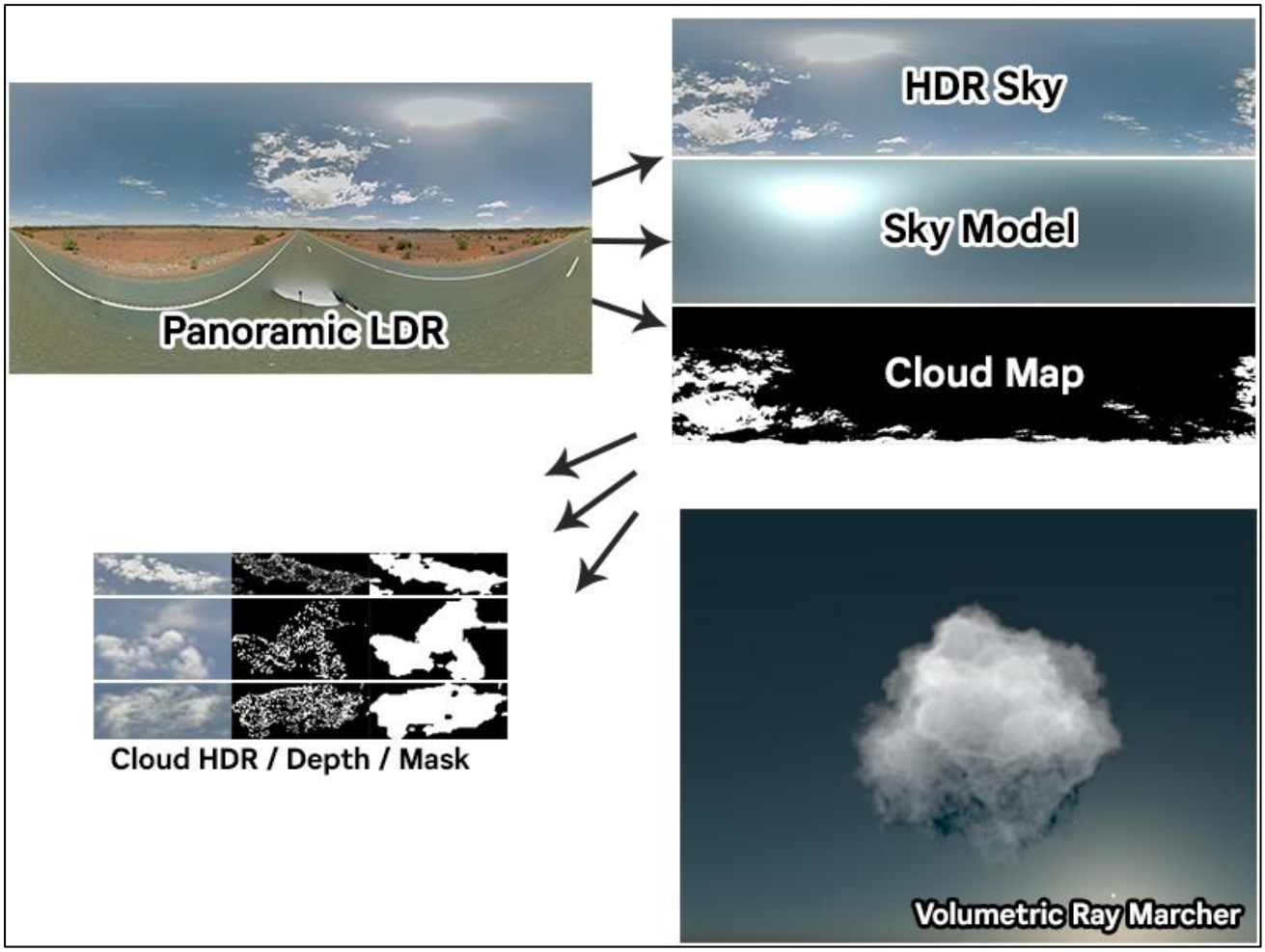
CTP Final Report – Matt Filer (17021871)

A toolkit to produce inexpensive and realistic skyboxes for games



Abstract

The purpose of this report is to detail the final outcomes of a research project which aimed to create a toolkit that can produce inexpensive and realistic skyboxes for games. The project's results were of interest and solved some novel problems: cloud data collection, image analysis, and upscaling HDR imagery. The project also produced various tools such as a panoramic image collection API, and a ray marching renderer for volumetric objects. This report will demonstrate results from these components, and evaluate their pros and cons.

