MODELING AND RECOGNIZING ASSEMBLY ACTIONS

by

Jonathan D. Jones

A dissertation submitted to Johns Hopkins University

in conformity with the requirements for the degree of

Doctor of Philosophy

Baltimore, Maryland

October 2021

Abstract

We develop the task of assembly understanding by applying concepts from computer vision, robotics, and sequence modeling. Motivated by the need to develop tools for recording and analyzing experimental data for a collaborative study of spatial cognition in humans, we gradually extend an application-specific model into a framework that is broadly applicable across data modalities and application instances. The core of our approach is a sequence model that relates assembly actions to their structural consequences. We combine this sequence model with increasingly-general observation models. With each iteration we increase the variety of applications that can be considered by our framework, and decrease the complexity of modeling decisions that designers are required to make.

First we present an initial solution for modeling and recognizing assembly activities in our primary application: videos of children performing a block-assembly task. We develop a symbolic model that completely characterizes the fine-grained temporal and geometric structure of assembly sequences, then combine this sequence model with a probabilistic visual observation model that operates by rendering and registering template images of each assembly hypothesis. Then, we extend this perception system by incorporating kine-

matic sensor-based observations. We use a part-based observation model that compares mid-level attributes derived from sensor streams with their corresponding predictions from assembly hypotheses. We additionally address the joint segmentation and classification of assembly sequences for the first time, resulting in a feature-based segmental CRF framework. Finally, we address the task of *learning* observation models rather than constructing them by hand. To achieve this we incorporate contemporary, vision-based action recognition models into our segmental CRF framework. In this approach, the only information required from a tool designer is a mapping from human-centric activities to our previously-defined task-centric activities.

These innovations have culminated in a method for modeling fine-grained assembly actions that can be applied generally to any kinematic structure, along with a set of techniques for recognizing assembly actions and structures from a variety of modalities and sensors.

Primary advisor: Sanjeev Khudanpur

Co-advisor: Gregory Hager

Acknowledgments

During my Ph.D. I have been lucky to have good advisors, collaborators, and friends. I would like to thank my advisor, Sanjeev, for first giving me a chance by admitting me to the program, and for giving me an appreciation for mathematical rigor, an attention to detail, and the diligence to keep working on something until it is right. I would also like to thank my co-advisor Greg for his support during the long second half of my Ph.D., where I began to apply all the theories and frameworks I had gathered from books and classes. From Greg I learned the art of pragmatism: to focus on the concrete over the abstract, to be creative in solving new problems, and to communicate methods and motivations effectively.

I would also like to thank our collaborators in the project that motivated my dissertation: Barbara Landau, Amy Shelton, Cathryn Cortesa, and Emory Davis. Collaborating on this project has allowed me to experience almost the full range of engineering development, from designing sensing systems to developing formal languages, and their perspective as experts in human cognition has been invaluable to me as I tried to give similar abilities to machines. Without their ideas, this document would never have existed.

Finally, I would like to thank my friends and family. Thanks to my parents,

Dan and Lisa Jones, for first encouraging me to study engineering, and for always supporting me regardless of how poorly or well things were going for me at any moment. Thanks to my wife, Sadhwi Srinivas, for being there to share all the grad school experiences—with you, everything is better. And thanks to the many friends I made while at Hopkins, who made this education into something truly fun.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Enabled by large datasets collected from YouTube [58, 36, 41] and powerful neural classifiers, the field of action recognition has made impressive progress in "web-scale" applications. Web-scale applications usually involve the classification of a very large set of short video clips (on the order of tens of seconds to a minute) drawn from a diverse set of high-level action categories—for example, the benchmark dataset UCF101 contains classes like `playing guitar`, `playing piano`, `cutting in kitchen`, and `skydiving`. Because they perform classification at a semantic (as opposed to physical) level, these systems can take advantage of contextual information in the scene, such as the appearance of the background or the presence of an object.

Web-scale action recognition is an important task, and is critical for supporting video search and retrieval tools. However, action recognition systems also have use in situated environments—application-specific settings like collaborative robotics, surgical training, or manufacturing [46]. These applications require systems that are capable of finer-grained distinctions. Rather than deciding whether a video clip is an instance of `cutting in kitchen` or

`skydiving`, action recognition systems that analyze cooking videos might need to distinguish between slicing off the stalk of an onion and dicing it.

This dissertation focuses on a particularly complex application of situated action recognition: perceiving the assembly of an object from its constituent parts. In this task, a system must make predictions about whether individual parts are joined and how they are oriented with respect to one another. Instead of using contextual cues to disambiguate between action categories, these assembly processes require systems that can reason geometrically and temporally, relating the structure of an assembly to the manipulation actions that created it.

1.1 Motivation

Applications of assembly action understanding fall into two broad categories: automation and analysis. In the automation of assembly activities, a perception system provides information about the current state of a spatial assembly to a higher-level planner. In this capacity its function is similar to that of an observer in the closed-loop control paradigm. For example, when a collaborative robot is working with a human partner to build a piece of furniture or an industrial part, the robot must be able to perceive which parts are connected and where, and it must be able to reason about how these connections can change as the result of its human partner's actions.

In the analysis of assembly activities, the system's predicted action sequence is used to compute a set of downstream measures—for example, whether an action will result in an erroneous construction. This need can arise in indus-

trial process monitoring and manufacturing applications, where an automated system might verify a product as it progresses through the assembly line or be used to instruct workers on the proper assembly of a new part. Interestingly, it also can be applied to behavioral studies of spatial cognition. The methods described in this dissertation were motivated by the development of a system for automatic, quantitative measurement of spatial reasoning skills in children and adults.

1.2 Thesis statement and contributions

This dissertation is based on the idea that understanding assembly actions requires a spatio-temporal model that relates those actions to their structural effects. We show that such a model has several uses: it enables detailed behavioral analyses, aids in the development of multimodal assembly perception systems, and provides a prior model that can guide the output of neural classifiers in structured prediction tasks.

In Chapter 3 we present a general method for modeling assembly actions and their structural effects, and demonstrate its application to the visual perception and analysis of assembly behaviors. This chapter is based on a prior publication in the IEEE Winter Conference on Applications of Computer Vision (WACV), along with two applications published in Cognitive Science:

- J. Jones, G. D. Hager, and S. Khudanpur. "Toward Computer Vision Systems That Understand Real-World Assembly Processes". In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2019, pp. 426–434. DOI: 10.1109/WACV.2019.00051

- Cathryn S Cortesa et al. "Characterizing spatial construction processes: Toward computational tools to understand cognition". In: *Annual Meeting of the Cognitive Science Society*. 2017, pp. 246–251

- Cathryn S Cortesa et al. "Constraints and Development in Children's Block Construction". In: *Annual Meeting of the Cognitive Science Society*. 2017, pp. 246–251

In Chapter 4 we modify the application-specific perception system of Chapter 3 to address the more general task of sensor-based perception of assembly sequences. We develop a framework for constructing observation models that makes use of kinematic sensor-based attributes, and apply our method to videos of IKEA furniture assembly in addition to the block-assembly scenario of Chapter 3. The methods of this chapter were published in Robotics and Automation Letters:

- J. D. Jones et al. "Fine-Grained Activity Recognition for Assembly Videos". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3728–3735. DOI: 10.1109/LRA.2021.3064149

In Chapter 5 we further generalize the methods of Chapter 4, focusing on *learning* observation models using video-based action recognition. In doing so, we take the opportunity to revisit the relationship between assembly actions and a task-oriented state that was established in Chapter 3. This chapter contains previously-unpublished work that was was influenced by a collaborative publication at AAAI:

- Tae Soo Kim et al. "DASZL: Dynamic Action Signatures for Zero-shot Learning". In: *AAAI* (2021)

In summary, the main contributions of this dissertation are the following:

- A structured sequence model for assembly actions;
- A system for the collection, annotation, and analysis of multimodal data from a behavioral study of spatial cognition;
- Vision and sensor-based perception systems for recognizing assembly structures;
- A method for integrating vision-based neural classifiers with a structured assembly model.

Chapter 2

Background

2.1 Motivation: Cognitive study

The methods in this dissertation were originally motivated by studies in cognitive science, which examine the development of spatial reasoning skills in children using the assembly of DUPLO block structures as an experimental methodology [7, 9].

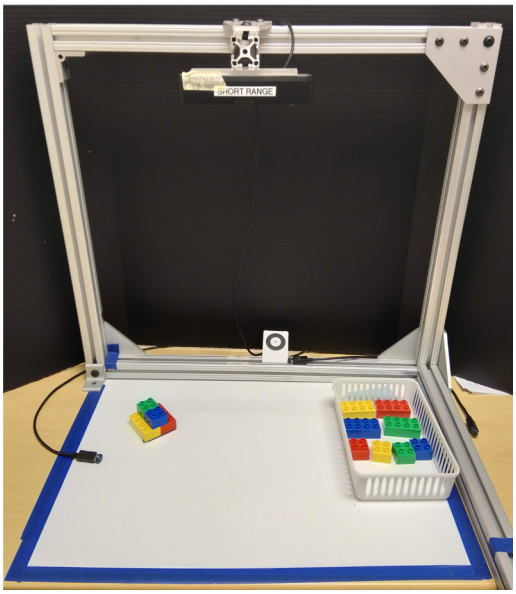

Figure 2.1: Video collection system

Figure 2.2: IMU in 3D-printed enclosure

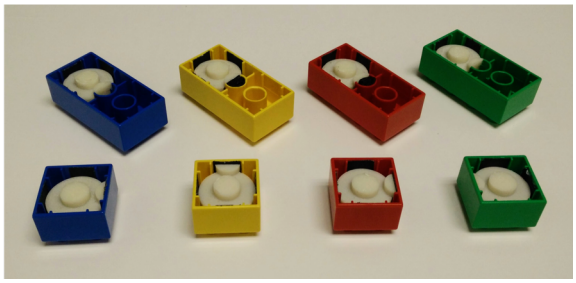

Figure 2.3: Blocks fitted with IMU enclosures

We developed a system that collects multi-modal data during such child behavioral studies. In these studies, children from 4–8 years of age are asked to copy one of the six models in Figure 2.4. Each model can be constructed from a set of eight blocks—four 2x2 square blocks and four 2x4 rectangular blocks, each in colors blue, yellow, red, and green. During an experimental trial, the child participant is given the exact set of blocks that is required for the model they have been asked to copy. A university ethics review board approved all study procedures, and participants and their legal guardians provided informed assent and consent, respectively.

Our data collection system records video using a Primesense Carmine RGBD camera mounted in a top-down orientation, as shown in Figure 2.1. It also wirelessly streams inertial measurements from each block using eight

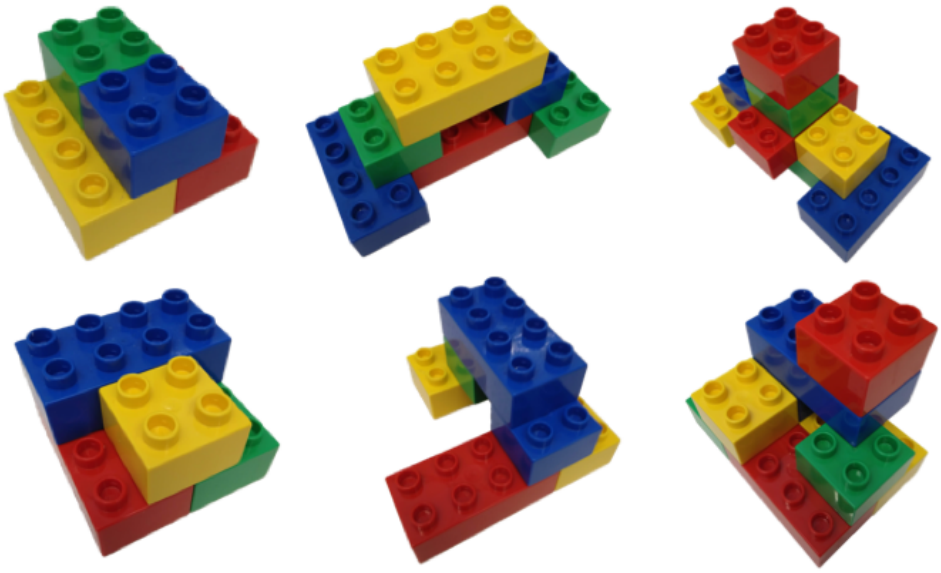

Figure 2.4: Target models to be assembled from the blocks

mBientlab metawear CPRO inertial measurement units (IMUs). Each DUPLO block was custom-modified to fit a 3D-printed enclosure that holds an IMU and implements the fittings that allow DUPLO blocks to connect to each other (see Figure 2.2).

The system records video at 30 frames per second, and streams IMU samples at 50 Hz. All samples are marked with a global timestamp at the time the data collection computer receives them, allowing signals to be synchronized for later analysis. After data collection, these recordings are analyzed with the goal of revealing individual differences between participants. The remainder of this dissertation is concerned with enabling these psychophysical analyses by first developing a representation that captures the fine-grained details of assembly behavior, then developing perception methods capable of automatically inferring these representations from recordings. In doing so, we develop

a framework for modelling and perceiving assembly activities that applies not just to the recordings from this study, but to assembly scenarios in general.

2.2 Related work

The assembly of an object from atomic parts is an application of significant interest in engineering fields, and several prior works in robotics and computer vision have addressed various forms of this task. However, our motivating study requires a level of detail and generality that existing methods cannot provide. In this dissertation we synthesize detailed models of spatial assemblies developed in robotic perception with general-purpose methods for vision-based action recognition.

2.2.1 Applications of assembly perception

Since the early 1990s, assembly processes have been examined in the planning community for the purpose of autonomous construction. Initially, de Mello and Sanderson [24] used an AND-OR graph to represent and search through the set of all (error-free) assembly plans. Knepper and colleagues [38] extended this framework to generate distributed assembly plans (*i.e.* plans for multiple workers) and applied their method to furniture pieces. An alternative line of work uses hierarchical task networks (HTN), which break down the top-level task into a series of sub-tasks using a tree-like data structure similar to a context-free grammar (CFG), and which include the previous AND-OR graphs as a special case [21]. When applied to the assembly of a structure, the sub-tasks of an HTN usually represent useful or semantically-meaningful

sub-assemblies (*e.g.* the nose or tail of an airplane). There is an interest in learning HTNs from human demonstrations, to do away with the need for detailed, task-specific plan programming [21, 50].

Several works in the human-robot interaction (HRI) community establish concrete applications of assembly action recognition systems. Moreover, since each system is developed *ad hoc* for the application, they demonstrate the need for a unifying framework for this task. Gupta and colleagues [16] developed a user interface that tracks the assembly of a DUPLO model. This work represents the model's state using each block's pose in a voxelized space, and updates the state as users add to it one block at a time. Although this system implicitly uses a grammar on block construction when it creates the set of update candidates, it does not represent the kinematic constraints imposed by block connections. Furthermore, it only maintains a single hypothesis when tracking the structure. This is feasible for their interface, whose users build one block at a time, carefully display the model in multiple viewpoints, and correct system errors if necessary. But a system that operates autonomously or works with less-engaged users (like a collaborative robot) will need a mechanism for operating under incomplete information and for correcting past inference errors.

Later, Hadfield and colleagues [17] implement an assembly parsing system for their experiments in child-robot interaction. In this application, a robot supervises children as they attempt to build planar, rectangular structures made up of four to six parts. Their system infers the assembly's structure by tracking each part's pose (6-DoF position and orientation) using RGB and

depth video and comparing against a set of pre-defined connection hypotheses, which they incorporate into the state of their tracker. Finally, Wang and colleagues [68] use an assembly parsing system in a human-robot collaborative assembly setting. They use a HTN to define an assembly plan for IKEA furniture, then track progress to assist human builders. Their system tracks part positions from first-person videos using fiducial markers, and determines part connections heuristically by comparing the relative pose between two parts to a set threshold. They also detect connecting and screwing actions, which the robot used to better orient the part for the convenience of its human partner.

2.2.2 Spatio-temporal structure for assembly perception

In robotic perception, methods have largely focused on discerning time-invariant kinematic structures by using instrumented sensing systems. Although prior work exists going back to the late '90s [10, 29, 55], the first complete probabilistic model for identifying kinematic systems was developed by Sturm and colleagues [62, 63, 60, 59, 61]. Using a graphical model to account for probabilistic dependencies between objects, this method estimates a time-invariant kinematic system from position data obtained using fiducial markers and, in one experiment, from plane-fitting on range data. Later, Niekum and colleagues [53] extend this model to perform changepoint detection of time-varying kinematic systems. Using their method, they identify configurations which can alter the kinematic behavior of a system (for example, the angle at which a stapler's hinge locks in place). In a related line of work, Martín-Martín and

Brock use a hierarchy of extended Kalman filters to estimate static kinematic systems by tracking features in an RGB-D video stream [49]. They also extend their method to multiple modalities by including a robot's end-effector motion as another observation signal.

These efforts have produced a rigorous and detailed theoretical framework for the estimation of kinematic structures from a variety of pose or location data (*i.e.* object poses, feature point locations, or end-effector configurations). However, even methods that analyze time-varying kinematic systems currently lack a framework for describing systems that evolve dynamically—for example, due to the effects of assembly actions.

