Fig. 44 illustrates the overall architecture of LEONARD. Segmentation and tracking procedures take a sequence of video frames as input and output a sequence of scenes. Model reconstruction takes this sequence of scenes as input and outputs a most-preferred model sequence. Event classification takes this sequence of models as input and outputs a set of event occurrences.

Model class	*pick up*	*put down*	*stack*	*unstack*	*move*	*assemble*	*disassemble*
a	241	456	374	161	224	308	310
b						616	
b_1	130	31	137	67	84		139
b_2	187	270	157	279	165		218
c	330	187	145	248	478		939
c_1						54	
c_2						246	
d						182	202
d_1					142		
d_2					176		
e					175	166	
e_1							140
e_2							204
f							713
f_1						42	
f_2						190	
g						98	264
Total intended	888	944	813	755	1444	1902	3129
	91%	90%	82%	83%	89%	69%	89%
Total frames	976	1047	988	908	1629	2776	3522

Fig. 41. A breakdown of the number of frames of each intended model class and event type. Model classes a–g indicate intended models and correspond to the model classes in Fig. 38.

Model class	*pick up*	*put down*	*stack*	*unstack*	*move*	*assemble*	*disassemble*
x_1	20 I	14 I	13 I	35 I	76 I	608 I	164 I
x_2	1 Vb	28 II	24 I	5 I	4 Vb	215 I, Vb	154 I
x_3	41 II	60 IIIa	16 I	7 I	1 ?	19 Vb	1 V
x_4	26 IIIa	1 IIIb	25 II	1 II	9 I	9 I	5 I
x_5			2 II	27 II	41 I	4 I	3 I
x_6			14 IIIa	3 II	20 II	15 I	53 I
x_7			37 IIIa	23 IIIa	2 II	2 ?	10 I
x_8			4 IIIa	48 IIIa	2 Vb	1 ?	1 I
x_9			40 IIIa	4 IIIa	30 II	1 Vb	2 I, IIIa
x_{10}			7 IIIa	31 IIIa	1 IIIa, Vb	7 II	30 I
x_{11}			1 I	1 IIIb	37 IIIa	18 II	1 VI
x_{12}			1 VI	2 II	36 IIIa	24 I	6 II
x_{13}						1 I	14 II
x_{14}						13 I	6 II
x_{15}						456 IV	45 IIIa
x_{16}						67 IIIa, IV	1 II
x_{17}						1 IIIb	10 IIIa
x_{18}						18 IV	34 IIIa
x_{19}						2 IIIb, IV	11 IIIa
x_{20}						9 Vb	1 IIIa
x_{21}						3 II	87 IIIa
x_{22}						4 II	1 VI
x_{23}						89 ?	2 VI
x_{24}						14 IIIa	23 II
x_{25}						16 IIIa	1 V
x_{26}						31 IIIa	20 IIIa
x_{27}						1 Va	17 IV
x_{28}						22 II	
x_{29}						21 IIIa	
Total unintended	88	103	175	153	185	874	393
	9 %	10 %	18 %	17 %	11 %	31 %	11 %
Total frames	976	1047	988	908	1629	2776	3522

Fig. 42. A breakdown of the number of frames of each unintended model class and event type. Model classes x_i indicate unintended models and correspond to the model classes in Figs. 39 and 40. Next to the frame counts for each unintended model class are the error mode(s) associated with that unintended model class. Error modes I, II, IIIa, IIIb, IV, Va, and Vb correspond to the error modes depicted in Fig. 43. Error mode VI indicates a segmentation and tracking error. A '?' indicates an unexplained error.

LEONARD recognizes event occurrences by parsing the state changes in force-dynamic relations between participant objects. These force-dynamic relations, i.e., support, contact, and attachment relations, are derived from the most-preferred models produced by model reconstruction. The force-dynamic relations are taken to be primitive event types. Compound event types are defined in terms of primitive event types using event-logic

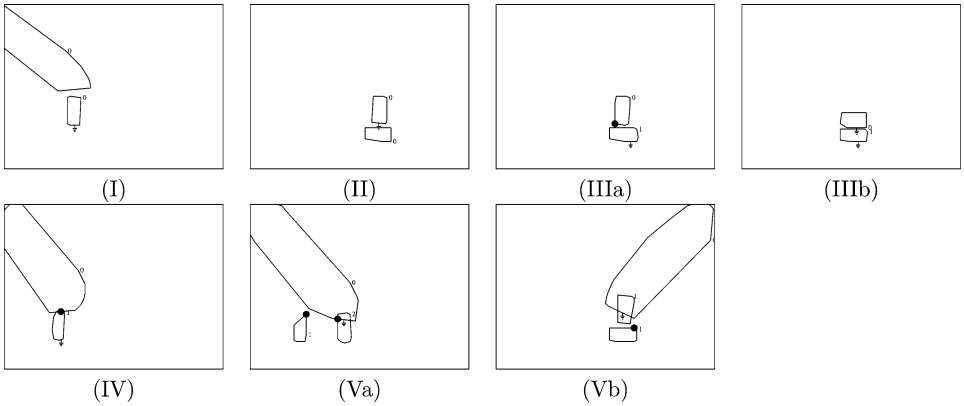


Fig. 43. The seven predominant error modes. (I) indicates an object being supported despite its center of mass being distant from the contact location. (II) indicates an object being supported by a substantiality constraint from above instead of below. (IIIa) and (IIIb) indicate failure to determine a support relation. (IIIa) indicates a joint replacing a support relation. (IIIb) indicates a groundedness assertion replacing a support relation. (IV) indicates a temporal-circumscription error. (Va) and (Vb) indicate unintended joints. (Va) indicates the hand being attached to an extra block. (Vb) indicates the hand being attached to the wrong block.

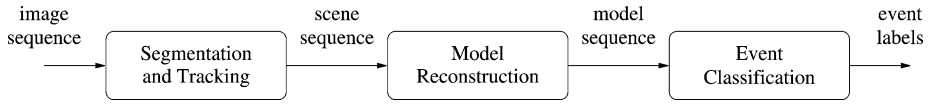


Fig. 44. The overall architecture of LEONARD. The model-reconstruction component from Fig. 32 corresponds to the middle box in this figure.

expressions. The event-classification component of LEONARD infers the occurrence of compound event types from the occurrence of primitive event types. Siskind [96] describes, in detail, how LEONARD determines the occurrence of primitive event types (i.e., force-dynamic relations) from the most-preferred model sequences, how compound event types are defined as event-logic expressions, and how compound event occurrences are inferred from primitive event occurrences.

Siskind [96] presented a limited experimental evaluation of the performance of LEONARD on a dataset of 35 movies comprising five movies for each of the seven event types *pick up*, *put down*, *stack*, *unstack*, *move*, *assemble*, and *disassemble*. Here we present an experimental evaluation of LEONARD on the larger dataset of 210 movies (30 movies for each of the seven event types) described earlier. For this experiment, a set of compound-event-type definitions was used that differs slightly from those reported in Siskind [96]. These new compound-event-type definitions are given in Figs. 45 and 46.

