Synthesising Abstract Deep Computer Vision Pretraining Data

Søren Winkel Holm, s183911

December 17, 2022

Contents

1 Introduction

In the field of computer vision, deep learning is often used to solve problems such as object detection and image segmentation. Two main problems are (1) the high cost of annotating the image by producing the label maps (such as this) (2) the high cost of training the deep learning models. Automatic generation of scenes with corresponding label maps might be usable even if the scenes are abstract. Such data could be used to pretrain the deep learning model and act as an inductive bias such that less training and less data is required when actually training on the real dataset. For this to work, the generated scenes should have objects with object having some assigned class which determines some visual characteristics of the object. The scenes do not have to have anything to do with the actural task but need to make the model learn som basics of object detection, classification and segmentation. An easy first solution would be having the class determine the colour of the object but it is much more realistic if the objects are textured with noise functions and the parameters of the procedurally generated noise were determined by the class.

2 Methods

2.1 Synthetic Scene Rendering

Task 1: Noise Functions and The Simple Scene In the fragment shader, two noise functions were implemented, both accepting three-dimensional vector input.

1. Improved Perlin noise (PN) using a permutation polynomial as McEwan, Sheets, Richardson, and Gustavson [McE+12]. 3D PN chooses pseudo-random gradients $\in \mathbb{R}^3$ at regular grid points in each axis, interpolating a smooth function consistent with the gradients [Gus05].

2. Sparse convolution noise (SC) as Frisvad and Wyvill [FW07]. SC targets minimal artifacts or patterns by approximates the computationally expensive white noise filtered with a Gaussian kernel using a cubic kernel instead.

Changing magnitude of the input vector – the point at which the noise landscape is queried – is equivalent to controlling the zoom level of the noise texture. The implementations were largely based on reference implementations [Fri16]: Only minor code

changes and commenting were performed along with using the functions to texture the scene objects with a given noise scale.

An initial scene was constructed by rendering a coloured sphere through 5-fold recursive subdivision of tetrahedra. The sphere was shaded using Phong shading from a positional light behind a perspective camera. The sphere was textured using the noise functions evaluated at normals scaled with a free parameter called noise scale. This texturing was perform by letting the noise intensity controlling the diffuse light intensity, see Listing 1.

Behind the sphere, a 2D noise textured background in black and white was placed.

The scene and noise functions can be inspected with user control at `file://` `/home/sorenmulli/Nextcloud/cand2/computer-graphics/afl/webgl-site/` `t1.html`.

Listing 1: Fragment shader snippet related to texturing of foreground objects

```
// Use normal vectors to get smooth noise over sphere
vec3 p = vec3(noiseScale * n.x, noiseScale * n.y, noiseScale * n.z);
float noise = do_sc ? scnoise(p) : pnoise(p);
float noiseIntensity = 0.5 * noise + 0.5;

// Phong lighting using noise intensity for diffuse radiance
float d = max(dot(normalize(n), normalize(w_i)), 0.0);
vec3 w_h = normalize(w_i + w_o);
vec3 lightSpecular = ks * Le * pow(max(dot(normalize(n), w_h), 0.0), s);
vec3 lightDiffuse = d * kd * Le * noiseIntensity;
vec3 lightAmbient = ka * Le;

gl_FragColor = vec4(lightDiffuse + lightAmbient + lightSpecular, 1.0);
```

Task 2: Random Scenes Add multiple spheres to the scene, randomly selecting placement and size for each.

Task 3: Object Classes Choose a number of classes (maybe 3) and give each class a distribution of shapes, of colours and a distribution of noise parameters (such as related smoothing and scale). Assign each sphere to a class and let its shape, colour and noise be sampled from the class prior. (I will be using spheres and for shape I will use a model matrix that transforms the sphere into an ellipsoid by stretching it. The amount of stretching should thus be determined by object class)

Task 4: The Full Scene With Label Maps Add the possibility of producing the label image by having a rendering mode which does not have any lighting or texturing, just colours each object according to its class. This image will then have the class in

the pixel position according to the object at that pixel position. Allow for saving the current canvas to a png.

Generated Datasets

2.2 Deep Image Segmentation

Baseline Learning Task

Addition of Synthetic Pretraining Data

3 Results

4 Discussion

1. Powerful benchmark for different methods generalization towards different difficulties: Explainability

References

[Fri16] Jeppe Frisvad. *Exploring noise functions with WebGL*. Online. Accessed: 16/12 2022. 2016. URL: https://people.compute.dtu.dk/jerf/code/noise/.

[FW07] Jeppe Revall Frisvad and Geoff Wyvill. "Fast High-Quality Noise". In: *Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia*. GRAPHITE '07. Perth, Australia: Association for Computing Machinery, 2007, pp. 243–248. ISBN: 9781595939128. DOI: 10.1145/1321261.1321305. URL: https://doi.org/10.1145/1321261.1321305.

[Gus05] Stafan Gustavson. *Simplex Noise Demystified*. Online. Accessed: 16/12 2022. Sweden, 2005. URL: http://www.itn.liu.se/~stegu/simplexnoise/simplexnoise.pdf.

[McE+12] Ian McEwan, David Sheets, Mark Richardson, and Stefan Gustavson. "Efficient Computational Noise in GLSL". In: *Journal of Graphics Tools* 16.2 (2012), pp. 85–94. DOI: 10.1080/2151237X.2012.649621.