Modelling scene data for render time estimation

Harsha K Chidambara
DNEG, London, UK

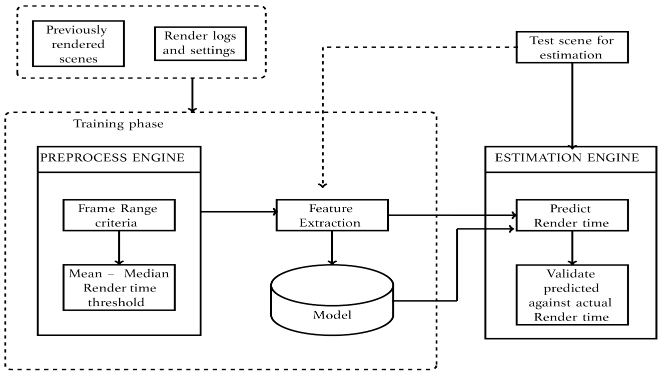

Figure 1: Render time estimation architecture

ABSTRACT

Rendering a scene is the most repeated and resource intensive task, at the core of a VFX facility. Each render can consume significant compute resources, which are expensive and finite. The number of renders (iterations) required to *final a shot* varies based on the creative and technical complexity of the scene. Getting a reliable estimate of the render time beforehand could prove useful from a budgeting and scheduling perspective. In this poster we present a novel approach to estimate render time of a scene based on a machine learning model built upon previous renders on a show. Each input vector for training is encoded from *direct* constituents of a scene like *assets*, *looks*, *lights* and *render* parameters like number of samples and resolution. Renders are categorized into two buckets: *less than an hour* and *greater than an hour* and two models are built for estimation. The estimated results for *test* scenes are verified against the actual render time for measuring accuracy.

CCS CONCEPTS

• **Computing methodologies → Supervised learning by regression**.

KEYWORDS

Data analysis, Regression, Render time, Estimation

ACM Reference Format:
Harsha K Chidambara. 2019. Modelling scene data for render time estimation. In *SIGGRAPH Asia 2019 Posters (SA '19 Posters), November 17–20, 2019, Brisbane, QLD, Australia.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3355056.3364578

1 INTRODUCTION

Rendering algorithms have been a topic of extensive research over the last few decades. The world has seen significant advancement in techniques ranging from scan-line renders to path tracing, and most recently real-time rendering of light fields. Render time analysis[Wimmer and Wonka 2003] have mostly been conducted on isolated hardware with an aim to benchmark performance of a certain algorithm, or to compare speedup across hardware/GPU. There is sparse literature on estimating render times at a macro scale for a post-production facility to help make informed decisions for scheduling[Ricklefs et al. 2017]. This poster is an attempt to share *a* solution which is currently being trialled at DNEG.

2 MOTIVATION AND CONTRIBUTION

Estimating render times are a *critical* task within a VFX facility. Owing to the complex workflows involved in rendering an image and the variety of inputs which impact the compute time, it can be quite challenging. The novel approach here is to parse a scene to be rendered and collate *information* about the parameters which are proven to influence render times. By proven, we mean that it either has a strong correlation on the *dependent* variable or influences another independent variable statistically. The proposed approach involves transforming any *user setting* applied either directly or indirectly to a render to a measurable entity. The current approach is focused towards Isotropix Clarisse $^{\text{TM}}$ as the primary renderer, but could be extended to other renderers.

3 METHODOLOGY

The architecture to estimate render time is described in Figure 1 The two main data sources are rendered *scenes* and render logs. It is probably a fair assumption to make that the number of *hero* assets during the lifetime of a show would not change drastically, at least by half-way into production. More often than not, each creative iteration is improving the *overall look*, lighting or tweaks in a scene to final a shot for full resolution renders. Hence, quantifying looks gets trickier than geometry. This poster tries to establish average render time as a function of constituents that *make* a render.

$$Render\ time = f(Geometry,\ Looks,\ Lighting,\ User\ Settings) \quad (1)$$

3.1 Preprocessing engine

Application of machine learning involves use of a good dataset and extraction of relevant features. The goal of the preprocessor is to prune outliers - the *mean* is influenced by occasional spike (or dip) of render times across a few frames. Limiting the absolute difference of mean and median render times across the entire frame range helps eliminate such *outliers*. In addition, since the *mean* can be influenced by the size of the denominator, a minimum frame range criteria was introduced - only renders with at-least 50 frames were chosen for training.