Brief biography

I am a final year Games Technology student with an interest in systems and tools programming. I conducted this project because I saw it as a challenge, having never attempted a research project before or explored many of the technologies that I knew the project would cover. I have expanded my knowledge, tested my problem solving skills, enhanced my research capabilities, and as a result diversified my portfolio; which can be found online: mattfiler.co.uk.

How to access the project

This project can be accessed online via GitHub at: github.com/MattFiler/CTP-2019-20.

Use the following credentials to gain access:

- Username: MattFiler-CTP-2019-20
- Password: Cl0udProject

The repository contains all source code, as well as built binaries for each component of the project. Installation instructions are available in the README file, detailing how to set up an Anaconda environment to perform the image processing, and how to set up OpenVDB to re-compile the raytracer. All code is commented where other sources have been used.

Introduction

The intention of this project was to create a toolkit that can provide a way to produce realistic skies through the utilisation of real-world data. These realistic skies would be offline renders that are inexpensive to implement in a live rendering application, such as a game, thereby solving the issue of generating expensive realistic skies at runtime. For applications already utilising static skies, this project intended to cut production time for artists by requiring little manual input to produce effective results.

It can be argued that realistic environments are more important than ever in the current gaming landscape, with a growing focus on graphical fidelity in modern triple-A titles. Although this may be the case, it is often difficult to achieve realistic visuals for a multitude of technical reasons, notably in virtual reality where applications are often required to fully render a scene upwards of three times while targeting a consistent average of ~80 frames per second. For this reason, pre-computing (or "baking") elements such as lighting and shadows are instrumental in maintaining performance; yet to achieve realistic skies, realtime solutions are still preferred. This project sought to offer an attractive offline alternative to realtime skies by solving the issue of realism, with the aim of minimising the static sky trade-off: a loss of dynamic options such as live day/night cycles; this itself being a non-issue in instances such as virtual reality due to shorter play sessions.

Through the course of this research project, a series of avenues were explored. Particularly interesting steps were made towards solving the novel problems of HDR upscaling and cloud depth calculation from imagery. Its final deliverables are:

- A pipeline for downloading and processing panoramic imagery for training
- A machine learning model trained to attempt to reproduce skies
- A volumetric raytracer able to load and render VDB objects

Practice

Due to consistent work throughout the year, this project has developed considerably since the previous report, with functional elements produced for every component that was planned to be explored. As well as this a range of interesting areas were researched, even if they didn't ultimately make it into the final project. Initial research was useful to gain an understanding for particular areas of the project, for example the cloud depth approximation methods, where significant prior work had been completed by NASA.

To keep the project on track during its development regular meetings were held. These meetings were not only used to assess the current objectives, but to plan ahead in the larger scope of the project and retarget elements appropriately. This resulted in a flexible workflow where the scope of different elements in the project could be rapidly altered to suit availability, deadlines, and other time constraints. Ultimately as this was a research project, the time some elements would take to complete was unknown, so adopting this flexible attitude to milestones proved to be useful. Regular meetings were also a good opportunity to discuss issues encountered with the research elements, allowing a space to explore alternative options to blockers; although due to the linear nature of the project this was not always possible.

A few key issues were overcome during development. One of these was managing the broad scope of the project; being solved by weekly meetings to distil current aims into achievable weekly goals. Another key issue was the speed at which some elements were required to be produced; this was solved by researching applications and libraries produced for such requirements, such as Keras for machine learning and MATLAB for mathematic scripting.

Outcomes

While the original plan for this project was to collect a dataset and utilise machine learning to generate sky imagery, an alternative approach was considered during development to potentially improve results. This approach proposed a volumetric raytracer which could produce skyboxes using depth values calculated through machine learning, allowing for more accurate lighting and clarity in the final image. For that reason, the research project has produced a range of notable outcomes, including: a pipeline for collecting and processing large quantities of HDR cloud imagery, a trained neural network utilising collected cloud imagery, and a volumetric ray marching renderer. The following section will explore each of these in detail.