Conversely, a thread of work in activity recognition that was roughly contemporary with the aforementioned publications in robotics focus on the temporal structure of assembly. These approaches address a simplified version of the assembly setting in which there is at most one way to connect each pair of parts and in which once connected, parts are never disconnected. In other words, they treat assembly actions at the object level, rather than at the level of intra-object contact points. This prevents systems from recognizing the full range of assembly actions that may be encountered in realistic deployment scenarios.

Summers-Stay and colleagues [64] apply an action grammar to the task of parsing a handful of kitchen and craft activities—namely `cooking vegetables`, `making sandwich`, `sewing a toy`, `card making`, and `assemble a machine`. This work is part of a larger effort to develop a formal grammar on manipulation actions in analogy to the Minimalist Program (an influential framework

in the field of theoretical syntax) [72, 71, 15]. Central to their approach is the MERGE operator, which they use to represent actions and compute their effects. They observe that just as a noun and a determiner can be combined to form a noun phrase, distinct object instances are combined into composite entities during manipulation actions. However, where the non-terminal nodes in a syntactic parse tree represent interpretable linguistic concepts, the non-terminals in their activity trees simply represent the order in which objects were combined. Additionally, this grammar has no mechanism for handling the disassembly of an object—they note that future work should study "the ability to build up trees based on disassembly and transformation as well as assembly".

A similar effort by Vo and Bobick [66] uses hand-defined probabilistic context-free grammars (PCFGs) to instantiate a graphical model for the purpose of segmenting and classifying compositional action sequences. They evaluate their method on a dataset of of toy-airplane construction videos, which they created and introduced in the same work, along with cooking and human activity datasets. Like the work of Summers-Stay and colleagues, this system, given an assembly video, produces a parse tree whose terminal nodes represent primitive objects and whose non-terminals represent composite entities (*i.e.* partial assemblies of the final product). However, this work additionally required that non-terminals correspond to interpretable sub-parts, such as the nose, wings, body, and tail of an airplane. While this requirement suits the hierarchical structure of a PCFG, it may be unrealistic for actual assembly parsing applications—any build sequence that does not conform to the pre-

specified sequence of sub-parts cannot be recognized by their model, even if it results in a final assembly that is present in the dataset. For instance, their framework has no way to represent an unexpected assembly sequence in which someone builds the body, then builds part of the nose, then builds the tail, and finally comes back to finish the nose.

Outside of assembly applications, recent work in computer vision has begun to address the estimation of structured variables. Within these structured problems, the degree of precision varies. As an example from action recognition, Koppula and colleagues [39] model human-object interactions using a spatio-temporal graph. These interactions are described at a relatively coarse level (for example, "subject opening microwave"). At a more abstract level, some recent work in computer vision has focused on parsing *semantic* graphs from images [30, 70]. Motivated by image retrieval applications, these methods seek not only to identify and localize objects, but also to estimate *relationships* shared between pairs of objects (for example, "man riding horse"). At the finest-grained level are methods which estimate *physical* image parses. The sequential scene parsing system developed by Hager and Wegbreit [18] estimates a graph whose vertices represent objects and whose edges represent support relationships between objects. Similarly, Jahangiri and colleagures [27] and Wu and colleagues [69] estimate structured, physical representations from images, although these representations are not posed as graphs explicitly.

These methods in computer vision and action recognition ultimately fall short in actual assembly parsing applications because of their coarse-grained representations of assembly actions. Our application adds a level of detail

beyond object-level action recognition or adjacency relationships: in addition to describing *which* two objects are connected, we also must describe precisely *how* they are connected by specifying the exact kinematic relationship between the two objects. By relating action sequences to an assembly's kinematic structure, our representation captures the full variability of assembly situations.

2.2.3 Domain-general vision systems for assembly

Most contemporary action recognition research has focused on very large datasets of human-centric actions that give a high-level summary of the events occurring in a video. As representative examples, consider two prominent, publically-available benchmark datasets: Something-Something [13] and Kinetics [6]. Each item in these datasets is a short (<30 seconds) video clip with a single (semantic) action label—for instance, `picking up a shoe` or `washing dishes`. The goal in these evaluations is to assign every clip its correct action label.

The most successful methodologies for this task have centered on learning high-quality representations. In most state-of-the-art methods (for example, I3D [6], TSM [47], and SlowFast [12]), a pretrained object-classification architecture such as ResNet [22] is inflated in the temporal dimension and fine-tuned on the target dataset. These methods require minimal input from the user and, in most circumstances, perform very accurately after they are trained. Fundamental assumptions in this approach are that the items in the output vocabulary are abstract, disjoint labels with nothing in common, and that

model outputs can be treated independently given the neural representation. Because of this, these methods are rarely applied in situations where video labels have complex structure, and none of these methods directly supports user-defined model structure.

Another family of approaches in action recognition focuses on *fine-grained* video understanding. This subfield was comprised of smaller datasets with more complex label structure, which focus more on application-level analysis than high-level human behavior. For example, an item in the JIGSAWS dataset is comprised of a video of a practice surgery along with the entire sequence of actions that occur [2]. Methods in this area tend to focus more on capturing the structure of the task through symbolic reasoning: stochastic parsers based on regular or context-free grammars captured temporal structure [26], while attribute grammars represented part-based structure [11, 48]. In general, these symbolic methods provide a direct way to impose constraints that come from first-principles knowledge about the problem structure. However, most existing methods require *too much* user input in the form of observation models or symbolic grammars. In many applications of fine-grained action recognition, researchers may not yet have a detailed understanding of how observations are generated or what structure governs the evolution of an action sequence. This can lead to models with weak representations and an arbitrary or non-intuitive grammar. For example, the previously-mentioned work by Vo and colleagues augments a sequence model with a latent variable that represents task progress. This purely-symbolic method is highly robust [66], but it requires a special-purpose model crafted by hand for every appli-

cation, and the relationship of their latent variable to the task is never made explicit—thus there is no general principle for applying this method to a new application domain.

As neural representation learning methods improved, a handful of methods combined ideas from both pattern recognition and symbolic reasoning techniques—most notably the segmental CNN [43]. This approach uses a segmental CRF to guide the output of a VGG-inspired vision module. This model operates directly on the action sequence, learning a dense set of pairwise action-transition parameters which have no direct interpretation in the context of the task. In Chapter 5 we build on these approaches to produce a flexible method that can be trained to recognize the assembly of any structure. We equip this hybrid model with modern neural architectures, eliminating the need for handcrafted observation models. We also equip it with a part-based sequence model that jointly tracks the action sequence along with a theoretically meaningful state variable representing task progress. This plays to the strengths of symbolic sequence models while minimizing the need for *ad hoc* modeling.

In summary, the modeling and recognition of assembly sequences is a broadly-useful task in robotics and computer vision, with a variety of established applications. However, most solutions to date have been either too specific (completely solving a single problem instance), or too general (partially solving many problem instances) to constitute a reusable framework that adequately captures an unconstrained setting like that of our motivating study. In this dissertation we combine concepts from symbolic sequence modelling,

sensor-based robotics, and deep computer vision into a single framework that applies not only to our application scenario, but to the assembly of kinematic structures in general.

Chapter 3

A first-principles model for assembly activities

3.1 Introduction

Motivated by a desire to enable better human-robot collaboration and finer-grained behavioral analyses, researchers in computer vision and robotics have recently begun to approach the challenging problem of assembly action recognition [16, 17, 68, 33, 25]. In assembly activity recognition, a perception system must understand the construction of a structure (*e.g.* a piece of furniture or a toy block tower) as it is built up from a set of primitive objects. This task requires perception at a finer level of detail than most activity recognition research to date. Typical action recognition methods are concerned with modeling scene-level information, and usually involve the classification of a very large set of short video clips, each on the order of tens of seconds to a minute, and each exemplifying one of a diverse set of high-level action categories. For example, the benchmark dataset UCF101 entails recognizing actions like `playing guitar`, `playing piano`, `cutting in kitchen`, and

`skydiving`. In this setting, a video's background is sufficient to distinguish between most actions (*e.g.* `skydiving` and `playing guitar`). In *fine-grained* action recognition, on the other hand, there is less inter-class variability between categories. [54]. For instance, a system that automatically generates a recipe from a cooking video will need to distinguish between action categories defined at a finer level than `cutting in kitchen`—it needs to understand *what* is being cut, and *when*. These approaches are generally concerned with perception at the level of objects, rather than scenes.

Understanding assembly actions requires perception at a level of geometric detail even finer than object-level, which, to our knowledge, has not been attempted in the action recognition literature to date. Systems must perceive not only *which* objects are connected to each other and *when*, but also *where* they are connected—that is, which specific contact points (screw holes, for example) are connected between two parts. In short, assembly action recognition requires a framework that unifies both semantic and geometric computer vision.

In this chapter, we describe a method for performing fine-grained structural parsing of time-series derived from spatial assemblies. As a case study, we apply this method to the specific task of parsing the construction of block structures from video. This process provides an interesting setting to test methods for very fine-grain geometric inference, time-series video parsing, and occlusion-robust image parsing.

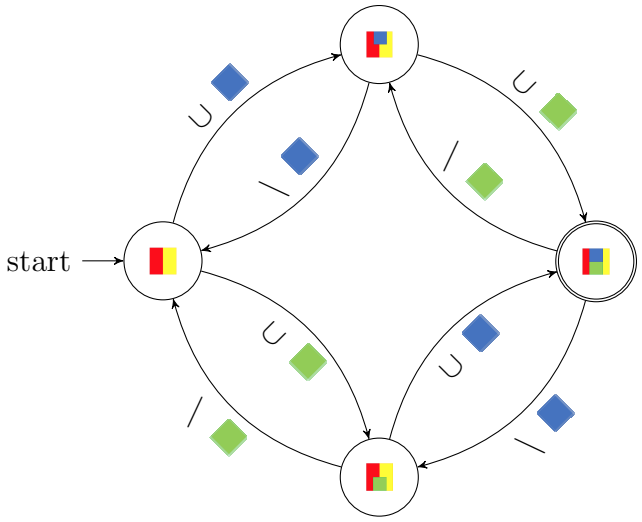

Figure 3.1: A finite-state machine that accepts action sequences, and whose states are kinematic linkages. This system represents all the assembly sequences that have the same initial structure as the sequence in Figure 3.2, and whose actions add or remove the blue and green square blocks.

3.2 Representation of assembly processes

Previous approaches to recognizing assembly actions (or equivalently, assembly structures) are scattered across several sub-disciplines in vision and robotics, and each addresses its own task independently. However, in this chapter we observe that there is a fundamental structure to the problem that is shared by all methods. We develop a framework that unifies approaches based on action recognition and approaches based on kinematic model estimation. We treat an assembly process as a sequence of kinematic structures which transition from one to another as the result of a (human) user's action. We define a variety of comparisons between kinematic linkages and assembly sequences, and describe how this framework has been used to understand build patterns in studies of spatial cognition. Then, we use our model to recognize assembly actions in a

toy block building task.

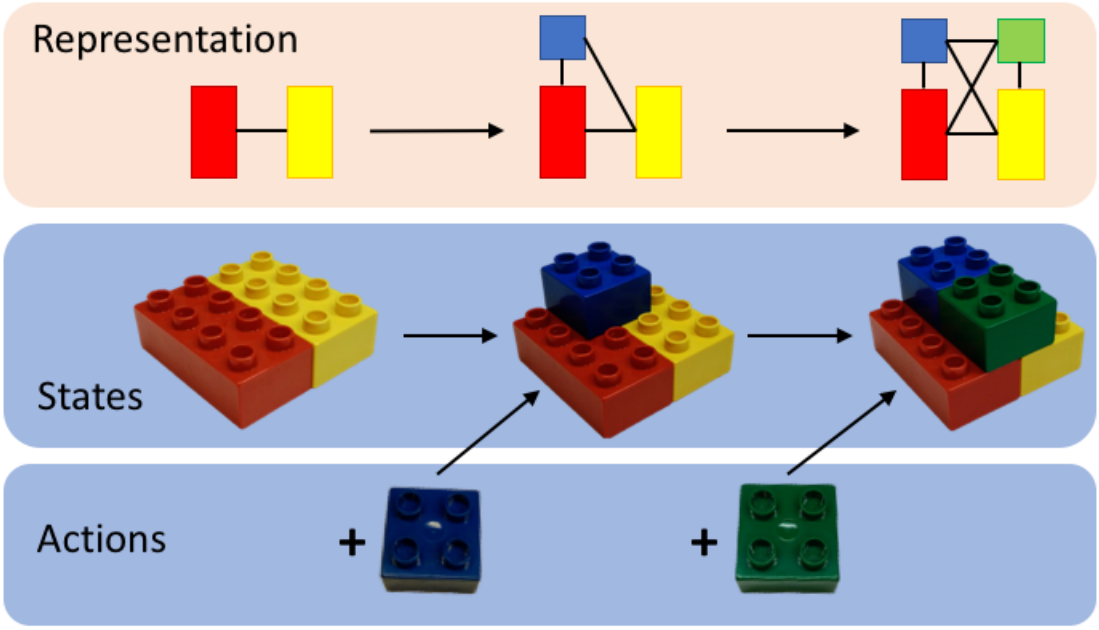

Figure 3.2: Partial example of an assembly process. The builder alters the state by adding or removing blocks in specific ways.

3.2.1 Assembly structures as kinematic constraint graphs

Consider as an example the assembly sequence shown in Figure 3.2, in which a person begins with two rectangular blocks aligned along their long sides and consecutively adds two square blocks. Each action the builder takes causes the assembly to transition to a new state, with a different set of kinematic links.

Generalizing the example above, we represent a spatial assembly as a graph whose vertices correspond to primitive objects (*i.e.* its component parts). If two objects are joined to each other, the vertices representing them are joined by an edge representing the constraint imposed by that joint. Finally, each edge is associated with a label that parameterizes its constraint.

Formally, we define the state of a spatial assembly as a edge-labeled graph $G = (\mathcal{V}, \mathcal{E}, \psi, \mathcal{P}, \mathcal{J}, L)$.

- The vertex set $\mathcal{V} = \{v_1, \ldots, v_n\}$ is a collection of labels identifying objects.
- The edge set $\mathcal{E} = \{e_1, \ldots, e_m\}$ is collection of labels identifying undirected edges that join pairs of distinct vertices.
- The incidence function $\psi : \mathcal{E} \to \mathcal{V} \otimes \mathcal{V}$ is a mapping from edge labels to the undirected edges they identify. We write each undirected edge as $e_i = (v_j, v_k)$ with $v_j < v_k$.
- The vocabulary of connection points $\mathcal{P}$ identifies all the locations on each object where another object can be connected—for example, a screw hole in a piece of furniture, or a stud or hole in a DUPLO block.
- The joint vocabulary $\mathcal{J} \subseteq \mathcal{P} \otimes \mathcal{P}$ parameterizes all possible physical connections between any pair of objects. Impossible connections, such as one between two DUPLO studs, are not present in this vocabulary.
- The edge labeling function $L : \mathcal{E} \to \mathcal{J}$ is a mapping from edges to their physical connections, represented in the joint vocabulary.

In the case of our Blocks study, we represent the connection point vocabulary using a discretized coordinate system whose axes are aligned with the studs on each part. We define the joint vocabulary to be a discretized version of SE(3), representing the relative pose of one block in the coordinate

frame of the other (rotations are constrained to ninety-degree increments in the horizontal plane).

In some situations we will encounter assemblies whose objects form multiple, disjoint sub-parts. In terms of our graph representation, each sub-part corresponds to a separate connected component in the graph.

3.2.2 Actions and assembly sequences

A builder assembles an object over time by taking actions that change that object's kinematic structure. In general, kinematic changes can take many forms: for example, many folding mechanisms such as chairs or gates are capable of locking into a stable configuration with fewer degrees of freedom. They could also describe changes in physical support relationships between objects in a scene, as in the work of Hager and Wegbreit [19].

In this chapter we focus on the assembly (or disassembly) of kinematic structures. We define assembly actions as kinematic changes that add or remove parts from the structure. Each action consists of an action type (`connect`, `disconnect`, or `identity`) and a partial kinematic graph δs. Unlike our representation of spatial assemblies, the *partial* graph δs only contains edges for joints that will change. With this information, we can compute the successor set of any state s:

- $\texttt{connect}(s, \delta s) \coloneqq s \cup \delta s$
- $\texttt{disconnect}(s, \delta s) \coloneqq s \setminus \delta s$
- $\texttt{identity}(s) \coloneqq s$

Thus, given an spatial assembly and an action, we can construct the resulting spatial assembly. We can also perform the *inverse* operation: given a final assembly s' and an initial assembly s, we can construct the action that caused this change,

$$s' - s \coloneqq \begin{cases} \texttt{connect}(\cdot, s' \setminus s) & \mathcal{E}_s \subset \mathcal{E}_{s'} \\ \texttt{disconnect}(\cdot, s \setminus s') & \mathcal{E}_{s'} \subset \mathcal{E}_s \\ \texttt{identity}(\cdot) & \mathcal{E}_{s'} = \mathcal{E}_s. \end{cases} \tag{3.1}$$

Note that this operation is only defined in the case that $\mathcal{E}_{s'} \subseteq \mathcal{E}_s$ or vice versa: in any other situation, there is not a single action that transforms s into s'. Instead, this transformation would require multiple actions.

Although the assembly process's start and end states can in principle be any states, in our experiments we will always begin at the state in which no objects are connected (we call this the *empty* state, because its edge set is empty).