Note that a *put down* occurs whenever there is a *stack*, a *pick up* occurs whenever there is an *unstack*, both a *pick up* and *put down* occur whenever there is a *move*, both a *put down* and a *stack* (and thus another *put down*) occur whenever there is an *assemble*, and both a *pick up* and an *unstack* (and thus another *pick up*) occur whenever there is a *disassemble*. An event occurrence consists of an event type, such as *pick up*, the participant objects, such as PICKUP(**hand**, **red-block**, **green-block**), and a timestamp, the interval during which

$$\textsc{PickUp}(x, y, z) \triangleq \left(\begin{array}{l} \neg\Diamond x = y \wedge \neg\Diamond z = x \wedge \neg\Diamond z = y \wedge \\ \textsc{Supported}(y) \wedge \neg\Diamond\textsc{Attached}(x, z) \wedge \\ \left\{ \begin{array}{l} \left[\begin{array}{l} \neg\Diamond\textsc{Attached}(x, y) \wedge \neg\Diamond\textsc{Supports}(x, y) \wedge \\ \textsc{Supports}(z, y) \wedge \textsc{Contacts}(z, y) \wedge \\ \neg\Diamond\textsc{Supported}(x) \wedge \neg\Diamond\textsc{Attached}(y, z) \wedge \\ \neg\Diamond\textsc{Supports}(y, x) \wedge \neg\Diamond\textsc{Supports}(y, z) \wedge \\ \neg\Diamond\textsc{Supports}(x, z) \wedge \neg\Diamond\textsc{Supports}(z, x) \end{array} \right] \\ \left[\begin{array}{l} \textsc{Attached}(x, y) \wedge \textsc{Supports}(x, y) \wedge \\ \neg\Diamond\textsc{Supports}(z, y) \wedge \\ \neg\Diamond\textsc{Supported}(x) \wedge \neg\Diamond\textsc{Attached}(y, z) \wedge \\ \neg\Diamond\textsc{Supports}(y, x) \wedge \neg\Diamond\textsc{Supports}(y, z) \wedge \\ \neg\Diamond\textsc{Supports}(x, z) \wedge \neg\Diamond\textsc{Supports}(z, x) \end{array} \right] \end{array} \right\}_{\wedge\{<,\mathsf{m}\}} \end{array} \right)$$

$$\textsc{PutDown}(x, y, z) \triangleq \left(\begin{array}{l} \neg\Diamond x = y \wedge \neg\Diamond z = x \wedge \neg\Diamond z = y \wedge \\ \textsc{Supported}(y) \wedge \neg\Diamond\textsc{Attached}(x, z) \wedge \\ \left\{ \begin{array}{l} \left[\begin{array}{l} \textsc{Attached}(x, y) \wedge \textsc{Supports}(x, y) \wedge \\ \neg\Diamond\textsc{Supports}(z, y) \wedge \\ \neg\Diamond\textsc{Supported}(x) \wedge \neg\Diamond\textsc{Attached}(y, z) \wedge \\ \neg\Diamond\textsc{Supports}(y, x) \wedge \neg\Diamond\textsc{Supports}(y, z) \wedge \\ \neg\Diamond\textsc{Supports}(x, z) \wedge \neg\Diamond\textsc{Supports}(z, x) \end{array} \right] \\ \left[\begin{array}{l} \neg\Diamond\textsc{Attached}(x, y) \wedge \neg\Diamond\textsc{Supports}(x, y) \wedge \\ \textsc{Supports}(z, y) \wedge \textsc{Contacts}(z, y) \wedge \\ \neg\Diamond\textsc{Supported}(x) \wedge \neg\Diamond\textsc{Attached}(y, z) \wedge \\ \neg\Diamond\textsc{Supports}(y, x) \wedge \neg\Diamond\textsc{Supports}(y, z) \wedge \\ \neg\Diamond\textsc{Supports}(x, z) \wedge \neg\Diamond\textsc{Supports}(z, x) \end{array} \right] \end{array} \right\}_{\wedge\{<,\mathsf{m}\}} \end{array} \right)$$

Fig. 45. Part I of the lexicon of compound event types used by LEONARD for the results in this paper. These definitions are a slight variant of those in Siskind [96].

the event occurred. For evaluation purposes, the timestamps were ignored. This leads to a set of expected event occurrences for each movie. These are illustrated in Fig. 47. For each movie, the set of event occurrences recovered by LEONARD, with duplicates removed after ignoring timestamps, was compared against the set of expected event occurrences. These sets matched (i.e., no false positives or negatives) on 30/30 *pick up*, 29/30 *put down*, 28/30 *stack*, 28/30 *unstack*, 8/30 *move*, 6/30 *assemble*, and 27/30 *disassemble* movies for a total of 156/210 (74.3%).

The errors fall largely into two systematic classes. All but four of the *assemble* misclassifications (assemble-left-qobi-04, assemble-left-qobi-06, assemble-right-qobi-03, and assemble-right-qobi-14) result from temporal circumscription mistakingly grounding the **red-block** instead of the **hand** during the first phase of the *stack* subevent. And all but two of the *move* misclassifications

(`move-left-qobi13` and `move-right-qobi11`)

result from overly general definitions of *pick up* and *put down* that trigger false positives of

PICKUP(**green-block**, **red-block**, **blue-block**)

$$\textsc{Stack}(w,x,y,z) \triangleq \left(\begin{array}{l} \neg\Diamond w = x \wedge \neg\Diamond y = w \wedge \neg\Diamond y = x \wedge \\ \neg\Diamond z = w \wedge \neg\Diamond z = x \wedge \neg\Diamond z = y \wedge \\ \textsc{Supported}(x) \wedge \neg\Diamond\textsc{Attached}(w,y) \wedge \\ \left\{\begin{array}{l} \left[\begin{array}{l} \textsc{Attached}(w,x) \wedge \textsc{Supports}(w,x) \wedge \\ \neg\Diamond\textsc{Supports}(y,x) \wedge \\ \textsc{Supports}(z,y) \wedge \textsc{Contacts}(z,y) \wedge \\ \neg\Diamond\textsc{Attached}(z,y) \wedge \\ \neg\Diamond\textsc{Supported}(w) \wedge \neg\Diamond\textsc{Attached}(x,y) \wedge \\ \neg\Diamond\textsc{Supports}(x,w) \wedge \neg\Diamond\textsc{Supports}(x,y) \wedge \\ \neg\Diamond\textsc{Supports}(w,y) \wedge \neg\Diamond\textsc{Supports}(y,w) \end{array}\right] \wedge_{\{<,\mathsf{m}\}} \\ \left[\begin{array}{l} \neg\Diamond\textsc{Attached}(w,x) \wedge \neg\Diamond\textsc{Supports}(w,x) \wedge \\ \textsc{Supports}(y,x) \wedge \textsc{Contacts}(y,x) \wedge \\ \textsc{Supports}(z,y) \wedge \textsc{Contacts}(z,y) \wedge \\ \neg\Diamond\textsc{Attached}(z,y) \wedge \\ \neg\Diamond\textsc{Supported}(w) \wedge \neg\Diamond\textsc{Attached}(x,y) \wedge \\ \neg\Diamond\textsc{Supports}(x,w) \wedge \neg\Diamond\textsc{Supports}(x,y) \wedge \\ \neg\Diamond\textsc{Supports}(w,y) \wedge \neg\Diamond\textsc{Supports}(y,w) \end{array}\right] \end{array}\right\} \end{array}\right)$$