3.2 Feature extraction

The *feature extractor* outputs an encoded vector by parsing every scene. The key is to identify components/settings from a scene that can be measured as well as prove to influence render time. For example, every material has a variety of inputs that can be controlled to achieve a certain look under certain lighting. Hopefully this poster encourages thoughts on ways to *quantify and measure* parameters from user input (scene). Finally, only those features that have a high *correlation* on a given movie, are selected for training a machine learning model. Below is a list of *most common* features that are extracted per scene:

- *Geometry* based metrics : Polygon count/primitive count
- *Look* based metrics :
  - Count of texture files or maps
  - Count of materials
  - *Clarisse* specific : Count of shading layers, count of material assignments, depth of shading graph, number of nodes in the shading graph
- *Lighting* related : Global Illumination, Count of lights
- *Render settings* : Resolution of image, Number of anti-aliasing samples, number of channels (AOV), Render quality
- *Camera* based metrics : Motion blur sample count

3.3 Training

Training was conducted once on *each movie*, by splitting renders into a mutually exclusive train and test set. All renders that made the dataset were from production. Calculation of *correlation coefficient* across input dimensions and also against the dependent variable gives a notion of importance. It was found that classifying renders into two *bags*: *less than an hour* and *greater than an hour* was more efficient in estimation, over the unpartitioned dataset. These categories could seem highly oversimplified to begin with, however they were certainly intuitive and valuable from a scheduling perspective. The training phase builds two models per show, a linear and a support vector regression (SVR) model. A Grid-search technique was used to vary the input parameters to the SVR. The model with the least error was chosen.

Table 1: Accuracy using Support Vector regression model

Movie code	Category	Accuracy(%)
Movie 1	Greater than an hour	70.3
Movie 1	Less than an hour	65.8
Movie 2	Greater than an hour	69.4
Movie 2	Less than an hour	67.5
Movie 3	Greater than an hour	54.8
Movie 3	Less than an hour	66.6

Table 2: Accuracy using Linear regression model

Movie code	Category	Accuracy(%)
Movie 1	Greater than an hour	61.5
Movie 1	Less than an hour	62.1
Movie 2	Greater than an hour	53.7
Movie 2	Less than an hour	63.5
Movie 3	Greater than an hour	35.5
Movie 3	Less than an hour	64.4

3.4 Estimation

The estimation engine uses the same feature extractor as in *training*, to parse a scene which it hasn't encountered before. Using regression model(s) from training phase, the predictor outputs an estimated *average render time per frame* for both categories. The more accurate result is chosen.

4 RESULTS

The results are verified across *all* renders on a show. Movie 1 was FX heavy, Movie 2 had predominantly creature based shots, whereas Movie 3 was *digi-double* and CG environment creation focused. The regression models used *mean absolute percentage error* (MAPE) as the error metric over a five-fold cross validation. The accuracy of the techniques, per category using SVR and linear model are shown in Table 1 and Table **??** respectively.

5 CONCLUSION AND FUTURE WORK

The initial results by testing the hypothesis on three movies show that the support vector model works well irrespective of the *category* of the render. Linear models work reasonably well on renders less than an hour per frame, which is quite acceptable. Further steps could be taken to incorporate *qualitative* aspects of a scene; for example material categories. Investigating other prediction algorithms for better accuracy and fine tuning learning parameters is definitely something which will be refined in the future.

REFERENCES

Hannes Ricklefs, Stefan Puschendorf, Sandilya Bhamidipati, Brian Eriksson, and Akshay Pushparaja. 2017. From VFX Project Management to Predictive Forecasting. In *ACM SIGGRAPH 2017 Talks (SIGGRAPH '17)*. ACM, New York, NY, USA, Article 6, 2 pages. https://doi.org/10.1145/3084363.3085036

Michael Wimmer and Peter Wonka. 2003. Rendering Time Estimation for Real-time Rendering. In *Proceedings of the 14th Eurographics Workshop on Rendering*. Eurographics Association, 118–129. http://dl.acm.org/citation.cfm?id=882404.882422