Towards Automatic Extraction of Tile Types from Level Images

Sam Snodgrass
Northeastern University
Boston, MA USA
s.snodgrass@northeastern.edu

Abstract

In recent years, the use of machine learning for procedural content generation (PCGML) has been growing. These PCGML approaches require a training corpus of levels, often annotated or represented in some abstracted way. Due to the manual effort required to annotate or translate a sufficient training corpus, most PCGML techniques have only been explored in a handful of domains. In this paper we take a step towards addressing this core issue of PCGML by presenting an unsupervised method for automatically extracting a representation for a level domain, given only images of the levels. This approach is a move towards making PCGML more broadly applicable by reducing the effort required to create a training corpus. We evaluate our approach by comparing the automatically extracted tile representation against existing PCGML training level corpus representations.

Introduction

Procedural content generation via machine learning (PCGML) (Summerville et al. 2018) is a growing field of research that automatically extracts models from existing game content, and uses those learned models to generate new content. These approaches rely on a corpus of training data from which to estimate their models. However, the creation of such training data (often through manual annotation or domain-specific scripts) can require a large time commitment as well as expert domain knowledge in order to reason about the representation of the data for a given domain. This requirement of annotating training data is in direct opposition to one of the core benefits of PCGML: reducing the amount of domain knowledge that must be encoded by users. Subsequently, most PCGML approaches have only been tested in a handful of domain where training data is readily available (e.g., *Super Mario Bros.* (Guzdial and Riedl 2016; Snodgrass and Ontañón 2016b; Summerville and Mateas 2016), *The Legend of Zelda* (Summerville and Mateas 2015), and *Lode Runner* and *Kid Icarus* (Snodgrass and Ontañón 2016b)).

In this paper we begin research into relieving PCGML techniques' reliance on manual annotations and domain specific knowledge from users. We present a proof of concept unsupervised approach for extracting a representative set of tile types from video game level images, which can then be used to represent levels from the given game. Our unsupervised approach attempts to find groups of functionally similar objects using only positional and structural level information. Our goal is to further increase the usability of PCGML techniques and broaden the applicability of such techniques to new domains by reducing the amount of domain knowledge required to explore a new domain.

The remainder of this paper is organized as follows: first, we discuss the relevant related work; we then present our approach for extracting tile sets; next, we present our experimental set-up, including the domain in which we test our approach and how we evaluate our approach; then we present and discuss our results; finally, we close by drawing our conclusions, and suggesting avenues of future work.

Related Work

Most of the current PCGML techniques for level generation techniques require annotated or abstracted training levels, often derived from level images (Summerville and Mateas 2016; Snodgrass and Ontañón 2016b; Summerville and Mateas 2015). A notable exception is Guzdial and Reidl's (Guzdial and Riedl 2016) approach which leverages a spritesheet and gameplay videos in order to automatically identify structures and construct its own internal graphical representation of the levels. This is an interesting approach that is able to leverage its representation to create remarkable results. Additionally, the use of gameplay videos can be reduced to using a static level image where each frame is either considered separately as a level, or concatenated together to form a full level. Therefore, methods that rely on gameplay videos can also benefit from an unsupervised representation learning approach.

We are not the first to recognize the tension of needing annotated training data in PCGML. Summerville et al. (Summerville et al. 2016) created and maintain the Video Game Level Corpus, a repository of video game levels represented in a variety of formats, including graphical and tile-based for the purpose of video game research. Regardless of these efforts, there are a vast number of video games, and it is infeasible to convert many of their levels using the current manual or domain specific methods. Others have attempted to sidestep the need for annotated training data in new domains by combining models for various domains (Guzdial

