Compressing and Comparing the Generative Spaces of Procedural Content Generators

1st Oliver Withington
Queen Mary University of London
London, UK
o.withington@qmul.ac.uk

2nd Laurissa Tokarchuk
Queen Mary University of London
London, UK
laurissa.tokarchuk@qmul.ac.uk

***Abstract*—The past decade has seen a rapid increase in the level of research interest in procedural content generation (PCG) for digital games, and there are now numerous research avenues focused on new approaches for driving and applying PCG systems. An area in which progress has been comparatively slow is the development of generalisable approaches for comparing alternative PCG systems, especially in terms of their generative spaces. It is to this area that this paper aims to make a contribution, by exploring the utility of data compression algorithms in compressing the generative spaces of PCG systems. We hope that this approach could be the basis for developing useful qualitative tools for comparing PCG systems to help designers better understand and optimize their generators. In this work we assess the efficacy of a selection of algorithms across sets of levels for 2D tile-based games by investigating how much their respective generative space compressions correlate with level behavioral characteristics. We conclude that the approach looks to be a promising one despite some inconsistency in efficacy in alternative domains, and that of the algorithms tested Multiple Correspondence Analysis appears to perform the most effectively.**

I. Introduction

Procedural Content Generation (PCG) for games, the algorithmic generation of digital artifacts, has in the past decade developed into a lively and diverse research field, with a proliferation of new works exploring novel implementations and use-cases of PCG systems. As the volume of new work has increased, it has become increasingly important to develop methods for comparing alternative generators and their outputs. Without them it is hard to identify when a new approach is a useful advance for the field, or for game designers to select the best approach for their purposes. For some forms of analysis this is more straightforward. PCG systems can be deployed in settings with quantifiable goals such as providing training material for Reinforcement Learning (RL) agents [1] [2], or more niche ones like generating high-density weather resistant cities [3]. For use-cases of PCG where the purpose of the generated artifacts is to be consumed by players or designers then comparing PCG systems is much more complex, as the role the artifacts are playing is fundamentally a subjective and artistic one. This makes it hard to assess the extent to which a generator is achieving its intended purpose.

A central concept for when the goal is to compare PCG systems in terms of their output, is that of 'generative space'. This refers to the conceptual volume that represents 'the theoretical space of all possible output of a generator [4]. When the goal is to compare PCG systems in terms of their output the underlying goal is to be able to draw useful comparisons between their generative spaces, but this can be extremely challenging. The total size of the spaces in terms of the number of possible artifacts can be extremely large, such as the over 18 quintillion procedurally generated planets that No Man Sky boasted at launch [5]. The relationship between the contents of a generative space and the parameterisation of the generator can also be unpredictable, especially if there are stochastic elements to the generation process. Furthermore, the applicability of a given generative space to a given domain can also be a subjective and artistic one. In some settings it might be acceptable to have undesirable or even non-functional artifacts in their generative space so long as the average output is diverse and interesting enough, such as when a filtering process is involved. For another, any unusable artifacts being present may be unacceptable, for example in settings where all generated content is 'necessary' [6].

A common approach for aiding designers in understanding generative spaces is to convert the extremely high dimensional uninterpretable volume which contains the direct encodings of the generated artifacts into something lower dimensional and human understandable. The most common way of doing this in prior research on game level generation is Expressive Range Analysis (ERA) [7]. This approach visualises generative spaces by calculating and mapping emergent Behavioral Characteristics (BCs) of the generated levels such as heuristics for difficulty or aesthetic qualities. ERA has been widely adopted as a metric for qualitatively comparing different generative spaces (See [8], [9], [10] for recent examples). However, it is also possible to directly compress the encoded representations themselves to produce similar low dimensional representations without the need to decide on BCs of interest as in ERA. This is commonly done as an intermediary step in a larger generative process, often to produce a low dimensional form of levels that can be understood by a neural network [11], [10], [12]. However we argue this direct compression can be more intrinsically valuable for producing useful representations of generative space which can help bridge the gap between designers and understanding of their generators.

The approach we explore in this paper is to compress high dimensional encoded representations of levels from a generative space to produce two dimensional projections that capture as much of the variance in the level set as possible. This projection which represents the underlying generative space can then be qualitatively understood and compared to alternatives in terms of the types and variety of levels that can be produced. If effective, this could let designers more easily understand and compare generative spaces without needing to make any decisions about the types of diversity which are of interest. We explore this approach in the context of generators for 2D tile-based games as there is a large amount of prior work and research that we can leverage such as level generators, pre-generated level corpuses [13], [14] and research frameworks [15], [16]. By taking samples of encoded generated levels from a system or selection of systems and treating them as sets of variables in which each variable represents a portion of the level, we can then use dimensionality reduction algorithms to represent that set using a smaller number of new compressed variables. The goal is that this compression approach will make it significantly easier to understand and compare the types and variety of content that can be produced by alternative PCG systems in a way that is easy to configure and domain agnostic.

