Visual Datasets for Artificial Intelligence Agents

by

Erwin Hilton

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Masters of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

Author .....
Department of Electrical Engineering and Computer Science
May 25, 2018

Certified by .....
Tomaso Poggio
Eugene McDermott Professor at the Department of Brain and Cognitive Sciences
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Visual Datasets for Artificial Intelligence Agents

by

Erwin Hilton

Submitted to the Department of Electrical Engineering and Computer Science
on May 25, 2018, in partial fulfillment of the
requirements for the degree of
Masters of Engineering in Computer Science and Engineering

Abstract

In this thesis, I designed and implemented two visual dataset generation tool frameworks. With these tools, I introduce procedurally generated new data to test VQA agents and other visual AI models on. The first tool is Spatial IQ Generative Dataset (SIQGD). This tool generates images based on the Raven's Progressive Matrices spatial IQ examination metric. The second tool is a collection of 3D models along with a Blender3D extension that renders images of the models from multiple viewpoints along with their depth maps.

Thesis Supervisor: Tomaso Poggio
Title: Eugene McDermott Professor at the Department of Brain and Cognitive Sciences

Acknowledgments

I would like to thank Qianli Liao and to the members of CBMM for their support in this thesis. The datasets and tools implemented are the result of many conversations between Qianli and me throughout the year. I would also like to thank my mom for her support throughout my MIT career, and to all the friends near and far who have supported me throughout the entire process.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis presents the Spatial IQ Generative Dataset **(SIQGD)** system, along with the **ShapePerspective** framework. In this introductory chapter, I will explain the motivations for building SIQGD and ShapePerspective, two visual dataset generation tools, in the context of artificial intelligence research. Then I will provide a high level overview of both systems before delving into their details.

1.1 Motivation for creating visual dataset generation tools

The centuries old goal of artificial intelligence has been to develop computer systems that can reason and answer questions about the world around it at the same level as human intelligence [12].

What is intelligence? Intelligence is the ability to acquire knowledge and skills. Fundamentally, defining intelligence is a difficult task. Marvin Minsky famously called intelligence a *suitcase* word [8] - a word densely packed with many different definitions.

In order for researchers to assess a computer system's 'intelligence' as comparable to a human being's, researchers need access to standardized metrics of human cognitive skills and intelligence. Perhaps not surprisingly, the study of human cognition and intelligence is an even older field than the study of artificial intelligence. For the

last century and a half, psychologists have developed the field of psychometrics to measure humans' mental capacities and processes.

Psychometrics use standardized tests such as the IQ (Intelligence Quotient) test to assess cognitive abilities. While not a complete assessment of all types of cognitive ability, IQ tests have shown strong correlations as predictors of human cognitive performance and later success in life. There is also a strong correlation between the performance on IQ tests and performance on other cognitive tasks. This is known as the g factor [4], a particular measure of *general* intelligence.

SIQGD aims to provide artificial intelligence researchers with free access to a subset of quality psychometric evaluations that cognitive researchers and psychologists have already used for decades.

On the other hand, **ShapePerspective** gives access to an automated rendering tool and model repository useful for providing data for another spatial problem humans currently outperform machines in - 3D reconstruction.

1.2 Overview of SIQGD

SIQGD is a customizable framework for the generation of spatial intelligence reasoning problems. While users of SIQGD will normally only interact with the basic dataset generation script, under the hood is a system for procedurally generating a combinatorially enormous amount of images and visual question answering (VQA) questionnaires. We generate the dataset using combinations of the Raven's Progressive Matrices' rule set, a spatial IQ test, and modifications of the ground truth attributes within the Raven Progressive Matrices. We will provide more background information on Raven's Progressive Matrices and VQA in chapter 2, and explain many of the attribute modifications made within the matrices in chapter 3.

1.3 Overview of ShapePerspective

ShapePerspective is a Blender extension designed to allow users to easily take renders of objects from hundreds of perspectives and from various distances. ShapePerspective also includes an ever growing repository of 3D objects hosted in a Harvard Dataverse repository. All of the repository's 3D models are assigned a Creative Commons license, and can be freely used for research purposes.

Further background information on Blender and 3D reconstruction is provided in chapter 2, and we will further explain many of ShapePerspective's design decisions in chapter 4.

1.4 Thesis Structure

In chapter 2, I briefly explain many of the concepts and background information necessary to understand my thesis work.

In chapter 3, I explain the logic behind SIQGD and many of the design decisions that drive the framework.

In chapter 4, I describe ShapePerspective which includes the extension coded for Blender and the repository of models compatible with the extension.

In chapter 5 I review my contributions to the field of AI research as a whole, and make recommendations for future work.

We will now dive into the details and research decisions of both SIQGD and ShapePerspective.

Chapter 2

Background Information

In this section we will provide background information to provide context for some of the concepts and technologies used in our dataset generation. Sections 2.1-2.3 focus on providing background for SIDG, while section 2.4-2.5 explains the core technology behind ShapePerspective.

2.1 Visual Question Answering (VQA)

Multi-disciplinary research problems have gained popularity in recent years as the computer vision research community has progressed beyond basic recognition tasks. Visual question answering is a field of research proposed by [9] that combines computer vision, natural language processing, knowledge representation and reasoning.