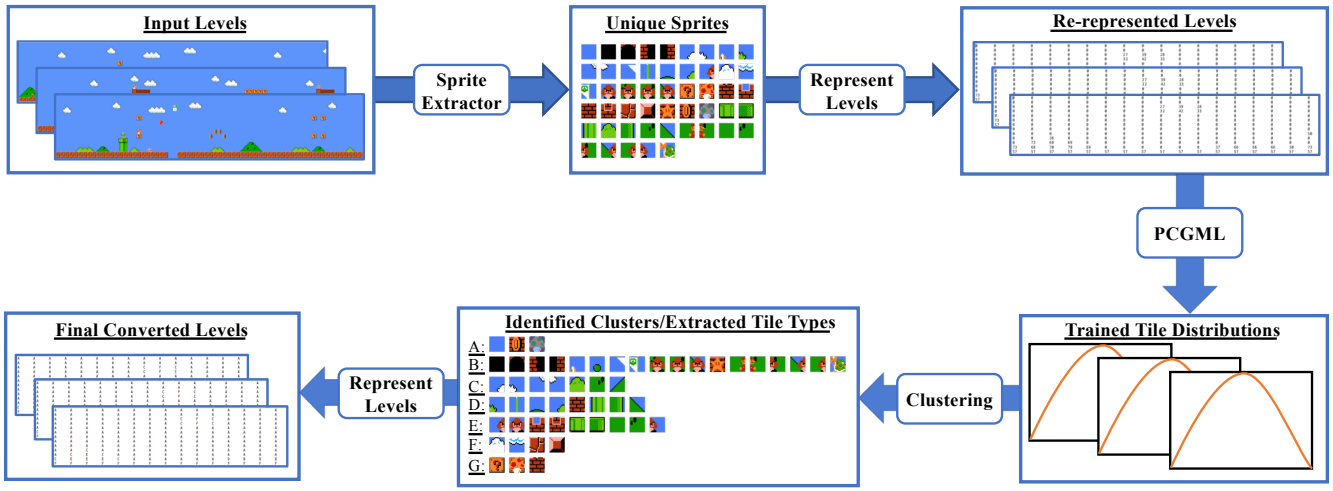


Figure 1: This figure shows the flow of our approach. We start with a set of level images; extract a set of unique sprites from those images; re-represent the levels using a unique identifier associated with each unique sprite; train a model on those represented levels; perform clustering on the sprites using the distance between the trained distributions as the metric, yielding a set of clusters corresponding to tile types; and finally we represent the input levels using the extracted tile types.

and Riedl 2018) or by transferring a learned model from one domain which has training data to another domain with more limited training data (Snodgrass and Ontanon 2016a). However, they still require training data for some (ideally functionally similar) domain, and thus do not address the root of the problem.

Some have explored the role of different structures and tile types in different game domains through interaction. In the General Video Game AI competition (Pérez-Liébana et al. 2016) the various agents needed to analyze and determine what the different elements in the given levels were without prior knowledge. The agents in this case were able to interact with the provided level and build up a world model this way. The most closely related work is that of Summerville et al (Summerville et al. 2017) which tries to determine what the elements in a *Super Mario Bros.* level do by analyzing gameplay traces and in-game events, and clustering player and object interactions modeled as probabilistic events. Notice, each of these approaches relies on the use of agent interactions in order to extract the function of the tiles, while we are first interested in seeing how far we can go only analyzing the structural elements of the level in order to determine the functional groupings.

Approach

In this section we propose our approach for automatically determining a representative set of tile types for a domain given a set of level images. At a high level our approach works in three phases: first, we automatically *label* the images using the set of unique sprites found in the level images; next, we *train* a Markov random field (Cross and Jain 1983) model on the levels treating each of those unique sprites as a temporary tile type; finally, we *cluster* the sprites using the distances between their learned probability distributions.

Figure 1 shows the flow of our approach. We discuss each of the above stages in more detail below.

Labeling

In this stage we first parse the input level images in order to extract a set of unique $x \times y$ pixel sprites. We then treat each of those unique sprites as a temporary tile type, and use them to re-represent the input levels in an intermediate tile-based representation, resulting in a set of tile-based levels that can be passed to the next stage of the pipeline.

