Measuring 3D Mesh Saliency using an Eye Tracker

THOMAS SWEETMAN

SUPERVISOR: YUKUN LAI
MODORATOR: PAUL L ROSIN
BSc Computer Science
School of Computer Science and Informatics
Cardiff University
May 2018

Abstract

For uses in 3D graphics and product design it can be greatly beneficial to have an accurate method of measuring saliency. These methods can be used in place of eye tracking experiments to produce an estimate of what a human might perceive as salient without the need for expensive equipment or to gather a large number of volunteers. These saliency methods are not always completely accurate and so this report aims to measure the accuracy of several different existing methods of detecting saliency and then learn a new model of measuring saliency using existing methods.

Acknowledgements

I would like to thank my supervisors, Dr Yukun Lai and Professor Paul Rosin for their support and knowledge throughout this project.

I would like to thank Dr hantao Liu and Lucie Lévêque for assisting me with the design of the eye tracking experiment and supplying the eye tracking equipment.

I would also like to thank my friends and family and members of Cardiff university for their participation in the experiment that made this project's goals possible.

Table of Contents

Table of Figures

Introduction

In recent years 3D data has become increasingly popular especially as 3D media technologies have improved. Some of the major applications of 3D data include special effects for film, gaming, 3D printing and 3D architecture modelling. When manipulating a 3D model via downsizing, stretching or deforming in some way salient (i.e. important) areas on the model need to be preserved to maintain the overall quality of the model.

This dissertation will develop a method for measuring salience on a 3D model. Salience will be measured by running experiments with an eye tracker which will produce a saliency measurement that is accurate to human visual judgement. using the data gained from the eye tacking experiment existing methods can be evaluated by comparing the importance the existing method gives each vertex (i.e. the saliency map) with the experiment data saliency map. The saliency map comparison will be conducted by SSIM which has been modified to take saliency maps rather than 2D images. Existing methods of measuring saliency are based on low level geometric features of the 3D mesh. this dissertation will use machine learning methods with the eye tracking data and the existing method data to optimise a new method of measuring 3D salience to better predict human perception of salience.

Measuring 3D saliency with an eye tracker could be used by advertising companies that wanted an insight into how their products are perceived and what stands out on their products, as is done on two dimensional advertisements. Producing a learned method of measuring saliency that better represents human perception would allow for more accurate feature preservation when manipulating a model.

This project was started as a Cardiff University Research Opportunities programme project, during the programme I managed to produce a method that took in 3D models and produced images for the eye tracking experiment. Once eye tracking data had been captured it then remaps the data to a saliency map on the original model. Some preliminary experiment data was collected during this project.

Background

Saliency

The concept of saliency is well documented and researched and is described as the “distinct subjective perceptual quality which makes some items in the world stand out from their neighbours and immediately grab our attention” (Itti, 2007). There is no one quality an object can have to make it salient as an objects salience in a scene is dependent on how unique its attributes are to the other objects that are around it. Work in 2D image saliency has made substantial progress towards measuring low-level features in images like colour, intensity and orientation using these features in different spatial scales and taking the surrounding area into account (Itti, 2007). In simple scenes these low-level features would be enough to predict where a person would look, however there is semantic information that would immediately grab a person’s attention like an image containing a person or text that aren’t necessarily picked up by the low-level features. The same multi-scale and centre-surround methodologies exist in model saliency as image saliency however, the low-level features change from colour and intensity to geometric information like distance between points and curvature.

Evaluating methods of measuring saliency

There are many methods of detecting saliency in a 3D model but there is little in the way of evaluating the effectiveness of these methods. Many papers use heatmaps to show salient areas on a model or use saliency guided mesh simplification to show the method preserving interesting areas. While these methods are effective at showing how a method works on a high level it makes it difficult to compare the effectiveness of different methods as these are only subjective methods of evaluation. This project aims to provide a new way of comparing methods of measuring saliency in a quantifiable manner by generating a ground truth using an eye tracker and comparing the ground truth saliency map with the method produced saliency maps using SSIM as the quality assessment function. Once these methods have been evaluated a new method can be learnt which combines several different methods to hopefully produce a more accurate model. A very similar endeavour was done on 2D images by (Judd, et al., 2009) where they took eye tracking data and evaluated existing models. While evaluating their methods they found several areas that existing models were inconsistent with the human eye tracking data, meaning they can add primitive methods to account for the inconsistencies for their learnt model.

SSIM

This project uses a modified version of an image quality assessment model called structural similarity (SSIM). This quality assessment model, developed by (Wang, et al., 2004) assesses the structural similarity between two images, used in the context of quality assessment a distorted image can be compared to the original image to compute how visually accurate the distorted image is to the original. The SSIM method combines three components to produce a similarity measure between two images; luminance, contrast, and structure. Comparing two images A and B the SSIM method takes a spatial patch around each pixel to get local information around the pixel, pixel intensity is then scaled by a gaussian function to give points nearer the centre of the patch more influence. the mean intensity of the patches is then calculated producing μ_x and μ_y where x and y are spatial patches of pixels in the same spatial position in images A and B respectively. From these means a luminance comparison function can be made

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$

Where the constant C_1 is included to stabilise the function when $\mu_x + \mu_y$ approaches zero. The standard deviations σ_x and σ_y of the spatial patches are used as a measurement of the contrast, with this measurement a contrast comparison function similar to the luminance comparison function can be made

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

Where the constant C_2 is another stabilising constant. To calculate the final component needed a slightly different function is needed

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$

Finally, to compute the SSIM index of a pixel, following the same procedure laid out by Wang et al. in their paper; taking $C_3 = C_2/2$ produces the following function

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_x + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

More detail on how these functions were developed and how they sit together can be found in the paper written by Wang et al. In this project the only change made is how the information is delivered to the SSIM function and how the spatial patch is evaluated. Rather than sending a two-dimensional image a list of values is sent with one value for each vertex representing the level of saliency at that vertex. The spatial patch around each vertex is found as every other vertex within a set distance from the initial vertex, this provides the local information that neighbouring pixels would supply in a 2D image. The area a pixel covers in an image is uniform for all pixels in that image, however this is not necessarily true for 3D models, each vertex will be connected to faces of different sizes and as such each vertex has a different level of influence on how the model looks. To account for varied vertex influence, each vertex's saliency value will be multiplied by the vertex's Voronoi area which for a triangular mesh is the sum of 1/3 of the area of each face the vertex is a part of.

Existing methods of measuring saliency

There are several existing methods of measuring saliency of 3D models, some of which I will evaluate in this dissertation and attempt to learn a new model using them, I am also using two primitive methods of detecting saliency. I have a centre bias model which simply makes vertices closer to the centre of the model have a higher saliency value, this accounts for peoples' tendency to look at the centre of an image as noted by Judd et al. The second method takes the gaussian curvature of the model at each point, a lot of the existing methods include some form of gaussian curvature information but having this primitive gives my learning methods a little more flexibility.

One of the methods used is a method by (Lee, et al., 2005) that uses the idea that areas that are expressing different geometric characteristics to the surrounding areas would be salient as people pick out things that are out of place. For example, a large spike in the middle of a flat surface would attract a lot of attention, but equally a flat section in an area populated by large spikes would be just as eye grabbing. A centre-surround operation is used which takes the difference between mean curvatures around each vertex then filtered with a gaussian. For each gaussian a gaussian weighted average of the curvatures of vertices within a radius that varies between gaussians, the different gaussians are then aggregated together using a nonlinear normalisation to produce the final computed saliency. More detail on this method can be found in the paper written by Lee et al.

The second method utilised is a method which is a multi-scale computational model similar to lee et al. however it uses spectral processing rather than a difference in gaussian to measure saliency. This method developed by (Song, et al., 2014) takes the set of meshes being a group of meshes simplified to different degrees and computes the scale saliency map for each scale by computing the spectral mesh saliency for each scale. The scale saliency maps are then summed together to generate a final saliency map, further detail on this method can be found in the paper by Song et al.