The VQA method involves giving an artificial intelligence model on an image and a natural language question about the image, and the task is to provide an accurate natural language answer.

Correctly answering the questions posed by the VQA task involves many perceptual abilities such as recognizing image context, objects, attributes, and spatial relationships between objects in the image. VQA also involves the use of higher level skills such as counting, logical inference, comparing, and using real world common sense knowledge.

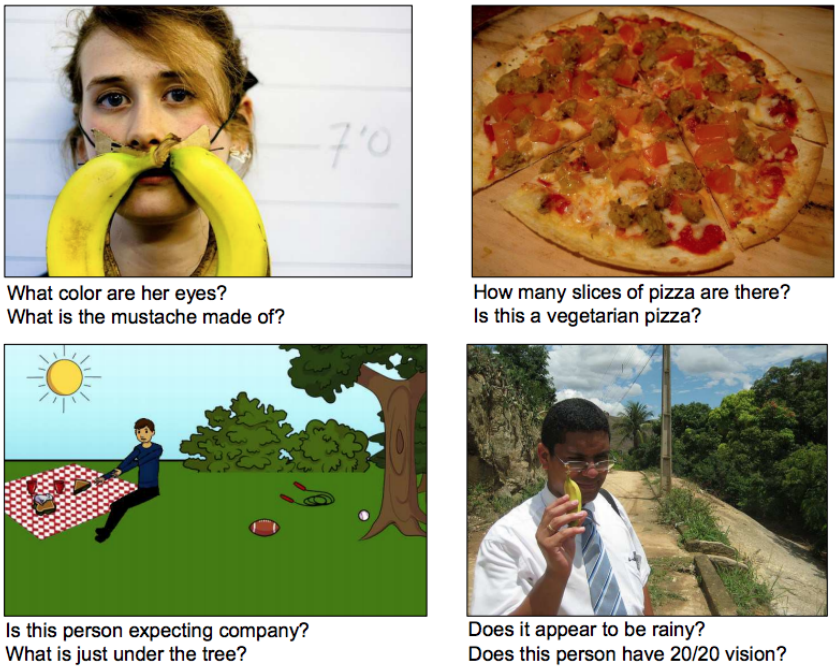

Figure 2-1: Examples of visual question answering problems.

2.2 Psychometric Spatial IQ Tests

2.2.1 Definition and Usage

Currently, psychometric tests are used to evaluate cognitive ability over a wide range of abilities. The definition of intelligence is thus reduced to excelling at a group of established intelligence tests. Evaluating intelligence by a set of intelligence tests also extends to AI agents, as Selmer et al showed [1]. If the various aspects of intelligence are split up and evaluated using different tests, then the development of human-level intelligence AI models can be encouraged by dividing and conquering different intelligence aptitude tests. This goes beyond the Turing Test, which focuses on the *appearance* of intelligence, and instead evaluates each currently known and previously tested aspect of intelligence.

Thus, by developing freely available psychometric datasets for artificial intelligence researchers, we can use SIQGD to evaluate part of the nonverbal analytical 'intelligence' of a machine by the same metrics humans are evaluated at.

2.2.2 Raven's Progressive Matrices (RPM)

Raven's Progressive Matrices are a nonverbal IQ test designed to test analytical reasoning. They are the most popular IQ test administered to demographic groups ranging from children to the elderly [6]. Raven's Progressive Matrices has arguably gathered more attention in the cognitive literature than any other psychometric measure of fluid intelligence, largely because it is an induction task that can be modeled computationally [3][13].

Raven's Progressive Matrices is designed to minimize relying on declarative knowledge from prior education and experience. Raven's Progressive Matrices are thought to be a good metric for *fluid intelligence*, the ability to deal with novel problems and adapt one's thinking to a new problem. This is in contrast to testing for *crystallized intelligence*, which are skills and knowledge acquired through previous experience.

Beyond testing for human analytical skills, we believe these kind of fluid intelligence metrics are ideal for testing artificial intelligence agents as they require no prior knowledge. Any model trained to solve Raven's Progressive Matrices would only be constrained by the quality of the training set.

Previous research has found that five main rules govern the variation in the entries in the Raven's Progressive Matrices examination [3]. The set of rules governing the variation in a problem is not unique, sometimes different combinations of the rules are interchangeable and can explain how a problem was derived. These rules are described as follows:

1. **Constant in a row** - the same categorical attribute occurs in the same row, but changes down a column. For example, one row could be made up of squares, the second circles, and the third diamonds.

2. **Quantitative Pairwise Progression** - a quantitative increment (size, position, number, etc) occurs between adjacent pairs of images. In this case, using the example for constant in a row would be the shapes gradually scale down as you go along each element in the row.

3. **Distribution of Three** - three values from a categorical attribute, such as the shapes used, are distributed throughout a row. For example, each row could contain three different shapes but the permutation changes for each row.

4. **Figure addition or subtraction** - a figure from one column is added or subtracted from a figure in another column to produce a new figure.

5. **Distribution of Two** - three values from a categorical attribute are distributed throughout a row, but for each image only a subset of two values are used while the third value goes unused.