Cloud data collection

A key requirement from the outset of the project was the need to gather a large quantity of high-quality cloud imagery for training. To source this, a custom API was developed which utilised publicly available panoramic images and presented them in a useable format. This API was queried by a local toolkit, through which low dynamic range (LDR) imagery could be easily downloaded. It was quickly identified however that the majority of software-processed panoramic imagery typically utilises heavy post-processing to average brightness across the frame, and so it was decided that the downloaded LDR images would be converted to high dynamic range (HDR) in order to utilise as much data as possible.

The conversion of LDR to HDR was perhaps one of the biggest areas of research for the project, as although this issue has already been solved by past papers such as ExpandNet (Marnerides *et al*, 2018), the results do not support images of the resolution required for this project. To solve this novel issue, an implementation of LDR to HDR conversion as developed by Zhang *et al* (2017) was used as a base to generate HDR values, and an upscaling script was developed utilising MATLAB. While experiments were conducted into conversion to the YCbCr colour space for adjusting luminance values when upscaling, ultimately the best solution was to calculate relative luminance values from the original full-resolution LDR image and the low resolution generated HDR image using $y = 0.2126R + 0.7152G + 0.0722B$. With these relative luminance values calculated, plotting them both to a histogram provided a way to map the machine learning generated HDR luminance values to the original LDR luminance values by simply stepping through the histogram's bins and finding a best match. Applying this mapping to the original chroma values produced HDR images of original full LDR resolution. The results of this process were positive, reliably producing high quality HDR imagery.

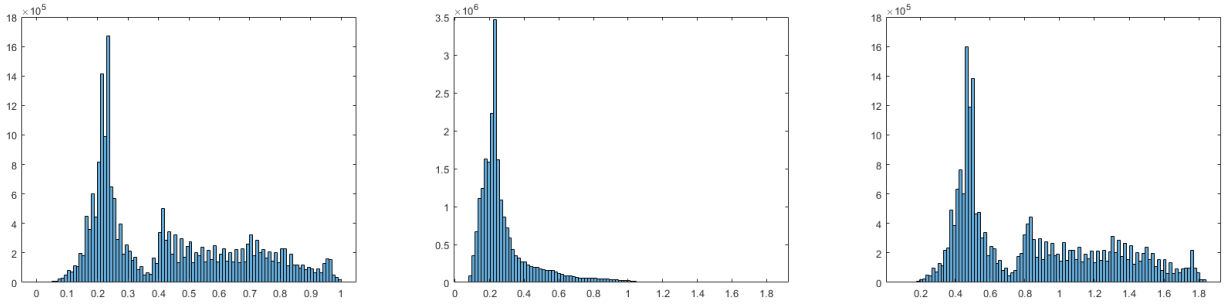


Figure 1 - Left to right: original LDR luma values, ML produced HDR luma values, LDR mapped HDR values.

Images were also processed to identify important metadata. Sun position was calculated by averaging brightness values in horizontal and vertical columns across the top of the image, taking the average position of the highest average brightness values in each column. Ground position was calculated by taking multiple processing passes across the whole image. The first of these passes involved checking the brightness of pixels vertically in each column of the image, logging brightness differences, and finding the best guess of ground location based on the average position of highest differences. Working across the image, the previous column's ground location guess then informs the next, to bias the guess towards a theoretically more accurate result. Multiple extra passes then run to discredit any outlying values (for example, a ground value that spikes away from its neighbours), before an average is taken across all remaining values to give a straight line along the ground. The result gives a high degree of accuracy, and the entire process can be seen illustrated in figure 2.

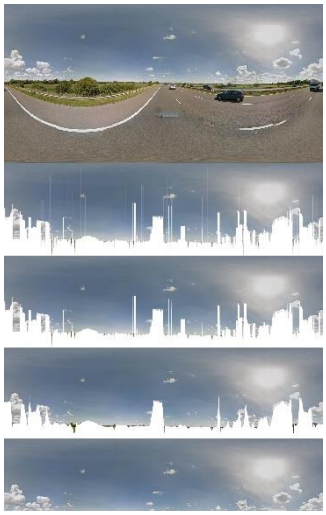


Figure 2 - Top to bottom: original, pass 1, pass 2, pass 3, final average.