3.3 Measuring differences between spatial assemblies

Usually, a model's state space is considered to be a set without any extra structure. This makes the comparison of states straightforward: the only possibility is to measure whether they are the same item or not. However, because its state is grounded in the fine-grained geometric structure of a spatial assembly, our framework admits analyses at multiple levels. This allows us to evaluate not only *if* two states differ, but also to measure *how*.

The canonical evaluation of equivalence in the kinematics literature measures if two mechanisms have the same general connectedness structure—that

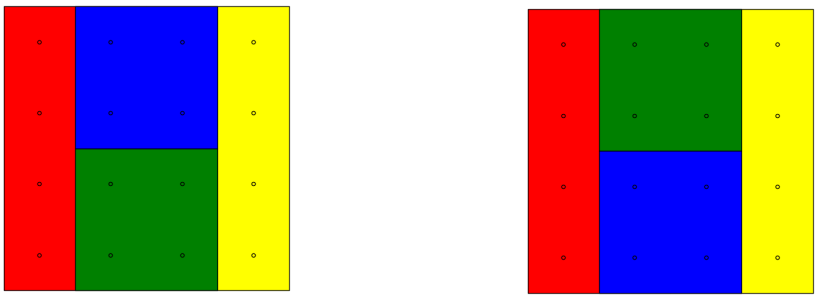

Figure 3.3: Schematic diagram illustrating a correct four-block model (left) and a commonly-encountered, incorrect copy (right). These assemblies are topologically equivalent, but they are not geometrically equivalent.

is, if their edge sets are equivalent. We call this type of measurement *topological* equivalence. This type of measurement characterizes the degrees of freedom in a linkage, but will not distinguish between two states that differ in the *location* of a single block placement. In the context of our experimental study it is critical to note the exact placement of each block, because minor errors such as the one illustrated in Figure 3.3 (in which the locations of the blue and green blocks are exchanged) are frequently encountered. To disambiguate between these assemblies, we introduce the concept of *geometric* equivalence: two linkages s and s' are geometrically equivalent if they are topologically equivalent (thus they have the same connected components $\mathcal{C}$), and for each connected component $c \in \mathcal{C}$, there exists a rigid transformation that registers s'_c to s_c.

3.4 Application to behavioral study

This framework, along with associated data collection, annotation, and analysis systems, has enabled qualitative and statistical analyses for a project

studying the development of human spatial cognition [7, 9]. In these studies, our framework was used to annotate assembly actions for more than 600 videos and parse them into the kinematic structures they produce. Then, these structure sequences were aggregated into group-level state machines for the purpose of visualization and quantitative analysis. Measures used in downstream statistical analyses include the in/out degree of a state in the state machine, the edit distance between each path and a selected reference path, and whether a state is "correct" or not—*i.e.* , whether it is a sub-assembly of the goal state. Details of this study, including preliminary findings, have been published in [7] and [9].

3.5 A first-principles model for classifying assembly sequences

In this section we provide a probabilistic model of an assembly process, from which we derive an algorithm for recognizing an assembly process from video. We base our model on a time-series structure inspired by a partially-observable Markov decision process (POMDP), which originates in the optimal control literature [4]. However, unlike typical control problems, our task is not to *take* a sequence of *actions* which optimizes some reward function, but rather to *recognize* the sequence of *states* resulting from actions that are being taken by an agent.

3.5.1 Observation model

Our graph-based representation of an assembly is sufficient to differentiate between assemblies with different structures, but not to render an image of the assembly process. We must additionally specify the global pose $p_i \in \mathrm{SE}(3)$ of each sub-part (*i.e.* each connected component in the state graph—recall that we define the set of a graph's connected components as $\mathcal{C}$). For notational convenience, we collect these poses into the variable $p = \big(p_1, \ \ldots, \ p_{|\mathcal{C}|}\big)$.

We model the joint probability of an image I, generated by state s in pose p, as

$$\mathrm{P}(I_t, p_t, s_t) = \mathrm{P}(I_t \mid p_t, s_t)\, \mathrm{P}(p_t \mid s_t)\, \mathrm{P}(s_t)\,. \tag{3.2}$$

If we have an appearance and a shape model for each object, and the camera parameters are known, we can generate a template T by rendering each sub-part using the poses in p. Disconnected sub-parts may be rendered independently if we ignore collisions and occlusions.

To account for the error incurred by rendering an idealized template, we model each pixel of the observed image as an independent Gaussian random variable, with mean given by the template pixel value and variance σ^2 (which we treat as a hyperparameter). Letting the mask $\mathcal{M}$ represent nuisance pixels in the image,

$$\mathrm{P}(I_t \mid p_t, s_t) = \prod_{i,j \notin \mathcal{M}} \mathcal{N}(I_t(i,j); T(i,j;p_t), \sigma^2). \tag{3.3}$$

To model the pose distribution of each sub-part, we assume the orientation is uniformly distributed on the unit sphere and the translation is uniformly distributed on a bounded volume of $\mathbb{R}^3$ visible to the camera.

3.5.2 Process model

In an assembly process, the state s_t is constructed or deconstructed through the actions $a_0, \ldots, a_{t-1}$ of some agent. In the case that we have partial information about the actions that were taken (for example, the posterior distribution from a black-box action recognition system), we can use it to construct the state transition probabilities:

$$\mathrm{P}(s_t \mid s_{t-1}) = \sum_{a_{t-1} \in \mathcal{A}} \mathrm{P}(s_t \mid s_{t-1}, a_{t-1}) \, \mathrm{P}(a_{t-1} \mid s_{t-1}) \tag{3.4}$$

However, it is common that such information is not available. In this case we can treat the assembly process as a Markov chain and estimate the state transition distribution directly.

Combining this with our observation model in Section 3.5.1, we obtain a final expression for the joint probability of all observed and inferred variables:

$$\mathrm{P}(I_{1:T}, p_{1:T}, s_{1:T}) = \prod_{t=1}^{T} \mathrm{P}(I_t \mid p_t, s_t) \, \mathrm{P}(p_t \mid s_t) \, \mathrm{P}(s_t \mid s_{t-1}) \tag{3.5}$$

This is represented by the graphical model in Figure 3.4.

3.5.3 Inference

Given $I_{1:T}$, we estimate the state sequence using a Viterbi-style decoding algorithm on the graphical model in Figure 3.4. That is, we solve the problem

$$p^*_{1:T}, s^*_{1:T} = \arg\max_{p_{1:T}, s_{1:T}} \mathrm{P}(I_{0:T}, p_{1:T}, s_{1:T}) \tag{3.6}$$

using max-sum message passing [40]. The solution to this problem may be explained as a hypothesize-and-test or analysis-by-synthesis approach. For

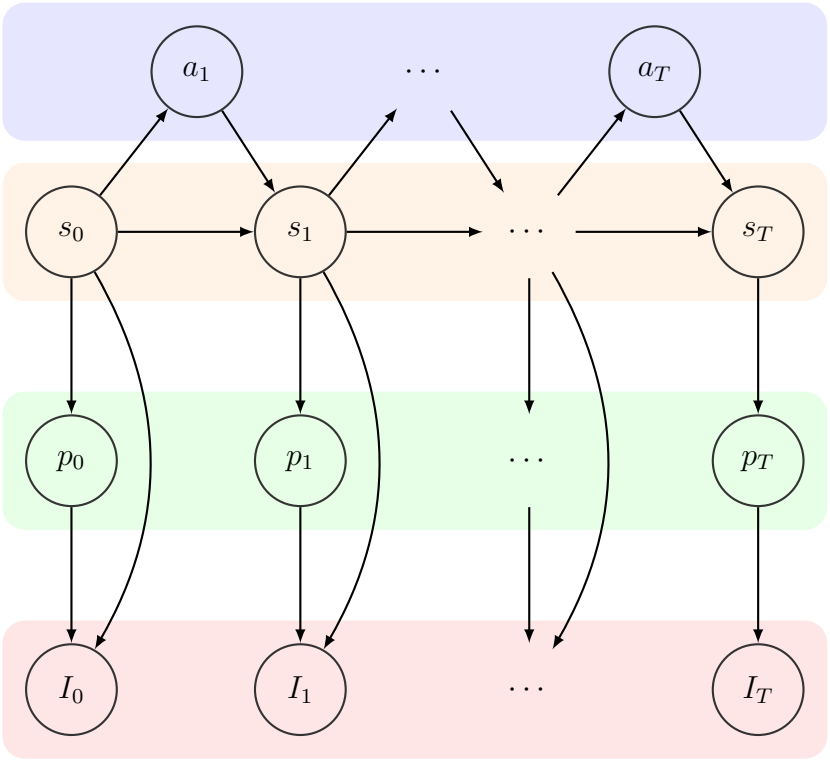

Figure 3.4: Graphical model corresponding to Equation 3.5. Each shaded region depicts a sequence of input, intermediate, or output variables. From top to bottom: assembly actions, assembly states, block poses, and observed keyframes.

each image, we generate a set of hypotheses about the assembly state. We evaluate each hypothesis *locally* by rendering a template and registering it to the observed image. Finally, we decode *globally* by choosing the most probable sequence of hypotheses according to our model.

3.5.3.1 Hypothesis generation

The time-series structure in our problem allows us to efficiently select and decode hypotheses using the Viterbi algorithm with beam search [28]. At each time step t, we prune any hypothesis with low probability—*i.e.*, any state s_t with

$$\max_{p_t} \mathrm{P}(I_{1:t}, p_{1:t}, s_{1:t}) < G \max_{p_t, s_t} \mathrm{P}(I_{1:t}, p_{1:t}, s_{1:t}) \,, \tag{3.7}$$

where G is a fixed constant. We then construct the hypothesis set for the next sample by running our model forward one step in time and including any state with nonzero prior probability. This adaptive pruning method allows the system to ignore most states when it is certain about a sample, but to consider more hypotheses when its uncertainty increases.

Every assembly process begins with a special state in which no objects are connected to each other. We call this state the *empty state* because its edge set is empty, and initialize the hypothesis set with only this state.

3.5.3.2 Template registration

At each time step we evaluate state hypotheses locally using $\mathrm{P}(I_t, p_t^* \mid s_t)$, the joint probability of the image and the best assembly pose under the hypothesis. We compute this best pose using template registration: once we have a hypothesis for the block state, we render a template T in a canonical pose

p_c for each sub-part in the assembly. Under the assumption that all object motion is planar, we can optimize over the pose in the pixel coordinates of the image rather than in the world coordinate frame.

Since the pose is uniformly distributed, the log probability $\log \mathrm{P}(I_t, p_t \mid s_t)$ of the image in a particular pose is proportional to the log likelihood $\log \mathrm{P}(I_t \mid p_t, s_t)$, which itself is proportional to a simple sum-of-squared-errors (SSE) distance metric. Thus, the registration problem for each sub-part's template is

$$R^*, \tau^* = \arg\min_{R,\tau} \sum_{(i,j)\notin\mathcal{M}} \left\| I((i,j)) - T(R(i,j)+\tau; p_c, s) \right\|^2, \qquad (3.8)$$

where (R, τ) is a rigid motion in the space of pixel coordinates. This is a non-linear least-squares optimization problem, which can be solved using standard methods. For implementation details, see Section 3.6.3.2.

3.6 Parsing Block Construction Processes

We apply the algorithm in Section 3.5.3 to the task of parsing videos of DUPLO block building activity, previously described in Chapter 2.

3.6.1 Dataset description

3.6.1.1 Controlled dataset

To evaluate our system under conditions which match the core modeling assumptions, we constructed a controlled dataset. We collected video and inertial measurement data from 30 experimental trials. We performed five examples of each of the six models in Figure 2.4 ourselves, taking care to ensure that there was an unobstructed view of each assembly as it was being created. Each video

in the dataset was annotated with the actions that occurred, along with their start and end times. These actions were parsed to construct a state sequence for each video. In total, 46 unique states were encountered in this dataset. For each state in a video, a video frame that captured an unobstructed view for each assembly state was hand-picked. In what follows, we refer to these frames as *keyframes*.

3.6.1.2 Child's play dataset

To evaluate the robustness of our system, we constructed a dataset from our recordings of child behavioral experiments, previously mentioned in Chapter 2. In each of the 145 videos, a child participant attempted to copy one of the six models shown in Figure 2.4. The data collection and annotation setup was identical to the one for the controlled dataset. When marking keyframes, if there was no clear view available for a state, the least-occluded frame was selected. See [8] for more information about the data collection and annotation process.

Figure 3.5: Example keyframes from a block assembly video. Each image represents the clearest view of a block assembly in the video.

As any parent can attest, children don't always do things the way you expect. This dataset contains many confounding factors, such as significant and frequent occlusion of the spatial assembly by children's hands and arms. Furthermore, 311 unique states were encountered in this dataset compared to 46 for the controlled dataset, 60.8% of these states have only one observed

example, and the empty state accounts for 13.8% of all observations. This results in a highly skewed prior on the state space (see Figure 3.6).

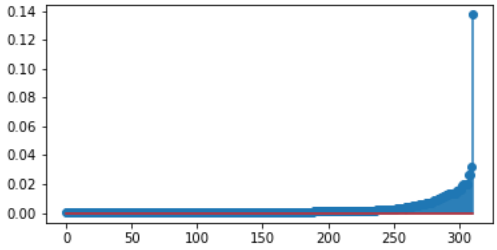

Figure 3.6: Prior distribution of states in the child's play dataset. Horizontal axis shows the (sorted) state index, vertical axis shows the state probability.

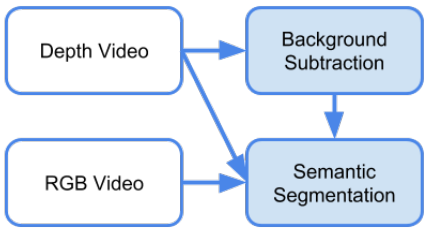

Figure 3.7: Image preprocessing pipeline

3.6.2 Image pre-processing

3.6.2.1 Background subtraction

We remove the background by fitting a plane to the depth image and masking all pixels within a set distance from the plane. We also mask the left-most portions of each image, since this region almost never contains the block assembly and frequently contains distracting objects.

3.6.2.2 Semantic segmentation

Semantic segmentation is necessary due to the presence of hands and other confounding objects in the frame. We use a simple method based on color-space segmentation and majority voting.

Pixel-level classification: To mitigate the effects of partial occlusions, we need a classifier that predicts whether pixels in the foreground of an image belong to blocks or hands. We define c_0 to be the class of blocks pixels, and c_1 to be the class of hands pixels. Since we do not have a dataset that is manually annotated with these labels, we constructed a proxy by combining the keyframes from Section 3.6.1.1, which we assume do not contain hands, and a set of images from the child's play dataset, which we assume all contain some portion of a hand. We constructed the child's play image set by randomly selecting 30 videos, then randomly selecting one frame for each state graph in the video.

For classification we chose a latent-variable mixture model. The latent variable k indexes a set of K Gaussian distributions with unit covariance, and is shared between both pixel classes. The probability of an image under this model is

$$\mathrm{P}(I) = \prod_{(i,j)\notin\mathcal{M}} \sum_{n=0}^{1} \sum_{k=1}^{K} P(c_n)\,\mathrm{P}(k \mid c_n)\,\mathrm{P}(I(i,j) \mid \mu_k)\,. \tag{3.9}$$

We estimate the means μ_k of each Gaussian distribution by training a mini-batch K-means [57] model on the proxy dataset described above. We estimate the conditional probabilities $\mathrm{P}(k \mid c_n)$ by computing histograms of Gaussian components for the clear-view and obstructed-view images, and $\mathrm{P}(c_0) =$

$P(c_1) = 0.5$ by our construction of the training set.

Using this model, we classify each pixel by assigning it to the nearest Gaussian component and assigning the Gaussian component to its most probable image class. To mitigate potential failures in the background model, we also assign pixels to the background class if their saturation or depth values are below set thresholds.

Segment-level classification: We segment frames into superpixels using the implementation of SLIC [1] in scikit-image [67]. Then, we assign segments larger than a set threshold to the background to avoid detecting obviously incorrect objects such as sleeves or arms. Finally we group contiguous pixels assigned to the same class, giving a set of segments which we use as object proposals.

3.6.3 Other implementation details

3.6.3.1 Parameter estimation and hyperparameters

We estimate the state-to-state transition probabilities of the model in Figure 3.4 using the empirical distribution of the training set. We set the appearance model of each block to maximally-saturated colors: red, blue, yellow, or green. For RGB data we set the the observation variance σ^2 to 1, and for depth we set it to 100. Finally, we choose $K = 32$ Gaussian components for the mixture model used during semantic segmentation.

3.6.3.2 Inference

Template registration: We solve the optimization problem associated with registering a template to an image frame (described in Equation 3.8) using the

Trust Region Reflective algorithm [5] implemented in SciPy [31]. We initialize τ at the centroid of each of the image segments classified as blocks in Section 3.6.2.2 and sample 25 uniformly-spaced values on the unit circle for R. We treat any pixels classified as hands as *missing data*, leaving them out when computing the value of the SSE objective. In the case that a block assembly has more than one connected component, we compute the best assignment of components to image segments using the Hungarian algorithm [51].

Decoding: To compensate for preprocessing errors and occluded segments, we apply add-one smoothing [28] to the HMM state transition probabilities when inferring the state sequence the child's play dataset. We add one extra count to each transition leading into the empty state, and one extra count to each transition leading out of the empty state. We set the Viterbi pruning coefficient G to zero when evaluating the system in Sections 3.7.1.1 and 3.7.1.2, and investigate its effect separately in Section 3.7.1.3. When doing combined RGBD inference, we register RGB and depth templates separately and send the sum of their resulting log probabilities as the input to the Viterbi decoder. This is equivalent to assuming that the color and depth modalities are independent.