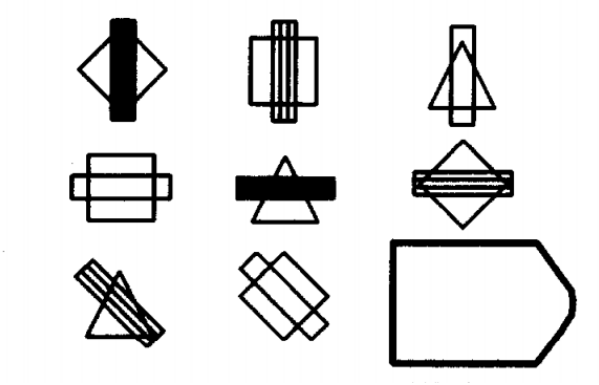

Figure 2-2: Example of a Raven's Progressive Matrix, from AuthorCarpenter. The task is to identify the best possible candidate figure for the space in the bottom right of the image, out of a set of multiple choice candidate figures. The rules used in the creation of this sample Raven's Progressive Matrices are the **distribution of three** and **constant in a row** rules. The geometric shapes used in each row (diamond, square, triangle) follow a distribution of three. The texture of the bars intersecting the geometric shapes also follow a distribution of three (solid, striped, or clear). The rotation of the bar follows the constant in a row rule (vertical, horizontal, or diagonal).

2.3 Boolean Operations on Polygons

As mentioned in the rules listed above, constructing a generative dataset based off of Raven's Progressive Matrices requires a way to produce figure addition and subtraction. We solve that by implementing the F.Martinez algorithm introduced in [7] for computing Boolean operations on polygons.

"Boolean operations on polygons" is synonymous with the application of the logical boolean operators (AND, OR, NOT, XOR, etc) on a set of polygons. As shown in the figure below, each operation corresponds to graphical transformations of the polygons.

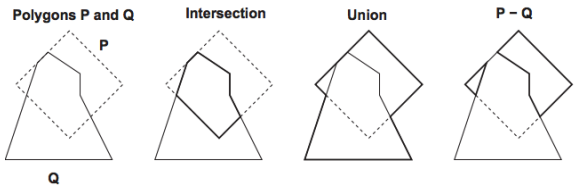

Figure 2-3: Boolean operations on a polygon. The logical operation P **AND** Q corresponds to the intersection of the two polygons, P **OR** Q to the union, and P-Q to P **NOT** Q. Not shown are Q-P (Q **NOT** P, the opposite of P-Q), and P **XOR** Q (everything but the intersection of the two polygons).

2.4 3D Reconstruction

Three dimensional reconstruction is the ability to infer and reproduce 3D object shapes using a variety of data such as images, depth maps, silhouettes, and point clouds.

While it is a difficult problem, learning how to deduce the shape of a 3D object from a single image is necessary for human level scene understanding. Humans and other animals have the unique ability to distinguish a variety of objects in a scene from different angles using just one or a few sample images.

Three dimensional model reconstruction of real world objects is a subset of computer graphics and computer vision research problems that has seen a good amount of recent progress by artificial intelligence researcher community [10][2][11].

There is always the need for more data, and it is the purpose of ShapePerspective to provide a useful tool for the production of image data content. Online 3D model repositories continue to grow on a daily basis and a revolution in scanning methods is creating increasingly faithful 3D reconstructions of real environments. As described below, ShapePerspective capitalizes on the ever growing community of Blender artists

and content makers as a resource to use for the artificial intelligence community.

2.5 Blender

Blender is a free and open source application for 3D creation. Blender supports the entire 3D pipeline - modeling, animation production, simulation, rendering, model rigging, rendering, compositing, motion tracking, video editing, and game creation. It uses the OpenGL library to display its interface, which allows it to be used on many platforms and operating systems.

Blender has a large online community made up with artists of different skill levels. Many of these artists provide their 3D models online for free or for a price. As the community grows ever larger, there is an increasing amount of free content becoming available.

2.5.1 Blender Extensions

Due to its open source nature, Blender has a flexible Python controlled interface. Blender has a functional Python API that allows developers to automate and customize a lot of the functionality in the 3D creation suite. Hundreds of add-ons and extensions designed by the community of Blender users are available for download.

ShapePerspective is an add-on that has been created to automate the process of generating content from Blender 3D models, and is freely available for researchers to use.

Chapter 3

SIQGD Architecture

In this section, we introduce the architecture of SIQGD. We first give a high level overview of SIQGD and of the software that generates the image dataset. We then describe each of the components of the SIQGD system in greater detail.

3.1 Architecture Overview

The SIQGD architecture is comprised of a procedural image and text generation system that generates images in the fashion of Raven's Progressive Matrices questions along with VQA questionnaires about each of the RPMs generated.

The image generation system uses the set of rules previously described in section 2.2.2 as modifiers to generate unique RPMs. A unique rule set is generated for the creation of each generated Raven's Progressive Matrix. Along with a unique rule set, attributes are chosen as parameters that the rules conditionally modify or statically set. While duplicate RPM images in the dataset are possible and unaccounted for, they are highly unlikely as the amount of combinations possible given the number of rules and attributes available for the generation of each RPM is a nontrivial amount.

The text generation system generates a unique VQA questionnaire about the generated RPM image.

3.2 Image Generation

3.2.1 Python Imaging Library (PIL)

The Python Imaging Library is a free library useful for opening, manipulating, and saving many different image file formats. SIQGD uses PIL to draw and save the images generated for the dataset.