To experimentally explore this approach we assess four commonly used dimensionality reduction algorithms (PCA, SVD, MCA and T-SNE) in the domain of compressing the generative spaces PCG systems for three 2D tile-based games: Super Mario, Lode Runner and Boxoban [13], an open source version of Sokoban. To aid in the comparison between the alternative algorithms, we assess the extent to which diversity in the compressed low dimensional space correlates with diversity in terms of behavioral characteristics (BCs) of the game levels. We conduct analysis on the linear correlation between the distances between levels in the compressed generative space against the difference between their BCs. The more that there is a correlation between the two, the more credibly we can claim that we are compressing the generative spaces of PCG systems while preserving behavioral information which would be useful to game designers. While the idea of applying dimensionality reduction algorithms to encoded game levels is not itself novel, we believe this is both the first to use it to compare alternative generative spaces, as well as the first to explore the correlation between the compressed space and the behavioral features of the levels.

The rest of this paper is laid out as follows. In Section II we discuss the most relevant related work and how this project builds on its ideas. In Section III we introduce and discuss the approach used, and the system we have implemented to assess it. In Section IV we explain the experimental design of the experiments presented, and in Section V we present the results from these experiments. In Section VI we discuss the implications of the results, as well as the limitations of the underlying approach and the future work that should be done to further explore it. In Section VII we conclude that this appears to a promising approach for understanding and comparing generators which is worthy of further examination in alternative domains and configurations.

II. Related Work

The concept of generative space appears in a majority of works focused on PCG systems, either directly or indirectly. Depending on the researcher and the context, many different terms can be used to refer to a PCG system's generative space. They can be referred to as 'search spaces' in the context of generate-and-test PCG systems [17], [18], or as 'possibility spaces' in work using stochastic PCG systems [3], [19]. Researchers investigating Quality-Diversity (QD) algorithm based approaches for PCG often refer to 'behavioral space' as QD approaches rely on characterising generative spaces in terms of emergent artifact behaviors. In each case the underlying concept is largely the same. They are all different ways of conceptualising the total set of possible outputs from a PCG system. In this work we use the term 'generative space' as it is widely understood as well as generalisable to different domains.

The other concept used frequently in this paper is 'Behavioral Characteristic' (BC). This term is commonly used in PCG works based on Quality-Diversity search (See [10], [20], [21] for recent examples) and it refers to emergent characteristics of generated artifacts which can be quantified to motivate the search for output diversity. Similar concepts often appear in PCG research under other names such as behaviors [22], or more simply as 'metrics' [23], [24]. These BCs can be derived from the encoded artifacts directly [25], [23], or derived from simulated play by an agent [21], [26].

The most prevalent method which aims to aid with the understanding and analysis of full generative spaces is Expressive Range Analysis (ERA). ERA was introduced by Smith and Whitehead in 2010 [7] and has since become a dominant method for understanding and comparing the generative spaces of PCG systems. To use it a designer selects two or more BCs of the generated levels which are then calculated for a sample of generated levels. They can then be visualised in lower dimensional space, typically on a 2D graph or heat map, allowing designers to visualise the generative spaces of their PCG systems in terms of BCs which are of most interest. This can have many benefits, such as highlighting where BCs are in conflict with each other and how different parameterisations of the same generator change the location and size of the resultant generative space in BC space [25], [23]. It is also used as a heuristic for comparing the output diversity of alternative generators [8].

The primary limitations of ERA are that it can only be used to visualise two dimensions of diversity simultaneously, and that it can be challenging to determine and quantify what diversity is of interest. The first limitation can be offset using the approach of Summerville [27] who used Corner Plots to visualise multiple BCs simultaneously, though the majority of works using ERA still opt to use sets of 2D visualisations based on two BCs for ease of readability. The second is more problematic as it speaks to the subjectivity of analysing the underlying artifacts. The recent work of Herve and Salge partly mitigates this weakness by exploring the relationship between commonly used BCs in PCG and expert evaluations of game content in the game Minecraft [24]. Their finding that there was significant correlation between perceptual differences between game artifacts and commonly used BCs bolsters the use of ERA, as well as the use of BCs as a heuristic for generative space diversity. However it does not address the issue that BCs and the heuristics that assess them need to be redesigned for each new game domain. As we will discuss later in this paper, the hope is that the approach presented here can realise many of the benefits of ERA while mitigating or avoiding these limitations.

The approach discussed in this paper relies on applying dimensionality reduction algorithms to representations of game levels. This idea has been used as a preliminary or intermediary step in several pieces of PCG research which were direct inspirations for this work. In 'Sampling Hyrule' from Summerville and Mateas [28] principal component analysis (PCA), a widely used dimensionality reduction algorithm, was used to compress representations of Zelda levels to construct a low dimensional representation of the space which could be sampled from to generate new levels. In 2018 Justesen et al used PCA to visualise how their generated level sets were distributed in relation to levels from the original games [1]. Variational Auto-Encoders, a neural network designed for dimensionality reduction, have also become widely used in PCG for level generation [29], [8]. These works and our own relate back to the landmark paper 'Eigenfaces for Recognition' [30], which found that PCA applied to raw image data of human faces could be used as the basis for accurate facial recognition software using only eight new variables. The insight that image data containing thousands of variables can be compressed while maintaining real world useful information adds weight to the idea that a similar approach could work for compressing generative spaces.