Training

In this stage we train a statistical model on the newly created tile levels. Many machine learning approaches can be used here in order to extract a generalized representation of the input levels, but in our experiments we leverage a Markov random field approach (Cross and Jain 1983).

We train a Markov random field using a neighborhood of the four surrounding sprites in the level, as shown in Figure 2. Using the Markov random field, we estimate $P(c|t)$ from the set of levels represented in the tile format described above, where c is a configuration of surrounding tile types at a given position in the level, and t is the tile type at the center of that configuration. A similar modeling approach using Markov random fields has previously been used by Snodgrass and Ontañón 2016b to model and generate game levels. The key difference here is that Snodgrass and Ontañón estimated $P(t|c)$ in order to capture the proper placement of tile types within a level and replicate it during generation, whereas we estimate $P(c|t)$ so that we can more easily compare which configurations and patterns occur around specific tile types, thus allowing us to reason more directly about how different tile types occur with different patterns in the input levels.

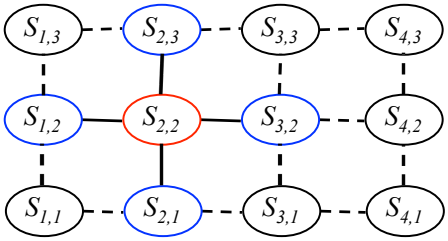


Figure 2: This figure shows the network structure used when training our Markov random field approach. The red cell indicates the current tile and the blue cells indicated the surrounding configuration or neighborhood.

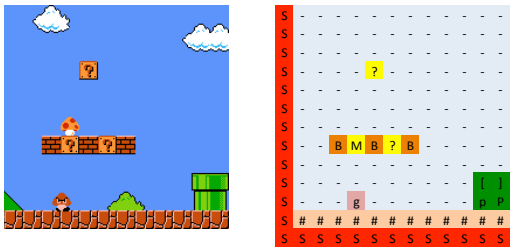


Figure 3: This figure shows a section of a *Super Mario Bros.* level represented using the complex manual tile set.

Clustering

In this stage we cluster the sprites based on the learned probability distributions corresponding to each sprite. To achieve the clustering, we leverage a hierarchical clustering approach implemented in R (Maechler et al. 2013). We use a hierarchical clustering approach so that we can easily inspect the clusters at varying levels of granularity.

For our distance metric, we compute the total variation distance (Verdú 2014) between the probability distributions learned for the given sprites. The total variation distance can be thought of as the maximum distance between two probability distributions for any one event. Specifically, we compute:

$$max(|P(c|t_i) - P(c|t_j)|) \quad \forall c \in C,$$

where the P are the probability distributions trained by the MRF, and t_i and t_j are the tiles for which the distributions are being compared, and C is the set of all possible surrounding tile configurations.

This hierarchical clustering approach results in a dendrogram where each leaf corresponds to a unique sprite. Thus, once the clustering is complete we can experiment with cutting the tree at different heights to investigate the clusters resulting from that cut. It is beneficial to be able to explore different granularities of clusters for our analysis, but other common methods for estimating the ideal number of clusters can be employed in place of manual inspection (e.g., the average silhouette method (Kaufman and Rousseeuw 2009)). Additionally, other common clustering techniques can be employed here, such as k-medoids or DBSCAN.

Experimental Evaluation

In this section we discuss our experimental design including the domain explored, the evaluations metrics used, and the results of our experiments.

Domain

We test our approach with the first level of *Super Mario Bros.*, a platforming game that has commonly been used as a testbed in PCG (Marino, Reis, and Lelis 2015; Shaker et al. 2011; Mawhorter and Mateas 2010) and PCGML (Dahlskog, Togelius, and Nelson 2014; Guzdial and Riedl 2016; Snodgrass and Ontañón 2016b; Summerville and Mateas 2016). We use only the first level in our experiments in order to explore the feasibility of our approach by using a limited number of unique sprites. Many current PCGML approaches that have been tested in this domain have leveraged a tile-based representation of the levels, and there have been several different tile representations used with varying degrees of fidelity. In our experiments, we compare our automatically extracted tile sets against two manually-defined tile sets:

- **Simple Manual**: This is the tile set used by the VGLC to represent levels in this domain. It consists of 11 tile types, and abstracts different enemy types to a single tile type, and represents above ground levels without treetops, bridges, or moving platforms. For the level we use in our experiments 7 of these tile types used.
- **Complex Manual**: This is the tile set used by Snodgrass and Ontañón in their more recent work (Snodgrass and Ontañón 2016b). This tile set consists of 45 tile types. It distinguishes between the different enemy types, it distinguishes blocks based on their contents, and is able to represent all above ground levels, castle levels, and underground levels. Figure 3 shows a section of a level represented in this format. For the level we use in our experiments 15 of these tile types are needed.

The mappings of the set of unique extracted sprites to these sets of tile types can be seen in Tables 1 (left) and 2 (left), respectively.

Evaluation Methods

Using the approach outlined previously, we extract a set of unique sprites from the input level image, and then cluster them in order to automatically determine different sets of tile types. We evaluate the results of our clustering by investigating the cluster statistics and through manual inspection of the clusters themselves. For the cluster statistics, we compute the average silhouette of the clusters created by cutting the dendrogram created by the hierarchical clustering method at several levels. This measures how well, on average, each sprite fits within its own cluster versus other clusters. We also note the largest and smallest cluster sizes for each clustering. This metric shows us the spread of the clusters and can help inform users about what a desirable number of clusters may be. For the manual inspection, we explore the differences between the found clusters and the manually defined tile types. For our evaluation we cut the

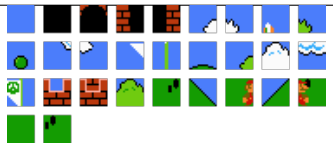
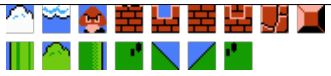
Simple Manual		Simple Clustering	
Tile Type	Sprites	Cluster	Sprites
Empty		*Cluster 1*	
Enemy		*Cluster 2*	
?-Block		*Cluster 3*	
Brick		*Cluster 4*	
Solid		*Cluster 5*	
Left Pipe		*Cluster 6*	
Right Pipe		*Cluster 7*	

Table 1: This table shows the *Simple Manual* set tile types and their corresponding sprites from the training level (left), and the *Simple Clustering* set of automatically extracted tile types determined via clustering (right). The arrangement of the clustering results does not indicate a relation to the manual tile type.

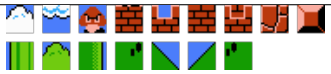
Complex Manual		Complex Clustering	
Tile Type	Sprites	Cluster	Sprites
Empty		*Cluster 1*	
Flagpole		*Cluster 2*	
Goomba		*Cluster 3*	
?-Block		*Cluster 4*	
Brick		*Cluster 5*	
Powerup		*Cluster 6*	
Solid		*Cluster 7*	
Extra Life		*Cluster 8*	
Top-Left Pipe		*Cluster 9*	
Top-Right Pipe		*Cluster 10*	
Coin Brick		*Cluster 11*	
Star Block		*Cluster 12*	
Left Pipe		*Cluster 13*	
Right Pipe		*Cluster 14*	
Koopa		*Cluster 15*	

Table 2: This table shows the *Complex Manual* set tile types and their corresponding sprites from the training level (left), and the *Complex Clustering* set of automatically extracted tile types determined via clustering (right). The arrangement of the clustering results does not indicate a relation to the manual tile type.

dendrogram to produce 7 clusters and 15 clusters, so that we can compare these clustering results to the manually defined 7 and 15 tile types that have been used previously to represent the chosen level.