3.2.2 Shapes

Objects in the image are defined by a **Shape** class comprised mostly of polygons and lines. The shapes are randomly selected from a set of arbitrarily determined initial shapes. The shape class contains many of the attributes that describe the shape, such as the color, number of sides, orientation, and center. Consequently, the **Shape** class also contains many of the methods used to modify the attributes of each shape.

3.2.3 Complexity

The difficulty of a generated RPM is assigned according to a complexity metric $C \in [1, 2, 3]$, which corresponds to the amount of rules used to generate the RPM. For example, if $C = 2$ then any combination of two of the rules in section 2.2.2, allowing for duplicates, can be used to generate the RPM.

While this is an arbitrarily set limit to the amount of rules that can be used to generate a matrix, the current complexity range can easily be modified in the code through the DatasetGenerator class. The range is set to $C \in [1, 2, 3]$ as realistically there is a limit to the amount of rules used to generate a matrix in an RPM test administered to humans.

3.2.4 Rule Generation

The rules are generated respective to the rules described in section 2.2.2.

As mentioned in the rule set description, each of the rules depended on selecting categorical attributes and modifying quantitative values of these attributes. Only a

subset of the categorical attributes are able to be quantitatively modified.

The categorical attributes used in the RPM generation are *independently determined* for the rules 'constant in a row', 'distribution of three', 'distribution of two', and 'figure addition and subtraction'. The categorical attributes are listed as follows:

- **Shape** - a 2d form such as polygons, circles, ellipses, arcs, lines, etc
- **Size** - the shape is set to a certain scale
- **Count** - the amount of similar shapes in the same figure
- **Color** - the color of the shape. The set of colors is denoted by $C = [white, black, blue, red, green, yellow]$
- **Orientation** - the orientation (set rotation) of the shape
- **Position** - where the shape is positioned relative to its background. The background of the shape is constrained to its sub-section of the RPM, where a standard RPM has nine separate sections divided into a 3x3 matrix.

Below is the set of applicable quantitative transformations for the rule 'quantitative pairwise progression', which is slightly different from the other rules as it requires a *conditional change between images* for each attribute:

- **Scaling** - increase or decrease in the size of the shape, where the scale factor $S \in [-1.5, 1.5]$
- **Tally** - the amount of shapes increases or decreases (duplicates or very similar shapes with slight transformations applied). The tally of shapes is a number in $T = [1, 2, 3, 4]$
- **Color change** - the color of a shape changes, to any other color in the set of colors C
- **Rotation** - the shape can rotate around its orientation by an angle increment. The possible angle increments range in $R \in [0, 2\pi)$

- **Polygonal side count change** - if the shape is a polygon, the amount of sides can be incremented or decremented. A polygon's side count is constrained to a number in $P_c = [3, 4, 5, ..., 9]$
- **Translation** - the shape moves to a new position

A matrix M of nine different figures is generated using these rules and attribute modifiers, in the style of Raven's Progressive Matrices.

The matrix M has the form

$$\begin{matrix} \mathrm{M}_{1,1} & \mathrm{M}_{1,2} & \mathrm{M}_{1,3} \\ \mathrm{M}_{2,1} & \mathrm{M}_{2,2} & \mathrm{M}_{2,3} \\ \mathrm{M}_{3,1} & \mathrm{M}_{3,2} & ? \end{matrix}$$

The figure in the bottom right of the matrix, at indice $M_{3,3}$ =?, is the missing component of the RPM left as a task for the examinee (and artificial intelligence model) to predict. While still generated as an answer using the ruleset, $M_{3,3}$ is omitted from the *question image* and replaced with a black square in its place.

Instead, the correct solution to $M_{3,3}$ is included in a separate set of candidate images that comprise an answer key. Three other images created using incorrect yet similar variations of the rules and categorical attributes are included in the answer key. **Figure 3-1** shows a sample RPM generated by our code, along with the corresponding answer that correctly fills in the black square.

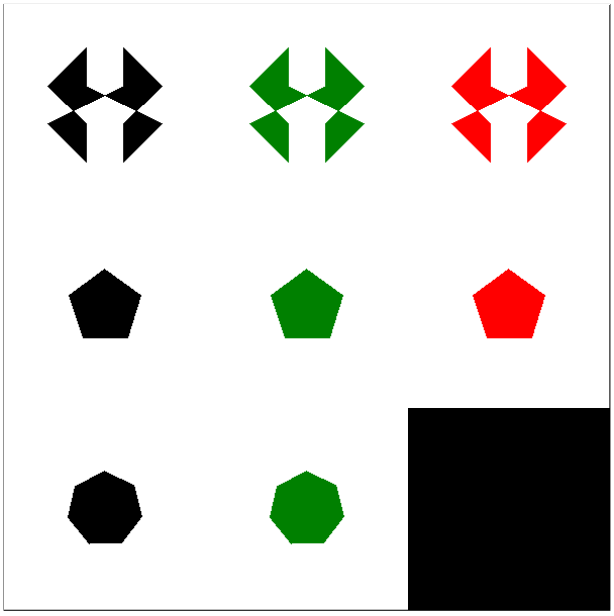

Figure 3-1: A generated RPM using our code. This RPM employs the **quantitative pairwise progression** and **constant in a row** rules. Quantitative pairwise progression is applied through the color change between each item in the row (each shape goes from black to green to red). Setting a single, constant shape for each row employs the 'constant in a row' rule. The correct answer that fills in the bottom right corner is a red heptagon as shown in the answer image in Figure 3-2.