To identify clouds in the sky, initially it was planned to use a project developed by Satilmis (2016). This classifier utilises a machine learning model trained on fisheye images, outputting a semantic map of cloud types from the original image. ImageMagick scripts by Weinhaus (2018) were utilised to warp trimmed HDR panoramic skies to and from fisheye, able to be passed through the classifier, with the resulting semantic map being applied as a lookup table (LUT) over the original image. While this process worked, its results were unsatisfactory and often unreliable. The semantic map rarely matched the original sky image, producing values that couldn't be relied upon to be used in the project; this can be seen in figure 3.

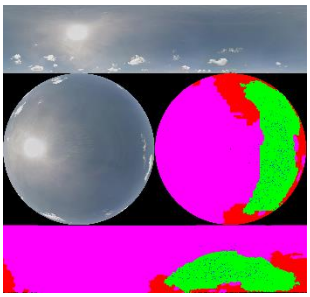


Figure 3 - A panorama sky converted to fisheye, classified using tools by Satilmis (2016), and dewarped back as a useable semantic map.

As an alternative, a method of identifying clouds similar to that of Yang *et al* (2017) was developed by analysing average chroma values across the image, then stepping through each pixel and evaluating it against the averages. Pixels were considered cloud if the following criteria was met:

$$(r > \bar{r}\ \&\ g > \bar{g}\ \&\ b > \bar{b}\ \&\ b < g\ \&\ b < r)\ or\ (\frac{r}{b} > \frac{\overline{r}}{\overline{6.5}})$$

With this initial cloud mask produced, a further refinement was performed by linearly checking each pixel in the mask. If cloud, the pixel's neighbours were then recursed to capture the full identified cloud region, with its bounds saved to a list. When creating new mask bounds the list was checked to make sure that none overlapped. With a completed list of unique mask bounds, sizes were evaluated to discredit bounds below and above a given threshold, leaving the most likely quality cloud bounds to keep. These bounds are then further evaluated to calculate their quality. A quality cloud is determined by a high average brightness, a high lowest brightness, and high average blue colour values through the bounds. If a cloud meets all of these criteria it is put into a revised mask, which is saved. Examples of quality clouds can be seen in figure 4.

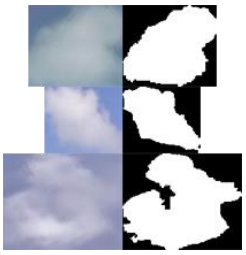


Figure 4 - Sections of mask marked as good quality.

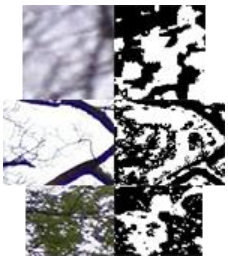


Figure 5 - Sections of mask marked as poor quality.

This process generated a reliable (although cautious) mask over the sky, of which all cloud pixels were considered to be of type stratocumulus. An improvement to this process would be to not simply discredit larger identified clouds in the refinement steps, and instead categorise them as different types, such as cumulonimbus. This could expand the dataset available, similar to the initial intention with the semantic map.

Figure 6 - A sky with its final generated cloud map.

Utilising this cloud map, the final stage of processing then attempted to calculate depth and inscattering within clouds; another novel issue in this context. To do this, a clear sky is generated using a method developed by Hosek *et al* (2013), known as the Hosek-Wilkie model. This model takes parameters for sky turbidity, ground albedo, and sun position; returning a HDR image which approximates a sky (as explained in the research report). Using this sky as a base, cloud depth data can then be approximated by calculating differences between it (Ls) and the generated HDR panorama sky cut out (L). To do this, each pixel in the image is iterated through, taking neighbouring pixels within a range and picking the closest in colour value. Using these two pixels, depth can be calculated using the formula:

$$d_a = \frac{-\log\left|\frac{L_a - L_b}{Ls_a - Ls_b}\right|}{\sigma s}$$

Where subscript a is the current pixel, and subscript b is the neighbour with the closest colour value. σs represents the scattering and absorption coefficient for the current cloud type, as detailed in a report by Satilmis (2016), as such $\sigma s = \sigma_{sca} + \sigma_{abs}$.