3.7 Experimental Setup and Results

To test the system's performance in a setting that matches the modeling assumptions, we first evaluate it on the controlled dataset. We train using the full child dataset and test on the full controlled dataset.

To test the system's usefulness in a real-life setting, we evaluate it on the

child's play dataset. In this dataset, state sequences are annotated for every video, but keyframes are only annotated for 72 videos. This means we can train the HMM's state transitions on all 145 videos, but our test set is limited to the 72 videos with annotated keyframes. We use leave-one-video-out cross validation and report average metrics across folds.

3.7.1 Evaluation

We evaluate accuracy, precision, and recall at three levels of granularity. At the *block* level, we ask whether the system has correctly estimated which blocks have been incorporated into the model, *i.e.* correctly detected vertex membership in each connected component. The *edge* level measures whether the system has correctly estimated which pairs of blocks are joined in the model, *i.e.* correctly identified each edge present. The *state* level is the most precise, measuring if the system has correctly estimated every block, edge, and edge *label* (corresponding to relative block poses).

3.7.1.1 Controlled dataset

Table 3.1 shows the results of our system on the controlled dataset. State accuracy is above 90% when parsing RGB data, with precision and recall nearly matching this performance. These results show that our system works well when the observed data match the expectations of our model. As may be expected, performance is worse when parsing depth data. Since half the blocks look identical to each other when color is ignored, the system has no way of distinguishing between different states with the same adjacency structure. However, results on combined data show that including depth frames along

modality	state acc.	state prec.	state rec.
rgb	92.53	91.14	91.14
depth	59.58	51.36	51.36
combined	92.50	91.14	91.14
	edge acc.	edge prec.	edge rec.
rgb	97.11	99.54	97.48
depth	78.55	85.02	84.22
combined	97.11	99.54	97.48
	block acc.	block prec.	block rec.
rgb	98.00	99.62	98.33
depth	87.83	91.60	92.09
combined	98.00	99.62	98.33

Table 3.1: Results, controlled dataset (macro-averages)

with RGB does no worse than RGB on its own—in this situation, it may be the case that the <8% of confusions from the RGB modality simply do not have corroborating information in the depth modality.

3.7.1.2 Child's play dataset

Table 3.2 shows the results of our system on the child's play dataset. From these results we see that although system performance degrades at the finest-grained level of detail, the system's multiple hypotheses and implicit prior world model enable robust estimates at a coarser level. Specifically, an edge precision better than 91% indicates that the system usually predicts a state that is similar to a subset of the ground-truth state. Again, system performance is lower when parsing depth data. The system's performance is worse

modality	state acc.	state prec.	state rec.
rgb	62.02	62.84	56.14
depth	32.92	21.42	20.49
combined	59.72	53.47	53.63
	edge acc.	edge prec.	edge rec.
rgb	67.01	91.28	69.19
depth	41.27	50.18	52.80
combined	69.05	84.90	72.25
	block acc.	block prec.	block rec.
rgb	71.44	93.20	73.47
depth	60.40	70.15	77.25
combined	76.05	90.84	79.36

Table 3.2: Results, child's play dataset (macro-averages)

using combined RGB and depth data than using RGB data alone, which may be a sign of a sub-optimal fusion strategy—rather than providing corroborative information, the depth modality likely assigns enough confidence to incorrect predictions that it decreases the RGB modality's overall performance.

3.7.1.3 Effect of Viterbi pruning

Although we did not prune any states during inference in sections 3.7.1.2 and 3.7.1.1 to obtain the best possible system performance, we investigate its effect here. Figure 3.9 and Figure 3.11 show our system's performance on the controlled and Child's Play datasets (respectively) as the Viterbi pruning coefficient G is varied. In both cases, it is possible to ignore large proportions of the state space before noticeable performance degradation is incurred. Fig-

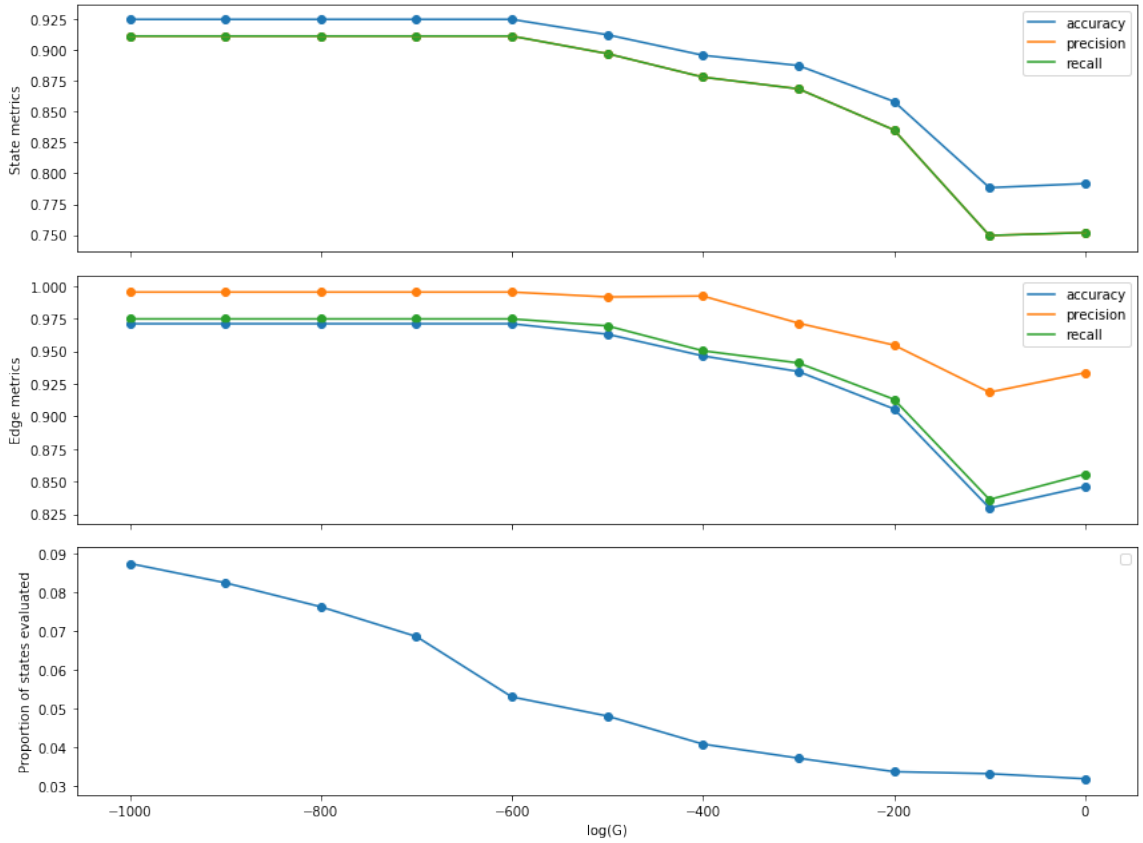

Figure 3.8: Controlled dataset

Figure 3.9: System performance on our controlled dataset as a function of the Viterbi pruning coefficient. From top to bottom: state metrics, edge metrics, and proportion of states visited.

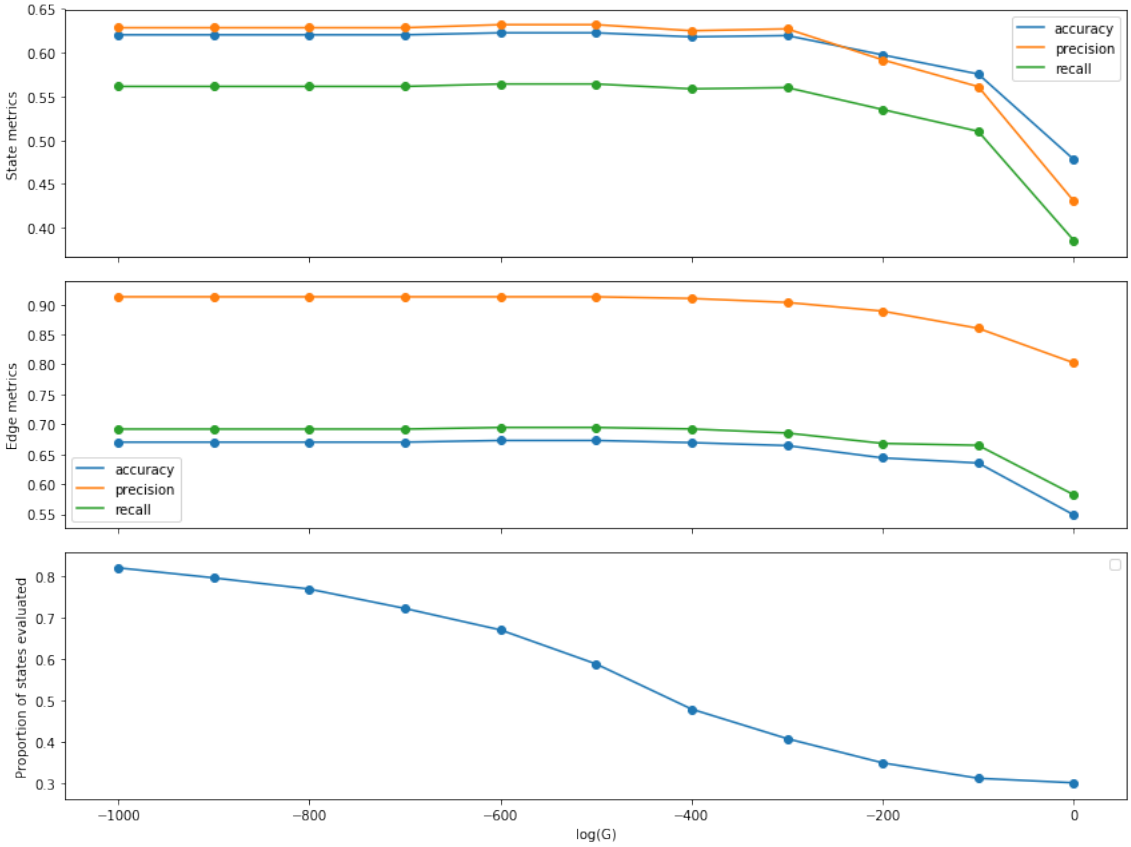

Figure 3.10: Child's play dataset

Figure 3.11: System performance on the Child's Play dataset as a function of the Viterbi pruning coefficient. From top to bottom: state metrics, edge metrics, and proportion of states visited.

ure 3.9 shows that the system's high confidence on the clear evidence of the controlled dataset leads to significant efficiency gains. Even at the most conservative value for G, it visits fewer than 9% of the possible states on average. This trend continues, though to lesser effect, in the child's play dataset results in Figure 3.11. The challenging nature of the child's play dataset, in addition to the transition smoothing we apply into and out of the empty state, lead to much greater uncertainty during inference. However, the state space can still be pruned by about 60% on average before performance degrades noticeably.

3.8 Conclusion

In this chapter we have outlined a model for recognition of assembly processes, derived a probabilistic inference algorithm, and applied it to parsing RGB videos of a block construction task. We evaluated on two datasets: one in a controlled setting, and another consisting of unconstrained data collected from child behavioral experiments. Results show that our system performs almost perfectly when data conditions match the modeling assumptions, and continues to give sensible results under much more challenging data conditions.

Results from these experiments suggest new research directions, particularly in fine-grained action recognition and in occlusion-robust computer vision. In this work the state parser operates on its own, but this model can work alongside an explicit action recognition system to enable more efficient and more accurate inference.

The methods and experiments presented here rely on one manually identified keyframe per state. When there is no single unoccluded view available

for a state, it may be possible to reconstruct most of the model image by aggregating information across multiple consecutive frames. Furthermore, removing the reliance on manual keyframe extraction is a natural next step in developing truly autonomous systems.

Finally, the multiple sensor modalities in our datasets offer a unique chance to explore methods for RGBD + IMU data fusion. For instance, IMU signals could be used as input for the explicit action recognition system described above. Alternatively, orientation estimates could be derived from the IMU signals to improve template registration. This is addressed in the subsequent chapter.

Chapter 4

Multimodal perception for assembly activities

4.1 Introduction

In Chapter 2 we established that most previous applications for perception of assembly structures have consisted of *ad hoc* systems for a particular application instance. In many of these cases, the application domains make use of instrumented sensors such as augmented-reality tags, the pose of a robotic end-effector, or embedded inertial measurement units (IMUs). A higher-level framework that unifies these sensor models would reduce the amount of expertise required to construct these systems, making them more general and easier to use for practitioners.

In Chapter 3 we developed a generic framework for modeling assembly sequences, but our perception system used an observation model that was developed for our particular case study. This limited the generality of our approach. In this chapter we develop a part-based framework for defining observation models that is well-suited to applications with instrumented sensors.

We demonstrate the generality of this method by evaluating on a new dataset of IKEA furniture assembly, which consists of egocentric RGB videos and AR markers, in addition to our primary case study.

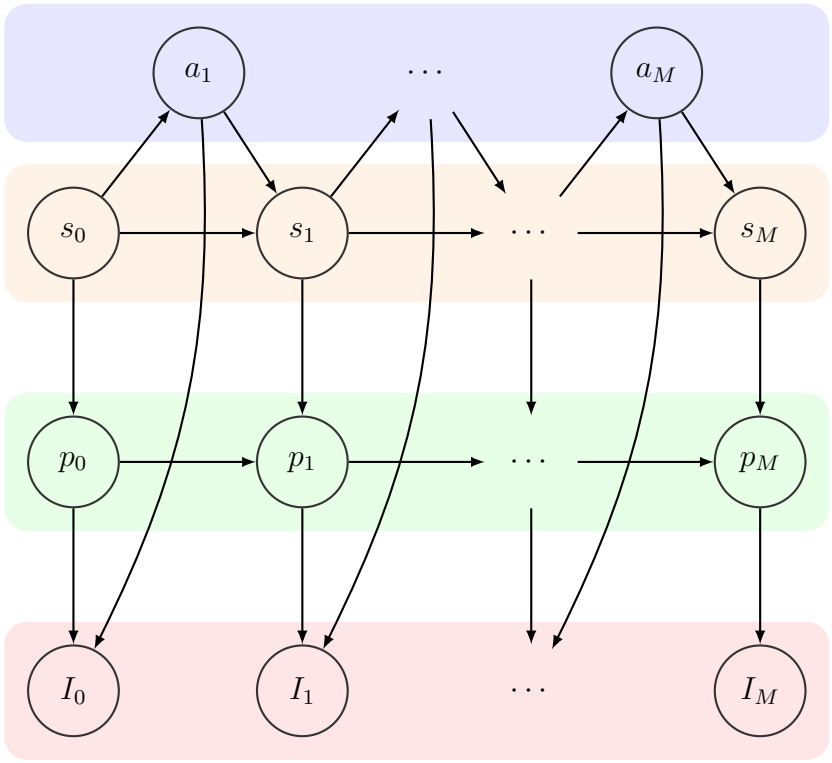

Figure 4.1: Graphical model depicting the generation of a video sequence from an assembly process. The shaded regions depict the various processing stages into which the task is frequently decomposed. From bottom to top: object recognition, object tracking, kinematic perception, action recognition. Each variable in the figure corresponds to a segment of the video—*i.e.* variables indexed with 0 correspond to the video segment depicting the assembly in its initial state and so on.

4.2 A discriminative model for joint classification and segmentation

In the assembly action recognition setting, we observe a sequence of discrete-time samples $x = x_1, \ldots, x_T$. Here x usually represents a video, but could also be a sequence of object poses or other intermediate attribute predictions. Each individual assembly s_m is realized as a segment of these samples: the m-th assembly generates the observation segment $x_{b_m}, \ldots, x_{e_m}$, where b_m is the beginning index of segment m and e_m is its ending index. We will use $s = s_1, \ldots, s_T$ to refer to the assembly labels for each individual sample $1, \ldots, T$. In this chapter, our systems are tasked with segmenting and classifying an observation sequence, producing as output a sequence of assembly structures $s = s_1, \ldots, s_T$.

In Figure 4.1 we illustrate a graphical model which depicts the generation of a video sequence from an assembly process[1].

We implement our models using a segmental conditional random field (CRF). This family of models is useful in the assembly action recognition setting because it generalizes both traditional probabilistic methods (*e.g.* hidden Markov models, which we use in Chapter 3, or their segmental generalizations) and contemporary neural methods (complex, learned features that feed into a logistic classifier, which we make use of in this chapter and in Chapter 5) [65]. Segmental CRFs provide the flexibility to implement first-principles, probabilistic methods for applications with few to no examples to

[1] In defining this generative process we are not prescribing that all approaches to assembly action recognition be strictly generative, or even probabilistic—rather, we use it as a tool to examine the problem and compare methods.

train on, to incorporate statistically-learned classifiers (*e.g.* neural networks) when datasets are big enough that learning can improve performance, and to enforce temporal consistency in the model's output.