The final method used in this project is from a paper by (Pickup, et al., 2015) written to find similar models to a given model. This is done by converting the model into a canonical form by taking several feature points on a model and moving them as far apart from each other while maintaining edge length so as not to overly distort the model. These feature points are selected using the conformal factor to find the extremities of the model. This report will be utilising this feature point selection part of this paper by pickup et al. The conformal factor used in a paper on generating a new 3D shape descriptor by (Mirela & Gotsman, 2008). The conformal factor can be described as a scalar function on the model that will produce a surface with a constant gaussian curvature. using the conformal factor is pose invariant which means it should allow the learnt method to account for more dynamically posed models.

Aims

The aims of this project are, to generate a 3D heatmap of salient areas of a model using eye tracking data, evaluate how accurate current methods of measuring saliency is to the eye tracking data, and develop a new and improved method of measuring saliency using machine learning.

Research question(s)

In order to achieve these aims this project will gather eye tracking data from participants and develop a method for mapping the eye tracking data onto the 3D mesh, evaluate existing methods of measuring saliency by comparing their saliency maps with the ground truth using SSIM, learn a new method of measuring saliency using least squares regression and a feed forward neural network.

Approach and Implementation

3D heatmap of eye tracking data on a model

Approach

To accurately measure the saliency of a 3D model using an eye tracker several things need to be considered. Firstly, showing the model to the participant in such a way that the entire model is displayed, and no areas of the model are shown more than others. Secondly mapping eye tracking data back to a model to produce a heatmap to show the overall 3D saliency of a model.

To generate eye tracking data, stimuli is needed. in a 2D context this is straight forward, measuring 3D saliency of an entire model raises the question of how to display a model to a participant without biasing any part of the model. you could show the participant an animated image of the model rotating however, this can introduce a bias as people will lose focus towards the end of viewing an image. To show all parts of a model without introducing a bias this project takes 20 2D images of a model from different positions around the model. To ensure an even coverage the images are taken from the centre of different faces of an icosahedron scaled to surround the model. After these images are generated they will be used in an eye tracking experiment to get saliency data for each image.

This project used 20 models as the data set for the eye tracking experiment. For each model a simplified version was added to the data set to see what impact simplifying a mesh had on its saliency. This means that a total of 40 models were used for the eye tracking experiment, each having 20 images to show participants. This was a large set of images, so it was split into 5 sub-sets, each participant would view a subset which contained 4 views of each model. It was decided to split the data set by view rather than by model as if a participant has already seen a model from several angles they would be likely to ignore certain features they have already seen. Each image is shown to a participant for 5 seconds then shown a grey screen for 2 seconds to remove any fixations based on the previous image.

To map the saliency data back to the mesh, we need to know where the mesh vertices are in an image. To find this image to vertex mapping an image is taken from the same position the original image is taken from on the icosahedron with each vertex colour coded so that each vertex can be identified once the testing has been done. Once a vertex has been assigned a fixation, vertices around it will also gain saliency based on their distance from the original vertex this produces a smooth saliency map and accounts for some error in the eye tracking data. If several people look at the same point on a mesh but the eye tracker places their fixations around the true point, then this

smoothing will make the true point more salient. Once all the fixations are mapped onto the mesh and smoothed the final salience heat map will be complete.

Implementation

to produce images for the eye tracking experiment the selected models need to be displayed to the screen. To achieve this functionality this project uses toolbox graph by (Peyre, 2007) taken from MathWorks file exchange to read and plot meshes form file. All of the models selected for use in this project are in the .obj format however this project should work with any file format the toolbox supports.

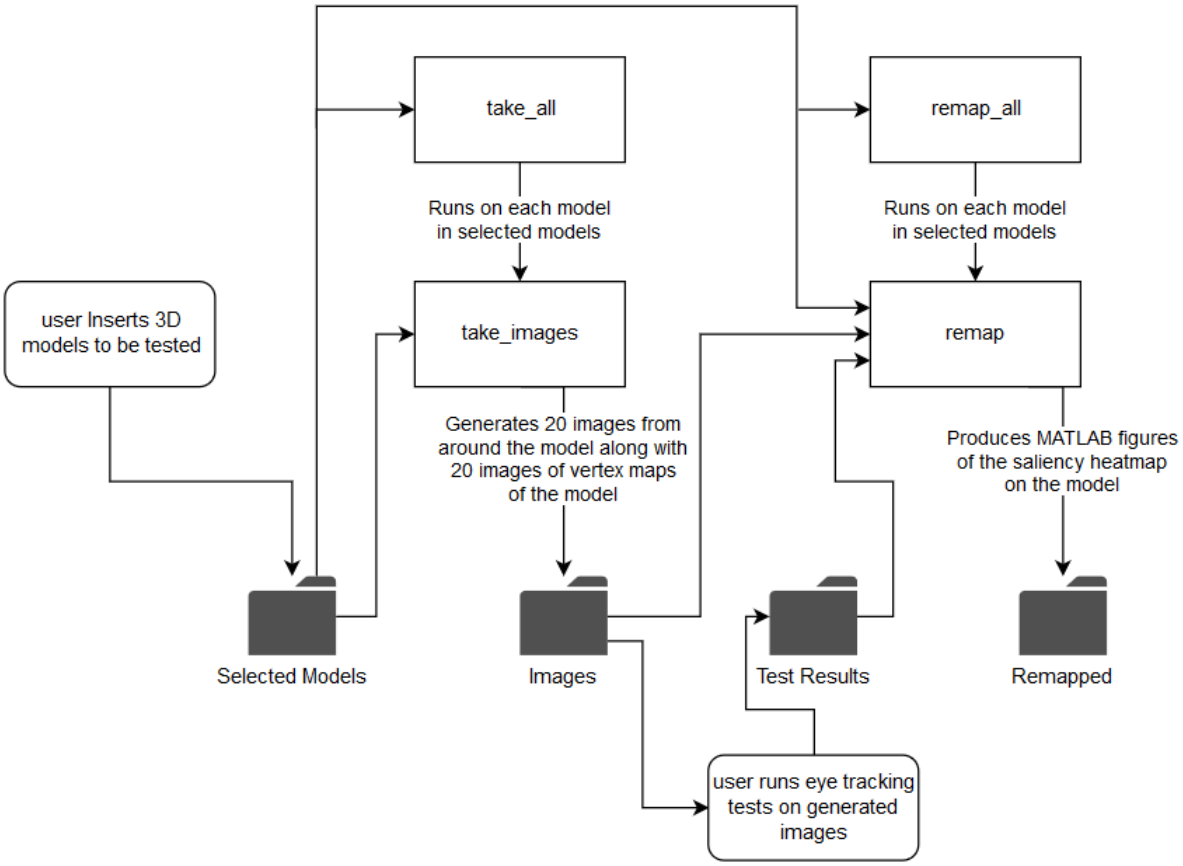


Figure 1 data flow for eye tracking stimuli and remapping scripts

Once the user has chosen the models they wish to use they should be placed in the selected models folder. The next step then is to generate the images needed for the eye tracking data. The script *take_all* calls the *take_images* script for each model in selected models which outputs images from around the model and the corresponding vertex maps. These images can then be taken to run eye tracking experiments and the results of those experiments should be placed in the test results folder. Once the test results are ready *remap_all* will call *remap* on each model in the selected model's folder which will output a MATLAB figure of each model with its saliency heatmap and a comparison figure with the heatmap before and after normalising the views.

take_images produces images as stimuli for the eye tracking experiment. To ensure complete coverage of a mesh without biasing the data 20 images are taken from different points of view around the mesh. These points are the centre of the faces of an icosahedron. These centre points are then scaled by the furthest vertex from the origin on the mesh so that the mesh nicely fits in the centre of the screen. For each point three lights are places around the camera to ensure each image

