Toward computer vision systems that understand real-world assembly processes

Jonathan D. Jones Gregory D. Hager Sanjeev Khudanpur
Johns Hopkins University
jdjones@jhu.edu, hager@cs.jhu.edu, khudanpur@jhu.edu

Abstract

Many applications of computer vision require robust systems that can parse complex structures as they evolve in time. Using a block construction task as a case study, we illustrate the main components involved in building such systems. We evaluate performance at three increasingly-detailed levels of spatial granularity on two multimodal (RGBD + IMU) datasets. On the first, designed to match the assumptions of the model, we report better than 90% accuracy at the finest level of granularity. On the second, designed to test the robustness of our model under adverse, real-world conditions, we report 67% accuracy and 91% precision at the mid-level of granularity. We show that this seemingly simple process presents many opportunities to expand the frontiers of computer vision and action recognition.

1. Introduction

Many applications of computer vision require systems to be deployed in relatively uncontrolled environments that evolve in time. As an example, consider the perception system guiding a collaborative robot working on a manufacturing task with a human partner. As parts are being assembled, this system must do more than recognize the actions its partner has taken—it must also understand and reason about the way these actions will change the part.

In assembly tasks such as the one just described, the manufactured part is a complex, dynamic object. The agents assemble this part by combining smaller entities (or subparts) in specific ways. For example: as part of assembling a table, one must select the correct leg, orient it properly with respect to the top, and then insert an appropriate screw to fasten the two together.

Understanding these processes is a challenging problem for computer vision, and can be better understood by examining inferences at three levels of precision. At the coarsest level, a system must identify which entities have been integrated into the assembly. Furthermore, it must identify which of the entities in the assembly are connected to each other. Finally, it must identify the precise manner of each connection.

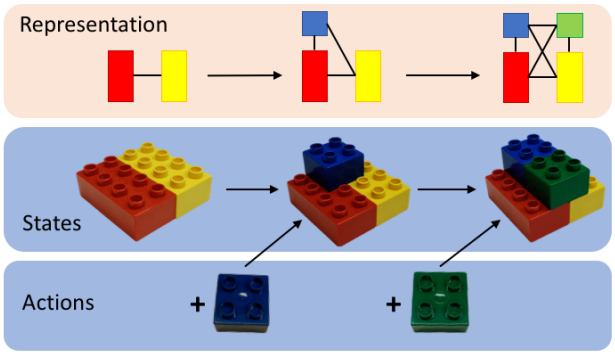


Figure 1: Partial example of an assembly process. The agent alters the state by adding or removing blocks in specific ways.

Much of the recent progress in activity recognition has focused on improving performance on a variety of benchmark datasets whose labels consist of coarse-grained video descriptions (*e.g.*, recognizing that an activity is furniture building instead of weightlifting) [16, 19, 2, 13, 6]. Since these methods focus on inferring from a relatively small, closed set of high-level labels, they are not well-suited to capturing the fine-grained details that are important in an assembly process. In addition, data acquisition in real-world environments is never as carefully controlled as the processes by which benchmark datasets are constructed. It is common for the entities or actions of interest to be partially, or even completely, occluded consistently throughout a procedure. The fine-grained complexity, combinatorial nature, and unpredictable observations in this task require different approaches from the current paradigm of computer vision research.

In this paper, 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 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.

The remainder of this paper is structured as follows. In Section 2, we review related work. We define a representation of spatial assemblies and the actions that modify them in Section 3. In Section 4 we derive probabilistic models and inference algorithms for the assembly process. We present the block building task as an instance of spatial assembly parsing in Section 5. We evaluate our approach empirically and conclude, respectively, in Sections 6 and 7.