4.2.0.1 Scoring assembly sequences

The total score of a sequence under a segmental CRF is constructed by combining the individual scores of each segment with sequence-level information produced by score_{seq}. This function takes two consecutive spatial assemblies as input, and produces a vector of transition features which model the relationship between assembly actions and kinematic structure as output. The total score of a sequence can be expressed mathematically as

$$\begin{aligned} \text{score}\,(x, s) = \sum_{m=1}^{M} &\text{score}\,(x_{b_m:e_m}, s_m, s_{m+1}) \\ &+ W_{seq}^{T}\,\text{score}_{seq}\,(s_m, s_{m+1})\,. \end{aligned} \tag{4.1}$$

In Equation 4.1, b_m is the beginning index of segment m, e_m is the ending index, and $x_{b_m:e_m} = x_{b_m}, \ldots, x_{e_m}$ is the part of the observed sequence corresponding to that segment. W_{seq} is a vector of real-valued coefficients that scales the effect of each transition feature. In our experiments we use a single transition feature: the empirical log transition probabilities estimated from a set of training data (in contrast to our previous work in Chapter 3, these probabilities are not smoothed).

We score each individual segment using a weighted combination of observation features, produced by the function score_{obs}. These scores model the relationship between observations and kinematic structures (or actions), and

usually include object poses as an auxiliary variable:

$$\text{score}\left(x, s_m, s_{m+1}\right) = \sum_t W_{obs}^T \, \text{score}_{obs}\left(x_t, s_m, s_{m+1}\right). \tag{4.2}$$

Again, W_{obs} is a vector of coefficients that determines the contribution of each observation feature. Our observation features score state *transitions*: they can depend on both the current assembly structure s_m and the next assembly s_{m+1}. This allows us to score action-related evidence and assembly-related evidence using the same framework—for example, identifying a screwdriver in use is a good indication that a `screwing` or `unscrewing` action is happening at that instant. On the other hand, observing that two parts move with a constant relative pose means they are probably connected on the same structure. For action features, we use the notion of assembly difference $a = s' - s$ defined in Section 3.2.2 to compute an action score $\text{score}_{obs}\left(x_t, s_m, s_{m+1}\right) = \text{score}(x_t, s_{m+1} - s_m) = \text{score}(x_t, a_{m+1})$. For assembly features, $\text{score}_{obs}\left(x_t, s_m, s_{m+1}\right) = \text{score}(x_t, s_m)$. The exact observation will depend on the characteristics of the application data—for example, we use distances between estimated object poses for a furniture-assembly task with AR markers, and we use pixel-level template registration scores along with IMU-based attributes for the block-play dataset.

As we just established, a segmental CRF takes advantage of temporal structure by breaking down a sequence's score into a sum over conditionally-independent segment transitions. Our representation of a spatial assembly has a similar graphical structure, but in this case edges represent *physical* dependencies rather than temporal ones. For instance, we could measure each part's pose independently as a set of unary features corresponding to the vertices of

the spatial assembly. Likewise, we could measure the relative pose between every pair of parts as a set of pairwise features corresponding to the edges of the spatial assembly. This decomposition is well-suited to data collected from instrumented sensors, because it is usually possible to obtain high-quality, direct measurements from each part (in our experiments, datasets feature per-part IMU sensors or AR markers).

We use this notion to define a part-based observation model. We factor the score of a spatial assembly into a sum over the scores of each possible edge, *i.e.*

$$\text{score}_{obs}\left(x_t, s_m\right) = \sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{i-1} \text{score}\left(\hat{f}_{ij}\left(x_t\right), f_{ij}\left(s_m\right)\right). \tag{4.3}$$

To score each pair of vertices we estimate the value of some physical attribute from our observation x_t, like the difference in translation between parts i and j, or whether parts i and j are on the same rigid body. Then, we predict the value of that same attribute for a hypothesized spatial assembly s_m and compare the similarity between the estimated and predicted values. We define $\hat{f}_{ij}\left(x_t\right)$ to represent the estimated value, and $f_{ij}\left(s_m\right)$ to be the predicted value.

As Sturm and colleagues point out [62], edges in a kinematic constraint graph are only conditionally independent when the graph is tree-structured (for example, in a kinematic chain). Our observation model can be interpreted as a second-order approximation, where higher-order dependencies are ignored for reasons of computational efficiency. Experimentally, we will show this approximation performs quite well.

4.2.1 Segmenting and classifying assembly sequences

To recognize assembly actions we need to find $s^*(x)$, the best labeling of assembly structures for a particular observation sequence $x = x_1, \ldots, x_T$, *i.e.*

$$s^*(x) = \arg\max_{s \in \mathcal{S}^{\otimes T}} \sum_{m=1}^{M} \sum_{t=b_m}^{e_m} W_{obs}^T \, \text{score}_{obs}\left(x_t, s_m, s_{m+1}\right) + W_{seq}^T \, \text{score}_{seq}\left(s_m, s_{m+1}\right). \tag{4.4}$$

In some cases segment boundaries for these labels are already known—for example, they may be provided by an upstream change point detector or by hand, as was done in Chapter 3. In this case, the Viterbi algorithm solves Equation 4.4 in $O(|\mathcal{S}|^2 T)$ time, where $\mathcal{S}$ represents the vocabulary of spatial assemblies. If boundaries are not known *a priori*, the sequence can be jointly segmented and classified using a segmental generalization of the Viterbi algorithm with worst-case runtime complexity $O(|\mathcal{S}|^2 T^2)$. The quadratic dependence in the sequence length can be reduced to a constant factor $O(k|\mathcal{S}|T)$ if there is an upper bound on the duration of a segment [56] or the number of segments in a sequence [44].

4.3 Experiments

In this section we present two experiments evaluating the performance of our method on datasets that were collected during studies targeted at human-robot interaction or spatial cognition. As yet, no public benchmarks exist for testing assembly action recognition methods in the general case. However, the datasets we consider illustrate two realistic application scenarios.

4.3.1 IKEA furniture

We first apply our model to the furniture-assembly demonstration dataset of Wang, Ajaykumar, and Huang [68]. In this dataset, 12 participants demonstrated the assembly of an IKEA chair in two different conditions, resulting in a total of 24 videos. The authors made 18 of these videos available to us (corresponding to both conditions for 9 of the 12 participants). In the first condition participants were asked to assemble the structure as efficiently as possible, and in the second they were asked to demonstrate its assembly "like a YouTuber". The chair consists of six different parts: the left and right sides, two support beams, a seat, and a backrest. The left and right sides have three contact points (screw holes), while the rest of the parts have two. In Figure 4.2 we show the component parts laid out before assembly, and the finished chair after assembly. In these trials first-person video was collected from two Logitech C930 cameras which were stacked vertically and mounted on the participant's head. To facilitate easier tracking, every furniture part was also marked with several AR tags.

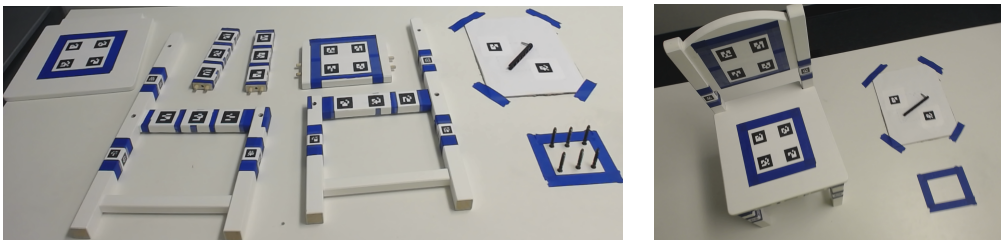

Figure 4.2: LEFT: IKEA parts before assembly. RIGHT: Assembled IKEA chair.

We labeled each video with the assembly actions that happen in it. For

ease of implementation, the partial kinematic graph δs that defines an action is annotated one edge at a time. All labels have the following format:

$$\texttt{action}(\texttt{object}_1, \texttt{object}_2, \texttt{contact points}).$$

The `action` can be one of `connect` or `disconnect`. Each of the `object` arguments identifies one of the six available parts in the setup. The contact points identify which screw holes in each part are connected to each other.

We parse these action sequences to produce a corresponding sequence of assembly configurations. Because the beams of the chair are symmetric and identical, some assemblies are indistinguishable from each other. When parsing, we consider assemblies and actions equivalent up to a swap in the placements of the two beams or a 180-degree rotation of a beam in the X-Y plane. In total, there are 11 unique assembly structures in the dataset.

4.3.1.1 Observation features

We obtain an estimated part pose from each bundle of AR markers and each camera using the ALVAR augmented-reality software library. We then average these pose estimates to obtain a single pose for each part: we compute the Euclidean mean for the position, and the L2 chordal mean for the orientation [20].

Using the estimated part poses, we compute the translation between each pair:

$$\hat{f}_{ij}(I) = \hat{\tau}_i(I) - \hat{\tau}_j(I) = \delta\hat{\tau}_{ij}(I) \tag{4.5}$$

Similarly, for each hypothesis assembly structure s, we predict the translation

between each pair of parts:

$$f_{ij}(s) = \tau_i(s) - \tau_j(s) = \delta\tau_{ij}(s) \tag{4.6}$$

We compare estimated part transforms with predicted ones using the Euclidean distance (scaled by a constant factor λ). To prevent the model from preferring assembly structures with more degrees of freedom, we impose a constant penalty α if a pair of parts is not connected in the hypothesis assembly structure s:

$$\begin{aligned}\text{score}_{obs}\left(I_t, s_m\right) = -\sum_{i=1}^{|\mathcal{V}|}\sum_{j=1}^{i-1}\lambda\|\delta\hat{\tau}_{ij}\left(I_t\right) - \delta\tau_{ij}\left(s_m\right)\|E_{ij}(s) \\ +\alpha(1 - E_{ij}(s)).\end{aligned} \tag{4.7}$$

In Equation 4.7, $E_{ij}(s_m)$ represents element i, j of s_m's adjacency matrix $E(s_m)$, and parameters λ and α are learned during training.

4.3.1.2 Experimental Setup

We use a leave-one-video-out cross-validation setup. For each fold, we estimate the log transition probabilities, along with parameters α and λ, on the 17 training videos, and evaluate performance on the single held-out test video. We set both W_{obs} and W_{seq} to one for every observation and sequence feature. We report frame-level accuracy and edit score averaged across folds, along with their standard deviations. Frame-level accuracy measures the proportion of samples that were labeled with the correct state sequence, and evaluates the system's overall performance in the joint segmentation and classification task. The edit score is defined in [42] as $1 - d(s, s')/\max\{|s|, |s'|\}$, where $d(s, s')$ is the Levenshtein edit distance. This metric evaluates performance

at the segment level: it penalizes sequences with mis-classified or mis-ordered segments, but not sequences that only differ in misaligned segment boundaries.

4.3.1.3 Results

Table 4.1: IKEA furniture-assembly results

	Frame Accuracy	Segment Edit Score
assembly	69.0±13.3	94.0±9.0

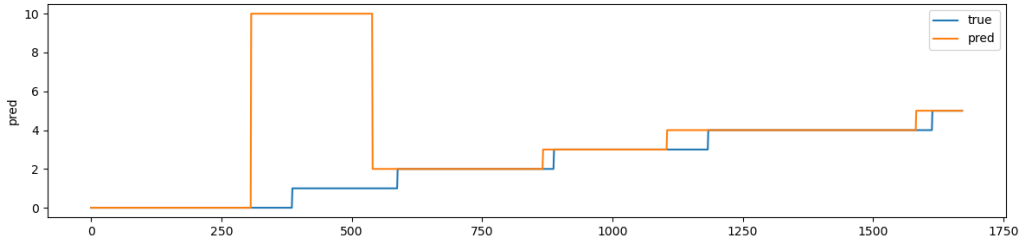

Figure 4.3: Illustration of a recognition experiment for the IKEA task. In this example, the model confuses the order in which beam parts were connected to the left side of the chair. The system's performance on this sequence is close to is average performance: frame accuracy is 74.8% and segment edit score is 83.3%.

In Table 4.1 we report metrics evaluated on assembly sequences. The system's average assembly edit score of 94% shows that it almost always retrieves the correct assembly sequence. This is due in part to the sequence model and the constrained nature of the dataset—errors that might occur if the observation model were operating independently (*e.g.* due to part occlusion or tracking errors) get corrected when when we require that each assembly action must be consistent with the last. The average framewise accuracy of the system is around 70%, which shows that the system is not operating solely

on sequence-model information. When our system makes an error, it usually misclassifies an action near the beginning of a sequence or makes small mistakes in identifying segment boundaries. In Figure 4.3 we present an example sequence exhibiting both of these error types.

4.3.2 Toy Blocks

Next, we return to the DUPLO-assembly dataset of Chapter 3. This dataset is based on the construction of six abstract DUPLO assemblies, and exhibits some interesting challenges in computer vision that have not yet been solved: the block model is frequently occluded during the build process, the assembly and parts can be moved around arbitrarily in the frame, extremely high precision (<15mm) is necessary in order to determine the exact nature of each connection (*i.e.* which studs are joined), there is much more variability in the attested assemblies; more than 300 unique assemblies result from 145 videos.

We labeled these videos using the same format as the IKEA dataset. In total, there are 311 unique assemblies in the dataset.

4.3.2.1 Observation features

Our experiments in this chapter build on the system we developed in Chapter 3, which recognizes assemblies from videos alone by rendering template images of spatial assemblies and registering them to observed video frames. The former method was precise, but not accurate: the visual modality captures the complete scene, which contains all the fine-grained detail necessary to disambiguate different connections between the same pairs of parts. However, the scene also contains noise, like distracting and occluding objects, and there

are several opportunities for upstream failures in the processing chain. We augment this video model with features from the inertial data source, in the belief that these modalities should complement each other well. The inertial modality is accurate, but not precise: our inertial signals are not subject to noise or occlusion, but they can only determine if pairs of blocks are moving together on the same rigid body. However, including information from the visual modality should resolve the inertial model's ambiguities.

Image features As in Chapter 3, we use a pixel-level observation model for our video features. We render an image of the hypothesis, then register it to the segment using a sum of square errors objective. We use the score of the best registration to represent an image's compatibility with a given assembly hypothesis:

$$\text{score}_{obs}^{IMG}(I_t, s_m) = \max_{p \in \mathcal{P}(s_m)} - \sum_{(r,c) \notin \mathcal{M}(I_t)} \|I_t(r,c) - T(r,c;x)\|^2 \tag{4.8}$$

In Equation 4.8, $\mathcal{P}(s_m)$ is the set of object poses consistent with the constraints imposed by assembly structure s_m, $T(\cdot;x)$ is a rendered template image corresponding to object poses x, and $\mathcal{M}(I_t)$ is a mask that identifies skin pixels in the image. We ignore skin pixels in the registration score because they can occlude the block models.

Inertial features We incorporate information from the IMU signals by predicting which blocks are connected (*i.e.* on the same rigid body). We represent this quantity using an indicator variable c_{ij}. Its value is 1 if parts i and j are on the same rigid body, and 0 if they are not. Using a learned classifier,

we estimate the value of c_{ij} from observed angular velocity measurements ω:

$$\hat{f}_{ij}(\omega) = \mathrm{P}(c_{ij} \mid \omega_i, \omega_j)\,. \tag{4.9}$$

We also predict its value for a hypothesized structure s:

$$f_{ij}(s) = \mathrm{P}(c_{ij} \mid s)\,, \tag{4.10}$$

$c_{ij,t}$ can be determined from the assembly structure s by checking whether object i and object j are in the same connected component (*i.e.* whether a path exists connecting vertex i and vertex j). Thus, $f_{ij}(s)$ is a one-hot vector.

We rescale both attributes so their entries are in the range (-1, 1) instead of (0, 1), and we compare estimated part transforms with predicted ones using the inner product. This way, estimated and predicted attributes contribute a positive score if their predictions agree, and contribute a negative score if their predictions do not:

$$\mathrm{score}_{obs}^{IMU}\left(\omega_t, s_m\right) = \sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{i-1} \left\langle 2\hat{f}_{ij}(\omega_t) - 1, 2f_{ij}(s_m) - 1 \right\rangle. \tag{4.11}$$

Connection classification We train a temporal convolutional network (TCN) [45] to predict c_{ij}. We use 5-fold cross-validation on the all 145 videos of the blocks dataset, and hold out 25% of the training data to use as validation in each fold. We train using cross-entropy loss for 15 epochs, and choose the model with highest F1 score on the heldout set. We use the Adam optimizer with a learning rate of 0.001. Our network has 6 layers with the following number of channels: [8, 8, 16, 16, 32, 32], and kernel size 25. We use a dropout rate of 0.2. Overall, the performance of this model is quite good: it retrieves block connections with an accuracy of 92.8 ± 1.5%, precision 87.1 ± 3.1%, recall 93.6 ± 1.8%, and F1 score of 90.2 ± 1.6%.

4.3.2.2 Experimental Setup

In this experiment, we evaluate our system on the subset of 61 videos with keyframes using leave-one-out cross-validation. We create the observation feature vector by concatenating image and inertial features, and we set both W_{obs} and W_{seq} to one for every observation and sequence feature. We use ground-truth segments and keyframes, making this a sequential classification task rather than joint segmentation and classification. For each fold, we estimate the model's state space and state transition parameters using 61 videos and test on the remaining one. Since videos are pre-segmented in this experiment, we only report the edit score (and not the frame-level accuracy).