$$\textsc{Unstack}(w,x,y,z) \triangleq \left(\begin{array}{l} \neg\Diamond w = x \wedge \neg\Diamond y = w \wedge \neg\Diamond y = x \wedge \\ \neg\Diamond z = w \wedge \neg\Diamond z = x \wedge \neg\Diamond z = y \wedge \\ \textsc{Supported}(x) \wedge \neg\Diamond\textsc{Attached}(w,y) \wedge \\ \left\{\begin{array}{l} \left[\begin{array}{l} \neg\Diamond\textsc{Attached}(w,x) \wedge \neg\Diamond\textsc{Supports}(w,x) \wedge \\ \textsc{Supports}(y,x) \wedge \textsc{Contacts}(y,x) \wedge \\ \textsc{Supports}(z,y) \wedge \textsc{Contacts}(z,y) \wedge \\ \neg\Diamond\textsc{Attached}(z,y) \wedge \\ \neg\Diamond\textsc{Supported}(w) \wedge \neg\Diamond\textsc{Attached}(x,y) \wedge \\ \neg\Diamond\textsc{Supports}(x,w) \wedge \neg\Diamond\textsc{Supports}(x,y) \wedge \\ \neg\Diamond\textsc{Supports}(w,y) \wedge \neg\Diamond\textsc{Supports}(y,w) \end{array}\right] \wedge_{\{<,\mathsf{m}\}} \\ \left[\begin{array}{l} \textsc{Attached}(w,x) \wedge \textsc{Supports}(w,x) \wedge \\ \neg\Diamond\textsc{Supports}(y,x) \wedge \\ \textsc{Supports}(z,y) \wedge \textsc{Contacts}(z,y) \wedge \\ \neg\Diamond\textsc{Attached}(z,y) \wedge \\ \neg\Diamond\textsc{Supported}(w) \wedge \neg\Diamond\textsc{Attached}(x,y) \wedge \\ \neg\Diamond\textsc{Supports}(x,w) \wedge \neg\Diamond\textsc{Supports}(x,y) \wedge \\ \neg\Diamond\textsc{Supports}(w,y) \wedge \neg\Diamond\textsc{Supports}(y,w) \end{array}\right] \end{array}\right\} \end{array}\right)$$

$$\textsc{Move}(w,x,y,z) \triangleq \neg\Diamond y = z \wedge \left[\textsc{PickUp}(w,x,y); \textsc{PutDown}(w,x,z)\right]$$

$$\textsc{Assemble}(w,x,y,z) \triangleq \textsc{PutDown}(w,y,z) \wedge_{\{<\}} \textsc{Stack}(w,x,y,z)$$

$$\textsc{Disassemble}(w,x,y,z) \triangleq \textsc{Unstack}(w,x,y,z) \wedge_{\{<\}} \textsc{PickUp}(x,y,z)$$

Fig. 46. Part II of the lexicon of compound event types used by LEONARD for the results in this paper. These definitions are a slight variant of those in Siskind [96].

and

PUTDOWN(**blue-block**, **red-block**, **green-block**)

for the *left* movies and

PICKUP(**blue-block**, **red-block**, **green-block**)

pick up		PICKUP(**hand**, **red-block**, **green-block**)
put down		PUTDOWN(**hand**, **red-block**, **green-block**)
stack		STACK(**hand**, **red-block**, **green-block**, **blue-block**) PUTDOWN(**hand**, **red-block**, **green-block**)
unstack		UNSTACK(**hand**, **red-block**, **green-block**, **blue-block**) PICKUP(**hand**, **red-block**, **green-block**)
move	*left*	MOVE(**hand**, **red-block**, **green-block**, **blue-block**) PICKUP(**hand**, **red-block**, **green-block**) PUTDOWN(**hand**, **red-block**, **blue-block**)
move	*right*	MOVE(**hand**, **red-block**, **blue-block**, **green-block**) PICKUP(**hand**, **red-block**, **blue-block**) PUTDOWN(**hand**, **red-block**, **green-block**)
assemble		ASSEMBLE(**hand**, **red-block**, **blue-block**, **green-block**) PUTDOWN(**hand**, **blue-block**, **green-block**) STACK(**hand**, **red-block**, **blue-block**, **green-block**) PUTDOWN(**hand**, **red-block**, **blue-block**)
disassemble		DISASSEMBLE(**hand**, **red-block**, **blue-block**, **green-block**) UNSTACK(**hand**, **red-block**, **blue-block**, **green-block**) PICKUP(**hand**, **red-block**, **blue-block**) PICKUP(**hand**, **blue-block**, **green-block**)

Fig. 47. The expected event occurrences, *sans* timestamps, for each movie type.

and

PUTDOWN(**green-block**, **red-block**, **blue-block**)

for the *right* movies. The remaining 14/210 (6.7%) errors are nonsystematic and result largely from model-reconstruction errors.

5. Related work

Bobrow [14] and ISSAC [72] solve physics word problems by constructing and solving a set of equations that represent a physical situation extracted from natural-language text. SHRDLU [116], MECHO [24,25,61], and Palmer [76] ground the semantics of natural-language fragments in diagrammatic representations. The physical models in these systems are constructed from text rather than visual input. Novak and Bulko [73] describe a system for interpreting drawings that depict physics problems. Their system uses the linguistic description of the problem as an aid to the process of understanding the image. Unlike LEONARD, it cannot correctly interpret an image without the help of an accompanying

linguistic description and thus cannot be taken as a model of visual perception. Blum, Griffith, and Neumann [12] and Fahlman [28] perform stability analysis on collections of blocks using a force-balancing approach instead of the kinematic approach presented here. Funt [43] performs stability analysis by way of simulation. Simulation is performed using a retinotopic representation of the scene with pixels organized along concentric circles centered around a fovea. This allows rotation of objects around the fovea but does not allow translation. None of these perform stability analysis in the presence of joints, none perform model reconstruction or event classification, and none operate on real video. Kramer [55,56] presents a system for simulating the kinematics of a mechanism using degree-of-freedom analysis. Cremer [26] presents a system for mechanism simulation using numerical methods. These systems can simulate the behavior of mechanisms with joints but do not perform model reconstruction or event classification and do not operate on video input. Forbus [33,34] and Forbus, Nielsen, and Faltings [35,36] discuss methods for reasoning qualitatively about kinematics.

Some linguists and cognitive scientists, such as Leech [57], Miller [69], Schank [84], Jackendoff [49,50], and Pinker [79], have formulated lexical-semantic representations for verbs based on the causal, aspectual, and directional qualities of the motion of participant objects. Badler [5], Adler [3], Nagel [70], Tsotsos [106], Tsuji, Morizono, and Kuroda [109], Okada [74], Tsotsos and Mylopoulos [107], Tsuji, Osada, and Yachida [110,111], Waltz and Boggess [114], O'Rourke and Badler [75], Rashid [81], Tsotsos, Mylopoulos, Covvey, and Zucker [108], Abe, Soga, and Tsuju [1], Marburger, Neumann, and Novak [65], Waltz [113], Abe and Tsuji [2], Adorni, Boccalatte, and Manzo [4], Marr and Vaina [66], Neumann and Novak [71], Rubin and Richards [83], Thibadeau [105], Hays [46, 47], Feldman, Lakoff, Stolcke, and Weber [29], Weber and Stolcke [115], Suppes, Liang, and Böttner [103], Regier [82], Yamoto, Ohya, and Ishii [117], Brand and Essa [22], Pinhanez and Bobick [78], Starner [101], Siskind [92], Siskind and Morris [97], Brand [17–19], Brand, Oliver, and Pentland [23], Bailey, Chang, Feldman, and Narayanan [6], and Bobick and Ivanov [13], among others, describe approaches, some implemented, some not, for recognizing events from simulated or real video based on the motion of participant objects. They do not perform stability analysis or model reconstruction. Others linguists and cognitive scientists, such as Herskovits [48], Talmy [104], and Jackendoff and Landau [51], have argued that force-dynamic relations, such as support, contact, and attachment, are crucial for representing the lexical semantics of verbs and spatial prepositions. Borchardt [15,16] presents event definitions that are based on force-dynamic relations but does not present techniques for recovering those relations automatically from visual input. Brand, Birnbaum, and Cooper [21], Birnbaum, Brand, and Cooper [11], and Brand [20] present heuristic approaches to stability analysis that operate on real video but do not perform model reconstruction and event classification. Siskind [87–89,91] presents a heuristic approach to stability analysis based on kinematic simulation and uses this analysis to perform model reconstruction and event classification but applies these techniques only to simulated video. Siskind [90,93,95] presents earlier versions of the work presented here. This work is motivated by the experiments of Gibson, Owsley, Walker, and Megaw-Nyce [44], Shepard [85,86], DiSessa [27], Freyd [37,38], McCloskey [68], Spelke [98–100], Freyd and Finke [39,40], Baillargeon et al.[9], Finke and Freyd [31], Baillargeon [7,8], Finke, Freyd, and Shyi [32], Leslie [58,59], Freyd and Johnson [41],