2. Related Work

Action recognition from procedural data: Several tasks from the fine-grained action recognition literature originate from procedural data—that is, processes in which an agent is trying to change the state of some entity to achieve a particular outcome. For instance, people try to make a salad in[20] or build a toy airplane in [22]. These procedural datasets have so far been used as testbeds for segmenting and classifying *actions* ("peeling a cucumber," or "added propeller"). However, a recent paper in the natural language processing literature learns from procedural text data, specifically recipes, to predict *state changes* induced by actions [4]. We take the latter approach and parse state changes over time, although action recognition is implicit in our model.

Fine-grained inference in computer vision: As performance on traditional classification tasks has begun to saturate in both computer vision and action recognition, recent years have seen a trend toward estimating more complex, structured variables.

Even within these structured problems, the degree of precision varies. As an example from action recognition, [14] models 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 [11, 24]. 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 in [8] estimates a graph whose vertices represent objects and whose edges represent support relationships between objects. Similarly, [9] and [23] estimate structured, physical representations from images, although these representations are not posed as graphs.

Our application adds a level of detail beyond these traditional adjacency relationships: in addition to describing *which* two objects are connected, we also describe precisely *how* they are connected by specifying the exact 3D pose (rotation and translation) between the two objects.

3. Representation of Assembly Processes

We first define data structures representing the salient entities and actions, as well as a model which describes the way entities are changed by actions.

3.1. State representation

We represent the state of a spatial assembly as a graph whose vertices correspond to entities (*i.e.* its component parts). If two entities are physically adjacent to each other, the vertices representing them are joined by a directed edge. We denote the graph's vertex set as $\mathcal{V}$, and its edge set as $\mathcal{E}$. Each edge e is associated with a label $l_e \in SE(3)$, which defines an object's relative pose[1] in the coordinate frame of its neighbor. Figure 1 visually illustrates such a graph for a DUPLO block building task, albeit with labels omitted from the edges to simplify illustration.

During the course of assembly it is possible to combine objects into multiple, disjoint sub-parts. In terms of our graph representation, each sub-part corresponds to a separate connected component in the graph. We denote the set of connected components as $\mathcal{C}$.

3.2. Action representation

Over the course of an assembly process, an agent can take a variety of actions to change the state of the assembly. Each action must specify *what* entities in the state will change, and *how* they will change. We use $\mathcal{A}$ to represent the set of possible actions.

In this paper, $\mathcal{A}$ is restricted to one type of constructive action, and one type of deconstructive action:

- $connect$ takes a state, two objects, and a relative pose as arguments. It returns a state graph with an edge added between the two objects.

- $disconnect$ takes a state and two (connected) objects as arguments. It returns a state graph with an edge removed between the two objects.

Figure 1 illustrates the effect of two consecutive $connect$ actions.

4. A Probabilistic Graphical Model

In this section we provide a probabilistic model of an assembly process, from which we derive an algorithm for estimating an assembly process from video.

[1] $SE(3)$ refers to the special Euclidean group on $\mathbb{R}^3$, a rotation and translation in three-dimensional space.

4.1. Model definition

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 [3]. 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.

4.1.1 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 $p_i \in SE(3)$ of each sub-part (*i.e.* each connected component in the state graph). 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 $s = (\mathcal{V}, \mathcal{E})$ in pose p, as

$$P(I_t, p_t, s_t) = P(I_t|p_t, s_t)P(p_t|s_t)P(s_t). \tag{1}$$

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 $\mathcal{X}$ represent the pixel coordinates of I_t,

$$P(I_t|p_t, s_t) = \prod_{x \in \mathcal{X}} \mathcal{N}(I_t(x); T(x; p_t), \sigma^2). \tag{2}$$

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 by the camera.