Results from this calculation (while unproven against legitimate real-world data) appear to give useful results, returning depth values around the edges of marked clouds; as would be expected since these regions allow visibility through to the sky. Where clouds are identified but depth values are not returned, a pre-determined constant is used for the cloud depth, as it is within the thicker part of the object. In future it would be preferable to modulate this constant by the depths surrounding it to give a more accurate (and better looking) result.

Shown in figure 7 is the entire processing pipeline for one isolated cloud in the full panoramic image, with valid depth data locations represented by white points.

Figure 7 - Left to right: original panorama section, generated cloud mask, generated sky model (HDR may seem brighter than other images), generated depth value locations, generated inscattering RGB.

Trained neural network

While a neural network was a key deliverable outcome from the outset of the project to produce sky imagery, development on it started late due to blockers encountered while working on the dataset pipeline, and extra time given to exploration with the image processing. Ultimately this wasn't a failing in the project management, but a result of the research exploring novel issues which were producing (at the time) unknown and unconfirmed results.

Without a clear idea of the final output from the training dataset it was tough to make a positive start on this element, which is one reason why the volumetric ray marcher approach was put forward towards the end of the project. Even though this was the case, a fully convolutional network was produced and trained, building from an existing deep learning project named DeepDoodle (CodeParade, 2018). While results from this network were unsatisfactory for real world use due to the time constraints to bring the project together, they showed promise in the method for future endeavours to continue to explore given more time.

The deep learning model utilised Keras, a high level API for machine learning which is able to interface to Tensorflow and Theano. This API allows for a quick and easy way to construct a machine learning model, and simple methods to save the trained model which made it perfect for the rapid development of the network towards the end of the project. Theano was chosen for the backend framework due to its widespread support and simple install for CPU and GPU configurations (GPU being used in this case to speed up processing times).

The network utilised the collected dataset's cloud mask and cut-out sky from each processed panoramic image. Training with this data it could learn the expected visual output for clouds, using the mask as context. Once trained, this allowed a user generated mask to be input to the network to produce cloud formations, which could be layered on top of a pre-computed sky model. Unfortunately, due to time constraints the network had to be trained at a low resolution and with a limited dataset, so while functional the final results at the time of writing are too poor quality to be considered production ready. As seen in figure 8, when training the network over 500 epochs the test set's RMSE value steadily rises above the training set's value, showing that the poor image output is also likely a result of overfitting the data in the model.

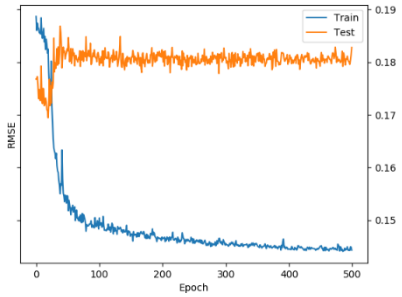


Figure 8 - Train and test RMSE values over 500 epochs.

With further training and refinements to the data or network structure, this component will likely produce useable results. As sun position was auto aligned to the same position across all images in the training dataset by the dataset tool, the network could also theoretically learn the lighting context to produce a believable final scene if trained further. As it stands, unfortunately the results from this network fall below the expected outcomes of the project, and further work will be required in this area.

Volumetric ray marcher

Proposed as an alternative to the machine learning generated sky imagery later into the project, a volumetric ray marcher was produced to utilise depth and light transport data which could be generated with a similar machine learning model, trained on depth values rather than RGB values. Using the predicted depth output from the model for user given masks volumetric objects could be produced to represent clouds, loaded through OpenVDB and rendered on top of a Hosek-Wilkie sky model. The advantage this would give is that you could accurately simulate lighting data through the renderer, rather than relying on the machine learning model to understand and generate it correctly itself; thereby giving a more realistic final result.

In the raytracer produced for this component OpenVDB is implemented and utilised to calculate the densities of volumetric objects along fired rays. Using this density data the brightness of the on-screen pixel is established, which is then modulated by density values calculated from the ray sample points through the volumetric object to the sun. This gives the effect of a sun shadow on the object, without performing any complex inscattering logic. Background sky is calculated using the Hosek-Wilkie model mentioned previously due to its accuracy and ease of use.