Kelly and Freyd [53], Leslie and Keeble [60], and Freyd, Pantzer, and Cheng [42] that suggest that the human perceptual system models the physics of the world, at least approximately.

Sugihara [102] is similar, in some respects, to the present work, while addressing a different problem, namely geometric-model reconstruction. In particular, Sugihara [102] formulates two main problems: determining whether a 2D line drawing can constitute a projection of a 3D polyhedral object and, if so, reconstructing that polyhedral object from the line drawing. In a fashion similar to the work in this paper, the former is formulated as a linear-programming problem. Sugihara [102] also discusses the relationship between the geometric model-reconstruction problem and the *skeletal-structure rigidity problem*, the problem of determining the rigidity of a 2D mechanism. More specifically, suppose that one is given a graph, along with a map from its vertices to planar coordinates. And one treats the edges of that graph as rods connected by revolute joints at the vertices. Is the mechanism depicted by that graph rigid? Sugihara [102] also gives a reduction from this question to linear programming. The stability-analysis method of the current paper can be viewed as extending that technique to support prismatic joints, gravity, and the substantiality constraint.

The skeletal-structure rigidity problem appears to be related to, but not identical to, the stability-analysis problem. The constructions, illustrated in Fig. 48, that map aspects of the LEONARD stability-analysis problem to the skeletal-structure rigidity problem demonstrate the similarities and differences between the two problems. In particular, these constructions can handle rigidity of polygons, grounded polygons, rigid joints, and revolute joints, but not prismatic joints, the substantiality constraint, or gravity. A polygon, such as that shown in Fig. 48(a), is transformed into a rigid skeletal structure by triangulation, as shown in Fig. 48(b). Triangulating a polygon ensures its rigidity. A grounded polygon, such as that shown in Fig. 48(c), is rigidly attached to a grounding rod, as shown in Fig. 48(d), by attaching two of its vertices to each of the two endpoints of the grounding rod. Rigidly attaching all grounded polygons to the same grounding rod ensures that there can be no relative motion between the grounded polygons. A rigid joint between two line segments, such as that shown in Fig. 48(e), is handled by adding a rod between each endpoint of one line segment to each endpoint of the other line segment, as shown in Fig. 48(f). Finally, a revolute joint between two line segments, such as that shown in Fig. 48(g), is handled, as shown in Fig. 48(h), by adding a vertex at the joint intersection point, breaking each line segment into two rods at the new vertex, and then adding additional rods between the two endpoints of each line segment that bypass the newly added vertex. These additional rod are depicted as dotted arcs in Fig. 48(h). I have not been able to formulate analogous transformations for prismatic joints, the substantiality constraint, and gravity. I believe that it is not possible to transform the substantiality constraint into a skeletal-structure rigidity problem, though it might be possible to transform that constraint into a *tensegrity-structure rigidity problem* [102, pp. 197–198]. Furthermore, I believe that it is not possible to transform prismatic joints or gravity into either skeletal-structure or tensegrity-structure rigidity problems.

The BUILD system of Fahlman [28] is also similar, in some respects, to LEONARD. BUILD is a blocks-world planning system that plans a sequence of block moves to reach a goal state from an initial state. Unlike most formulations of planning (e.g., STRIPS,

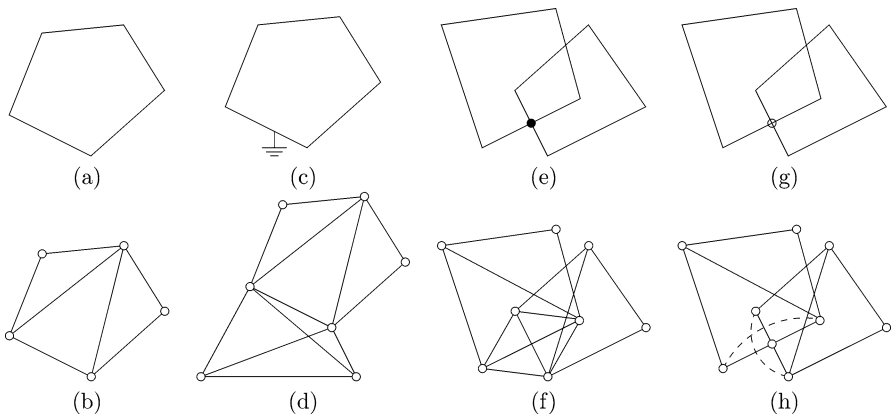
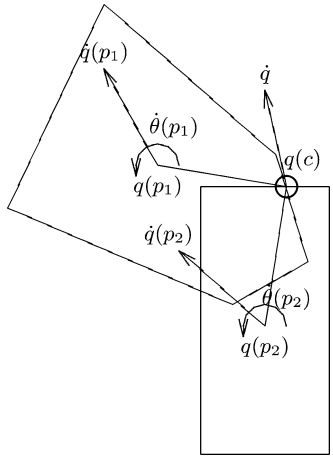


Fig. 48. A mapping from aspects of the LEONARD stability-analysis problem to the skeletal-structure rigidity problem. (a)–(b) Polygons to rods. (c)–(d) Groundedness. (e)–(f) Rigid joints. (g)–(h) Revolute joints.

Fikes and Nillson [30], and the situation calculus, Green [45]) which characterize states predicatively, BUILD characterizes states geometrically via the positions, orientations, shapes, and sizes of the blocks. An implicit background constraint adopted by BUILD is that all states must be stable. Thus BUILD incorporates a stability-analysis procedure in the planning process. This stability-analysis procedure differs from the one used by LEONARD in three crucial ways. First, it uses a force-balancing approach based on the techniques of Blum et al. [12] rather than the kinematic approach used by LEONARD. Second, for reasons of computational efficiency, it implements force balancing via heuristic search for a causal support chain rather than by a reduction to linear programming as done in Blum et al. [12] and Mann, Jepson, and Siskind [64]. It seems that, when BUILD was written, the linear-programming approach was too costly. Today, computers are fast enough so that performing stability analysis by linear programming is not the limiting factor to the performance of LEONARD. Rather, circumscriptive search through the space of interpretations to find a minimal one, performing stability analysis on each interpretation, is the limiting factor. Third, LEONARD performs stability analysis relative to interpretations that incorporate attachment relations which BUILD does not do. BUILD differs from LEONARD in ways beyond stability analysis. BUILD, unlike LEONARD, plans sequences of actions. LEONARD, unlike BUILD, performs model reconstruction to recover unobservable information, operates on video input, and recognizes events.