Results

Tables 1 and 2 show the manually defined tile sets and the sets of sprites represented by each tile type (left) and the clusters found by our approach (right). In many cases our approach clusters tiles that are functionally similar. For ex-

k	Avg. Silhouette	Max. Size	Min. Size
2	0.3778221	46	7
7	0.1788397	21	1
11	0.1791865	21	1
15	0.1825074	20	1
20	0.09250035	16	1
26	0.08664249	10	1

Table 3: Cluster statistics for the clusters found by cutting the dendrogram for various numbers of clusters. 7 and 15 are used in the rest of our evaluation because the manually defined tile sets have represent the chosen training level with 7 (simple) and 15 (complex) tile types.

ample, in both settings, a cluster is found containing many of the brick and question mark tiles. Additionally, the pipe-top tiles are accurately grouped together in the simple setting (albeit with a background tile), and are accurately split in the complex setting. This is encouraging as it shows that these distinctions can be made automatically with only structural information (to some extent), but there are clear issues. Cluster 1 (in the simple setting) contains a mix of background sprites and enemies, and in general background sprites are interspersed with many of the clusters. This may be because while the background sprites all perform similar functions, the individual sprites (e.g., cloud sections, bush sections, etc.) often only appear in specific configurations, and thus have a very sparse (and very distinct) probability distribution, which makes them more likely to get grouped in with other sprites. For example, in the simple setting the bush and hill background sprites get clustered with the bottom pipe sections. Notably, all of these sprites typically appear just above the ground sprites. This behavior is reflected in the complex setting as well. This suggests a shortcoming of the distance metric used for clustering and potentially an insufficiency in using only the positional information of the sprites. In the future, more informative distance metrics could be considered which may encompass frequencies of the tiles appearances or perhaps the shapes on contiguous tiles similar to Guzdial and Riedl's clustering approach (Guzdial and Riedl 2016).

Tables 3 shows the average silhouette of the clusters, as well as the maximum and minimum cluster sizes. Recall that the average silhouette of a clustering approximates the spread of the clusters, and a smaller silhouette means that the elements within a cluster are more closely related. As expected, with more clusters the average silhouette typically decreases. An interesting exception here is that the silhouette when $k = 15$ (for the complex setting) is larger than when $k = 7$ (for the simple setting). This indicates that when $k = 7$ we get tighter, more similar clusters. This is somewhat reflected in our manual inspection of the clustering results discussed above.

Limitations and Future Work

As this was an initial step, there are two major limitations that we hope to address in the future. First, while our clustering approach was able to group some functionally similar sprites together (e.g., bricks, pipes) it struggled with the grouping of others (most notably the background sprites: sky, cloud, hill, and bush). This is partially a shortcoming of the distance metric and clustering employed, as several of the background sprites appear in similar configurations as other sprites (e.g., the hills and the pipes). We aim to first explore more robust modeling and distance metric options. However, as Guzdial et al. discuss (Guzdial, Sturtevant, and Li 2016), we likely need both static and dynamic analysis (i.e., analysis of the structural and gameplay elements, respectively) to get a complete understanding of the domain. Therefore, once we more fully explore our static analysis options (i.e., distance metrics, clustering options, modeling approaches), we will incorporate dynamic analysis. Dynamic analysis will help our model reason more clearly about the functionality of the sprites, and cluster them more cohesively. Incorporating dynamic analysis can easily undermine the goal of our work (i.e., reducing reliance on domain knowledge and domain specific scripts) by requiring a specific agent for each domain. To avoid this potential tension, we will explore the use of reinforcement learning agents, specifically those used in the General Video Game AI competition, that may be able to learn how to play the game without requiring much domain knowledge.

The second limitation of our work is that we have thus far only applied it to one domain, and only a fraction of that domain. To address this, we will first apply our approach to a larger subset of *Super Mario Bros.* levels, including those with different visual sprite sets, such as castle and underground levels. We are also interested in exploring our approach in domains unexplored by PCGML techniques thus far, such as *Metroid*, which do not have a defined set of tiles used by researchers that can be treated as the "ground truth" as we did in the *Super Mario Bros.* domain. This will help us explore the robustness of our refinements and force us to devise and explore more general methods of evaluation.