This work is also closely related to the emerging and popular subfield of Machine Learning-based PCG approaches for game levels, commonly referred to as PCGML (See [31] for a recent overview of this field). These approaches use neural networks to learn from sets of game levels and generate new ones. These techniques have been applied to many diverse goals, such as reproducing the style of expert designers [32], learning user preferences [33] and generating new levels for unseen games [8]. These works are related to this one in two key ways. Firstly, they typically aim to extrapolate useful information about game levels directly from their representations, and their success adds credibility to the idea that encoded forms of game levels contain sufficient real-world useful information about their form and function. Secondly, they are very relevant to the concept of generative space. PCGML can be conceptualised as a process of learning to reproduce a generative space in the case of training directly from sets of game levels, or as the process of producing an ideal generative space from diverse inputs as in the case of learning from user preferences.

III. Approach

The goal for this approach is to represent sets of generated levels from a PCG system in a compressed two dimensional space, while maintaining enough information about the levels such that levels close together in the compressed space have similar BC values. To achieve this we apply dimensionality reduction algorithms to sets of game levels to create new uncorrelated variables composed out of combinations of the variables that compose the encoded level representations. We can then select the two new variables which explain the most variance in the underlying level data and reproject the level set in this space, giving us a two dimensional visualisation of the original high dimensional generative space. This should help designers to answer questions such as:

- Whether a pair of generators produce similar levels
- What kind of outliers are present in a generative space
- What effect re-parameterisation of a generator is having on its output

The two requirements for this approach to be applied to set of levels are that every level be the same size and that they be assembled out of discrete parts in which each part can have one of a discrete set of values. Asides from these two requirements the approach is intended to be content agnostic and applicable to alternative content representations without significant configuration or domain specific tweaking.

The high level steps of the system we use for applying and validating the approach are as follows:

To start, sets of levels are produced or sourced from each system that we want to evaluate which serve as representatives of the underlying generative space that they came from. In this iteration of the system designed to work with tile-based 2D game levels, each level is loaded as a 2D matrix of characters in which each location in the matrix represents a tile in the level, and each character represents the corresponding tile type (i.e a solid block, empty space etc) that appears at that location.

The next step is to flatten the encoded levels into one dimensional arrays. If the compression algorithm being used uses categorical data then this is done in a single step, with each row of the character matrix combined horizontally into a single, ordered row. For algorithms which require numeric data we compress them into a 1D one-hot matrix in two steps. First, the character matrix is converted into a 3D one-hot matrix of size height x width x number of tile types, with every value set to 0 apart from those which indicate the tile type which appears at each location, which are set to 1. This 3D matrix can then be flattened to 1D in the same way as the categorical data. This one-hot conversion is the same that is used in many GAN-based PCGML works [32], [33]. The full set of 1D representations can then be stacked on top of each other to give a 2D matrix in which every row represents a level, and every column represents a location in the original level, or location and tile type in the case of one-hot encoding.

The compressed 2D matrix representing all levels to analyse is now ready to have a dimensionality reduction algorithm applied. In this work we implement and compare four different algorithms: Single Value Decomposition (SVD), Principle Component Analysis (PCA), Multiple Correspondence Analysis (MCA) and T-distributed Stochastic Neighbor Embedding (T-SNE). While they all operate differently (See Section IV-A), they are all designed to uncover underlying structures and dimensions in data so that it can be modelled using a new, smaller set of variables. The original data can then be reprojected using the top two most explanatory new variables produced by the respective algorithms. We note that all algorithms tested apart from T-SNE quantify the amount of variance explained by the generated variables. In typical uses of dimensionality reduction algorithms this is extremely important as it indicates how much of the mathematical variance of the underlying data set is being captured by the top n new variables. However, in this work we are interested in the behavioral difference between the generated levels, not in mathematical variance in their representations. As a result it is not the mathematical variance explained that we report on, but the amount that variance in the projected 2D space correlates with BC variance.

With the generative space visualisations now generated we can assess how effective each compressed projection is at capturing behavioral information about the levels. To assess this we calculate the linear correlation between BCs of the levels and the levels' relative positions in the compressed space using Spearman's rank coefficient. The claim we make is that the more that proximity between levels in the compressed space correlates with proximity in the levels's BC values the more credibly we can claim that the compression is conserving useful behavioral information about the levels and the more we are realising the benefits of ERA without many of its limitations. To calculate this we take every possible pair of levels and calculate the distance between their locations in the compressed spaces, and the difference between the values for commonly used BCs like number of enemies and linearity. We then look for correlation between the two values by calculating Spearman's ρ, which we then use as our heuristic for the performance of the compression algorithm that produced the compressed space.

IV. Experiment Design

In this section we provide an overview of the experimental design used in this work, as well as the justifications for the design decisions in the current implementation. The system is implemented in Python and is available on GitHub at https://github.com/KrellFace/Generative-Space-Compression

A. Compression Algorithms

We implement four data compression algorithms in this work: PCA, SVD and MCA. Kernal PCA [34] was also implemented but was found to not meaningfully outperform PCA in any configuration. For PCA we use the standard implementation provided by sklearn in its decomposition module. For SVD we use the TruncatedSVD implementation from the same module, though we note that this works identically to standard SVD. For T-SNE we use the implementation from the manifold module of sklearn and for MCA we use the implementation from Max Halfords prince module found at: (https://github.com/MaxHalford/prince). These four algorithms were chosen for several reasons, including their ease of implementation and the wide number of domains in which they have demonstrated usefulness [35]. More specifically, PCA was chosen as it has demonstrated utility at compressing game levels in prior works [1], [28] and it is commonly accepted as 'the most important linear dimensionality reduction technique' [36]. SVD was chosen as it operates similarly to PCA except without first centering the data and we wanted to observe the influence of this on the final data. MCA is used as it is regarded as the categorical data counterpart to PCA [37], and seeing as the levels are encoded as categorical rather than continuous data it could allow for similar analysis while avoiding a pre-processing step of converting to a one-hot matrix. Finally, T-SNE has also demonstrated utility in game level analysis [8] and is regarded as well suited specifically for visualising high dimensional datasets with nonlinear structures [38] . However, there are many alternative algorithms with similar goals and outputs that could be used here [35], and future work could usefully explore these further.