LEONARD is closest in spirit to the work of Mann, Jepson, and Siskind [63,64] and Mann and Jepson [62]. Both operate within the perceiver framework [52]. The major difference is that Mann et al. use a dynamic theory while LEONARD uses a kinematic theory. Mann et al. takes the observables to be the linear and angular, positions, velocities, and accelerations of polygons. It adopts an ontology that includes a same-layer relation, revolute joints, and three kinds of motors: ones that can apply a linear force, ones that can apply an angular torque, and ones that can apply both, between a pair of polygons. It takes Newton's second law ($\mathbf{F} = m\mathbf{a}$) and a Coulombic model of friction as the theory. And it uses a prioritized-circumscription process, minimizing motors and then revolute joints, as



$$\dot{q}(p_1) + (q(c) - q(p_1)) \times \dot{\theta}(p_1) = \dot{q} = \dot{q}(p_2) + (q(c) - q(p_2)) \times \dot{\theta}(p_2)$$

Fig. 49. The theory of Mann et al. will entertain a revolute joint between points on two polygons only when those two points have the same linear velocity.

the preference ordering. Mann and Jepson [62] adds a temporal-circumscription process that differs from the one incorporated into LEONARD.

The approach of Mann et al. suffers from some problems, however, that motivate the approach taken in LEONARD. One such problem is illustrated in Fig. 49. The theory in Mann et al. will entertain a revolute joint at contact c between polygons p_1 and p_2 only when the point c on each polygon p_i has the same linear velocity $\dot{q}$ as a result of the observed linear and angular velocities $\dot{q}(p_i)$ and $\dot{\theta}(p_i)$ of that polygon. Noisy estimates of those velocities will limit the ability to entertain revolute joints. Similarly, the theory in Mann et al. must entertain the presence of motors to induce forces and torques when gravity is insufficient to account for the observed accelerations of polygons. Noisy estimates of those accelerations will induce the approach to entertain spurious motors.

Velocity and acceleration, however, are not directly observable. They are calculated by approximating the derivative of position, which is observable, using finite differences. Current segmentation and tracking techniques, however, give noisy position estimates. The derivative operator amplifies this noise, making the velocity estimates unreliable and the acceleration estimates even more unreliable. An even more fundamental problem arises, however. Mann et al. assumed that all polygons were rigid. Their shape and size could not change. This was necessary to assign each polygon a well-defined position, namely its center of mass, from which its velocity and acceleration could be computed. Without this assumption of rigidity, the notion of 'the velocity of a polygon' becomes ambiguous. When a nonrigid polygon moves, the change in position of its center of mass can be attributed to an arbitrary combination of motion and shape change. Without some additional information, such as texture, it is impossible to disambiguate the motion and shape change to determine a meaningful velocity.

Nonrigid polygons arise for at least four reasons. First, some objects, such as hands, are inherently nonrigid. Second, the shape and size of objects will change as they are

occluded or enter and leave the field of view. Third, motion in depth will change the size of the silhouette of an object that results from projecting the 3D object onto the 2D image plane. Finally, out-of-plane rotation can change the shape of that silhouette. Mann et al. avoided these difficulties by considering only frontal-parallel scenes and using a tracker that imposed a rigidity constraint.

The theory used in Mann et al. is richer and more physically accurate than the theory used in LEONARD. However, using Newtonian physics as a model of perception relies crucially on the ability to recover the velocities and accelerations of visually observed objects, something which is unreliable at best and ill-defined at worst. Much research in psychology, e.g., McCloskey [68], has indicated that humans do not base their understanding of the world on Newtonian physics but rather use less accurate, or naive, physical theories. Perhaps the human visual system encodes something other than Newtonian physics precisely because of the difficulty of recovering object velocities and accelerations. On the other hand, the kinematic theory used in LEONARD is too weak. It lacks any notion of force and is thus unable to differentiate between truly grounded objects and agents that can support themselves by exertion of force. And its way of modeling friction via rotation of surfaces leads to errors as discussed in Section 4. This leads to a fundamental question: what kind of theory will admit a notion of force and friction yet not require accurate and unambiguous recovery of velocities and accelerations?

6. Discussion

It should be pointed out that the goal of this endeavor is not optimal performance on the specific task of force-dynamic model reconstruction from frontal-parallel movies of hands manipulating colored blocks. Many aspects of this particular task can be better handled by other, sometimes simpler mechanisms. For example, depth perception can be handled by a variety of techniques such as stereo, structure from motion, shape from shading, laser and/or sonar range finders, etc., instead of circumscriptive reasoning about support via substantiality vs. support by attachment. Similarly, hands can be distinguished from blocks by shape and/or color cues, instead of circumscriptive reasoning about groundedness. Such task-specific techniques, however, often fail when the task changes. And they often lack robustness. Moreover, they shed little light on the potentially rich and complex reasoning process underlying human perception. The goal of this endeavor is to investigate a more reasoning-intensive approach to perception, one that constructs interpretations that are consistent with deep knowledge of the physical world. It tries to determine how much hidden information the perceiver framework alone can recover within the confines of a limited physical theory. The hope is that, in the long run, such reasoning-intensive approaches to perception, when coupled with more extensive physical theories, will lead to more robust and flexible perceptive systems.

One problem with LEONARD is that the inference pipeline is very brittle. Any noise or mistake at any stage in the pipeline can lead to an incorrect judgment at the end of the pipeline. The current fashionable trend in AI research is to mitigate such brittleness by switching from a logic-based paradigm to one based on statistical methods. However, most current statistical models are chosen because they are amenable to closed-form parameter

estimation and classification methods, not because they are accurate causal or generative models of the underlying physical system. An interesting area for future work would be to devise a model that, on one hand, was as rich and accurate as that used by LEONARD, and, on the other hand, also ameliorated the brittleness of LEONARD by use of statistical methods. Perhaps the work of Koller, McAllester, and Pfeffer [54] and Pfeffer [77] can form the framework for such an approach.

The overall concern of the line of research behind this paper is to develop a theory for understanding the physical world through perception. Given the current formalist trend in AI research, it is fashionable to assess the strengths and weaknesses of a theory by formal analysis. It is not possible, however, to use formal methods to analyze how well a theory, such as the one incorporated in LEONARD, matches pretheoretic intuitions such as human cognition. The best we can do is assess the quality of the match by way of experiment, as was done in Section 4. However, that experiment only assesses the match to a tiny corner of human perception and cognition, namely frontal-parallel views of hands manipulating colored blocks. It is sobering to consider how much of human conceptualization of the physical world is left to model.