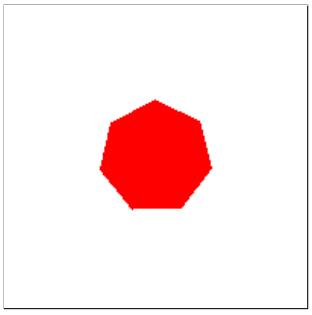

Figure 3-2: Answer to the Raven's Progressive Matrices, a red heptagon.

3.2.5 Answer Key

Along with the Raven's Progressive Matrices, an answer key is generated with the data. As stated before, the answer key contains four different images, among them is the real answer image. The answer key is assembled by including the correct answer image, and constructing the other three possible (erroneous) answers by replacing or

omitting one of the rules and/or attributes of the answer image.

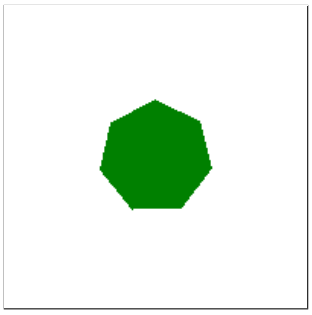

Figure 3-3: Example of a wrong answer generated with the answer key, a green heptagon. It is incorrect as while the heptagon follows the **constant in a row** rule correctly, it does not apply the **quantitative pairwise progression** rule (the change in color from green to red).

3.3 VQA Questionnaire

3.3.1 Objects and Relationships

The SIQGD universe is composed of all of the shapes declared in the Shapes class, which includes many polygon shapes (an infinite amount of n-sided regular polygons along with some arbitrary polygons). Each object can come in one of many combinations of different colors, sizes, rotation, and positions within the Raven's Progressive Matrix image.

Questions are generated by relating the ground-truth attributes within the answer image with the adjacent images, and between the objects inside the answer image. Relationships between adjacent images include comparing orientation, colors, number of sides between shapes, and so on. Relationships unique to the answer image include comparing shapes inside the answer image (if there is more than one shape),

Objects are spatially related via four relationships - a shape can either be 'above', 'below', 'to the right of', and 'to the left of' another object. Thankfully, as this dataset's universe exists only in the 2D world, constructing questions that follow accurate semantics of the spatial relationships of these objects is simpler than relating positions in a 3D world.

3.3.2 Question Families

SIQGD takes a page from the CLEVR dataset [5] through the method with which it converts the objects and relationships in an image to natural language. SIQGD uses several text templates that provide multiple ways to compose unique questions for an image. In order to minimize question-conditional bias, we implement **question families**, where each question family is composed of several text templates that provide multiple ways of expressing a question in natural language.

For an example of a text template, the question "How many 7-sided shapes are in the answer image?" can be composed by instantiating from the text template "How many <S> <C> shapes are in the answer image?", binding the parameters <S> and <C> (corresponding to types 'number of sides' and 'color') to the values *7* and nil. Similarly, another way of phrasing the same question as a separate text template in the question family is "In the answer image, how many <S><C> shapes are there?.

Position	Color	Side Count	Tally	Orientation
<P>	<C>	<S>	<T>	<O>

Table 3.1: The question family parameter types and their corresponding parameter character expressions

Currently, there are only a few question families along with their corresponding answer generator program templates. Even so, a small number of question families can generate a huge number of unique questions. This provides VQA researchers with a huge amount of data. SIQGD can easily be extended by adding new question families.

Below are examples of possible question family text templates and corresponding questions that can be generated for a matrix in the dataset.

- *How many <S1> shapes are <P> of the <S2> shape in the answer image?*

  How many 3-sided shapes are to the right of the green 7-sided shape in the answer image?

- *What orientation is the <C1> shape in the answer image relative to the <C2> shape in the image <P> of the answer image?*

  What orientation is the red shape in the answer image relative to the blue shape in the image to the left of the answer image?

- *How many more sides does the <C1> shape have than the <C2> shape in the answer image?*

  How many more sides does the green shape have than the yellow shape in the answer image?

- What set of rules were used to create the answer image?

  (General question, with one set of rules as an answer)

3.3.3 Question Generation

Generating a question for a matrix image is simple: a question family is chosen, values are selected for each of the question template parameters, one or more of the text templates from the question family are filled in the selected values, and the corresponding answer generator program is executed to find the answer.

The questions are machine-readable in the form of a generated text file in each of the generated Raven's Progressive Matrix set's folder as `vqa_questions.txt`. The respective answers are saved in `vqa_answers.txt`.

Chapter 4

ShapePerspective Architecture

In this section, we introduce ShapePerspective's architecture. We first give a high level overview of ShapePerspective. We then describe the workflow of a user using the ShapePerspective system and the Blender extension used to generate a dataset for each model in ShapePerspective's dataverse repository.