B. Game Domains

This work assesses generative spaces from three different game domains: Super Mario, Sokoban and Lode Runner, using open source level corpuses for each game.

For Super Mario we make use of the Mario AI Benchmark, an open source platform for AI research based on Super Mario [15]. The most up to date version of the platform can be found at (https://github.com/amidos2006/Mario-AI-Framework) courtesy of Ahmed Khalifa, and it comes packaged with pre-generated level sets from a selection of 9 generators which were generated as part of the work of Horn et al [39], along with 15 levels from the original game. Each generated set comes with 1000 generated levels. All 9000 generated levels are used in our analysis, while the 15 from the original game are excluded as they are of varying sizes unlike the generated sets which are all 16 by 200 cell grids. We use a simplified encoding system in which every tile value is mapped to one of five types: Empty Space, Enemy, Solid, Pipe and Reward. A variety of generative approaches are represented in the level sets used [39], and the Super Mario levels are the largest tested in terms of tile count by a significant margin. This should present the generative space compression approach with a challenge distinct from the other two domains, in which relatively distinct generative spaces are localised in large and sparse high dimensional spaces.

For Sokoban we use level sets provided as part of Guez et al's research into Boxoban using an open source variant called Bokoban [13]. Three sets of levels are provided: 'unfiltered', 'medium' and 'hard'. The unfiltered set were generated by Guez et al using the approach of Racaniere et al [40], and the other two were generated and selected the approach of Guez et al explained in [41]. For the medium set 500,000 levels are available, stored in sets of one thousand in individual text files. For the unfiltered set one million levels are available, also stored in individual text files containing a thousand levels. The hard set contains 3,332 levels. We use the same encoding used in the original work, with the only tile types available being solid block, empty space, pushable block, goal and player spawn location. The medium and hard sets were selected based on the failure of trained reinforcement learning agents at solving them. This means that the primary difference between the sets is not in the generative approach used in making them, but in the difficulty of solving them. This provides an interesting challenge for compressing the generative space of the three sets, as while they present significant variety in the gameplay experience provided in terms of their difficulty, they were all generated using a similar underlying approach before being filtered based on difficulty. This combined with the relatively small encoding sizes, with the smallest total size and tile variety of the three game domains, makes them an appealing challenge for testing this approach.

For Lode Runner we assess only the 150 levels found in the original game. These are retrieved in an encoded form from the Video Game Level Corpus (VGLC) [14], an open source repository of encoded tile-based game levels. We note that this means in this case we are not in fact assessing a generative space as this set was hand authored rather than coming from a generator. However, as there is little conceptual difference in compressing a generative with compressing a space of hand authored levels we feel that this will still be a valuable experiment. The small size and high aesthetic diversity of the level set also provide a different challenge to the compression technique than that provided by the other two game domains. We use the same encoding system as the VGLC in this work.

C. Level Set Selection

For the Super Mario and Boxoban domains, a subset of 4,000 levels are randomly selected for each experimental run and this selection is evenly distributed between the nine individual generator sets for Super Mario, and the three sets for Boxoban. A subset is taken for these domains to limit the computational resources required. A large sample size was chosen to increase the credibility of the Boxoban and Super Mario results as a larger set is presumed to be more representative of the underlying generative spaces than a small sample would be. However, initial experimentation as well as the Lode Runner results suggest that lower sample sizes could still produce effective results. For Lode Runner the full 150 levels are used in every run.

D. Behavioral Characteristics

For each game domain we calculate between two and three BCs (See Table I). Each BC was selected for both being quick to calculate based on an encoded representation. Both linearity and enemy count have appeared frequently in prior work using ERA [7], [42] and have also been found to reflect player perceptions [43]. Contiguity, a measure which rewards solid blocks being adjacent, has has appeared in prior work using QD algorithms in PCG for tile-based games [44] and is

TABLE I
BEHAVIORAL CHARACTERISTICS

Game	BC	How Calculated (Count)
Mario	Empty Space (ES)	Empty Tiles
	Linearity (Lin)	Horizontally Adjacent Solid Tiles
	Enemy Count (EC)	Enemy Tiles
Sokoban	Empty Space (ES)	Empty Tiles
	Contiguity (Contig)	Adjacent Solid Tiles
Lode Runner	Empty Space (ES)	Empty Tiles
	Linearity (Lin)	Horizontally Adjacent Solid Tiles
	Enemy Count (EC)	Enemy Tiles

intended to be a heuristic for how restricted player movement is in the level. Empty space amount, and other similar block count or ratio based BCs have also appeared in prior works implementing ERA [45].