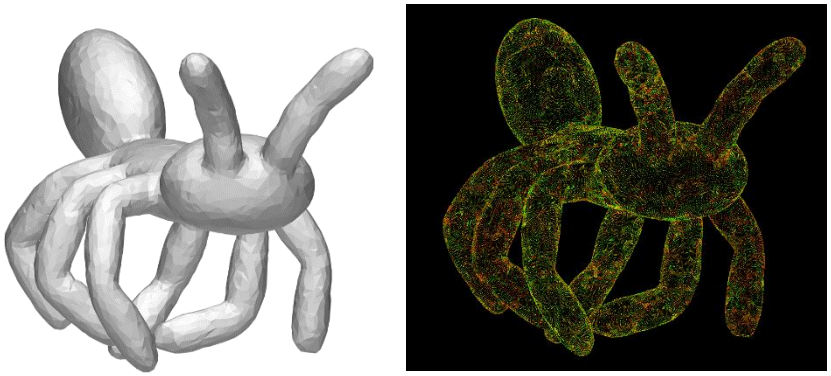


Figure 2 Example of an image from a view and its vertex map

has the same even lighting. Specular lighting strength was set to 0 as bright spots on a model could draw people's attention and give areas of little geometric interest high saliency. take_images outputs a list of where each vertex is on the image along with another set of images of the model from the same 20 positions as the experiment images were taken. The purpose of this second set of images is to aid remapping the saliency data back to the model. A three-dimensional scatterplot is made to mark the location of each vertex on the mesh, each vertex marker is colour coded by the index of the vertex using RGB encoding. Each marker takes up exactly one pixel of the image, so when given a fixation location in pixels the remapping code can identify which vertex is at that location. The mesh and the background of the image are then coloured black. This ensures that the only thing with any non-zero RGB values are the vertices, otherwise the remapping code could take the grey areas of the mesh as a colour coded vertex and either have array bound errors or just incorrect vertex selection. The benefit of having the mesh in the scene and blacked out Is that vertices behind the mesh that would not be visible to a viewer do not show up on the vertex map as they are blocked by the mesh. This stops the situation where a fixation may lie between two vertices and the remapping decides fixation is on a vertex on the opposite side of the mesh, where the participant could not see at all. One issue I encountered while developing the vertex mapping code was that for the simplified meshes the vertex map was so sparse that for some fixations there were no vertices nearby, so the algorithm decided it was 'outside the mesh' and as such an anomaly. To circumvent this issue, I took all the simplified meshes and subdivided them. This preserved their simplified features while also having enough vertices to not cause issue with the remapping code.

The remap script maps the eye tracking data to the 3D model. it iterates over each fixation in the experiment, for each fixation it takes the vertex map that corresponds to the image the fixation was on and taking the fixation x and y position in pixels and finds the nearest coloured pixel in the vertex map and decodes its RGB value into a vertex index. Once the fixation has been mapped to a vertex in this way its neighbours need to be found to smooth the saliency over the mesh. A quick method could be to check every other vertex to see whether it is within a set distance. While this method would work, for meshes with thin areas it would make the opposite side of the mesh salient. An alternative method would be to find the neighbours of the fixated vertex and add them to a border array. For each vertex in the border array check If it is within a set distance and if it is, then add it to the list of vertices to gain saliency and add its neighbours to the border array. This process continues until no more neighbours are within range of the initial vertex. This method can still allow salience to be added to the back side of a thin mesh if the point is near enough to the edge but for most meshes this is acceptable. An ideal method would be to include all vertices with a path to the initial vertex with a length less than the set distance however, for large meshes this method would take a very long time to compute for all fixations. The distance to include other vertices is set as the distance between the two farthest apart vertices divided by 20. This should provide around 5% of the mesh

around the vertex. Once all the vertices to gain saliency from the fixation has been calculated each vertex gain saliency according to the following formula

$$e^{d/d_{\max}} * t$$

Where d is the distance between the fixated vertex, d_{max} is the distance between the two farthest vertices and t is the time that the participant focused on that fixation, this means that quick scans over uninteresting areas of a model don't distort the saliency map.

Some views of a mesh will contain more interesting things than others which introduces an issue where when a participant is faced with an uninteresting view of a model they will still look at something, creating areas that are marked as salient but in truth are rather uninteresting. To tackle this issue a least squares regression is used to weight each view according to how much saliency a view gives to vertices that are common with neighbouring views. To achieve this an array Is made that contains view value pairs. For each fixation two views have in common two entries are made the first being the first view index and the saliency value of the common vertex from that view. The second entry is the second view index and the negative of the saliency value of the common vertex from the second view. This array is then solved by least squares using the MATLAB '\' operator. This outputs a set of weights to scale each view by. If a view has a lot of fixations on vertices that are common with neighbouring views, then its considered a less important view and its weighting will decrease while the neighbouring views that the fixations where in are more important. This functionality stops boring views having an unnaturally high saliency since people must look at something while also enhancing views that attract a lot of attention from other views. Once all the fixations have been processed and normalized the remap script creates two MATLAB figures for each model, one with the model with its normalised saliency map and one with a side by side comparison of the model with the saliency map with and without normalisation. Finally, the normalised saliency scaled so that the maximum saliency value is 2 and the minimum saliency value is -2. This means all saliency values can be easily compared as they are all in the same range. The saliency values are then output to a text file.

Evaluation of existing methods of measuring saliency

Approach

This project uses three existing methods of measuring saliency and two primitive methods. The first primitive method is a simple centre bias which gives vertices near the origin a higher saliency than points further away from the centre. The second primitive method calculates saliency as the gaussian curvature at each vertex. These two primitive methods are not likely to be very accurate on their own however, they can give the learning methods more flexibility. The three existing methods to be used are the three methods talked about in the background section by (Lee, et al., 2005) (Mirela & Gotsman, 2008) (Song, et al., 2014).

To evaluate existing methods of measuring saliency a method of comparing two heatmaps on the same mesh is needed. This comparison method can be used to evaluate accuracy by comparing the ground truth saliency map generated by the eye tracking experiment and the saliency map output by the method to be evaluated. If the two maps are similar, then the method is accurate. A simple mean squared error comparison between two saliency maps could provide a basic insight into similarity between the two maps, however two very similar saliency maps can have a very high MSE if the overall pattern of the map is the same but shifted slightly to one side, then MSE will score high as each vertex saliency is different even though the two heatmaps look virtually the same.

A comparison method that takes some local information from the surrounding vertices is needed to provide a more accurate assessment of method accuracy. This project will be using SSIM to assess methods accuracy to the ground truth. SSIM uses a centre-surround mechanism as part of its evaluation and as such is ideal for comparing saliency maps on meshes. The only issue with using SSIM is it is designed for 2D images and not 3D mesh heatmaps and so some work on redefining the neighbourhood from a set number of pixels around a pixel to all vertices within a certain distance. With SSIM ready to take in salience maps of 3D models a more meaningful evaluation of existing methods can be made. To evaluate each method the second half of the selected models are used to compare each methods saliency maps with the ground truth, the average of the SSIM scores will then be used to compare methods against each other. using the second half of the selected models means that when comparing the existing methods to the learnt methods which will use the first half of the models as training data both accuracy scores are evaluated across the same models and as such is an unbiased comparison.

Implementation

This project was designed with modularity in mind, and as such any new methods of measuring saliency can be added to the system by putting the source code somewhere accessible and adding a new folder under Saliency values that the method will write saliency data to. With a small code addition to existing_method_accuracy and the two learning methods a new method can be added to the system. With more time this process could be smoothed out further to simply link the code to be called and name a folder.