4.1 Architecture Overview

ShapePerspective is a system that is meant to be easy and helpful to use for researchers. Holistically, ShapePerspective is comprised of a 3D model repository of Blender files hosted in Dataverse along with a Blender extension called ShapePerspectiveRenderer that makes use of Blender's Python API to automate the process of taking renders of the 3D models.

The Blender extension has modifiable parameters that include choosing how close the render's camera is relative to the 3D model, the amount of pivot points (vertices the camera points to and renders in its view), and the amount of images rendered per pivot point.

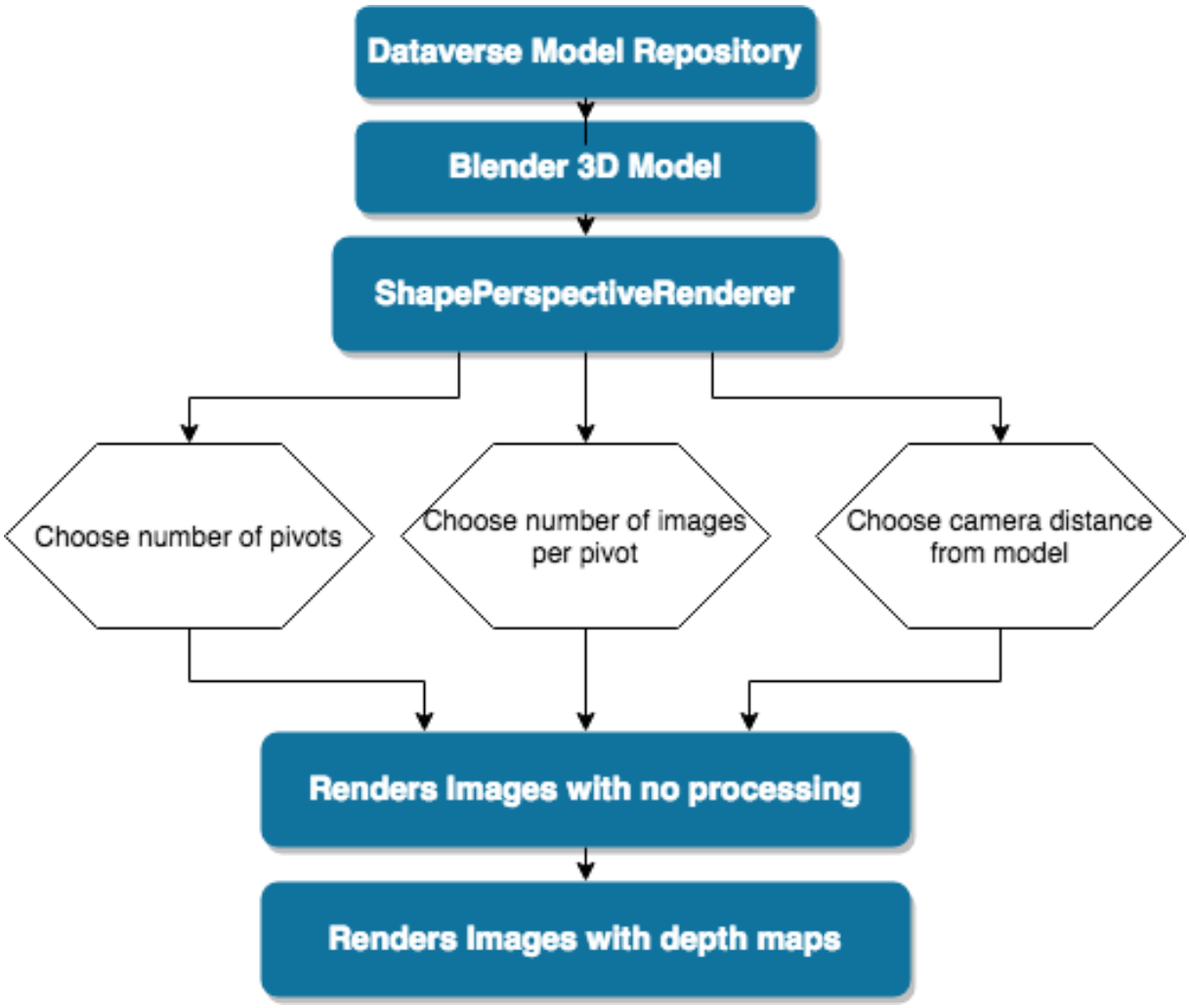

Figure 4-1: General workflow of the ShapePerspective dataset. Users access a 3D model from the Dataverse repository, then render and save the images using ShapePerspectiveRenderer.

4.2 Dataverse 3D Model Repository

The repository of 3D models is hosted using Harvard's Dataverse. The repository is built through careful curation of freely available Blender models on the internet.

All of the models have been vetted as licensed under a Creative Commons license (or equivalent) and work with ShapePerspectiveRenderer's rendering. The vetting process requires inspecting ShapePerspectiveRenderer's functionality with the model, as some Blender files can contain entire scenes with multiple objects or other extraneous data that interferes with rendering the object of interest inside the scene.