E. Linear Correlation

For every combination of compression algorithm and game domain BC we calculate the difference between the values for every pair of levels in each set. For the compression algorithm space this is the vector distance between the level pairs respective position in the compressed projection. For the BCs we take the absolute difference between the values for the level pair. We then calculate the linear correlation between these values for every level pair, using Spearman's ρ as we do not expect there to be a normal relationship between the two sets of values.

F. Number of Runs

To account for the influence of the random selection of levels for Super Mario and Boxoban, as well as the stochastic nature of T-SNE, ten runs are conducted for each game and algorithm.

G. Computational Resources Used

All experiments were run on an Intel i5-10310U CPU using a single thread and took approximately four hours to complete.

V. RESULTS

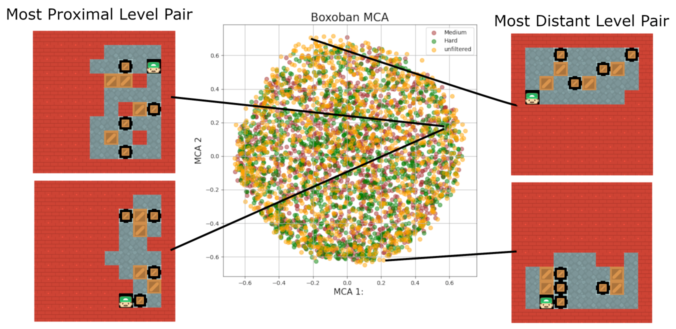

Fig. 1. Best Individual Boxoban Compression. T-SNE Run 9. Average BC Compression Correlation: 0.0380. Presented with the most proximal and most distant pair of levels in the compressed space

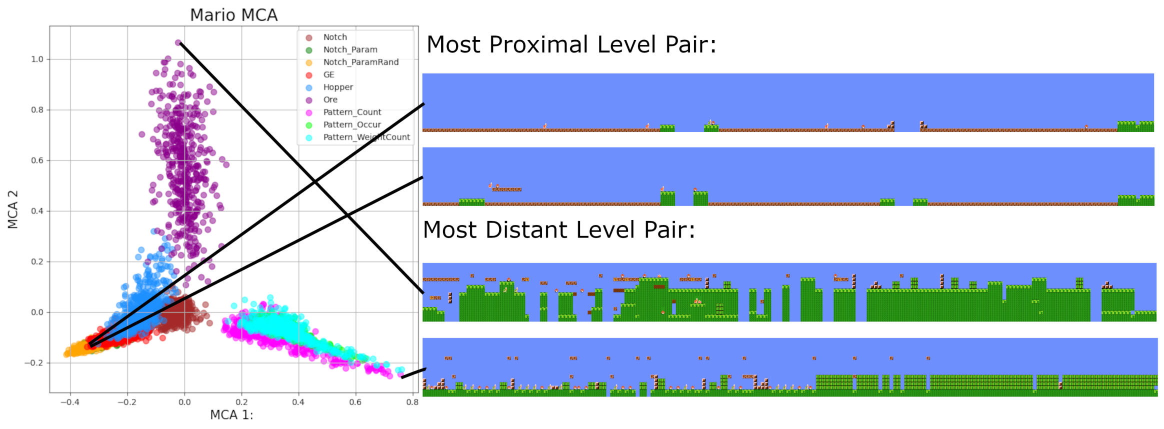

Fig. 2. Best Individual Super Mario Compression. MCA - Run 7. Average BC Compression Correlation: 0.497. Presented with the most proximal and most distant pair of levels in the compressed space

TABLE II
AVERAGE SPEARMAN'S RHOS AND ASSOCIATED P VALUES FOR EACH COMPRESSION ALGORITHM AND GAME BC. PRESENTED AS MEAN AVG +- STDDEV. BEST VALUES FOR EACH GAME BC HIGHLIGHTED IN BOLD

		PCA		SVD		MCA		T-SNE	
		Spearman's ρ	P Value	Spearman's ρ	P Value	Spearman's ρ	P Value	Spearman's ρ	P Value
Mario	ES	0.503±0.011	0±0	0.530±0.005	0±0	**0.765±0.003**	0±0	0.303±0.006	0±0
	Lin	0.417±0.005	0±0	0.382±0.006	0±0	0.497±0.006	0±0	**0.494±0.005**	0±0
	EC	**0.330±0.003**	0±0	0.301±0.002	0±0	0.295±0.004	0±0	0.288±0.006	0±0
Boxoban	ES	0.024±0.010	0±0	0.034±0.011	0±0	**0.049±0.010**	0±0	0.026±0.010	0±0
	Contig	0.019±0.010	0.017±0.052	0.032±0.010	0±0	**0.046±0.010**	0±0	0.020±0.009	0.011±0.034
Loderunner	ES	0.656±0	0±0	**0.818±0.000**	0±0	0.582±0.000	0±0	0.133±0.025	0±0
	Lin	0.440±0	0±0	**0.557±0.000**	0±0	0.440±0.000	0±0	0.046±0.032	0.036±0.115
	EC	-0.011±0	.266±0.001	0.008±0.000	0.408±0.0	0.008±0.000	0.390±0.005	**0.042±0.042**	0.121±0.260