4.3.2.3 Results

Table 4.2: Results: Block-building dataset

	Edit Score (assembly)
VIDEO	61.2±36.3
VIDEO + IMU	77.5±28.6

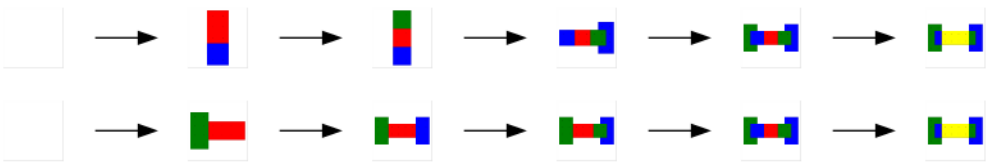

Figure 4.4: Example of an error produced by our system (model orientation is arbitrary in these images). TOP: Predicted sequence. BOTTOM: Ground-truth sequence. The edit distance is 50%, but the model's incorrect predictions are similar to the ground truth.

In Table 4.2, we compare the performance of the baseline video-only model

with the IMU-video fusion system. Including our IMU features improves the average assembly edit score by 16% absolute, corresponding to a 42% relative reduction in error rate. This performance is especially impressive in light of three observations: first, the system is selecting from more than 130 assemblies at every sample. Second, more than 20% of the assemblies in this dataset only occur once. Because we estimate the model's vocabulary from the assemblies attested in the training set, this imposes an upper bound of around 80% on our system performance. Finally, system errors tend to be similar to the true assembly in appearance and shape (in Figure 4.4 we visualize an example).

Table 4.3: Results: Ablation study

	Edit Distance	Rel. Improvement
IMU	59.7	
VIDEO	56.5	5.66
VIDEO + IMU	45.2	25.0
VIDEO + IMU + SEQ	22.5	101

In Table 4.3 we show the results of an ablation experiment examining the contribution of each feature in our model. The first column is the normalized assembly edit distance (*i.e.* one minus the edit score), and the second column is the improvement in edit distance relative to the IMU-only baseline. The inertial and visual modalities operate more-or-less equally alone at roughly 60% error rate. Fusing modalities improves the average error rate by 25% relative to the visual modality alone. Finally, decoding these fused scores with our sequence model gives an additional 100% improvement in error rate

relative to the IMU-only baseline. Taking these results together with those of Table 4.2, we can conclude that all three sources of information contribute to the overall performance of our system.

4.4 Conclusion

In this chapter we outlined a general method for recognizing assembly actions from observation sequences, along with features that take advantage of a spatial assembly's special structure. Then, we evaluated our method empirically on two application-driven data sources: on an existing IKEA furniture dataset, our system recognizes assembly actions with an average framewise accuracy of 70% and an average edit score of 90%, while on a block-building dataset that requires fine-grained geometric reasoning to distinguish between assemblies, our system attains an edit score of 77%.

In this chapter we show our methods work well in two settings: instrumented toy blocks and furniture pieces with AR markers. However, in a practical assembly-understanding application, it may not be desirable to augment video observations at all. In these situations, depth cameras could be used as the main sensing modality, and part identities and poses could be estimated through point-cloud registration. A similar method was used effectively in [19].

This work opens up interesting directions for future research. Representing all possible assembly structures can be both a blessing and a curse: while it allows us to parse assembly actions at the level of detail required for applications, it also causes the vocabulary that we must search through to explode

in size. In this work we have sidestepped the issue by limiting our vocabulary to the assemblies that we encounter at training time, but our blocks experiments show that further performance gains will require a way to recognize previously-unseen assemblies without searching exhaustively through all possible assemblies. Second, developing a robust and application-general observation model for video data could improve the performance and usability of assembly action recognition systems. This would require learned models that require relatively few training sequences and which are also robust to partial occlusions—since 3D models of each part are usually available in assembly scenarios, we believe that simulation-to-real transfer is an intriguing possibility for this line of work.

Chapter 5

Understanding assembly activities through video-based action recognition

5.1 Introduction

As computer vision techniques mature, researchers are beginning to consider applications in the analysis of everyday tasks. Rather than providing a high-level summary of what is happening in a video, systems for task-oriented analysis need to understand how events in a video relate to the goal of the task, and whether that goal is being achieved. This requires a level of understanding beyond simple pattern recognition.

To date, approaches to these tasks have tended to fall within two broad categories: in applications, researchers usually focus on *ad hoc* approaches that provide a satisfying solution for a particular problem instance. On the other hand, researchers focused on core methodology have provided solutions that apply in many scenarios, but which tend to fall short of the needs of an actual application instance.

In this chapter we take an intermediate approach. We give a complete and general solution to a ubiquitous application in the computer vision, human-computer interaction, and robotics literatures: namely, the task of understanding the assembly of an object from its parts. Our method is comprised of two components: one focused on application-generic action recognition, and another focused on application-specific symbolic reasoning. Our action recognition component eliminates the need for *ad hoc* observation models, which required manual specification and hand-tuning in our previous methods, presented in Chapters 3 and 4. Our symbolic reasoning component, on the other hand, prevents the system from needing to learn overly complex output mappings, and grounds the model output in the theoretically-motivated understanding of our application domain, developed in Chapter 3.

Our approach also delves into an interesting aspect of video understanding applications. We investigate the relationship between the *instantaneous actions* that occur in a video and a *persistent state* related to task progress. The majority of conventional tasks in video understanding, such as action recognition, human-object interaction, or video captioning, all occur instantaneously in time. In contrast, for task-focused video analysis it is essential to track some persistent state related to task progress, either explicitly or implicitly. For example, a system that evaluates cooking videos will benefit from the ability to understand the status of the dish—have the eggs been scrambled in the pan? Likewise, for furniture assembly a system should understand which parts are connected to each other. Given conventional methods, it is not possible to derive this information from an action recognition system directly: the

relationship between actions and the task state is not known.

In this chapter we show that a system combining out-of-the box neural action recognition models with a sequence model that relates actions to their structural effects performs well in tracking task progress. We represent spatial assemblies as discrete vectors whose elements are part connections, and action-recognition outputs as partial observations of the transitions between these vectors. We construct perception systems for spatial assembly sequences and assembly actions based on two principles: first, learn a representation that implicitly captures the dynamics of assembly videos. Then, use symbolic reasoning to explicitly enforce structure in the model output.

5.2 Methods

Similar to Chapter 4, in this chapter we are concerned with jointly segmenting and classifying an input video I, represented as a sequence of frames $I_1, \ldots I_T$. However, in this scenario our method produces a generic sequence of labels $y = y_1, \ldots, y_T$ as output. The label y_t could be the current action a_t, the current state of the spatial assembly s_t, or both: the pair (a_t, s_t). We use a hybrid spatiotemporal model with both neural and symbolic components, which we modify to take advantage of the structure of spatial assemblies and actions that modify the assembly. The basic approach of this model is twofold: first, use statistical learning to find a representation of the input video that implicitly captures the structure of the task. Then, use symbolic reasoning to constrain the output of the model by tracking task progress along with actions. This way it will only produce globally-consistent label sequences.

5.2.1 Symbolic representations for assembly processes

A core feature of our model is that it tracks task progress jointly with the action. In Chapter 3, we have established that a natural representation of task progress for assembly scenarios is the state of the current assembly structure s, which can be represented as a kinematic constraint graph [35]. We parameterize a spatial assembly as an edge-labeled graph where nodes represent parts, edges represent part connections, and edge labels specify the type of connection (if any) that is present between every pair of constituent objects. We likewise parameterize an assembly action as a difference between spatial assemblies. We use the same graph structure to represent which edges are changed by the action, but augment it with an additional symbol that represents whether the action is an addition (`connect`) or removal (`disconnect`) of parts. In the special case of a degenerate action (*i.e.* an action that changes nothing in the spatial assembly), we define the action to be `identity`. In what follows we will refer the to set of all realizable spatial assemblies as the assembly vocabulary $\mathcal{S}$, and the set of all realizable assembly actions as the action vocabulary $\mathcal{A}$.

For convenience, in this chapter we represent spatial assemblies as a fully-connected, undirected graph with N vertices (where N represents the number of potential parts in an assembly structure). If two parts are not connected, the corresponding edge is assigned a special background label ϵ. We use C_{ij} to represent the number of possible connections between parts i and j, and define the maximum number of possible connections between any two parts as $C \coloneqq \max_{i,j} C_{ij}$. Since an assembly action consists of a kinematic graph

and an action label, we can represent the kinematic graph component of an assembly action similarly. In this case connections represent changes from one assembly to another, and the background label ϵ means a joint is not changed by the action.

5.2.1.1 Human-centric and task-centric actions

As mentioned in Section 5.1, most computer vision datasets annotate actions at a coarser level than the assembly action representation we have just outlined. These labels usually give a high-level summary of what is happening in the video, rather than exactly identifying the status of the task. For want of better nomenclature, we will distinguish between these paradigms by calling the former 'human-centric' and the latter 'task-centric'. For example, a video segment that is labeled as the human-centric action `screw leg` might be labeled as the task-centric action `connect(leg screw 1, tabletop hole 3)`.

Since our objective in this chapter is to predict the task state from action recognition scores, we need a way to relate human-centric action labels to task-centric ones. Our primary observation is that human-centric actions can be thought of as partial observations of task-centric actions. Returning to the example of the last paragraph, the human-centric label tells us that a connection rather than a disconnection is occurring, and that one of the parts involved is the table leg. However, it gives no information about the other part involved in the connection or where the connection was made (for instance, if the second part is a tabletop it likely has several screw holes to choose from).

5.2.1.2 Part-based embedding of a spatial assembly

In this chapter we additionally introduce a method for constructing a vector embedding of a spatial assembly graph. Compared with the graph-based representation of Chapter 3, this vector representation facilitates visualizing and comparing sequences of spatial assemblies and is more compatible with the output of contemporary neural network frameworks. We base our representation on the observation that because its vertices are consistently ordered, a spatial assembly s can be uniquely identified by a vector of edge labels. We map spatial assemblies to their vector representations as follows: start with a matrix of edge labels $L \in \{0, \ldots, C\}^{N \times N}$, whose elements L_{ij} correspond to the index label of the connection between parts i and j. Since the spatial assembly graph is undirected, this matrix is symmetric. We flatten the upper-triangular portion of this matrix into a discrete-valued vector to obtain our representation $e \in \{0, \ldots, C\}^{\frac{N(N-1)}{2}}$. We embed assembly actions using the same approach, but include an additional dimension that encodes the type of the action (*i.e.* `connect`, `disconnect`, or `identity`)—thus for action embeddings, $e \in \{0, \ldots, C\}^{\frac{N(N-1)}{2}+1}$

For any ordered vocabulary $\mathcal{V}$, it is possible to construct what is called a "one-hot" vector embedding [52]. In this method, each item is embedded in $\{0, 1\}^{|\mathcal{V}|}$ by mapping it to the standard basis vector e_i corresponding to the item's position in the vocabulary. Thus, every entry but one in this vector is zero. Our method uses the spatial assembly's graph structure to reduce the size of the embedding from the number of items in the assembly vocabulary $|\mathcal{S}|$, to the maximum number of edges in the graph $\frac{N(N-1)}{2}$. Thus it scales

only quadratically in the number of parts, where the alternative could scale combinatorially.

5.2.2 Neural action recognition

We use a neural network to compute a task-specific representation of the input video, $z \coloneqq f(x; W) \in \mathbb{R}^{|\mathcal{A}| \times T}$, where $\mathcal{A}$ represents the action vocabulary. Our goal is for this representation to approximate the conditional distribution over assembly sequences: $z_{it} \propto \log \mathrm{P}(a_t = i \mid x)$.

We implement this process in two steps: first, we train a convolutional classifier to predict action labels from short clips extracted in overlapping windows. Then, we train a neural sequence model on the convolutional classifier's output. These models are trained independently due to GPU memory constraints, but the separation is also convenient for purposes of abstraction. The convolutional classifier produces short-term information about the scene, and acts as a low-level pattern recognizer. The role of the sequence model, on the other hand, is to perform mid-level temporal smoothing and/or sensor fusion. The output of this model is then passed to a structured inference layer, which performs high-level symbolic reasoning.

To demonstrate the extent to which our method does not depend on domain-specific modeling, we use the same hyperparameters for all neural models. All I3D models use a ResNet-50 backbone, and we set the size of each LSTM hidden state to be 512 dimensions. Models are trained with the cross-entropy loss and the Adam optimizer. Models are trained on manually labeled action segments, but in validation and test phases we process videos

in overlapping windows with fixed size and stride length.

5.2.3 Symbolic reasoning

Given the neural representation z, we use a segmental CRF to impose structural constraints on the system's output. For our application, we define the latent variable of the CRF to be the joint human-centric action and assembly sequence, $y = y_1, \ldots, y_T$ where $y_t = (s_t, a_t)$. Although the size of the joint action and assembly vocabulary (which we denote as $\mathcal{Y}$) in the worst case is $|\mathcal{S}||\mathcal{A}|$, in practice it is much smaller because many actions and assembly states are incompatible with each other.

5.2.3.1 Scoring

For any CRF, the posterior probability of the latent variable sequence y is computed by normalizing a real-valued score function

$$\log \mathrm{P}(y \mid z) = \mathrm{score}(z, y) - \log \sum_{y} \exp \mathrm{score}(z, y). \tag{5.1}$$

To simplify notation, we rewrite the framewise assembly predictions $y = y_1, \ldots y_T$ as a sequence of M segments with durations $d = d_1, \ldots, d_M$: $y = (a_1, s_1, d_1), \ldots, (a_M, s_M, d_M)$. The overall score of the sequence decomposes into a sum over each segment. We break this term into three parts: a transition term $\mathrm{score}(a_m, s_m, a_{m-1}, s_{m-1})$, a duration term $\mathrm{score}(d_m, a_m, s_m)$, and

an observation term $\sum_{t=b_m}^{b_m+d_m} \text{score}(z_t, a_m, s_m)$. Thus,

$$\text{score}(z, d, l) = \sum_m \text{score}(a_m, s_m, a_{m-1}, s_{m-1}, d_m, z) \tag{5.2}$$

$$= \sum_m \text{score}(a_m, s_m, a_{m-1}, s_{m-1}) + \text{score}(d_m, a_m, s_m) \tag{5.3}$$

$$+ \sum_{t=b_m}^{b_m+d_m} \text{score}(z_t, a_m, s_m).$$

We next focus on defining the individual scoring functions in Equation 5.2, beginning with the data term $\text{score}(z, a, s)$. Even though our primary interest is to operate on action scores as inputs, this framework is capable of scoring both actions and assemblies. We further decompose the data score as an independent combination of the action and assembly scores: $\text{score}(z, a, s) = \text{score}(z, a) + \text{score}(z, s)$. For the action term $\text{score}(z, a)$, we directly use the output of a neural action-recognition model such as I3D. We obtain the spatial assembly term $\text{score}(z, s)$ by introducing the edge-label vector e as a latent variable. In a conventional probabilistic model, we would use a log-likelihood as our data score and marginalize over the individual edge label predictions to obtain the final assembly score

$$\text{score}(z, s) = \log \text{P}(z \mid s) \tag{5.4}$$

$$= \sum_j \log \sum_{e_j} \text{P}(z_j \mid e_j) \, \text{P}(e_j \mid s) \, . \tag{5.5}$$

However, for undirected models like the CRF, we have more freedom in the comparison we choose. In our experiments we use the attribute-based framework we developed in Chapter 4. We compare edge labels predicted from the

observation x with those predicted from the assembly hypothesis s:

$$\text{score}(z, s) = \sum_j \langle \hat{e}_j(z), e_j(s) \rangle. \tag{5.6}$$

The transition prior $\text{score}(a, s, a', s')$ represents the probability that an action a taken when the assembly's state is s will produce state s', and action a' occurs next. Assuming that any action a' is equally likely, $\text{score}(a, s, a', s') \propto \text{score}(a, s, s')$. Our task in this situation is to determine if the human-centric action a can transform the spatial assembly from state s to s'. Recalling the discussion from Section 5.2.1.1, we can compute this by checking if the task-centric action $a_s \coloneqq s' - s$ is in $e(a)$, the set of task-centric actions that are compatible with a:

$$\text{score}(a, s, s') = \begin{cases} 0 & s' - s \in e(a) \\ -\infty & s' - s \notin e(a) \end{cases} \tag{5.7}$$

Thus, to transfer this approach to a new application, the only new information that must be supplied is the mapping $e(\cdot)$. If a dataset has both human-centric and task-centric labels, this mapping could be learned at training time; otherwise it must be specified manually. In our experiments we will address two situations. In the case of our blocks dataset we will recognize task-level actions directly, reducing $e(\cdot)$ to an identity mapping. In the case of the IKEA dataset we define this mapping manually, but in a part-based manner: we combine use each action's object and verb information (published alongside the dataset [3]), with the kinematic information of our assembly structure.

The duration scores for each state are defined similarly: any duration up to the maximum segment duration found in the training set is allowed with score $= 0$, but longer durations receive a penalty of score $= -\infty$.

5.2.3.2 Decoding

In this section we address the problem of decoding (*i.e.* producing jointly-most-probable) outputs from our model. In contrast with our assembly-decoding application of Chapter 4, here there are three scenarios of interest. We first consider the direct case, where the model inputs are joint action and assembly scores, and the desired model output is the *joint* action and assembly sequence. In this case, the output can be obtained through conventional Viterbi-style decoding on the original score function:

$$y^* = \arg\max_{s,a} \text{score}(z, s, a). \tag{5.8}$$

The conventional solution to this problem has runtime complexity $O(|\mathcal{Y}|^2 DT)$ (where D is the maximum allowable duration for any segment) [56].