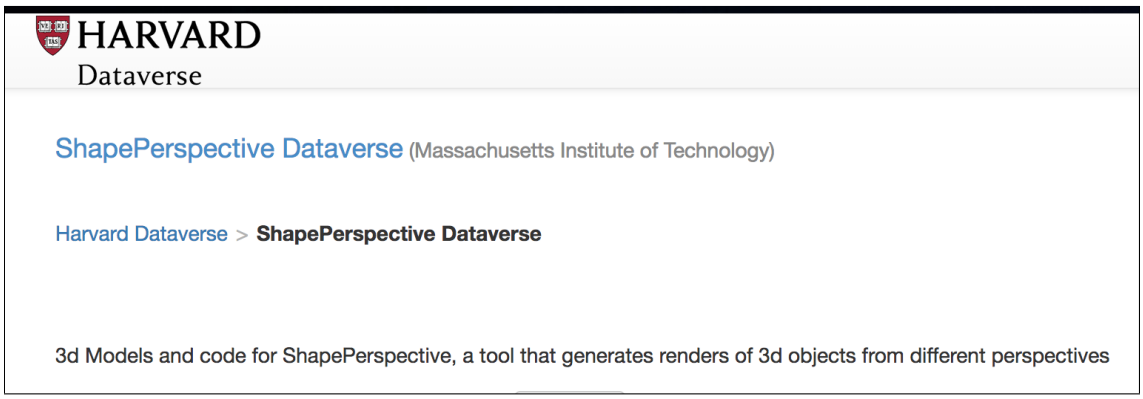

Figure 4-2: Models compatible with the addon are stored in a Dataverse repository.

4.3 ShapePerspectiveRenderer

We will now detail some of ShapePerspectiveRenderer parameter customizability as a Blender extension and the process through which the extension renders images.

4.3.1 Pivot Points

ShapePerspectiveRenderer starts off by selecting pivot points. The render camera will be focused on a pivot point for all rotations around the object model. ShapePerspectiveRenderer selects the pivot points according to the parameter **pvFractionOfVerts**, with which it randomly samples a corresponding fraction of the 3D model's vertices as pivot points.

4.3.2 Rendering

After the pivot points are selected, the renderer iterates over each of the pivot points. For each pivot point, the render camera is directed at that pivot in a line that passes through the center of the model. The render camera always maintains a set distance away from the pivot along that line, set by the variable **cameraDistance**.

The model is then rotated along each of its x, y, z axes. The model is fully rotated at an even interval determined by the cube root of the total amount of renders that will be taken for a given pivot point. For each rotation, the camera stays at the set **cameraDistance** distance from the pivot point even if the pivot point is occluded

from the camera's persepective. This cubic root, and therefore the amount of renders taken per full rotation around an axis, is set by the variable **stepCount**.

Two renders are taken at every rotation. One render has no processing applied, and is the 3D view of the object as it would be seen in a scene. The other render is a depth map of the object relative to the render camera's perspective.

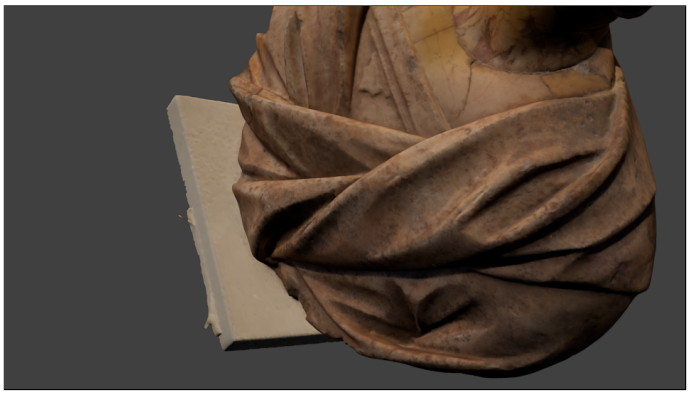

Figure 4-3: A single render with no processing of a 3D model.

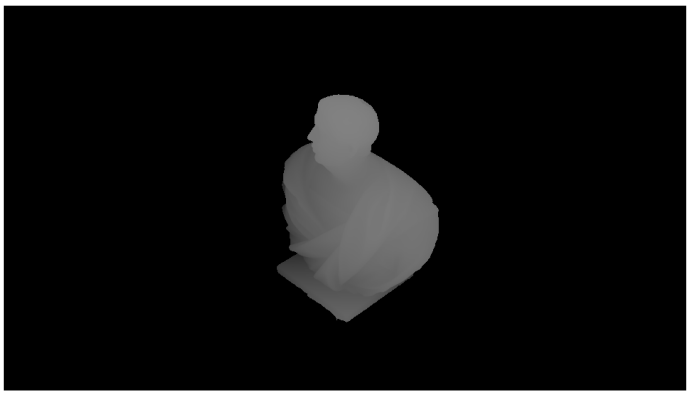

Figure 4-4: A single rendered depth map of a 3D model.