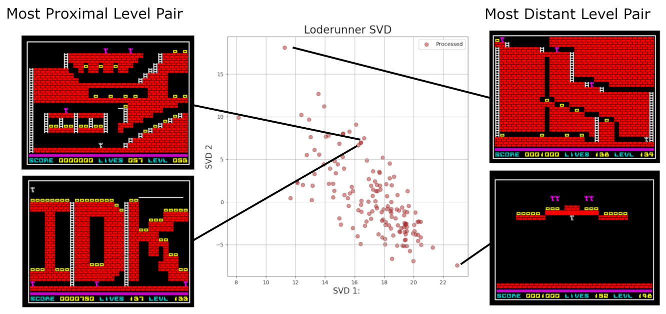

Fig. 3. Best Individual Lode Runner Compression. SVD Run 4. Average BC Compression Correlation: 0.461. Presented with the most proximal and most distant pair of levels in the compressed space

VI. DISCUSSION

Overall we would describe the experimental results as promising but inconsistent. Across all game domains the highest performing compression algorithms produced compressed spaces which correlated with the behavioral characteristics assessed. However, there were areas in which the approach struggled to produce meaningful compressions. For the Enemy Count BC for Lode Runner the approach was unable to produce compressed spaces that correlated despite the approach otherwise performing well in that domain, and in the Boxoban domain the approach under-performed across all BCs.

In terms of individual algorithm performance, MCA appeared to perform the most effectively out of the algorithms tested, producing the strongest correlations with BCs in 3 out of 8 domains. In this regard it only marginally outperformed SVD and T-SNE, which performed the best in 2 out of 8 domains. It also notably under-performed in specific domains, such as the Enemy Count BC for Mario, a phenomenon that requires more investigation to understand. In general while MCA performed the best, we would argue that the inconsistency of results should motivate more experimentation before alternatives are rejected.

Across the game domains assessed there was significant variance in performance of the approach. In both Super Mario and Lode Runner domains generative spaces which correlated substantially with BCs were found. In the Boxoban domain the performance for all algorithms and both BCs was significantly lower, with the best performing compression algorithm producing average Spearman's correlation coefficients of <0.05.

We believe there are two possible reasons for the Boxoban under-performance. Firstly, the levels may be too similar to each other. Though 5000 levels were assessed for each run all three sets come from a similar generative approach and share features such as having exactly four movable boxes and goals. This is supported by the generative space visualisation (Fig. 1) which indicates substantial overlap between the three level sets, in contrast with the Super Mario visualisation (Fig. 2) in which the levels from different generators occupied different spaces. The primarily difference between the three sets being their difficulty may simply not be present or detectable using the approach presented. The second possible reason is that the approach may perform worse in domains with smaller level encodings. Boxoban had the lowest number of tile types at only 5, and the levels were the smallest tested at only a 10 by 10 grid. More experimentation is required to explore whether there's a correlation between size of encoded level and the efficacy of the generative space compression.

Furthermore, the Boxoban results highlight a general weakness with the approach, which is the inability to detect structural similarities which appear in different locations of the level. To a human observer, the pair of levels which were flagged as being the most dissimilar in the compressed space (Fig. 1) appear to be substantially similar. However, as the similar structures appear in two different halves of the level they are incorrectly identified in the compressed generative space as being completely dissimilar.

As noted the approach also under-performed in the leniency BC within the Lode Runner game domain. We suspect this was a result of a combination of the relative paucity of input level data combined with the relative sparseness of enemies within the levels. The Lode Runner input set is both small at only 150 levels. Therefore it follows that the large scale structural differences between the levels would account for the majority of the variance within the set, leading to the low correlation with leniency due to the relatively low influence of enemy placement on the overall variance.

A. Limitations and Future Work

A primary limitation of this work is its focus on 2D tile-based games. While the focus on this domain made sense pragmatically due to the abundance of prior work we could use, tile-based games only make up a small portion of the contemporary games market. It is also limited by the small number of game domains assessed, and the fact that only the Mario level corpus contained sets from different generators. Future work could benefit from exploring this approach in domains with more complex content representations, with content produced from a wider range of similar and dissimilar generators. A first step could be applying it to the generative spaces of Minecraft map generators, as it is a popular 3D game whose maps are comprised of typed chunks.

A practical limitation of the approach presented is the requirement that every encoded level have the same encoded size. Future work could explore the approach used by Summerville and Mateas which used a graph cutting procedure to scale up the smaller Zelda levels to the size of the larger ones [14], though this necessarily involves creating new information that is not present in the original representation. Future work could usefully explore these and alternative methods for applying our approach to game levels of varying size.

As discussed when noting the aesthetic similarity of the two Boxoban levels flagged as being most dissimilar (Fig. 1), our approach can be weak in detecting structural similarities between levels. Future work could explore the use of alternative compression systems such as Convolutional Neural Networks that handle location independent structural similarity.

In this work we used correlations between the compressed spaces and BCs as the heuristic for whether the compressions were capturing useful information about the game levels. However, it would be valuable to explore this further with user studies to examine whether proximity between levels in compressed generative space correlates with player perceptions of the similarity of the levels. If they do correlate this would be robust evidence that the generative space compression retains useful information about the generative spaces.