In our experiments, we are usually interested in slightly different problem formulations than direct joint-decoding. For instance, one of the most useful applications to us is mapping outputs from a neural action recognition model to the best assembly sequence. In this case, model inputs are action scores, and the desired output is assembly labels. In this case the observation and duration models only depend on the current action a_m. We can eliminate the state sequence s by marginalizing over it. (Here we introduce $\psi(\cdot) \coloneqq \exp \text{score}(\cdot)$ to simplify notation)

$$\psi(z, d, s) = \sum_s \prod_m \psi(a_m, s_m, a_{m-1}, s_{m-1}) \psi(d_m, a_m, s_m) \prod_{t=b_m}^{b_m+d_m} \psi(z_t, a_m, s_m) \tag{5.9}$$

$$= \prod_m \sum_{a_m} \prod_{t=b_m}^{b_m+d_m} \psi(z_t, a_m) \psi(d_m, a_m) \psi(a_m, s_m, a_{m-1}, s_{m-1}). \tag{5.10}$$

Likewise, we can marginalize over the assembly sequence s to turn the sequence model into a decoder that only outputs physically-realizable action labels:

$$\psi(z, d, a) = \sum_{s} \prod_{m} \psi(a_m, s_m, a_{m-1}, s_{m-1}) \psi(d_m, a_m, s_m) \prod_{t=b_m}^{b_m+d_m} \psi(z_t, a_m, s_m) \tag{5.11}$$

$$= \prod_{m} \prod_{t=b_m}^{b_m+d_m} \psi(z_t, a_m) \psi(d_m, a_m) \sum_{s_m} \psi(a_m, s_m, a_{m-1}, s_{m-1}) \tag{5.12}$$

In each case, we compute the model output in two steps: first log-marginalize to obtain $\text{score}(z, d, s)$ or $\text{score}(z, d, a)$, then proceed with a conventional segmental-Viterbi decoding algorithm.

5.3 Experiments

In this section we show that neural models form a flexible and high-performing basis for assembly action recognition methods. Then we demonstrate that additionally including our symbolic parser allows these models to reason jointly about assembly actions and their effect on the state of the spatial assembly. We use this mechanism to improve performance on the action recognition task and to predict task progress when only human-centric action recognition labels are learnable.

We evaluate our method on two datasets of assembly videos: one focused on recognizing task progress, and the other focused on recognizing human activity. Our domain-agnostic approach allows us to use the same observation model for both datasets. While we use domain-specific knowledge to tailor the sequence model to each application, our joint action and assembly formulation

allows us to reuse the same model for both action recognition and assembly tracking tasks—the only difference is which variable is selected as output.

5.3.1 Evaluation metrics

We focus on five metrics when evaluating our systems: edit score, framewise accuracy, precision, recall, and F1 score. The edit score represents the number of insertions, deletions, or substitutions required to transform the system's output assembly or action sequence into the ground-truth sequence. It is normalized by the sequence length and subtracted from 1 to produce a number between zero and one, with higher scores representing better performance [43]. The framewise accuracy measures the fraction of frame-level labels that were predicted correctly. The precision and recall measure the system's performance on all labels except the background class (*i.e.* 'no action' or 'no connection'): they measure the proportion of ground-truth labels correctly retrieved by the system and the proportion of the system's labels that match ground truth, respectively. The F1 score is the geometric mean of the precision and recall.

When evaluating performance on the assembly recognition task, we use *edge-level* versions of precision, recall, and F1. Instead of treating each spatial assembly in the output or label sequence as an atomic unit, we use our method from Section 5.2.1.2 to represent the spatial assembly as a vector of edge labels. Our edge-level versions evaluate average precision, recall, and F1 scores over all the *edges* in the assembly sequence, rather than over the spatial assemblies as a whole. As a result, they measure the proportion of edge labels that were retrieved correctly, rather than the proportion of times the model produces an

output that perfectly matches the assembly label. When the model's output is not perfectly correct, these measurements give us an idea of how much overlap there is between the model's output and the ground-truth label.

We trained and evaluated all models using 5-fold cross validation, and report metrics averaged over all videos in the dataset.

5.3.2 Experiment 1: Blocks

In this section we evaluate on an expanded version of our Blocks dataset from Chapter 3. This version is made up of 185 videos of children performing a block-building copy task. As described in Chapter 3, action labels in this dataset represent the addition or removal of part connections—in other words, this dataset is labeled directly with task-centric actions, rather than human-centric ones. By parsing these actions in time, we produce a corresponding set of spatial assembly labels. This annotation scheme results in a challenging label set with a large vocabulary that captures fine-grained physical distinctions: overall there are 424 actions and 458 assemblies.

5.3.2.1 Results: Action recognition

We first performed an ablation experiment investigating the performance of our method in the action recognition task. Table 5.1 shows the results. The baseline I3D model identifies assembly actions fairly well, with an F1 score of nearly 50%. For this dataset, temporal smoothing with a neural sequence model does not improve performance in a meaningful way: the model slightly improves the edit distance at the expense of retrieval metrics. It is likely that instead of learning an improved model, this LSTM learns that the background

action tends to last for a long time. This would be in line with what we expect, given that the Blocks dataset is very long-tailed with an over-represented background class—it is likely that there are simply not enough instances in the dataset to learn a sequence model of anything else. Finally, our symbolic module outperforms the LSTM in every metric, with significant gains over the baseline classifier in edit score and precision. This shows that symbolic reasoning can improve performance in low-resource scenarios.

model	edit	acc.	prec.	rec.	F1
I3D	40.9	61.0	56.6	44.8	48.6
I3D+LSTM	48.4	62.3	40.2	34.5	34.4
I3D+CRF	59.7	63.3	61.6	44.5	50.1

Table 5.1: Action recognition results on the Blocks dataset

5.3.2.2 Baseline assembly recognition model

To compare application-agnostic models based on neural action classifiers with previous work based on task-specific modeling, we first develop a baseline model for joint segmentation and classification of assembly videos. The vision-based methods we used previously in Chapters 3 and 4 were based on template registration, and relied on hand-selected keyframes to function. Due to the high runtime complexity of the method, it cannot scale to the joint segmentation and classification setting: for each frame in the input, the previous method requires solving a nonlinear least-squares objective using an iterative method for every assembly in the vocabulary.

Our approach consists of a similar data-processing procedure to that of

Chapter 3. We first subtract the background by fitting a plane to the depth image. Then, we use Mask-RCNN [23] to remove any image segments that are detected as `person` class with confidence ≥ 0.5. However, rather than rendering-based template registration, our method uses a CNN-based object classifier to predict the label of each edge in the assembly. Thus its runtime scales only quadratically in the number of parts in an assembly, and can be sped up significantly by parallelizing operations on a GPU. Finally, we decode the model's output using the segmental CRF of [43]. This approach marks the first fully-automatic solution to the problem of joint segmentation and classification of spatial assemblies for the blocks dataset.

We used a ResNet-50 model pretrained on Imagenet as the backbone of our vision-based system. Our model outputs a vector $z := f(x; W) \in \mathbb{R}^{|\mathcal{C}|\times|\mathcal{V}|\times T}$. By training with a cross-entropy loss, we hope to produce a final output $z_{ijt} \propto \log \mathrm{P}(e_{it} = j \mid x)$, where e_{it} is the i-th element of the edge attribute vector e_t associated with the spatial assembly s_t.

We started by pre-training this model on synthetically-rendered images of all the spatial assemblies in our vocabulary. All rendered images are 128×128 pixels in size. When pre-training, we randomize the position and orientation of the spatial assembly and randomly apply occlusion masks. Next, we train this model on the videos in the blocks dataset. After removing the background and any occluding hands or other distracting objects from the frame, we compute proposal bounding boxes from the resulting image by taking a 128×128-pixel crop around the center of every contiguous segment. The final output of the

model is a sum over its output for each proposal segment x_p:

$$\text{score}(x, \cdot) = \sum_p f(x_p; W). \tag{5.13}$$

When decoding the final output of our model, we define our transition scores to be binary-valued: if two states differ by the addition or removal of one block, then a transition is allowed between them. If they differ by the addition or removal of more than one block, a transition is not allowed.

5.3.2.3 Results: Assembly recognition

We first perform an ablation study investigating the behaviour of our baseline model. Table 5.2 shows the results. The ResNet model alone achieves moderately high precision, though its recall is significantly lower. This behavior is expected, because in most frames the entire structure is not visible due either to partial occlusion or preprocessing errors. However, these missed detections are compensated for by the mid-level LSTM. This model significantly improves recall, leading to better performance across the board. However, the edit score is still quite low. By imposing some simple segment transition and duration rules, the segmental CRF dramatically improves this edit score at no significant cost to overall accuracy or edge retrieval.

Although the focus of this chapter is on vision, for the sake of comparison we additionally include the IMU-attribute features defined in Chapter 4 as input to our baseline model. We fuse the outputs of the IMU and video streams at the edge level by concatenating them and using them as input to the LSTM sequence model. Including these features leads to significant performance improvements: overall, it decreases most measurements of system

error by half.

Finally, in the last row of Table 5.2 we compare the performance of our application-agnostic action recognition model to the vision-only baseline. Despite incorporating far fewer domain-specific modeling decisions, our action recognition system performs on par: it has slightly higher edit and accuracy scores, but slightly lower precision. This likely indicates that our method still predicts the background class slightly more frequently compared to the baseline.

model	edit	acc.	prec.	rec.	F1
RESNET	3.5	35.7	78.0	49.1	58.7
+LSTM	15.4	50.1	83.1	82.1	81.4
+CRF	53.2	52.9	83.8	81.9	81.6
+IMU	64.5	73.5	90.4	93.0	91.9
I3D+CRF	55.5	55.7	79.9	81.3	79.8

Table 5.2: Assembly recognition results on the Blocks dataset

5.4 Experiment 2: IKEA dataset

We also evaluate our methods on the IKEA-ASM public benchmark dataset [3]. This dataset comes with its own unique characteristics and challenges: it consists of 372 videos with 33 unique action types. In each video, the builder assembles one of four furniture items: a TV bench, a coffee table, a side table, or a drawer. In contrast to the blocks videos, this dataset was labeled with human-centric actions rather than task-centric ones. We use this dataset to demonstrate the use of our method in predicting task progress from

human-centric actions. For the purpose of evaluation, we additionally labeled a subset of 14 videos (3.8% of the dataset) with ground-truth assembly actions. These videos were chosen uniformly at random from each furniture item in the dataset.

Since this dataset has no task-level labels, we need to construct assembly and action vocabularies. We define the assembly vocabulary as the set of all valid sub-graphs of the four furniture pieces. We define the action vocabulary as the set of all valid differences between spatial assemblies: $\mathcal{A} \coloneqq \{s' - s \; \forall s, s' \in \mathcal{S} \otimes \mathcal{S} : s \subseteq s'\}$. Finally, we define a part-based mapping from human-centric actions to task-centric actions by using information provided in the original dataset publication. This table provides the verb and object for involved every action. We map every verb to an assembly sign (*i.e.* `connect`, `disconnect`, or `no action`), then use this along with part information to construct $e(a)$, the set of task-centric actions that are compatible with the human-centric action a.

5.4.1 Results: Action recognition

We report the results of our action recognition experiment in Table 5.6. Our baseline I3D model gives moderate performance, with edit score and accuracy both around 45%. Smoothing predictions with an LSTM improves both metrics significantly—it gives a 45% and 57% relative relative reduction in error rate, respectively. However, postprocessing with our symbolic reasoning module does not have any meaningful effect on the model's performance. This is counter to the results shown on the Blocks task in the previous section,

and illustrates the effect of dataset conditions on the relative contributions of these models: while the blocks dataset consists of hundreds of unique actions that are directly related to the structure of the task, this IKEA dataset contains only a few dozen unique actions, and only a handful of those are relevant to task progress. This results in two main practical differences: first, the overall sequence model for this dataset will be much less sparse than that of the Blocks dataset, resulting in a model that rules out far fewer hypothesis sequences. Second, the smaller size of the action vocabulary will result in an easier task for the neural sequence model to learn—this could explain the difference in performance between the two LSTM modules.

model	edit	acc.	prec.	rec.	F1
I3D	44.3	45.6	48.5	90.3	61.0
+LSTM	64.4	71.7	73.0	96.9	82.1
+CRF	64.8	71.8	73.1	96.9	82.1

Table 5.3: Action recognition on the IKEA-ASM dataset

We additionally examine performance by furniture type in Table 5.4. We point out here that the system's performance on the shelf drawer is significantly lower than the rest of the furniture items. This is likely because the TV bench, coffee table, and side table all share a significant proportion of their sub-structure: all three are comprised of a tabletop and four legs, possibly with other parts. Thus, the assembly actions for the shelf drawer are underrepresented in training dataset.

model	edit	acc.	prec.	rec.	F1
Lack TV Bench	65.8	76.5	78.0	96.3	85.9
Lack Coffee Table	68.0	76.3	77.5	96.5	88.6
Lack Side Table	69.8	80.5	82.8	95.4	88.2
Kallax Shelf Drawer	53.9	53.9	53.9	100	69.1

Table 5.4: Action recognition results per furniture item

5.4.2 Results: Zero-shot assembly recognition

In these experiments we show that even when our sequence model does not significantly improve performance in human-centric action recognition, it can still be used to construct a high-performing assembly recognition system—even when there are no assembly labels to train on.

In the zero-shot setup, we train the neural models of the previous section to recognize human-centric actions on 80% of the videos in the dataset. The test set consists of the 14 videos that were labeled with assembly-level actions—a subset of the remaining 20% of heldout videos. At test time, we use the neural observation models along with our symbolic sequence model to produce assembly-level predictions. We evaluate by comparing these predictions to the ground-truth assembly labels.

We report our results in Table 5.5. Overall, our system performs extremely well. Even though the action recognition model's performance is imperfect (with accuracy less than 72%), on this task our method recognizes assemblies with nearly 80% frame-level accuracy and achieves 100% edit score—this means it perfectly identifies the assembly sequence every time, and the only source of error is the placement of segment boundaries.

model	edit	acc.	prec.	rec.	F1
I3D+LSTM+CRF	100	78.3	90.9	97.6	93.7

Table 5.5: Assembly recognition results on the IKEA-ASM dataset

model	edit	acc.	prec.	rec.	F1
Lack TV Bench	100	67.3	80.5	98.4	87.9
Lack Coffee Table	100	78.6	90.8	99.7	94.5
Lack Side Table	100	89.3	97.7	97.2	97.4
Kallax Shelf Drawer	100	70.5	90.1	95.3	92.5

Table 5.6: Assembly recognition results per furniture item

5.5 Conclusion

In this chapter we have described a general-purpose method that combines ideas from deep learning and symbolic reasoning to jointly segment and classify assembly videos. By jointly modeling assembly actions along with the task state, we can perform action recognition or track task progress with the same sequence model. We evaluate our approach on two assembly-oriented video datasets: one based on toy block play, and the other based on the assembly of IKEA furniture. On the first, we show our symbolic method outperforms an LSTM-based sequence model in the action recognition task, and gives comparable performance to a handcrafted baseline model in tracking task progress while requiring far less application-specific modeling. On the second, we give the first known results for joint segmentation and classification of assembly actions and demonstrate strong performance (perfect edit distance) in a zero-shot task progress tracking experiment. We hope this work will inspire further

research in neuro-symbolic action recognition and task-oriented video understanding.

Chapter 6

Conclusion

In this dissertation we have developed the task of assembly understanding by applying concepts from computer vision, robotics, and sequence modeling. Motivated by the need to develop tools for recording and analyzing experimental data for a collaborative study of human spatial cognition, we gradually extended an application-specific model into a framework that is broadly applicable across data modalities and application instances. The core of our approach is a sequence model that relates assembly actions to their structural effects. In each main chapter of this dissertation we combine this sequence model with increasingly-general observation models. With each iteration we increase the variety of applications that can be considered by our framework, and decrease the complexity of modeling decisions that system designers are required to make.

In Chapter 3 we presented an initial solution for modeling and recognizing assembly activities in our primary application: videos of children performing a block-assembly task. First, we developed a symbolic model that completely characterized the fine-grained temporal and geometric structure of assembly

sequences: it can be applied to any kinematic structure and distinguishes even such minute differences as reflection or shift errors in the placement of a part. Then, we combined this sequence model with a probabilistic observation model that operates by rendering and registering template images of each assembly hypothesis. Given a coarse 3D model of each block and a sequence of manually-specified keyframes, we were able to identify the spatial assembly sequence performed in each video with 62% accuracy.

In Chapter 4 we extended the perception system of Chapter 3 by incorporating sensor-based observations. The primary contribution of this chapter is a part-based observation model that compares mid-level attributes derived from sensor streams with their corresponding predictions from assembly hypotheses. We additionally addressed the joint segmentation and classification of assembly sequences for the first time in this chapter, resulting in a feature-based segmental CRF framework. In our experiments we demonstrated an edit score of nearly 95% in jointly segmenting and classifying assembly sequences for a new application: egocentric videos of IKEA furniture assembly with fiducial markers. Then, we showed that incorporating attributes derived from IMU sensors into the system from Chapter 3 reduced the error rate by more than 40% relative to our previous, video-only implementation.