4.1.2 Process model

In an assembly process, the state s_t is constructed or deconstructed through the actions 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:

$$P(s_t|s_{t-1}) = \sum_{a_{t-1} \in \mathcal{A}} P(s_t|s_{t-1}, a_{t-1})P(a_{t-1}|s_{t-1}) \tag{3}$$

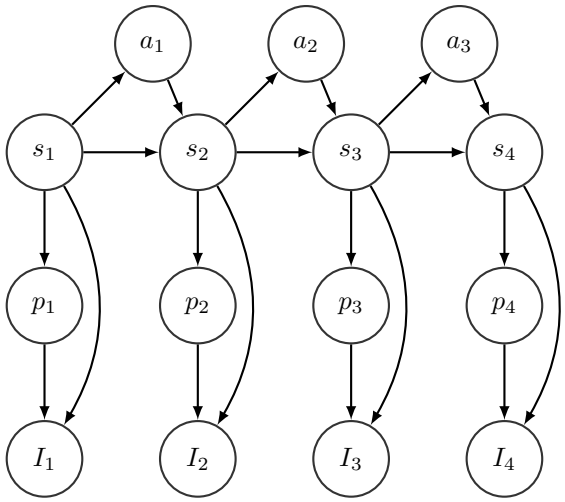


Figure 2: Graphical model corresponding to equation (4)

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 4.1.1, we obtain a final expression for the joint probability of all observed and inferred variables:

$$P(I_{1:T}, p_{1:T}, s_{1:T}) = \prod_{t=1}^{T} P(I_t|p_t, s_t)P(p_t|s_t)P(s_t|s_{t-1}) \tag{4}$$

This is represented by the graphical model in Figure 2.

4.2. Inference

Given $I_{1:T}$, we estimate the state sequence using a Viterbi-style decoding algorithm on the graphical model in Figure 2. That is, we solve the problem

$$p^*_{1:T}, s^*_{1:T} = \underset{p_{1:T}, s_{1:T}}{\arg\max}\, P(I_{1:T}, p_{1:T}, s_{1:T}) \tag{5}$$

using max-sum message passing [15]. The solution of this problem can be explained as a hypothesize-and-test approach. For 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.

4.2.1 Hypothesis generation

The time-series structure in our problem allows us to efficiently select and decode hypotheses using the Viterbi algorithm with beam search [10]. At each time step t, we prune

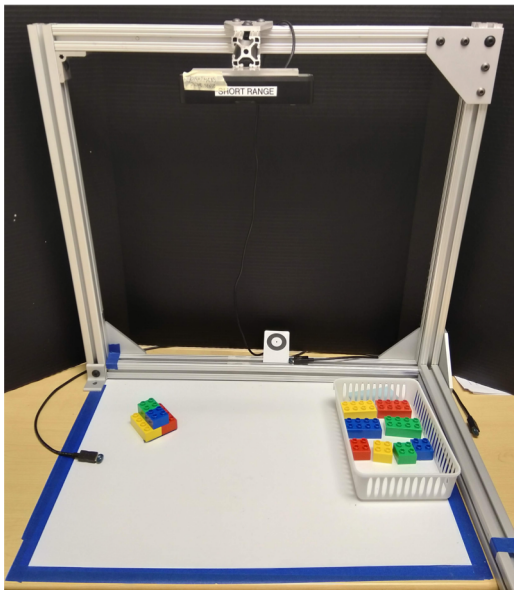
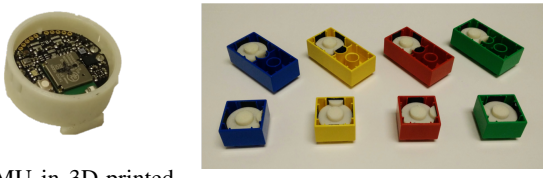