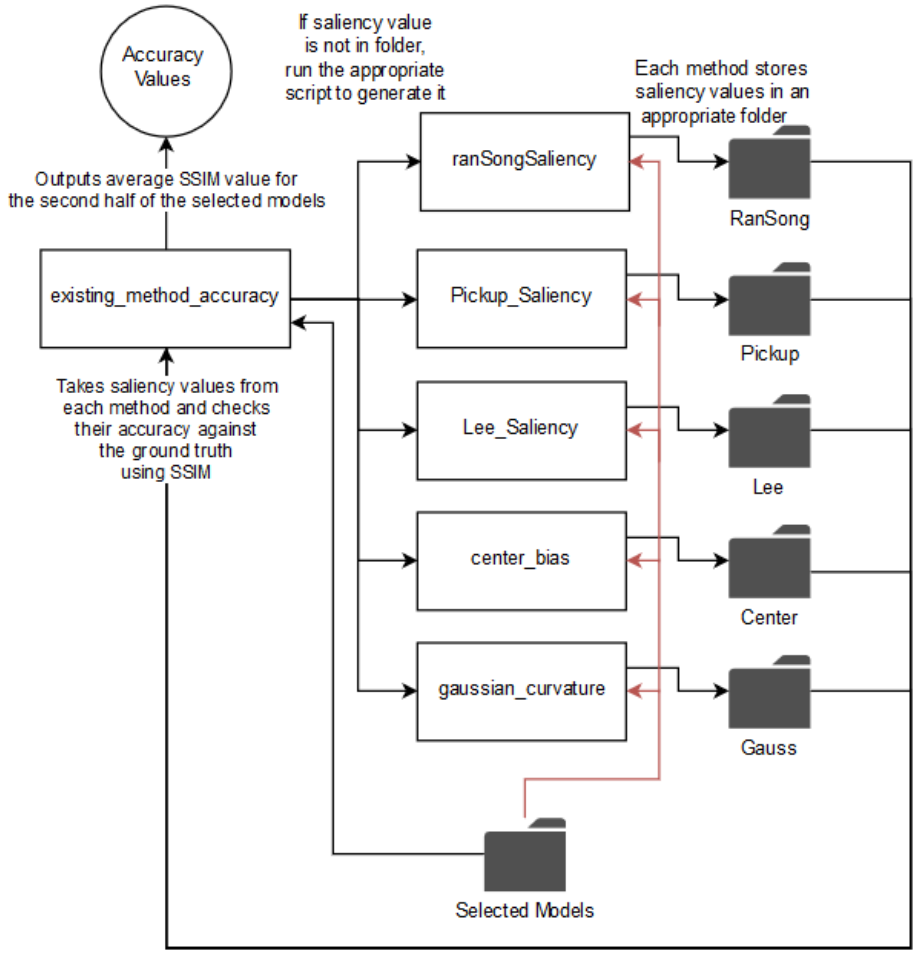


Figure 3 data flow for existing method and evaluating scripts.

The existing_method_accuracy script takes in a list of method names and for each method on that list it will check for the second half of the selected models if a saliency map has already been produced by the method and if it has not it calls the method to create one. Once a saliency map has been obtained it will call SSIM_mesh_helper which simply loads two saliency maps and then calls SSIM_for_3D_Saliency which runs the modified SSIM code which works on 3D meshes. The two maps being passed into the mesh helper will be the method being tested and the ground truth data from the eye tracking experiment. Once this process has been completed for the second half of the selected models each method is assigned the average of its SSIM values across the test set.

Not all the existing methods were straightforward to use, to use Lee et al. method the models first needed to be simplified as the method required too much memory otherwise. This then means that a saliency mesh is returned that is for a simplified mesh. to generate a saliency map for the initial mesh the simplified saliency map needs to be 'patched' onto the larger mesh. to achieve this, I find the closest vertices on the initial mesh to each of the vertices on the simplified mesh. these vertices are now the centre of each patch. While there are still unclaimed vertices each patch claims all unclaimed vertices on the border of the patch. Every vertex inside a patch is then given the saliency value of the vertex on the simplified mesh that started that patch, resulting in a full patched saliency map as shown in Figure 4 to the right.

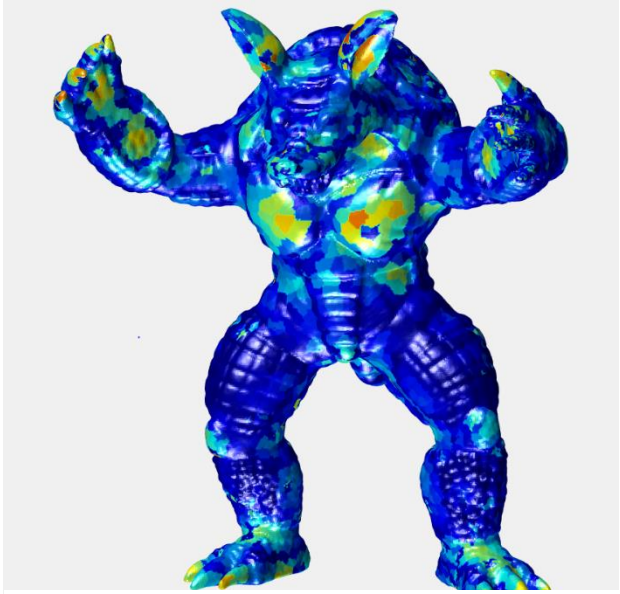


Figure 4 Example of Patched Lee et al. Saliency method

Modifying SSIM to work on 3D heat maps required a change in SSIM's neighbourhood. When working on 2D images it took a window around each pixel, but this is not sufficient for meshes. pixels are always uniformly spaced apart and are perfectly uniform in size, neither of these facts are true with vertices. I developed a new window for 3D SSIM where the neighbourhood was defined as vertices within a set distance from the initial vertex if it can be reached via vertices also within the distance. This neighbourhood is very similar in design to the remapping neighbourhood with the only exception being its size. The remapping neighbourhood uses approximately 5% of the mesh whereas the SSIM neighbourhood uses approximately 2%. Vertices differ from pixels in the area they cover as well so to address this rather than just multiplying the salience score by the gaussian distance from the initial vertex each score is multiplied by its vertex's Voronoi area. Which is the area around a vertex that is no closer to any other vertex. In a triangular mesh this is very easy to compute. The Voronoi area of a vertex in a triangular mesh is one third of the sum of the area of all the faces the vertex is attached to. Using the Voronoi area as a scale compensates for the fact that some vertices have more influence than others.

Learning a new method of measuring saliency

Approach

There will be two separate learning methods this project will use to produce a more accurate method of measuring saliency. Both will train on the first half of the selected models and test on the second half. The first method is least squares regression, a straight forward method that for its data set minimises the mean squared error of each method supplied to it against the ground truth by linearly scaling each method by a weight. These weights can then be applied to each methods saliency map of the test set models to produce a saliency map weighted by the learnt model. SSIM can then evaluate the accuracy of the weighted method against the ground truth for the test set.

The second method is using MATLAB's feedforward network in the neural network toolbox. As with the least square regression method all the ground truth data and the existing method data for the first half of the selected models is passed in. the feedforward network will then train on this data until it cannot improve anymore. the network will then simulate output for the remaining second half of the selected models and SSIM will assess its accuracy to the ground truth. As the feed forward network can take a different number of nodes, several different nodes will be tested to see what the optimum number of nodes are for this problem.