Finally, in Chapter 5 we focused on *learning* observation models rather than constructing them by hand. To achieve this we incorporated contemporary, vision-based action recognition models into the segmental CRF of Chapter 4. Under this approach, the only information required from a user is a mapping from human-centric activities to the task-centric activities we defined in

Chapter 3. Experimentally, we established the first joint segmentation and classification results for our primary application of block-assembly videos. We showed that a baseline neuro-symbolic system based on our original implementation in Chapter 3 achieved an edit score of 53%, and that including the sensor-based attributes of Chapter 4 additionally improved performance to 65%. Compared to the video-only baseline, our action-recognition system achieved comparable performance while requiring fewer modeling decisions. We additionally showed that the *same model* can be used to recognize actions rather than spatial assemblies, giving a 23% improvement in edit score relative to an LSTM-based sequence model. We then proceeded to apply the same framework to a second, larger dataset of IKEA furniture assembly. Here we showed that using our model, spatial assemblies can be recognized even when only human-centric action labels are available at training time: in a zero-shot assembly recognition experiment, we identified spatial assembly sequences with 100% edit score.

This work has culminated in a flexible model that blends contemporary approaches in deep learning with the classical theories of kinematics and probabilistic sequence modeling. Going forward, there are many opportunities for applying and improving this methodology. For instance, there is still an open question regarding how previously-unseen spatial assemblies may be reliably detected. Our part-based assembly model represents a step in this direction, but preliminary experiments have shown that conventional neural classifiers do not learn this structure during training. There are also many alternative neurosymbolic methods to consider: perhaps jointly training the LSTM and

CRF sequence models could result in better performance, or an LSTM with the Connectionist Temporal Classification loss [14] could give similar performance while reducing the complexity of the symbolic model. Finally, this framework could be implemented in robotic perception systems for more robust autonomous construction, or improved human-robot collaborative assembly.

Bibliography

[1] R. Achanta et al. "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.11 (2012), pp. 2274–2282. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2012.120.

[2] Narges Ahmidi et al. "A Dataset and Benchmarks for Segmentation and Recognition of Gestures in Robotic Surgery". In: *IEEE Transactions on Biomedical Engineering* 64.9 (2017), pp. 2025–2041. DOI: 10.1109/TBME.2016.2647680.

[3] Yizhak Ben-Shabat et al. "The IKEA ASM Dataset: Understanding People Assembling Furniture Through Actions, Objects and Pose". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2021, pp. 847–859.

[4] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. II*. 3rd. Athena Scientific, 2007. ISBN: 1886529302, 9781886529304.

[5] M. Branch, T. Coleman, and Y. Li. "A Subspace, Interior, and Conjugate Gradient Method for Large-Scale Bound-Constrained Minimization Problems". In: *SIAM Journal on Scientific Computing* 21.1 (1999), pp. 1–23. DOI: 10.1137/S1064827595289108. eprint: https://doi.org/10.1137/S1064827595289108. URL: https://doi.org/10.1137/S1064827595289108.

[6] Joao Carreira and Andrew Zisserman. "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[7] Cathryn S Cortesa et al. "Characterizing spatial construction processes: Toward computational tools to understand cognition". In: *Annual Meeting of the Cognitive Science Society*. 2017, pp. 246–251.

[8] Cathryn S Cortesa et al. “Characterizing spatial construction processes: Toward computational tools to understand cognition”. In: *Annual Meeting of the Cognitive Science Society*. 2017, pp. 246–251.

[9] Cathryn S Cortesa et al. “Constraints and Development in Children’s Block Construction”. In: *Annual Meeting of the Cognitive Science Society*. 2017, pp. 246–251.

[10] João Paulo Costeira and Takeo Kanade. “A Multibody Factorization Method for Independently Moving Objects”. In: *International Journal of Computer Vision* 29.3 (1998), pp. 159–179. ISSN: 1573-1405. DOI: 10.1023/A:1008000628999. URL: https://doi.org/10.1023/A:1008000628999.

[11] Dima Damen and David Hogg. “Attribute Multiset Grammars for Global Explanations of Activities”. In: 2009. DOI: 10.5244/C.23.123.

[12] Christoph Feichtenhofer et al. “SlowFast Networks for Video Recognition”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 6201–6210. DOI: 10.1109/ICCV.2019.00630.

[13] Raghav Goyal et al. “The ”Something Something” Video Database for Learning and Evaluating Visual Common Sense”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017.

[14] Alex Graves et al. “Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks”. In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML ’06. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006, 369–376. ISBN: 1595933832. DOI: 10.1145/1143844.1143891. URL: https://doi.org/10.1145/1143844.1143891.

[15] A. Guha et al. “Minimalist plans for interpreting manipulation actions”. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013, pp. 5908–5914. DOI: 10.1109/IROS.2013.6697213.

[16] Ankit Gupta et al. “DuploTrack: A Real-time System for Authoring and Guiding Duplo Block Assembly”. In: *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*. UIST ’12. Cambridge, Massachusetts, USA: ACM, 2012, pp. 389–402. ISBN: 978-1-4503-1580-7. DOI: 10.1145/2380116.2380167. URL: http://doi.acm.org/10.1145/2380116.2380167.

[17] J. Hadfield et al. “Object Assembly Guidance in Child-Robot Interaction using RGB-D based 3D Tracking”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 347–354.

[18] Gregory D. Hager and Ben Wegbreit. “Scene parsing using a prior world model”. In: *International Journal of Robotics Research* 30.12 (2011), pp. 1477–1507. ISSN: 02783649. DOI: 10.1177/0278364911399340.

[19] Gregory D. Hager and Ben Wegbreit. “Scene parsing using a prior world model”. In: *International Journal of Robotics Research* 30.12 (2011), pp. 1477–1507. ISSN: 02783649. DOI: 10.1177/0278364911399340.

[20] Richard Hartley et al. “Rotation Averaging”. In: *International Journal of Computer Vision* 103.3 (2013), pp. 267–305. ISSN: 1573-1405. DOI: 10.1007/s11263-012-0601-0. URL: https://doi.org/10.1007/s11263-012-0601-0.

[21] B. Hayes and B. Scassellati. “Autonomously constructing hierarchical task networks for planning and human-robot collaboration”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 5469–5476.

[22] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[23] Kaiming He et al. “Mask R-CNN”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322.

[24] L. S. Homem de Mello and A. C. Sanderson. “AND/OR graph representation of assembly plans”. In: *IEEE Transactions on Robotics and Automation* 6.2 (1990), pp. 188–199. ISSN: 2374-958X. DOI: 10.1109/70.54734.

[25] A. Hundt et al. “The CoSTAR Block Stacking Dataset: Learning with Workspace Constraints”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 1797–1804.

[26] Y.A. Ivanov and A.F. Bobick. “Recognition of visual activities and interactions by stochastic parsing”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.8 (2000), pp. 852–872. DOI: 10.1109/34.868686.

[27] E. Jahangiri et al. “Information Pursuit: A Bayesian Framework for Sequential Scene Parsing”. In: *ArXiv e-prints* (2017). arXiv: 1701.02343 [cs.CV].

[28] Frederick Jelinek. *Statistical Methods for Speech Recognition*. Cambridge, MA, USA: MIT Press, 1997. ISBN: 0-262-10066-5.

[29] Jingyu Yan and M. Pollefeys. “Automatic Kinematic Chain Building from Feature Trajectories of Articulated Objects”. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 1. 2006, pp. 712–719. DOI: 10.1109/CVPR.2006.66.

[30] Justin Johnson et al. “Image Retrieval using Scene Graphs”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3668–3678.

[31] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. 2001–. URL: http://www.scipy.org/.

[32] J. Jones, G. D. Hager, and S. Khudanpur. “Toward Computer Vision Systems That Understand Real-World Assembly Processes”. In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2019, pp. 426–434. DOI: 10.1109/WACV.2019.00051.

[33] J. Jones, G. D. Hager, and S. Khudanpur. “Toward Computer Vision Systems That Understand Real-World Assembly Processes”. In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2019, pp. 426–434. DOI: 10.1109/WACV.2019.00051.

[34] J. D. Jones et al. “Fine-Grained Activity Recognition for Assembly Videos”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3728–3735. DOI: 10.1109/LRA.2021.3064149.

[35] Jonathan D. Jones et al. “Fine-Grained Activity Recognition for Assembly Videos”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3728–3735. DOI: 10.1109/LRA.2021.3064149.

[36] Will Kay et al. “The kinetics human action video dataset”. In: *arXiv preprint arXiv:1705.06950* (2017).

[37] Tae Soo Kim et al. “DASZL: Dynamic Action Signatures for Zero-shot Learning”. In: *AAAI* (2021).

[38] Ross A. Knepper et al. “Distributed Assembly with AND / OR Graphs”. In: *Proceedings of the Workshop on AI Robotics at the International Conference on Intelligent Robots and Systems*. 2014.

[39] Hema Swetha Koppula, Rudhir Gupta, and Ashutosh Saxena. "Learning Human Activities and Object Affordances from RGB-D Videos". In: *Int. J. Rob. Res.* 32.8 (2013), pp. 951–970. ISSN: 0278-3649. DOI: 10.1177/0278364913478446. URL: http://dx.doi.org/10.1177/0278364913478446.

[40] Frank R. Kschischang, Brendan J. Frey, and Hans Andrea Loeliger. "Factor graphs and the sum-product algorithm". In: *IEEE Transactions on Information Theory* 47.2 (2001), pp. 498–519. ISSN: 00189448. DOI: 10.1109/18.910572. arXiv: arXiv:1011.1669v3.

[41] H. Kuehne et al. "HMDB: a large video database for human motion recognition". In: *Proceedings of the International Conference on Computer Vision (ICCV)*. 2011.

[42] C. Lea, R. Vidal, and G. D. Hager. "Learning convolutional action primitives for fine-grained action recognition". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1642–1649.

[43] Colin Lea et al. "Segmental Spatiotemporal CNNs for Fine-Grained Action Segmentation". In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, 2016, pp. 36–52. ISBN: 978-3-319-46487-9.

[44] Colin Lea et al. "Segmental Spatiotemporal CNNs for Fine-Grained Action Segmentation". In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, 2016, pp. 36–52. ISBN: 978-3-319-46487-9.

[45] Colin Lea et al. "Temporal Convolutional Networks: A Unified Approach to Action Segmentation". In: *Computer Vision – ECCV 2016 Workshops*. Ed. by Gang Hua and Hervé Jégou. Cham: Springer International Publishing, 2016, pp. 47–54. ISBN: 978-3-319-49409-8.

[46] Colin Stuart Lea et al. "Multi-Modal Models for Fine-grained Action Segmentation in Situated Environments". PhD thesis. Johns Hopkins University, 2017.

[47] Ji Lin, Chuang Gan, and Song Han. "TSM: Temporal Shift Module for Efficient Video Understanding". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019.

[48] Jingen Liu, B. Kuipers, and S. Savarese. "Recognizing Human Actions by Attributes". In: *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR '11. USA: IEEE Computer Society, 2011, 3337–3344. ISBN: 9781457703942. DOI: 10.1109/CVPR.2011.5995353. URL: https://doi.org/10.1109/CVPR.2011.5995353.

[49] Roberto Martín-Martín and Oliver Brock. "Coupled recursive estimation for online interactive perception of articulated objects". In: *The International Journal of Robotics Research* 0.0 (0), p. 0278364919848850. DOI: 10.1177/0278364919848850. eprint: https://doi.org/10.1177/0278364919848850. URL: https://doi.org/10.1177/0278364919848850.

[50] Anahita Mohseni-Kabir et al. "Interactive Hierarchical Task Learning from a Single Demonstration". In: *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*. HRI '15. Portland, Oregon, USA: Association for Computing Machinery, 2015, 205–212. ISBN: 9781450328838. DOI: 10.1145/2696454.2696474. URL: https://doi.org/10.1145/2696454.2696474.

[51] James Munkres. "ALGORITHMS FOR THE ASSIGNMENT AND TRANSPORTATION PROBLEMS". In: *Journal of the Society for Industrial & Applied Mathematics* 5.1 (1957). DOI: 10.1137/0105003.

[52] Kevin P Murphy. *Machine learning: a probabilistic perspective*. Cambridge, MA: MIT Press, 2012.

[53] S. Niekum et al. "Online Bayesian changepoint detection for articulated motion models". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 1468–1475. DOI: 10.1109/ICRA.2015.7139383.

[54] Marcus Rohrbach et al. "Recognizing Fine-Grained and Composite Activities Using Hand-Centric Features and Script Data". In: *International Journal of Computer Vision* (2015), pp. 1–28. ISSN: 0920-5691. DOI: 10.1007/s11263-015-0851-8. URL: http://dx.doi.org/10.1007/s11263-015-0851-8.

[55] David A. Ross, Daniel Tarlow, and Richard S. Zemel. "Unsupervised Learning of Skeletons from Motion". In: *Computer Vision – ECCV 2008*. Ed. by David Forsyth, Philip Torr, and Andrew Zisserman. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 560–573. ISBN: 978-3-540-88690-7.

[56] Sunita Sarawagi and William W Cohen. "Semi-Markov Conditional Random Fields for Information Extraction". In: *Advances in Neural Information Processing Systems 17*. Ed. by L. K. Saul, Y. Weiss, and L. Bottou. MIT Press, 2005, pp. 1185–1192. URL: http://papers.nips.cc/paper/2648-semi-markov-conditional-random-fields-for-information-extraction.pdf.

[57] D. Sculley. "Web-scale K-means Clustering". In: *Proceedings of the 19th International Conference on World Wide Web*. WWW '10. Raleigh, North Carolina, USA: ACM, 2010, pp. 1177–1178. ISBN: 978-1-60558-799-8. DOI: 10.1145/1772690.1772862. URL: http://doi.acm.org/10.1145/1772690.1772862.

[58] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. "UCF101: A dataset of 101 human actions classes from videos in the wild". In: *arXiv preprint arXiv:1212.0402* (2012).

[59] J. Sturm et al. "Operating articulated objects based on experience". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010, pp. 2739–2744. DOI: 10.1109/IROS.2010.5653813.

[60] J. Sturm et al. "Vision-based detection for learning articulation models of cabinet doors and drawers in household environments". In: *2010 IEEE International Conference on Robotics and Automation*. 2010, pp. 362–368. DOI: 10.1109/ROBOT.2010.5509985.

[61] Jürgen Sturm. "Learning Kinematic Models of Articulated Objects". In: *Approaches to Probabilistic Model Learning for Mobile Manipulation Robots*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 65–111. ISBN: 978-3-642-37160-8. DOI: 10.1007/978-3-642-37160-8_4. URL: https://doi.org/10.1007/978-3-642-37160-8_4.

[62] Jürgen Sturm, Cyrill Stachniss, and Wolfram Burgard. "A Probabilistic Framework for Learning Kinematic Models of Articulated Objects". In: *J. Artif. Intell. Res. (JAIR)* 41 (2011), pp. 477–526. DOI: 10.1613/jair.3229.

[63] Jürgen Sturm et al. "Learning Kinematic Models for Articulated Objects". In: *Proceedings of the 21st International Jont Conference on Artifical Intelligence*. IJCAI'09. Pasadena, California, USA: Morgan Kaufmann Publishers Inc., 2009, pp. 1851–1856. URL: http://dl.acm.org/citation.cfm?id=1661445.1661742.

[64] D. Summers-Stay et al. “Using a minimal action grammar for activity understanding in the real world”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 4104–4111. DOI: 10.1109/IROS.2012.6385483.

[65] Charles Sutton and Andrew McCallum. “An Introduction to Conditional Random Fields”. In: *Foundations and TrendsÂ® in Machine Learning* 4.4 (2012), pp. 267–373. ISSN: 1935-8237. DOI: 10.1561/2200000013. URL: http://dx.doi.org/10.1561/2200000013.

[66] Nam N. Vo and Aaron F. Bobick. “Sequential Interval Network for parsing complex structured activity”. In: *Computer Vision and Image Understanding* 143 (2016), pp. 147 –158. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2015.07.006. URL: http://www.sciencedirect.com/science/article/pii/S1077314215001599.

[67] Stéfan van der Walt et al. “scikit-image: image processing in Python”. In: *PeerJ* 2 (2014), e453. ISSN: 2167-8359. DOI: 10.7717/peerj.453. URL: http://dx.doi.org/10.7717/peerj.453.

[68] Yeping Wang, Gopika Ajaykumar, and Chien-Ming Huang. “See What I See: Enabling User-Centric Robotic Assistance Using First-Person Demonstrations”. In: *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*. HRI ’20. Cambridge, United Kingdom: Association for Computing Machinery, 2020, 639–648. ISBN: 9781450367462. DOI: 10.1145/3319502.3374820. URL: https://doi.org/10.1145/3319502.3374820.

[69] Jiajun Wu, Joshua B Tenenbaum, and Pushmeet Kohli. “Neural Scene De-rendering”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[70] Danfei Xu et al. “Scene graph generation by iterative message passing”. In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. 2017, pp. 3097–3106. ISBN: 9781538604571. DOI: 10.1109/CVPR.2017.330. arXiv: 1701.02426.

[71] Y. Yang et al. “Manipulation action tree bank: A knowledge resource for humanoids”. In: *2014 IEEE-RAS International Conference on Humanoid Robots*. 2014, pp. 987–992. DOI: 10.1109/HUMANOIDS.2014.7041483.

[72] Yezhou Yang et al. “Learning the Semantics of Manipulation Action”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, 2015, pp. 676–686. DOI: 10.3115/v1/P15-1066. URL: https://www.aclweb.org/anthology/P15-1066.