Figure 3: Video collection rig



(a) IMU in 3D-printed enclosure

(b) Blocks fitted with IMU enclosures

Figure 4: IMU-embedded DUPLO blocks

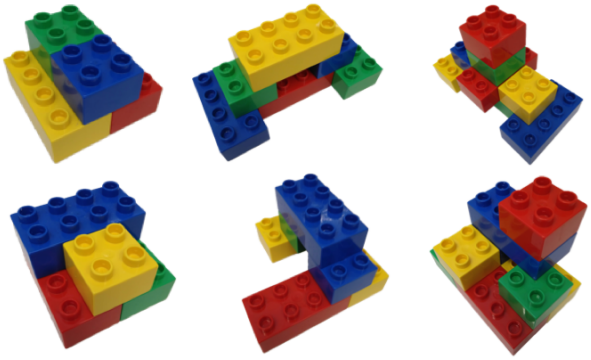


Figure 5: Target models to be assembled from the blocks

any hypothesis with low probability—*i.e.*, any state s_t with

$$\max_{p_t} P(I_{1:t}, p_{1:t}, s_{1:t}) < G \max_{p_t, s_t} P(I_{1:t}, p_{1:t}, s_{1:t}) \quad (6)$$

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.

4.2.2 Template registration

At each time step we evaluate state hypotheses locally using $P(I_t, p_t^*|s_t)$, 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 $\log P(I_t, p_t|s_t)$ of the image in a particular pose is proportional to the log likelihood $\log P(I_t|p_t, s_t)$, 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

$$R^*, \tau^* = \arg\min_{R,\tau} \sum_{x \in \mathcal{X}} \|I(x) - T(Rx + \tau; p_c, s)\|^2, \quad (7)$$

where (R, τ) is a rigid motion in the space of pixel coordinates. This is a nonlinear least-squares optimization problem, which can be solved using standard methods. For implementation details, see section 5.3.2.

5. Parsing Block Construction Processes

We apply the algorithm in Section 4.2 to the task of parsing videos of DUPLO block building activity. These data were recorded during behavioral experiments studying early childhood development of spatial skills.

5.1. Dataset description

5.1.1 Controlled dataset

To evaluate our system under conditions which match the core modeling assumptions, we collected 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 5 ourselves, taking care to ensure that there was an unobstructed view of each partial assembly as it was being created.

We recorded RGBD video using a Primesense Carmine camera mounted in an overhead position (see Figure 3),

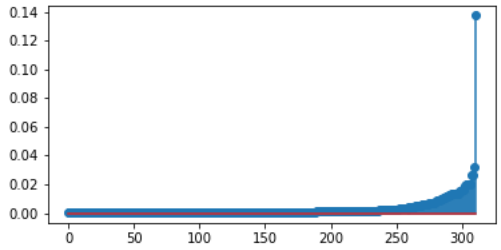


Figure 6: Prior distribution of states in the child's play dataset. Horizontal axis shows the (sorted) state index, vertical axis shows the state probability.

with 320x240 resolution and 30 Hz sampling rate. We recorded inertial measurements from MbientLab Metawear CPRO sensors. Each sensor was housed in a 3D-printed enclosure which was then fitted into a DUPLO block (See Figures 4a, 4b). We sampled linear acceleration and angular velocity from each device at a rate of 50 Hz.

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 was marked which captured an unobstructed view for each block state. In what follows, we refer to these frames as *keyframes*.

5.1.2 Child's play dataset

To evaluate the robustness of our system, we collected a dataset from child behavioral experiments [7]. In each of the 145 videos, a child participant attempted to copy one of the six models shown in Figure 5. A university ethics review board approved all study procedures, and participants and their legal guardians provided informed assent and consent, respectively.

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 [7] for more information about the data collection and annotation process.

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 block model 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 6).

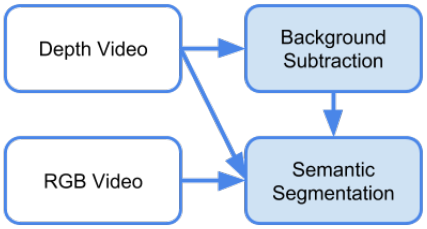


Figure 7: Image preprocessing pipeline

5.2. Image pre-processing

5.2.1 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.