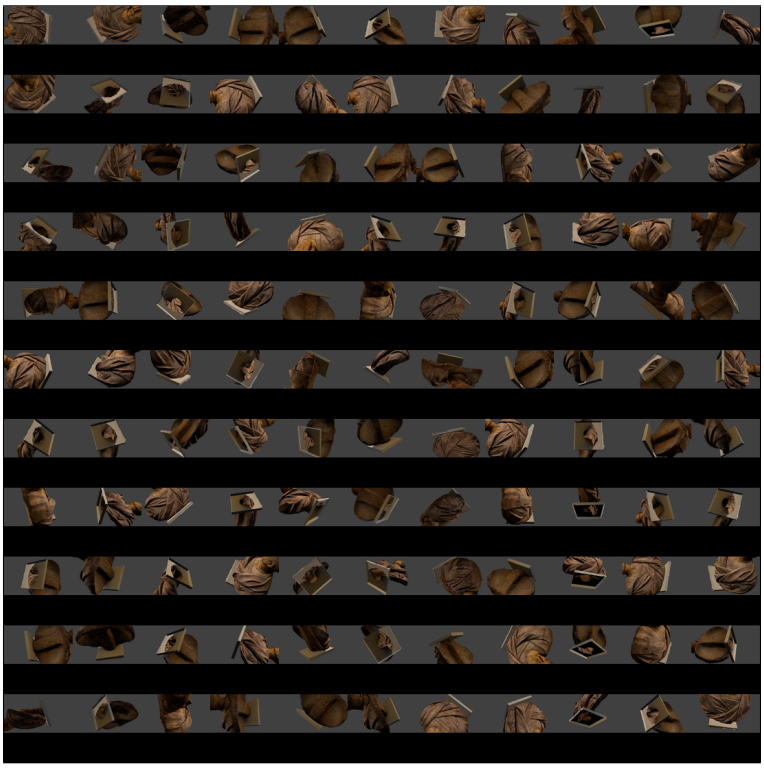

Figure 4-5: Stitched image of all the renders taken for a single pivot around one 3D Blender model. In total, there are nine pivot points, or about a fifth of the total amount of vertices of this model (by default, 20 percent of vertices are randomly sampled as pivots) . Both the fraction of vertices taken as pivots and the number of images taken per pivot are customizable parameters.

Chapter 5

Conclusion

5.1 Future Work

SIQGD is still in the process of expanding its VQA question family and text template framework. A useful next step would be the crowdsourcing of question family and text template creation as an open-ended task for the VQA research community. The more question families and text templates in the dataset, the less question-conditional bias will be present in the questionnaires generated. Additionally, more objects such as irregular polygons and modifiable attributes can be added to SIQGD in order to expand the universe of possible Raven Progressive Matrices generated.

ShapePerspective Dataverse is still expanding, and further scraping of freely available 3D models on the web is necessary to provide a deeper and richer taxonomy of models. A breadth of objects akin to the variety of objects humans interact with on a day to day basis is needed in order for artificial intelligence models to effectively train their visual 3D reconstruction abilities to the same level as a human's. Useful further additions to the Blender extension could include the addition of full object scenes, object masks, surface normal, and additional processing through the rendering tool.

The SIQGD and ShapePerspective frameworks requires further analysis from researchers to implement and train models on each of the datasets.

5.2 Contribution

For this M.Eng thesis, we developed two unique visual datasets accessible for artificial intelligence researchers to train and test their models on. Both of these datasets test unique visual cognition skills.

We developed SIQGD as a generative way of creating visual question answering problems. We designed a customizable way for VQA researchers to modify the natural language with which the questions are expressed through question family text templates. We also designed, developed, and vetted ShapePerspective as a 3D model repository and tool for the creation of 3D model image datasets. Lastly, we defined next steps to promote and augment the usefulness of both datasets for the artificial intelligence research community.

Appendix A

Dataset Links

A.1 SIQGD Link

The link to the SIQGD Github repository, which includes the image and VQA text generation tools.

https://github.com/throwninlie/IQ-spatial-dataset

A.2 ShapePerspective Link

The link to the ShapePerspective Github repository, which includes Blender extension tool.

https://github.com/throwninlie/ShapePerspective

A.3 ShapePerspective Dataverse

The 3D models hosted on Harvard Dataverse, freely available for research purposes, can be found at:

https://dataverse.harvard.edu/dataverse/shapepersp

Bibliography

[1] Selmer Bringsjord and Bettina Schimanski. What is artificial intelligence? Psychometric AI as an Answer. 2003.

[2] M. Brown and D. G. Lowe. Unsupervised 3d object recognition and reconstruction in unordered datasets. 2005.

[3] Patricia A. Carpenter, Marcel A. Just, and Peter Shell. What one intelligence test measures: A theoretical account of the processing in the raven progressive matrices test. 1990.

[4] Arthur R. Jensen. The g factor. the science of mental ability. 1998.

[5] J. Johnson, L. Fei Fei, B. Hariharan, C.L. Zitnick, L. van der Maaten, and R. Girshick.

[6] R. M. Kaplan and D. P. Saccuzzo. Standardized tests in education, civil service, and the military. psychological testing: Principles, applications, and issues. 2009.

[7] F. Martinez, A. J. Rueda, and F.R. Feito.

[8] Marvin Minsky. The emotion machine. 2006.

[9] J. Lu M. Mitchell D. Batra C. Zitnick S. Antol, A. Agrawal and D. Parikh. Vqa: Visual question answering. 2015.

[10] A. A. Soltani, H. Huang, J. Wu, T. D. Kulkarni, and J. B. Tenenbaum. Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networksy. 2017.

[11] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Multi-view 3d models from single images with a convolutional network. 2016.

[12] A.M. Turing. Computing machinery and intelligence. 1950.

[13] Tom Verguts and Paul De Boeck. The induction of solution rules in ravenâĂŹs progressive matrices test. 2002.