Conclusions

In this paper we presented a proof of concept approach for automatically extracting a set of representative tile types from a set of input level images without requiring domain or design knowledge from the user. This approach is meant as a step towards alleviating the reliance of PCGML users on domain specific scripts and manual annotation when creating training data. Using our approach, we automatically extracted 2 sets of tile types and found some similarities between the extracted tile sets and manually defined tile sets of the same size. The found clusters also contained quite a bit of noise and did not perfectly delineate the sprites by functionality. In the future we will explore methods for defining cleaner clusters and further exploring how this automatic approach performs more broadly, both in other domains and paired with a variety of PCGML techniques.

References

Cross, G. R., and Jain, A. K. 1983. Markov random field texture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* (1):25–39.

Dahlskog, S.; Togelius, J.; and Nelson, M. J. 2014. Linear levels through n-grams. *Proceedings of the 18th International Academic MindTrek*.

Guzdial, M., and Riedl, M. 2016. Game level generation from gameplay videos. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Guzdial, M., and Riedl, M. O. 2018. Combinatorial creativity for procedural content generation via machine learning. *"The First Knowledge Extraction from Games Workshop"*.

Guzdial, M.; Sturtevant, N.; and Li, B. 2016. Deep static and dynamic level analysis: A study on infinite mario. In *Experimental AI in Games Workshop*, volume 3.

Kaufman, L., and Rousseeuw, P. J. 2009. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons.

Maechler, M.; Rousseeuw, P.; Struyf, A.; Hubert, M.; and Hornik, K. 2013. *cluster: Cluster Analysis Basics and Extensions*.

Marino, J. R.; Reis, W. M.; and Lelis, L. H. 2015. An empirical evaluation of evaluation metrics of procedurally generated Mario levels. In *Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*.

Mawhorter, P., and Mateas, M. 2010. Procedural level generation using occupancy-regulated extension. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, 351–358. IEEE.

Pérez-Liébana, D.; Samothrakis, S.; Togelius, J.; Schaul, T.; and Lucas, S. M. 2016. Analyzing the robustness of general video game playing agents. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, 1–8. IEEE.

Shaker, N.; Togelius, J.; Yannakakis, G. N.; Weber, B.; Shimizu, T.; Hashiyama, T.; Sorenson, N.; Pasquier, P.; Mawhorter, P.; Takahashi, G.; et al. 2011. The 2010 mario AI championship: Level generation track. *TCIAIG, IEEE Transactions on* 3(4):332–347.

Snodgrass, S., and Ontanon, S. 2016a. An approach to domain transfer in procedural content generation of two-dimensional videogame levels. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Snodgrass, S., and Ontañón, S. 2016b. Learning to generate video game maps using Markov models. *IEEE Transactions on Computational Intelligence and AI in Games*.

Summerville, A., and Mateas, M. 2015. Sampling hyrule: Sampling probabilistic machine learning for level generation.

Summerville, A., and Mateas, M. 2016. Super Mario as a string: Platformer level generation via LSTMs. *Proceedings of 1st International Joint Conference of DiGRA and FDG*.

Summerville, A. J.; Snodgrass, S.; Mateas, M.; and Ontañón, S. 2016. The VGLC: The video game level corpus. In *Seventh Workshop on Procedural Content Generation at First Joint International Conference of DiGRA and FDG*.

Summerville, A.; Behrooz, M.; Mateas, M.; and Jhala, A. 2017. What does that?-block do? learning latent causal affordances from mario play traces. In *Proceedings of the first Whats Next for AI in Games Workshop at AAAI*, volume 2017.

Summerville, A.; Snodgrass, S.; Guzdial, M.; Holmgård, C.; Hoover, A. K.; Isaksen, A.; Nealen, A.; and Togelius, J. 2018. Procedural content generation via machine learning (PCGML). *IEEE Transactions on Games*.

Verdú, S. 2014. Total variation distance and the distribution of relative information. In *ITA*, 1–3. Citeseer.