LEONARD performs stability analysis and model reconstruction on the entire field of view. Its stability-analysis and model-reconstruction methods work well only when there are a small number of objects in the field of view. Methods are needed to focus the attention on a small portion of the field of view when it contains a large number of objects. LEONARD also adopts a restrictive layered 2D ontology. This limits LEONARD to frontal parallel views. LEONARD fails miserably when shown the same blocks-world sequences from a view that is not frontal parallel. One possibility for addressing this limitation is to extend the stability-analysis procedures to 3D. However, there appears to be a fundamental paradox regarding human depth perception. On one hand, humans appear unable to make precise absolute depth judgments. On the other hand, they appear to perform physical reasoning tasks that require precise depth judgment. Determining the right representation for depth information, one that can be reliably recovered from visual input and also support physical reasoning, is an important area for future research. LEONARD models all objects as convex polygons. However, the world is replete with nonconvex objects, the most prominent being containers of all forms. LEONARD's ontology of attachment relations is limited to revolute and prismatic joints. That prevents LEONARD from modeling containers with screw tops, for this requires helical joints and a 3D ontology. Beyond formal kinematic theory, humans conceptualize additional kinds of attachment relations with such substances as glue and tape, each with its distinctive physical properties. LEONARD's ontology considers one kind of substance: convex rigid polygons with uniform density and uniform density distribution. However, the world is full of objects with different densities as well as objects with nonuniform density distribution. Furthermore, much of the world consists of nonrigid objects: string, rubberbands, cloth, paper, pillows, liquids, powder, sand, gel, foam, smoke, gas, etc. All of these have distinct physical properties from the perspective of human cognition. The ontology of substances according to human cognition is much richer than the traditional solid-liquid-gas ontology of classical physics. Likewise, the ontology of forces is richer than LEONARD's simple model of gravity: friction, wind, light, electricity, heat, magnetism, etc. Such richer ontologies of substance and force are needed to model common everyday notions such as *roll*, *slide*, *fold*, *pierce*, *cut*, *insert*, *open*, etc.

The current work on LEONARD is just the beginning of a long research program whose goal is the codification of such physical notions in a fashion that supports perception of—and reasoning about—such verbal concepts.

7. Conclusion

In this paper, I have presented a method for reconstructing force-dynamic models from video input. The method is based on a kinematic stability-analysis procedure that determines the stability of a polygonal scene, under an interpretation, via a reduction to linear programming. A sequence of circumscription procedures searches the space of admissible interpretations to find the simplest stable interpretations. The techniques have been implemented in a system called LEONARD and have been tested on 210 movies comprising 11946 frames. On these, 83.4% were labeled with the intended interpretation. Model reconstruction has also been used as the first stage of an event-recognition procedure. LEONARD can label a movie as containing one or more *pick up*, *put down*, *stack*, *unstack*, *move*, *assemble*, or *disassemble* events. On this task, LEONARD correctly labeled 74.3% of the movies with the correct label set. These results illustrate the potential for building computational entities that exhibit deep physical understanding of the world as perceived from visual input.

References

[1] N. Abe, I. Soga, S. Tsuji, A plot understanding system on reference to both image and language, in: Proceedings of IJCAI-81, Vancouver, BC, 1981, pp. 77–84.
[2] N. Abe, S. Tsuji, A learning of object structures by verbalism, in: J. Horecky (Ed.), COLING 82, Elsevier North-Holland, Amsterdam, 1982, pp. 1–6.
[3] M.R. Adler, Computer interpretation of peanuts cartoons, in: Proceedings of IJCAI-77, Cambridge, MA, 1977, p. 608.
[4] G. Adorni, A. Boccalatte, M.D. Manzo, Cognitive models for computer vision, in: J. Horecky (Ed.), COLING 82, Elsevier North-Holland, Amsterdam, 1982, pp. 7–12.
[5] N.I. Badler, Temporal scene analysis: Conceptual descriptions of object movements, Technical Report 80, University of Toronto Department of Computer Science, 1975.
[6] D.R. Bailey, N. Chang, J. Feldman, S. Narayanan, Extending embodied lexical development, in: Proceedings of the 20th Annual Conference of the Cognitive Science Society, Madison, WI, 1998.
[7] R. Baillargeon, Representing the existence and the location of hidden objects: Object permanence in 6- and 8-month-old infants, Cognition 23 (1986) 21–41.
[8] R. Baillargeon, Object permanence in $3\frac{1}{2}$- and $4\frac{1}{2}$-month-old infants, Developmental Psychology 23 (5) (1987) 655–664.
[9] R. Baillargeon, E.S. Spelke, S. Wasserman, Object permanence in five-month-old infants, Cognition 20 (1985) 191–208.
[10] M. Berkelaar, ftp://ftp.ics.ele.tue.nl/pub/lp_solve/, 1998.
[11] L. Birnbaum, M. Brand, P. Cooper, Looking for trouble: Using causal semantics to direct focus of attention, in: Proceedings of the 4th International Conference on Computer Vision, 1993, pp. 49–56.
[12] M. Blum, A.K. Griffith, B. Neumann, A stability test for configurations of blocks, A.I. Memo 188, MIT Artificial Intelligence Laboratory, Cambridge, MA, 1970.
[13] A.F. Bobick, Y.A. Ivanov, Action recognition using probabilistic parsing, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Santa Barbara, CA, 1998, pp. 196–202.

[14] D. Bobrow, Natural language input for a computer problem solving system, Ph.D. Thesis, MIT, Cambridge, MA, 1964.
[15] G.C. Borchardt, A computer model for the representation and identification of physical events, Technical Report T-142, Coordinated Sciences Laboratory, University of Illinois at Urbana-Champaign, 1984.
[16] G.C. Borchardt, Event calculus, in: Proceedings of IJCAI-85, Los Angeles, CA, 1985, pp. 524–527.
[17] M. Brand, Transforming problems into imagery and solving them via visual computations, in: Proceedings of the 18th Annual Conference of the Cognitive Science Society, San Diego, CA, 1996.
[18] M. Brand, Understanding manipulation in video, in: Proceedings of the 2nd International Conference on Face and Gesture Recognition, Killington, VT, 1996, pp. 94–99.
[19] M. Brand, The inverse Hollywood problem: From video to scripts and storyboards via causal analysis, in: Proceedings of AAAI-97, Providence, RI, 1997, pp. 132–137.
[20] M. Brand, Physics-based visual understanding, Comput. Vision Image Understanding 65 (2) (1997) 192–205.
[21] M. Brand, L. Birnbaum, P. Cooper, Sensible scenes: Visual understanding of complex scenes through causal analysis, in: Proceedings of AAAI-93, Washington, DC, 1993, pp. 588–593.
[22] M. Brand, I. Essa, Causal analysis for visual gesture understanding, in: Proceedings of AAAI Fall Symposium on Computational Models for Integrating Language and Vision, 1995.
[23] M. Brand, N. Oliver, A. Pentland, Coupled hidden Markov models for complex action recognition, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1997.
[24] A. Bundy, L. Byrd, G. Luger, C. Mellish, R. Milne, M. Palmer, MECHO: A program to solve mechanics problems, Technical Report Working paper 50, Department of Artificial Intelligence, Edinburgh University, 1979.
[25] A. Bundy, G. Luger, M. Palmer, R. Welham, MECHO: Year one, in: M. Brady (Ed.), Proceedings of the 2nd AISB Conference, Edinburgh, 1998, pp. 94–103.
[26] J.F. Cremer, An architecture for general purpose physical system simulation—Integrating geometry, dynamics, and control, Ph.D. Thesis, Cornell University, Ithaca, NY, 1989. Available as TR 89-987.
[27] A.A. DiSessa, Phenomenology and evolution of intuition, in: D. Gentner, A.L. Stevens (Eds.), Mental Models, Erlbaum, Mahwah, NJ, 1983, pp. 15–33.
[28] S.E. Fahlman, A planning system for robot construction tasks, Artificial Intelligence 5 (1) (1974) 1–49.
[29] J.A. Feldman, G. Lakoff, A. Stolcke, S.H. Weber, Miniature language acquisition: A touchstone for cognitive science, in: Proceedings of the 12th Annual Conference of the Cognitive Science Society, Cambridge, MA, 1990, pp. 686–693.
[30] R. Fikes, N. Nilsson, A new approach to the application of theorem proving to problem solving, Artificial Intelligence 2 (3–4) (1971) 189–208.
[31] R.A. Finke, J.J. Freyd, Transformations of visual memory induced by implied motions of pattern elements, J. Experiment. Psychology: Learning Memory Cognition 11 (2) (1985) 780–794.
[32] R.A. Finke, J.J. Freyd, G.C.-W. Shyi, Implied velocity and acceleration induce transformations of visual memory, J. Experiment. Psychology: General 115 (2) (1986) 175–188.
[33] K.D. Forbus, Spatial and qualitative aspects of reasoning about motion, in: Proceedings of AAAI-80, Palo Alto, CA, 1980, pp. 170–173.
[34] K.D. Forbus, Qualitative reasoning about space and motion, in: D. Gentner, A.L. Stevens (Eds.), Mental Models, Erlbaum, Mahwah, NJ, 1983, pp. 53–74.
[35] K.D. Forbus, P.E. Nielsen, B. Faltings, Qualitative kinematics: A framework, in: Proceedings of IJCAI-87, Milan, Italy, 1987, pp. 430–435.
[36] K.D. Forbus, P.E. Nielsen, B. Faltings, Qualitative spatial reasoning: The CLOCK project, Artificial Intelligence 51 (1–3) (1991) 417–471.
[37] J.J. Freyd, The mental representation of movement when static stimuli are viewed, Perception and Psychophysics 33 (1983) 575–581.
[38] J.J. Freyd, Dynamic mental representations, Psychological Rev. 94 (1987) 427–438.
[39] J.J. Freyd, R.A. Finke, Representational momentum, J. Experiment. Psychology: Learning Memory Cognition 10 (1984) 126–132.
[40] J.J. Freyd, R.A. Finke, A velocity effect for representational momentum, Bull. Psychonomic Soc. 23 (1985) 443–446.