5.2.2 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 annotated with these labels, we constructed a proxy by combining the keyframes from section 5.1.1, 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 d indexes a set of M Gaussian distributions with unit covariance, and is shared between both pixel classes. The probability of an image under this model is

$$P(I) = \prod_{x \in \mathcal{X}} \sum_{i=0}^{1} \sum_{j=1}^{M} P(c_i) P(d_j|c_i) P(I(x)|d_j). \quad (8)$$

We estimate the means of Gaussian distributions d_j by training a minibatch k-means [18] model on the proxy dataset described above. We estimate the conditional probabilities $P(d_j|c_i)$ by computing histograms of Gaussian components for the clear-view and obstructed-view images, and $P(c_0) = P(c_1) = 0.5$ 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 [21]. 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.

5.3. Other implementation details

5.3.1 Parameter estimation and hyperparameters

We estimate the state-to-state transition probabilities of the model in Figure 2 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 chose $M = 32$ Gaussian components for the mixture model used during semantic segmentation.

5.3.2 Inference

Template registration: We solve the optimization problem given by (7) using the Trust Region Reflective algorithm [5] implemented in SciPy [12]. We initialize t at the centroid of each of the image segments classified as blocks in 5.2.2 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 [17].

Decoding: To compensate for preprocessing errors and occluded segments, we apply add-one smoothing [10] to the HMM state transition probabilities when predicting on 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 6.1.1 and 6.1.2, and investigate its effect separately in Section 6.1.3. 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 modalities are independent.

6. Experimental Setup and Results

To test the system's performance in a setting that matches the modeling assumptions, we 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.

6.1. 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).

6.1.1 Controlled dataset

Table 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 with RGB does no worse than RGB on its own.

6.1.2 Child's play dataset

Table 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-grain level of detail, the system's multiple hypotheses and implicit prior world model enable robust estimates at a coarser level. More 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

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 1: Results, controlled dataset (macro-averages)

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

is worse using combined RGB and depth data than using RGB data alone, likely because of lower confidence in the RGB modality producing a less skewed interpolation between RGB and depth results.

6.1.3 Effect of Viterbi pruning

Although we did not prune any states during inference in sections 6.1.2 and 6.1.1 to obtain the best possible system performance, we investigate its effect here. Figure 8 shows our system's performance on the child's play and controlled datasets as the Viterbi pruning coefficient G is varied. In both cases, it is possible to ignore large proportions of the state space before significant performance degradation is incurred. Figure 8a 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 8b. The challenging nature of this 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.

7. Conclusion

In this paper we have outlined a model for assembly processes, derived a probabilistic inference algorithm, and applied it to parsing 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 still gives sensible results under much more challenging data conditions.

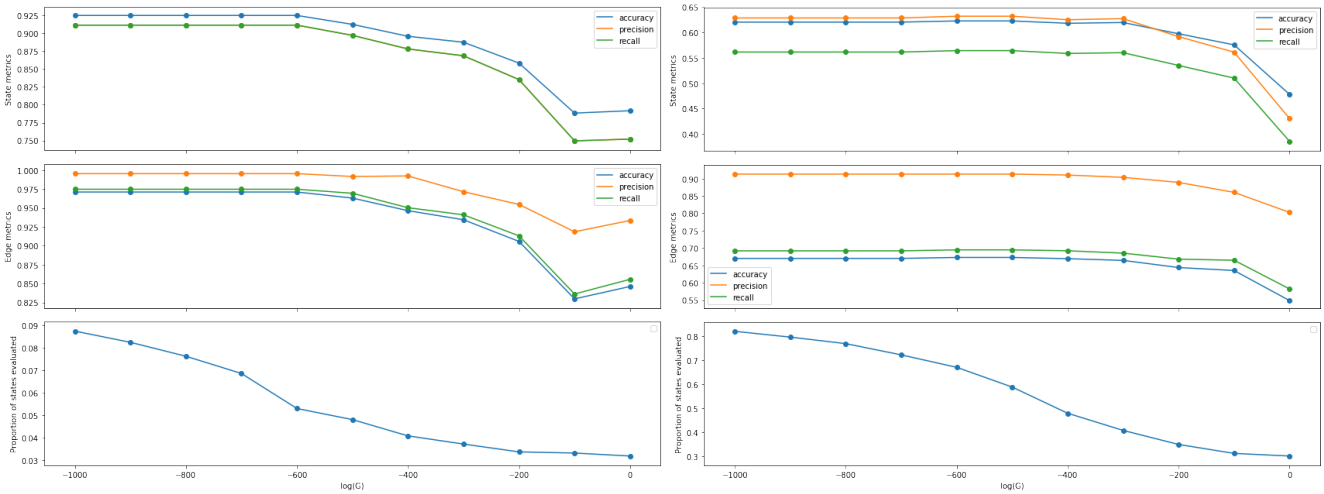
Results from these experiments suggest new research directions, particularly in fine-grained action recognition and in occlusion-robust computer vision. In this work the state parser operates on its own, but this model can work alongside an explicit action recognition system to enable more efficient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