Implementation

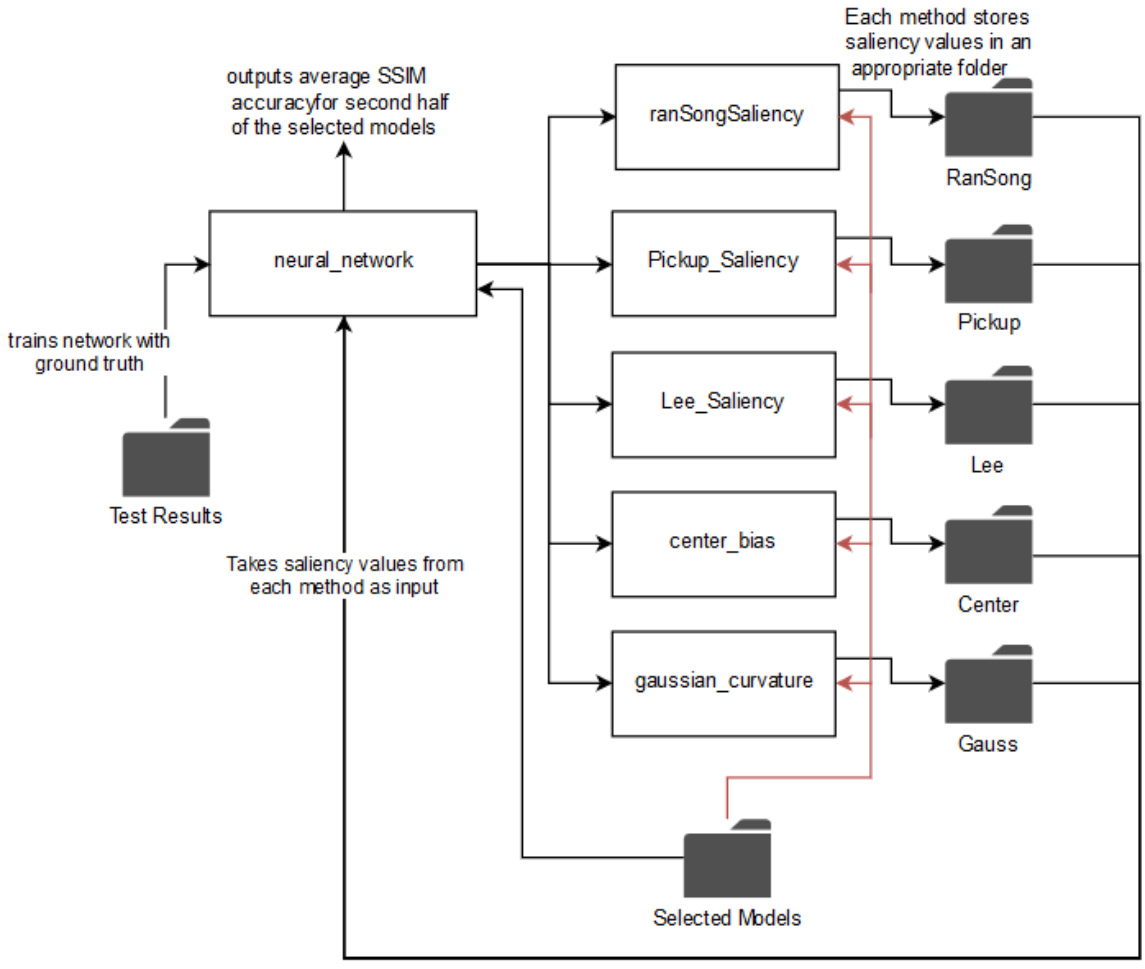


Figure 5 data flow for the neural_network script

The neural network being used for this learning method is a feed forward neural network. I decided to use a shallow network rather than attempt deep learning with such a large dataset given the time frame of the project I felt I would probably get better results working with a shallow network. I tested out MATLAB's newgrnn radial network, but it could only take a small subset of the data due to memory constraints. Using random down sampling seemed a little too arbitrary so I opted for MATLAB's feedforwardnet. several different runs of this network will be done to find what the optimum number of nodes to get the most accurate output is. The network will be trained on the first half of the network and tested on the second half of the network to keep things consistent with the other existing method evaluations.

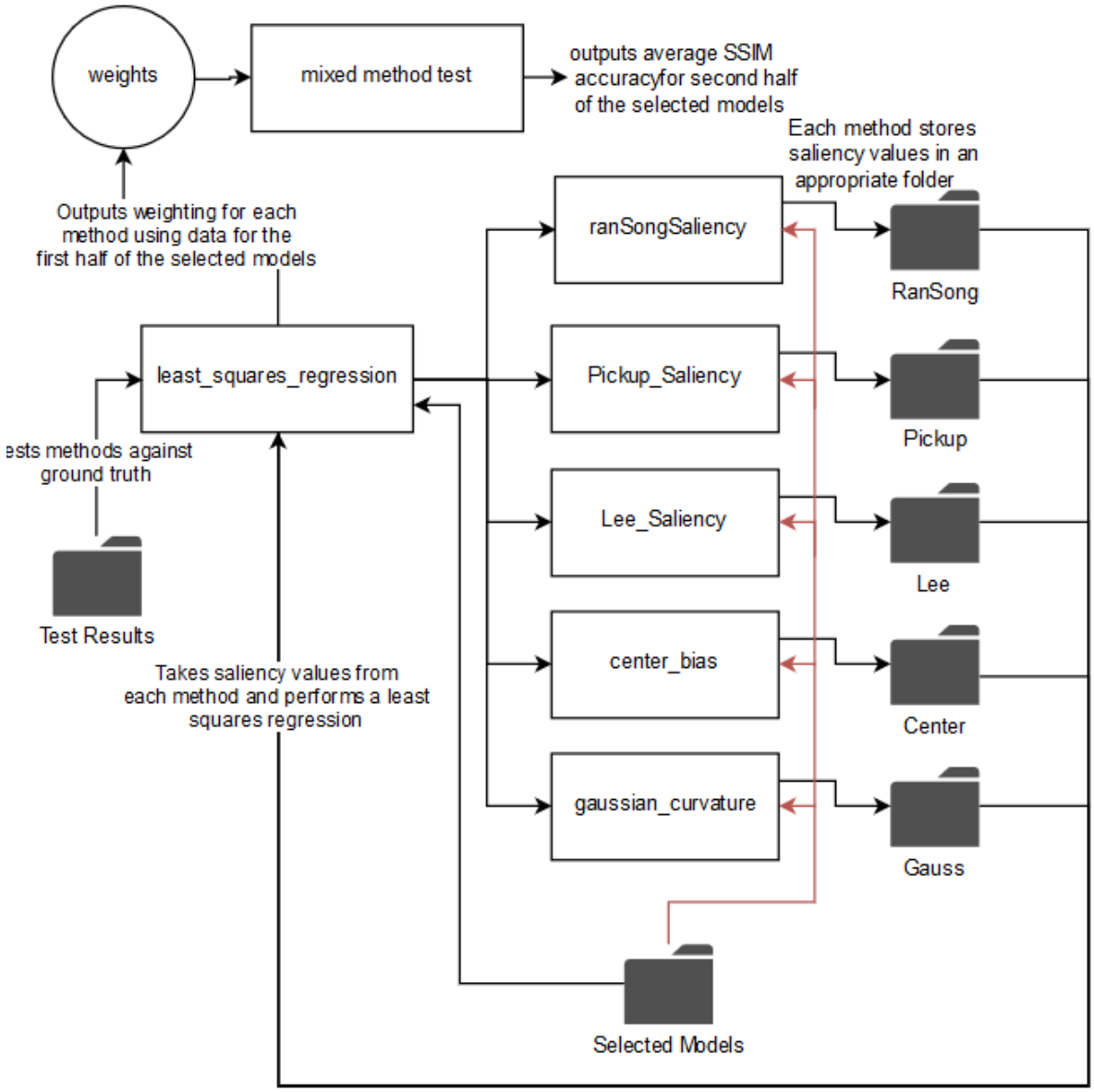


Figure 6 data flow for least square regression learning method

The least square regression learning method is relatively straight forward. Feed it the dependent data being the eye tracking experiment data and the independent data being the method saliency maps for the first half of the selected models. Then find the weighting for each method and an intercept that minimises the squared error to the eye tracking data. The least_squares_regression script then outputs an array of weights with one value corresponding to each method passed in and one extra intercept variable. To compute the array of weights a least squares fit is applied to the data. To do this all the existing method saliency maps for the first 20 models are loaded in as the independent variables and the ground truth for the first 20 models is loaded in as the dependent variable. A covariance matrix is made so that each method or ground truth data has a covariance value with every other method. Two matrices are made from the covariance matrix. A is all but the last column and row of the covariance matrix and C is the last column of the covariance matrix. The weights are computed by the following formula

$$weights = A^{-1} * C$$

The intercept is then calculated as

$$intercept = mean(dependent) - mean(independent) * weights$$

Once these calculations are complete the weights array will then be passed to the mixed_method_test script which combines each methods saliency map linearly scaled with the corresponding weight. The mixed saliency map will then be sent along with the ground truth map to SSIM to be evaluated. Just like the existing methods this will be done for each model in the second half of the selected models. My biggest concern with this method is that to keep things accurate with the comparison model the data must be normalised to -2 to 2 which makes the intercept part of the least squares method relatively irrelevant.