Figure 9 - The raytracer output utilising the Disney Moana dataset (Thacker, 2018), with depth-to-sun calculation enabled.

Although inscattering logic is not applied, work was completed towards doing so. A method known as Woodcock tracking was explored which involves stepping through a volume, and randomly deciding to stop or continue (Marschner, 2012). Upon stopping, a phase function created by Henyey *et al* (1940) would be sampled, which was successfully implemented into the project. This phase function when sampled returns a new direction to cast the ray in, allowing for the ray to scatter within the volume until it leaves, or hits a maximum length.

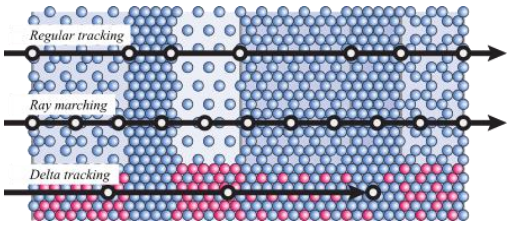


Figure 10 - Ray methods demonstrated.

Ultimately, the results from the raytracer as-is are suitable for the current progression of the project, and were it to continue further, this inscattering logic could be easily implemented to improve visuals. Another useful addition would be proper transform controls per volumetric instance to allow for cloud positioning in the environments. The image output from the raytracer should also be adapted into a panoramic image or environment map, to allow use in a game or other rendering application.

Discussion of outcomes

Although up to this point the project ultimately hasn't generated production ready results for its initial proposed outcomes, the crafted components cover a considerable amount of the proposal, with some extra additions such as the volumetric ray marcher. All components produced to date while not considered final are functional and reliable. The dataset pipeline is an easy and efficient way to collect large volumes of processed imagery for training; and while both lacking features the deep learning project and ray marcher produce interesting results that go towards solving the challenge set in this project's initial proposal.

Due to this project's originality, its outcomes to this point could be of interest in a wider professional context. The methods developed for HDR upscaling, and work towards depth/inscattering calculation from cloud imagery are both novel issues and could well be useful for other similar projects. As detailed in a dissertation by Eilertsen, G (2018), low resolution sensors in HDR cameras are often a problem with retaining quality at the gain of the extra dynamic range in the image. The methods of HDR upscaling developed in this project could be useful to solve this issue, with LDR imagery being utilised in conjunction with LDR to HDR methods.

Although the results for the depth/inscattering calculations are unproven for accuracy, the values generated are close to what was expected, and appear to merit the method for at least making a step towards solving the issue. Again, while the HDR upscaling has not been compared with a real full-resolution HDR image to gauge an accuracy figure, the results that were produced are high quality and appear to solve the novel issue when compared against LDR counterparts.

Conclusion

While this project isn't currently at a point where it can be regarded as complete, it is intended to be continued past this point. Key improvements are planned to be made to the machine learning component to improve the quality of results, and to train with depth data to allow the raytracer to be utilised. If this utilisation proves useful, the raytracer can then be expanded upon to properly calculate inscattered light as detailed in this report.

With these improvements, the project should be functional to the point that it can produce quality useable results, capable of being utilised for projects on hardware such as the Oculus Quest, for example. Having spoken to developers in the research phase an interest was shown towards the project, so it would be appropriate to try and refine the generated results further.

Reviewing the question posed in the initial research report: "how can pre-computed static skies be brought up to modern standards utilising recent advances in modern science and technology?" it can be argued that this project has made strong steps to solve this problem. While the results that the project currently produces wouldn't be considered production ready, with further work completed it is expected to be useable. The early results from the raytracer are promising and paired with the progress in machine learning with work completed to sample cloud depths, there are signs that this is achievable.