In future work it will also be important to explore how best to make use of our generative space projections, ideally in cooperation with experienced level designers. While the original motivation for this project was to be able to assess the output diversity of a generative system, if the projections are sufficiently information rich and human understandable they could be more widely useful. For example, future work could explore their utility as the basis for mixed-initiative design tools, or automatic generator parameter tuners by allowing designers to target specific areas of the projected space.

VII. Conclusion

In this paper we have presented an approach for compressing the generative spaces of PCG systems using dimensionality reduction algorithms. The approach appears to be a promising basis for developing generative space visualisation and comparison tools as despite its simplicity it was able to produce generative space projections which correlated significantly with behavioral features. Of the algorithms we tested MCA performed the most reliably, but not by a margin that eliminates alternatives from further investigation. Though more work is required to confirm the efficacy of this approach in more complex spaces, we hope this work could form the basis for new qualitative tools for aiding designer understanding of the generative spaces of PCG systems.

Acknowledgments

This work was supported by the EPSRC Centre for Doctoral Training in Intelligent Games & Games Intelligence (IGGI) [EP/S022325/1]. Thanks to Marko Tot and Michael Saiger for their helpful advice on an earlier draft of this work.

References

[1] N. Justesen, R. R. Torrado, P. Bontrager, A. Khalifa, J. Togelius, and S. Risi. Illuminating Generalization in Deep Reinforcement Learning through Procedural Level Generation. [Online]. Available: http://arxiv.org/abs/1806.10729

[2] S. Risi and J. Togelius, "Increasing generality in machine learning through procedural content generation," *Nature Machine Intelligence*, vol. 2, no. 8, pp. 428–436, 2020.

[3] T. Galanos, A. Liapis, G. N. Yannakakis, and R. Koenig, "ARCH-Elites: Quality-diversity for urban design," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 2021, pp. 313–314.

[4] M. Guzdial *et al.*, "Tabletop Roleplaying Games as Procedural Content Generators," in *International Conference on the Foundations of Digital Games*, ser. FDG '20. Association for Computing Machinery, 2020.

[5] R. Khatchadourian, "World Without End," *The New Yorker*, 2015. [Online]. Available: https://www.newyorker.com/magazine/2015/05/18/world-without-end-raffi-khatchadourian

[6] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-Based Procedural Content Generation: A Taxonomy and Survey," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 172–186, 2011.

[7] G. Smith and J. Whitehead, "Analyzing the expressive range of a level generator," in *Proceedings of the 2010 Workshop on Procedural Content Generation in Games - PCGames '10*. ACM Press, 2010, pp. 1–7.

[8] M. Jadhav and M. Guzdial. Tile Embedding: A General Representation for Procedural Level Generation via Machine Learning. [Online]. Available: http://arxiv.org/abs/2110.03181

[9] Y. Zakaria, M. Hadhoud, and M. Fayek, "Procedural Level Generation for Sokoban via Deep Learning: An Experimental Study," 2021. [Online]. Available: https://tinyurl.com/yzbhyamm

[10] A. Sarkar and S. Cooper, "Generating and Blending Game Levels via Quality-Diversity in the Latent Space of a Variational Autoencoder," in *The 16th International Conference on the Foundations of Digital Games (FDG) 2021*, ser. FDG'21. Association for Computing Machinery, 2021.

[11] S. Thakkar, C. Cao, L. Wang, T. J. Choi, and J. Togelius, "Autoencoder and Evolutionary Algorithm for Level Generation in Lode Runner," in *2019 IEEE Conference on Games (CoG)*. IEEE, 2019, pp. 1–4.

[12] A. Sarkar, Z. Yang, and S. Cooper. Controllable Level Blending between Games using Variational Autoencoders. [Online]. Available: http://arxiv.org/abs/2002.11869

[13] A. Guez *et al.*, "An investigation of Model-free planning: Boxoban levels." [Online]. Available: https://github.com/deepmind/boxoban-levels/

[14] A. J. Summerville, S. Snodgrass, M. Mateas, and S. Ontañón. The VGLC: The Video Game Level Corpus. [Online]. Available: http://arxiv.org/abs/1606.07487

[15] S. Karakovskiy and J. Togelius, "The Mario AI Benchmark and Competitions," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 55–67, 2012.

[16] D. Perez-Liebana *et al.*, "The 2014 General Video Game Playing Competition," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 8, no. 3, pp. 229–243, 2016.

[17] A. Liapis, G. N. Yannakakis, and J. Togelius, "Constrained Novelty Search: A Study on Game Content Generation," *Evol. Comput.*, vol. 23, no. 1, pp. 101–129, 2015.

[18] D. Gravina, A. Khalifa, A. Liapis, J. Togelius, and G. N. Yannakakis, "Procedural content generation through quality diversity," in *IEEE Conference on Games 2019, CoG 2019*, ser. IEEE Conference on Computatonal Intelligence and Games, CIG. IEEE Computer Society, 2019.

[19] S. Mason, C. Stagg, and N. Wardrip-Fruin, "Lume: A system for procedural story generation," in *Proceedings of the 14th International Conference on the Foundations of Digital Games*. ACM, 2019, pp. 1–9.

[20] M. Charity, M. C. Green, A. Khalifa, and J. Togelius, "Mech-Elites: Illuminating the Mechanic Space of GVG-AI," in *International Conference on the Foundations of Digital Games*. ACM, 2020, pp. 1–10.