Results and Evaluation

Eye tracking results

I used 20 different models as the dataset for this project some models came from the Stanford repository and some others came from the AIM@SHAPE-VISIONAIR shape repository. A full list of models and where they were acquired can be found in the appendix section of this project.

Running experiments on the eye tracker for this project was done in two parts. The first half of the eye tracking data was collected during the CUROP project and the second half was collected during the dissertation period. In the end I managed to get 23 participants to do the experiment. For some meshes the saliency map seemed to be very concentrated in one or two areas and yet for some other meshes the saliency map was very spread out across the mesh.

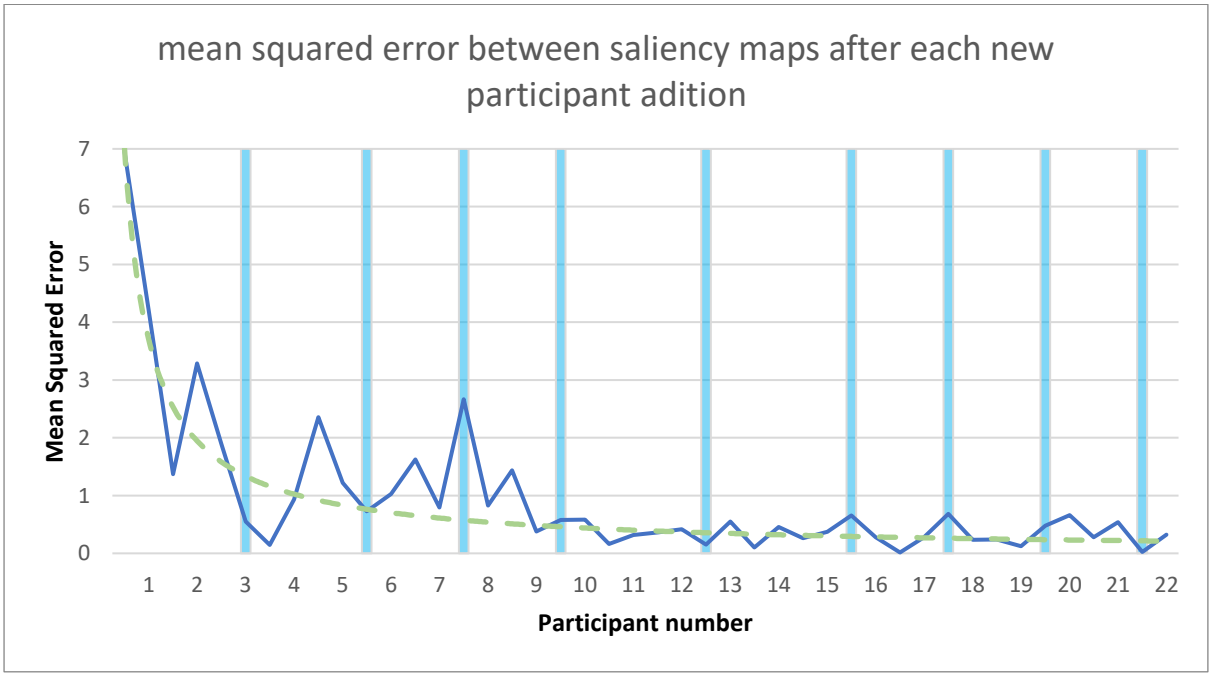


Figure 7 Graph showing error convergence of eye tracking data. blue bars represent when a new set of views is recorded.

To see if my data set was large enough to get reliable results I calculated the mean squared error between the saliency map before and after each participants data was added during the remapping stage. The resultant graph of the mean squared error for each addition to the saliency map shows that the dataset is converging after around 10-15 participants. This means that I had enough participants to confidently produce an accurate human saliency map of each model. It is even potential as the dataset converged so early that more models could be introduced to increase the training sets for the learnt methods. Due to the way the remapping code runs it processes all data

from one group of the experiment before moving on to the next. As each group contains certain views of a model, once a new group is being processed the error will increase drastically as a whole new set of vertices are visible to participants in different groups. The blue bars in Figure 7 Graph showing error convergence of eye tracking data. blue bars represent when a new set of views is recorded. where these group changes occur.

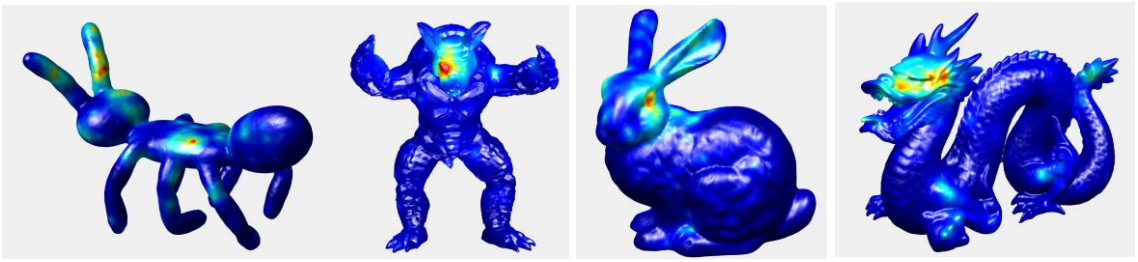


Figure 8 examples of remapped eye tracking data, reds and yellows are salient areas, greens and blues are non-salient areas. from left to right ant, simplified armadillo, bunny and dragon

Models with obvious facial features drew most of the attention from participants. As shown in Figure 8 above, the ant's saliency map is far more spread out than the other three models that all have a very high salience around the face. This result is to be expected however, on models with faces very little else is regarded as salient. This could be due to one of two things. The first possibility is that as each participant will look at the face at some point but the secondary points they look at are not as commonly shared with other participants the saliency values around the face massively outweigh any other values. This means that when the saliency map is normalised to -2 to 2 areas that are not in the facial region have a negligible salience value. A way to circumvent this would be to use a non-linear scaling so that super salient areas don't drown out the rest of the dataset. The second possibility is that people naturally look at the most salient part of a scene first. In the case of these models the most salient area might well be the face, but as participants only have 5 seconds to view each image they may not have time to look at other slightly less salient areas on the mesh. a possible solution for this would be to increase the exposure time of each image and give fixations at the beginning of a viewing a higher saliency weight so that the first things viewed are given more salience than a point viewed at the end of the viewing time.