References

- Academy Software Foundation (2012) OpenVDB (2019) [computer program]. Available from: https://www.openvdb.org/download/ [Accessed 26 March 2020]
- CodeParade (2018) *DeepDoodle* [computer program]. Available from: https://github.com/HackerPoet/DeepDoodle [Accessed 5 April 2020]
- Eilertsen, G (2018) *The high dynamic range imaging pipeline.* Dissertation, Linkoping University.
- Henyey, L *et al.* (1940, p.117) *Annales d'Astrophysique.* Volume 3. NASA Astrophysics Data System, USA.
- ImageMagick Studio (1999) ImageMagick (2019) [computer program]. Available from: https://imagemagick.org/ [Accessed 01 April 2020]
- Marnerides, D *et al.* (2018, section 2, part 1) *EUROGRAPHICS 2018.* Volume 37, Number 2.
- Marschner, S (2012) *Volumetric Path Tracing.* Cornell University.
- MathWorks (1994) MATLAB (2020) [computer program]. Available from: https://www.mathworks.com/products/matlab.html [Accessed 25 March 2020]
- Satilmis, P (2016) *High Fidelity Sky Models.* PhD, University of Warwick.
- Thacker, J (2018) Download Disney's data set for Motunui island from Moana. *CG Channel* [blog]. 4 July. Available from: http://www.cgchannel.com/2018/07/download-disneys-data-set-for-motunui-island-from-moana/ [Accessed 28 September 2019]
- Weinhaus, F (2017) pano2fisheye/fisheye2pano (2018) [computer program]. Available from: http://www.fmwconcepts.com/imagemagick/ [Accessed 20 March 2020]
- Wilkie, A *et al.* (2013) *Predicting Sky Dome Appearance on Earth-like Extrasolar Worlds* [SIGGRAPH 2012], Los Angeles. 5 August.
- Yang, J *et al.* (2017, p.1191-1201) *Atmospheric Measurement Techniques.* Volume 10. State University of New York, USA.
- Zhang, J *et al.* (2017) *Learning High Dynamic Range from Outdoor Panoramas* [ICCV 2017], Venice. 22 October.

Project Log

Date	Tasks completed prior to meeting	Questions arisen in meeting	Tasks to complete for next meeting
9th January	Next week is presentation, so this week have been starting work on the video. Tidied up codebase to get a demo together.	N/A	Will complete video by next week, as demo is on Wednesday.
23rd January	This week I presented my CTP demo, which went well, and spoke to Tom about next steps. We discussed the current state of the project, and its current achievements.	How will the HDR upscaling progress? This is currently the biggest blocker.	Look into conversion to YCbCr colour space for the HDR upscaling – should be doable in MATLAB.
6th February	This week began working on the raytracer component of the project, and looked into YCbCr conversion in MATLAB. Ultimately, the libraries are all outdated and didn't seem to work well.	Good work with start on raytracer. What alternatives can be used for MATLAB libraries? Get the images loading into histograms.	Find alternative libraries, or alternate methods to using YCbCr. Try and load the data into histograms and match the plots.
20th February	Initial HDR upscaling implemented, seems to be semi-working. Plots are matched, however results seem to be slightly off. Using luma value matching rather than YCbCr colour space.	How reliable is this method? Test it on a few images.	Test reliability of method. Double check the data being used as it seems to be off in some places on the histogram. Implement the script into the pipeline.
5th March	Script implemented into pipeline. Upscaling fixed and working – were some bugs with brightness in previous iteration. Can now reliably work using luma from full-res LDR and low res HDR.	Can images be converted to fisheye for use in Pinar's classifier? This is currently a blocker.	Continue with raytracer. Images need to be converted to fisheye to work with Pinar's classifier properly. Also, get OpenVDB working in the raytracer.
19th March	Managed to compile OpenVDB, and get it implemented in a basic form to the raytracer. Fisheye conversion working, but not back the other way.	Are the results from Pinar's classifier good? Can depth calculation be implemented at this point?	Implement the depth/inscattering formula researched previously. Get de-fisheye working to be able to use the classifier output.
2nd April	Conversion back from fisheye working – but results of classifier are poor. Alternate method development in progress analysing red/blue colour channels. OpenVDB expanded to load into raytracer with depth values returned.	Can alternate method be developed to be reliable enough? Can an output be generated from the raytracer?	Make a start on the machine learning component to have something for submission. Output an image from the raytracer.
16th April	Cloud map refined to a point where it is reliable. Large dataset collected with pipeline, and DeepDoodle training in progress. Image out from raytracer, utilising depth to sun. Work begins on final report.	Can the machine learning component be ready for submission?	Finish final report. Round up the machine learning model to have something for submission.