[21] M. C. Fontaine, J. Togelius, S. Nikolaidis, and A. K. Hoover, "Covariance Matrix Adaptation for the Rapid Illumination of Behavior Space," *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pp. 94–102, 2020.

[22] A. Alvarez, S. Dahlskog, J. Font, and J. Togelius, "Empowering quality diversity in dungeon design with interactive constrained map-elites," in *IEEE Conference on Games 2019, CoG 2019*, ser. IEEE Conference on Computatonal Intelligence and Games, CIG. IEEE Computer Society, 2019.

[23] M. Cook, J. Gow, G. Smith, and S. Colton, "Danesh: Interactive Tools For Understanding Procedural Content Generators," *IEEE Transactions on Games*, 2021.

[24] J.-B. Hervé and C. Salge, "Comparing PCG Metrics with Human Evaluation in Minecraft Settlement Generation," in *The 16th International Conference on the Foundations of Digital Games (FDG) 2021*, ser. FDG'21. Association for Computing Machinery, 2021.

[25] T. Smith, J. Padget, and A. Vidler, "Graph-based generation of action-adventure dungeon levels using answer set programming," in *Proceedings of the 13th International Conference on the Foundations of Digital Games*. ACM, 2018, pp. 1–10.

[26] A. Khalifa, M. C. Green, G. Barros, and J. Togelius, "Intentional Computational Level Design," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '19. Association for Computing Machinery, 2019, pp. 796–803.

[27] A. Summerville, "Expanding Expressive Range: Evaluation Methodologies for Procedural Content Generation," in *Fourteenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2018.

[28] A. Summerville and M. Mateas, "Sampling hyrule: Multi-technique probabilistic level generation for action role playing games," in *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2015.

[29] S. Snodgrass and A. Sarkar, "Multi-Domain Level Generation and Blending with Sketches via Example-Driven BSP and Variational Autoencoders," in *International Conference on the Foundations of Digital Games*. ACM, 2020, pp. 1–11.

[30] M. Turk and A. Pentland, "Eigenfaces for Recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[31] J. Liu, S. Snodgrass, A. Khalifa, S. Risi, G. N. Yannakakis, and J. Togelius, "Deep Learning for Procedural Content Generation," *Neural Computing and Applications*, vol. 33, no. 1, pp. 19–37, 2021.

[32] V. Volz, J. Schrum, J. Liu, S. M. Lucas, A. Smith, and S. Risi. Evolving Mario Levels in the Latent Space of a Deep Convolutional Generative Adversarial Network. [Online]. Available: http://arxiv.org/abs/1805.00728

[33] J. Schrum, J. Gutierrez, V. Volz, J. Liu, S. Lucas, and S. Risi, "Interactive Evolution and Exploration within Latent Level-Design Space of Generative Adversarial Networks," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, ser. GECCO '20. Association for Computing Machinery, 2020, pp. 148–156.

[34] S. Romdhani, S. Gong, and A. Psarrou, "A Multi-View Nonlinear Active Shape Model Using Kernel PCA," in *Procedings of the British Machine Vision Conference 1999*. British Machine Vision Association, 1999, pp. 48.1–48.10.

[35] S. Ayesha, M. K. Hanif, and R. Talib, "Overview and comparative study of dimensionality reduction techniques for high dimensional data," *Information Fusion*, vol. 59, pp. 44–58, 2020.

[36] L. Van Der Maaten, E. Postma, and J. Van den Herik, "Dimensionality reduction: A comparative review," *J Mach Learn Res*, vol. 10, pp. 66–71, 200.

[37] M. Greenacre and T. Hastie, "The Geometric Interpretation of Correspondence Analysis," *Journal of the American Statistical Association*, vol. 82, no. 398, pp. 437–447, 1987.

[38] L. van der Maaten and G. E. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[39] B. Horn, S. Dahlskog, N. Shaker, G. Smith, and J. Togelius, "A comparative evaluation of procedural level generators in the Mario AI framework," in *Proceedings of the 9th International Conference on the Foundations of Digital Games, FDG 2014, Liberty of the Seas, Caribbean, April 3-7, 2014*, M. Mateas, T. Barnes, and I. Bogost, Eds. Society for the Advancement of the Science of Digital Games, 2014.

[40] S. Racanière *et al.*, "Imagination-Augmented Agents for Deep Reinforcement Learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Curran Associates Inc., 2017, pp. 5694–5705.

[41] A. Guez *et al.*, "An investigation of model-free planning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2464–2473.

[42] N. Shaker, M. Nicolau, G. N. Yannakakis, J. Togelius, and M. O'Neill, "Evolving levels for Super Mario Bros using grammatical evolution," in *2012 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2012, pp. 304–311.

[43] A. Summerville, J. R. H. Mariño, S. Snodgrass, S. Ontañón, and L. H. S. Lelis, "Understanding mario: An evaluation of design metrics for platformers," in *Proceedings of the 12th International Conference on the Foundations of Digital Games*. ACM, 2017, pp. 1–10.

[44] O. Withington, "Illuminating super mario bros: Quality-diversity within platformer level generation," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*. ACM, 2020, pp. 223–224.

[45] C. Jemmali, C. Ithier, S. Cooper, and M. El-Nasr, "Grammar based modular level generator for a programming puzzle game," *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 2020.