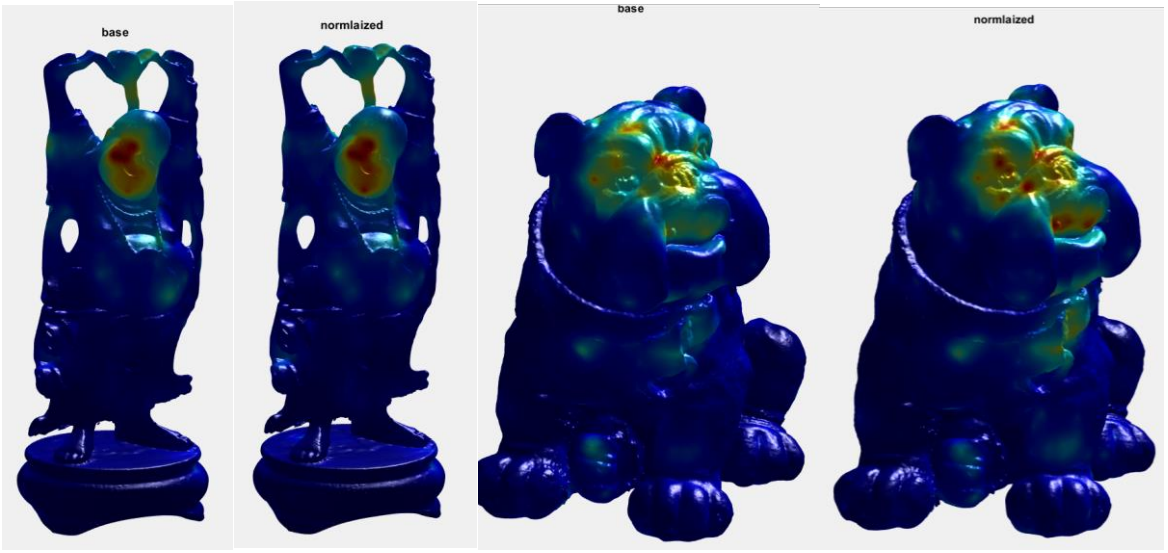


Figure 9 comparison of saliency maps before and after normalising views. From left to right happy(before), happy(after), bulldog(before) and bulldog(after)

Normalising views seemed to work quite nicely, salient areas that get a lot of attention from multiple different views are enhanced like both the faces of the examples in Figure 9. Views where people's attention was not directed towards the centre of the image had a weaker impact, for example happy buddha's sleeve must have been fixated on from a view that consistently looked at other areas and so what is intuitively a boring area of the model is considered less salient as other fixations from that view looked elsewhere.

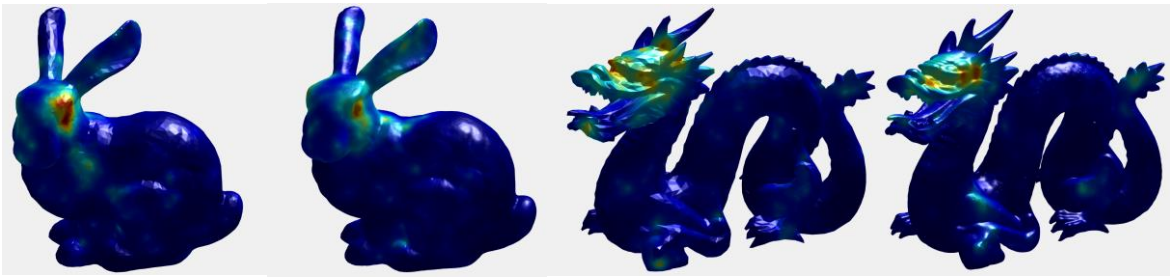


Figure 10 comparison between original and simplified saliency maps. from left to right: simplified bunny, bunny, simplified dragon, dragon.

It seems that there is little to no difference in how people perceive meshes weather they are simplified or not. As can be seen above in figure 10 both the original mesh and the simplified mesh have very similar saliency maps. there are slight differences in where the minor saliency points are like the simplified bunny's shoulder vs the original bunny's upper back however I feel that with a large enough dataset these minor saliency points would even out and both saliency maps would look near identical. This would imply that fine detail does not play a major role in where people look for these kinds of meshes. if this fact holds true it could improve run times for methods for measuring saliency as they can work on simplified meshes without losing accuracy.

Overall, I think measuring saliency of 3D models using the eye tracker worked well. Having a better idea on detailed meshes of some mid-level saliency rather than one massively salient area would be a major improvement on these saliency maps. the biggest downfall is the run time, remapping the eye tracking data to the models took an incredibly long time. Most of this run time is down to finding the neighbourhood around a vertex, finding 5% of the model every time a fixation is made resulted in code that for larger more detailed models took a very long time to run.

Existing method Evaluation

Due to time constraints and how long running the SSIM for 3D saliency can take for the larger models for evaluating existing methods and the learnt methods I will be using 8 models rather than the planned 20. The models chosen to be used are kitten, spot, teapot and sheep along with their simplified versions.

	Gaussian Curvature	Centre Bias	RanSong	Lee	Pickup
SSIM average	0.083	0.007	0.063	0.067	0.137
Standard deviation	0.110	0.139	0.079	0.081	0.112

Figure 11 table of average SSIM values and standard deviation for each method of measuring saliency

Centre bias performed very poorly however I do not think this is due to people not looking at the centre of a model. The only mesh in the test set I had time to run that does not have a face is the teapot which is spherical in shape centred on the origin meaning that most of the mesh is a long way from the centre. This means the centre bias method has very little influence on this dataset. As all of the meshes in this test set are quite smooth the centre surround mechanisms in Lee et al. difference of gaussian and spectral processing by Song et al are of less use as there is very little variance in surface curvature. Due to this consistent curvature of the limited dataset the Gaussian curvature primitive was quite accurate as there are only a few areas where curvature is different, and these areas typically are salient as they are the only areas on the mesh that are not a consistent round shape. The conformal factor saliency method by Mirela et al. as shown in Figure 12 below the conformal method leaves the majority of the mesh with no salience value at all. This will contribute to a higher SSIM score as the best part of the ground truth is close to zero salience. The edge areas that the conformal factor measures as salient are consistent with the eye tracking data in this case however there are a lot of salient points around the base of the spout that the conformal factor does not account for.

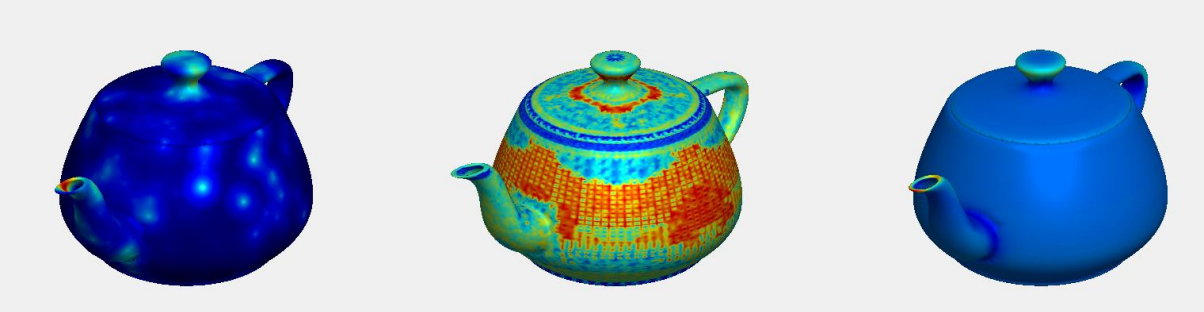


Figure 12 SSIM comparison between ground truth and conformal factors method by Mirela et al. from left to right: ground truth, SSIM index map, conformal factor, this comparison had an SSIM score of 0.1030

Learnt method Evaluation

The least squares regression method achieved an average SSIM score of 0.0857 which out-performs all but the conformal factor method. The reason for least squares underperforming the conformal factor could be because in the training data the conformal factor offered very little accuracy in terms of salience. The saliency map produced by leas squares shown in Figure 13 however does to fit the general pattern of the ground truth data aside from the lid being marked as salient when virtually no one looked at it. I feel with a better scaling function to get the data in the range of -2 to 2 to preserve the low salience zones the least squares regression model would be a lot more accurate.