The results in this paper 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 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.

8. Acknowledgements

This work was supported by NSF award No. 1561278. We also gratefully acknowledge our collaborators Cathryn Cortesa, Barbara Landau, and Amy Shelton for their leading role in the behavioral experiments underlying this paper.



(a) Controlled dataset

(b) Child's play dataset

Figure 8: System performance as a function of the Viterbi pruning coefficient. From top to bottom: state metrics, edge metrics, and proportion of states visited.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, Nov 2012.

[2] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[3] D. P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. II*. Athena Scientific, 3rd edition, 2007.

[4] A. Bosselut, C. Ennis, O. Levy, A. Holtzman, D. Fox, and Y. Choi. Simulating action dynamics with neural process networks. In *International Conference on Learning Representations*, 2018.

[5] M. Branch, T. Coleman, and Y. Li. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM Journal on Scientific Computing*, 21(1):1–23, 1999.

[6] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[7] C. S. Cortesa, J. D. Jones, G. D. Hager, S. Khudanpur, A. L. Shelton, and B. Landau. Characterizing spatial construction processes: Toward computational tools to understand cognition. In *Annual Meeting of the Cognitive Science Society*, pages 246–251, 2017.

[8] G. D. Hager and B. Wegbreit. Scene parsing using a prior world model. *International Journal of Robotics Research*, 30(12):1477–1507, 2011.

[9] E. Jahangiri, E. Yoruk, R. Vidal, L. Younes, and D. Geman. Information Pursuit: A Bayesian Framework for Sequential Scene Parsing. *ArXiv e-prints*, Jan. 2017.

[10] F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, Cambridge, MA, USA, 1997.

[11] J. Johnson, R. Krishna, M. Stark, L.-j. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-fei. Image Retrieval using Scene Graphs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3668–3678, 2015.

[12] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed November 20, 2018].

[13] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[14] H. S. Koppula, R. Gupta, and A. Saxena. Learning human activities and object affordances from rgb-d videos. *Int. J. Rob. Res.*, 32(8):951–970, July 2013.

[15] F. R. Kschischang, B. J. Frey, and H. A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[16] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[17] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial & Applied Mathematics*, 5(1), Mar 1957.

[18] D. Sculley. Web-scale k-means clustering. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 1177–1178, New York, NY, USA, 2010. ACM.

[19] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild, 2012.

[20] S. Stein and S. J. McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous*

Computing, UbiComp '13, pages 729–738, New York, NY, USA, 2013. ACM.

[21] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014.

[22] N. N. Vo and A. F. Bobick. From stochastic grammar to bayes network: Probabilistic parsing of complex activity. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[23] J. Wu, J. B. Tenenbaum, and P. Kohli. Neural scene derendering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[24] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei. Scene graph generation by iterative message passing. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pages 3097–3106, 2017.