[41] J.J. Freyd, J.Q. Johnson, Probing the time course of representational momentum, J. Experiment. Psychology: Learning Memory Cognition 13 (2) (1987) 259–268.
[42] J.J. Freyd, T.M. Pantzer, J.L. Cheng, Representing statics as forces in equilibrium, J. Experiment. Psychology: General 117 (4) (1988) 395–407.
[43] B.V. Funt, Problem-solving with diagrammatic representations, Artificial Intelligence 13 (3) (1980) 201–230.
[44] E.J. Gibson, C.J. Owsley, A. Walker, J. Megaw-Nyce, Development of the perception of invariants: Substance and shape, Perception 8 (1979) 609–619.
[45] C.C. Green, Applications of theorem proving to problem solving, in: Proceedings of IJCAI-69, Washington, DC, 1969, pp. 219–239.
[46] E.M. Hays, On defining motion verbs and spatial prepositions, in: C. Freksa, C. Habel (Eds.), Repräsentation und Verarbeitung räumlichen Wissens, Springer, Berlin, 1989, pp. 192–206.
[47] E.M. Hays, On defining motion verbs and spatial relations, Technical Report 61, Universität des Saarlandes, SFB 314 (VITRA), 1989.
[48] A. Herskovits, Language and Spatial Cognition: An Interdisciplinary Study of the Prepositions in English, Cambridge University Press, New York, 1986.
[49] R. Jackendoff, Semantics and Cognition, MIT Press, Cambridge, MA, 1983.
[50] R. Jackendoff, Semantic Structures, MIT Press, Cambridge, MA, 1990.
[51] R. Jackendoff, B. Landau, Spatial language and spatial cognition, in: D.J. Napoli, J.A. Kegl (Eds.), Bridges Between Psychology and Linguistics: A Swarthmore Festschrift for Lila Gleitman, Erlbaum, Mahwah, NJ, 1991.
[52] A.D. Jepson, W.A. Richards, What is a percept?, Technical Report RBCV-TR-93-43, University of Toronto Department of Computer Science, 1993.
[53] M.H. Kelly, J.J. Freyd, Explorations of representational momentum, Cognitive Psychology 19 (1987) 369–401.
[54] D. Koller, D.A. McAllester, A.J. Pfeffer, Effective Bayesian inference for stochastic programs, in: Proceedings of AAAI-97, Providence, RI, 1997, pp. 740–747.
[55] G.A. Kramer, Geometric reasoning in the kinematic analysis of mechanisms, Technical Report TR-91-02, Schlumberger Laboratory for Computer Science, 1990.
[56] G.A. Kramer, Solving geometric constraint systems, in: Proceedings of AAAI-90, Boston, MA, 1990, pp. 708–714.
[57] G.N. Leech, Towards a Semantic Description of English, Indiana University Press, 1969.
[58] A.M. Leslie, Getting development off the ground: Modularity and the infant's perception of causality, in: P. van Geert (Ed.), Theory Building in Development, Elsevier North-Holland, Amsterdam, 1986, pp. 405–437.
[59] A.M. Leslie, The necessity of illusion: Perception and thought in infancy, in: L. Weiskrantz (Ed.), Thought without Language, Oxford University Press, New York, 1988, Chapter 8, pp. 185–210.
[60] A.M. Leslie, S. Keeble, Do six-month-old infants perceive causality? Cognition 25 (1987) 265–288.
[61] G. Luger, Mathematical model building in the solution of mechanics problems: Human protocols and the MECHO trace, Cognitive Sci. 5 (1981) 55–77.
[62] R. Mann, A.D. Jepson, Toward the computational perception of action, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Santa Barbara, CA, 1998, pp. 794–799.
[63] R. Mann, A.D. Jepson, J.M. Siskind, The computational perception of scene dynamics, in: Proceedings of the Fourth European Conference on Computer Vision, Springer, Cambridge, UK, 1996, pp. 528–539.
[64] R. Mann, A.D. Jepson, J.M. Siskind, The computational perception of scene dynamics, Comput. Vision Image Understanding 65 (2) (1997).
[65] H. Marburger, B. Neumann, H. Novak, Natural language dialogue about moving objects in an automatically analyzed traffic scene, in: Proceedings of IJCAI-81, Vancouver, BC, 1981, pp. 49–51.
[66] D. Marr, L. Vaina, Representation and recognition of the movements of shapes, Proc. R. Soc. Lond. B 214 (1982) 501–524.
[67] J. McCarthy, Circumscription—A form of nonmonotonic reasoning, Artificial Intelligence 13 (1–2) (1980) 27–39.

[68] M. McCloskey, Naive theories of motion, in: D. Gentner, A.L. Stevens (Eds.), Mental Models, Erlbaum, Mahwah, NJ, 1983.
[69] G.A. Miller, English verbs of motion: A case study in semantics and lexical memory, in: A.W. Melton, E. Martin (Eds.), Coding Processes in Human Memory, V.H. Winston and Sons, Washington, DC, 1972, Chapter 14, pp. 335–372.
[70] H. Nagel, Analysing sequences of TV-frames, in: Proceedings of IJCAI-77, Cambridge, MA, 1977, p. 626.
[71] B. Neumann, H. Novak, Event models for recognition and natural language description of events in real-world image sequences, in: Proceedings of IJCAI-83, Karlsruhe, Germany, 1983, pp. 724–726.
[72] G. Novak, Computer understanding of physics problems stated in natural language, American J. Comput. Linguistics 53 (1976).
[73] G.S. Novak, W.C. Bulko, Understanding natural language with diagrams, in: Proceedings of AAAI-90, Boston, MA, 1990.
[74] N. Okada, SUPP: Understanding moving picture patterns based on linguistic knowledge, in: Proceedings of IJCAI-73, Tokyo, Japan, 1979, pp. 690–692.
[75] J. O'Rourke, N.I. Badler, Model-based image analysis of human motion using constraint propagation, IEEE Trans. Pattern Anal. Machine Intelligence 2 (6) (1980) 522–536.
[76] M. Palmer, Semantic Processing for Finite Domains, in: ACL Book Series, Cambridge University Press, New York, 1990.
[77] A.J. Pfeffer, IBAL: A probabilistic rational programming language, in: Proceedings of IJCAI-01, Seattle, WA, 2001.
[78] C. Pinhanez, A. Bobick, Scripts in machine understanding of image sequences, in: Proc. AAAI Fall Symposium Series on Computational Models for Integrating Language and Vision, 1995.
[79] S. Pinker, Learnability and Cognition, MIT Press, Cambridge, MA, 1989.
[80] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in C, 2nd Edition, Cambridge University Press, New York, 1992.
[81] R.F. Rashid, Towards a system for the interpretation of moving light displays, IEEE Trans. Pattern Anal. Machine Intelligence 2 (6) (1980) 574–581.
[82] T.P. Regier, The acquisition of lexical semantics for spatial terms: A connectionist model of perceptual categorization, Ph.D. Thesis, University of California at Berkeley, 1992.
[83] J.M. Rubin, W.A. Richards, Boundaries of visual motion, A.I. Memo 835, MIT Artificial Intelligence Laboratory, 1985.
[84] R.C. Schank, The fourteen primitive actions and their inferences, Memo AIM-183, Stanford Artificial Intelligence Laboratory, 1973.
[85] R.N. Shepard, Psychophysical complementarity, in: M. Kubovy, J.R. Pomerantz (Eds.), Perceptual Organization, Erlbaum, Mahwah, NJ, 1981, pp. 279–341.
[86] R.N. Shepard, Ecological constraints on internal representations: Resonant kinematics of perceiving, imagining, thinking, and dreaming, Psychological Rev. 91 (1984) 417–447.
[87] J.M. Siskind, Naive physics, event perception, lexical semantics and language acquisition, in: The AAAI Spring Symposium Workshop on Machine Learning of Natural Language and Ontology, 1991, pp. 165–168.
[88] J.M. Siskind, Naive physics, event perception, lexical semantics, and language acquisition, Ph.D. Thesis, MIT, Cambridge, MA, 1992.
[89] J.M. Siskind, Grounding language in perception, in: Proceedings of the Annual Conference of the Society of Photo-Optical Instrumentation Engineers, Boston, MA, 1993.
[90] J.M. Siskind, Axiomatic support for event perception, in: P. McKevitt (Ed.), Proceedings of the AAAI Workshop on the Integration of Natural Language and Vision Processing, Seattle, WA, 1994, pp. 153–160.
[91] J.M. Siskind, Grounding language in perception, Artificial Intelligence Rev. 8 (1995) 371–391.
[92] J.M. Siskind, Unsupervised learning of visually-observed events, in: The AAAI Fall Symposium Workshop on Learning Complex Behaviors in Adaptive Intelligent Systems, 1996, pp. 82–83.
[93] J.M. Siskind, Visual event perception, in: K. Ikeuchi, M. Veloso (Eds.), Symbolic Visual Learning, Oxford University Press, New York, 1997, Chapter 9.
[94] J.M. Siskind, Visual event perception, in: Proceedings of the 9th NEC Research Symposium, 1999, Also available as Technical Report 99-033, NEC Research Institute, Inc.

[95] J.M. Siskind, Visual event classification via force dynamics, in: Proceedings of AAAI-00, Austin, TX, 2000, pp. 149–155.
[96] J.M. Siskind, Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic, J. Artificial Intelligence Res. 15 (2001) 31–90.
[97] J.M. Siskind, Q. Morris, A maximum-likelihood approach to visual event classification, in: Proceedings of the Fourth European Conference on Computer Vision, Springer, Cambridge, UK, 1996, pp. 347–360.
[98] E.S. Spelke, Cognition in Infancy, in: Occasional Papers in Cognitive Science, Vol. 28, MIT Press, Cambridge, MA, 1983.
[99] E.S. Spelke, Where perceiving ends and thinking begins: The apprehension of objects in infancy, in: A. Yonas (Ed.), Perceptual Development in Infancy. Minnesota Symposia in Child Psychology, Erlbaum, Mahwah, NJ, 1987, pp. 197–234.
[100] E.S. Spelke, The origins of physical knowledge, in: L. Weiskrantz (Ed.), Thought Without Language, Oxford University Press, New York, 1988, Chapter 7, pp. 168–184.
[101] T.E. Starner, Visual recognition of American sign language using hidden Markov models, Master's Thesis, MIT, Cambridge, MA, 1995.
[102] K. Sugihara, Machine Interpretation of Line Drawings, MIT Press, Cambridge, MA, 1986.
[103] P. Suppes, L. Liang, M. Böttner, Complexity issues in robotic machine learning of natural language, in: L. Lam, V. Naroditsky (Eds.), Modeling Complex Phenomena, Springer, Berlin, 1991.
[104] L. Talmy, Force dynamics in language and cognition, Cognitive Sci. 12 (1988) 49–100.
[105] R. Thibadeau, Artificial perception of actions, Cognitive Sci. 10 (2) (1986) 117–149.
[106] J.K. Tsotsos, Some notes on motion understanding, in: Proceedings of IJCAI-77, Cambridge, MA, 1977, p. 611.
[107] J.K. Tsotsos, J. Mylopoulos, ALVEN: A study on motion understanding by computer, in: Proceedings of IJCAI-79, Tokyo, Japan, 1979, pp. 890–892.
[108] J.K. Tsotsos, J. Mylopoulos, H.D. Covvey, S.W. Zucker, A framework for visual motion understanding, IEEE Trans. Pattern Anal. Machine Intelligence 2 (6) (1980) 563–573.
[109] S. Tsuji, A. Morizono, S. Kuroda, Understanding a simple Cartoon film by a computer vision system, in: Proceedings of IJCAI-77, Cambridge, MA, 1977, pp. 609–610.
[110] S. Tsuji, M. Osada, M. Yachida, Three dimensional movement analysis of dynamic line images, in: Proceedings of IJCAI-79, Tokyo, Japan, 1979, pp. 896–901.
[111] S. Tsuji, M. Osada, M. Yachida, Tracking and segmentation of moving objects in dynamic line images, IEEE Trans. Pattern Anal. Machine Intelligence 2 (6) (1980) 516–522.
[112] A.J. Viterbi, Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, IEEE Trans. Inform. Theory 13 (1967) 260–267.
[113] D.L. Waltz, Toward a detailed model of processing for language describing the physical world, in: Proceedings of IJCAI-81, Vancouver, BC, 1981, pp. 1–6.
[114] D.L. Waltz, L. Boggess, Visual analog representations for natural language understanding, in: Proceedings of IJCAI-79, Tokyo, Japan, 1979, pp. 926–934.
[115] S.H. Weber, A. Stolcke, L_0: A Testbed for Miniature Language Acquisition, Technical Report TR-90-010, International Computer Science Institute, 1990.
[116] T. Winograd, Understanding Natural Language, Academic Press, New York, 1972.
[117] J. Yamoto, J. Ohya, K. Ishii, Recognizing human action in time-sequential images using hidden Markov model, in: Proceedings of the 1992 IEEE Conference on Computer Vision and Pattern Recognition, IEEE Press, 1992, pp. 379–385.