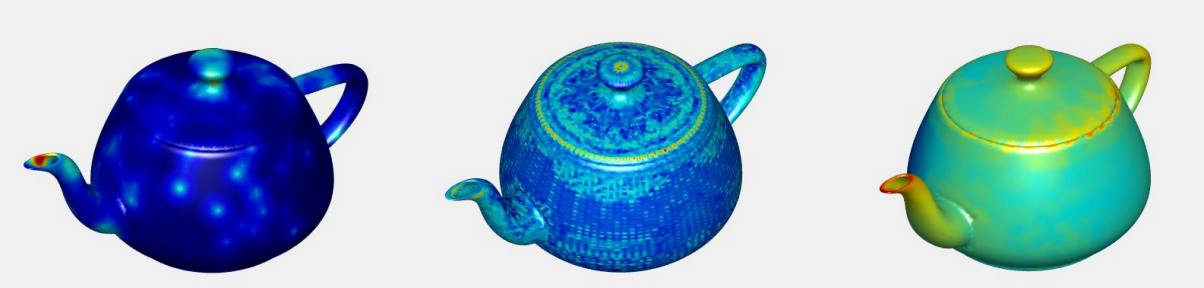


Figure 13 SSIM comparison between ground truth and least squares regression method. from left to right: ground truth saliency map, SSIM index map, least squares learnt method saliency map. This comparison had an SSIM score of 0.0346

The feed forward network, as demonstrated in Figure 14 below produces a saliency map that gives high saliency to sharp protrusions on the model and the area around the neck which is probably due to the necks gaussian curvature. The method gives very low saliency to flat areas or large protrusions

like the cow's legs. Using the most effective number of nodes which was found to be 6, the average SSIM score of the feed forward network was 0.1348 which is much better than the primitives and the difference of gaussians and spectral processing. It was still not quite on par with the conformal factor method, but this may simply be due to the subset of the test data I was able to use is all very smooth and the training data was not this smooth, so the network did not learn to make the most of the conformal factors method.

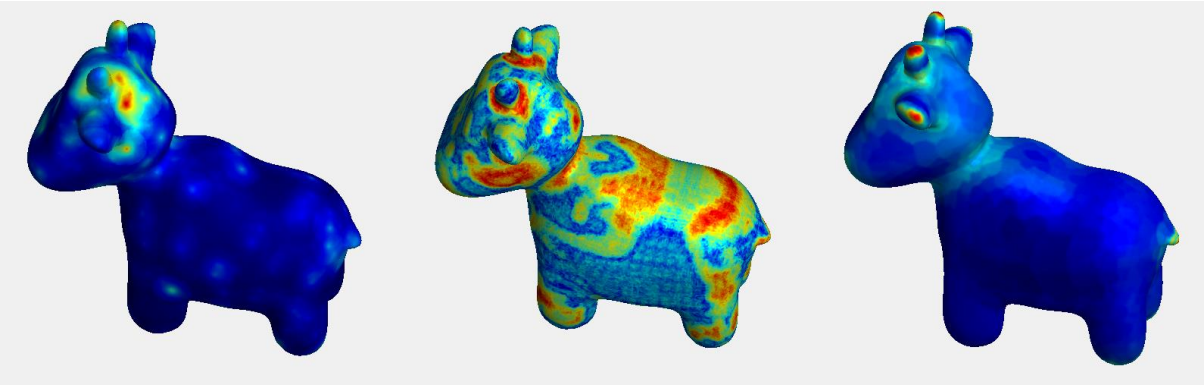
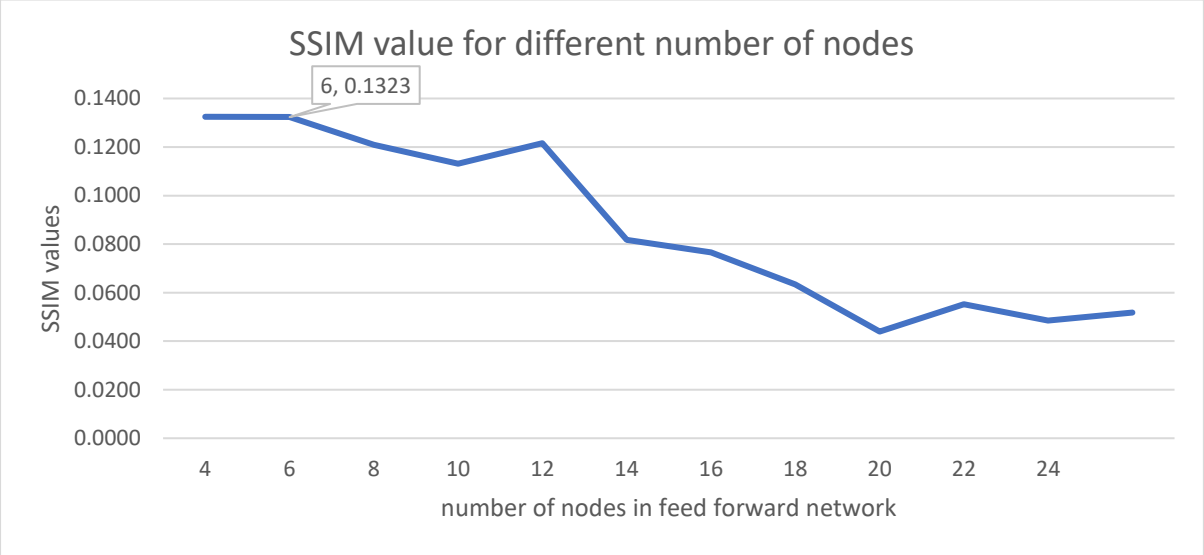


Figure 14 SSIM comparison between ground truth and feed forward network. from left to right: ground truth saliency map, SSIM index map, feed forward learnt method saliency map. this comparison had an SSIM score of 0.3235 this version of the network had 6 hidden nodes

The graph below shows the average SSIM performance over the test data set for the feed forward network trained on the first half of the selected models with different number of nodes to find what the optimal number of nodes for this problem is. The feed forward network seemed to overfit the data very quickly. There was an improvement from 4 nodes to 6 nodes, so we know that the network does not require even fewer nodes. The optimum number of nodes for this problem was 6, this network outputs plausible saliency maps that at times can be very competitive (SSIM score of 0.3235, highest attained score so far) but has a high standard deviation (0.112845) so it is not the most reliable method of measuring saliency, but It can predict human saliency very well for some models.



Future Work

There are several different directions that I feel this project could make progress towards. The first would be to create a simple UI interface to make using the various scripts this project relies on a lot simpler for other users. It would also be nice to have the time to run the evaluation tests on the remaining 12 models that could not be finished within the timeframe of this project.

I would be interested to attempt running 2D methods of measuring saliency on each of the 20 images produced for the eye tracking experiment and then remapping the 2D saliency map onto the 3D mesh. this would allow for a much broader range of potential methods to add to the learnt method. Being able to use 2D methods of measuring saliency would also mean access to object recognition methods, for example a method that detects faces. This would prove to be very useful for the learning methods as faces have proven to be very salient.

Conclusions

This project set out to take a group of 3D models and gather eye tracking data on them and remap the eye tracking data onto the 3D models to produce a saliency map on the 3D models. This functionality can be slow on large meshes but works well none the less. This project also aimed to evaluate existing models of measuring saliency and learn a new method of measuring saliency from existing models. existing models have been evaluated and new methods of measuring saliency have been developed, unfortunately time ran short and I could not evaluate the existing models and the learned methods on the entire test set however some tests were done that showed promising results. Perhaps over a larger dataset and with more methods to learn from, a learnt method could prove to be undoubtedly more accurate at predicting human saliency.

Reflection on Learning

Over both CUROP and this dissertation this project has challenged and intrigued me with several difficult and unique problems. Solving the compatibility issues and hardware limitations that came with running other people's research code has improved my ability to find compromises rather than just accepting that it wont work. By tackling this problem, I have developed my ability to break down a large problem into smaller parts that as a part are simpler to solve and allow for easier maintenance in the future.

Appendices

List of models

The following meshes were taken from the Stanford repository found at:
http://graphics.stanford.edu/data/3Dscanrep/

Armadillo
Bunny
Dragon
happy

Head = http://graphics.cs.williams.edu/data/meshes.xml#2
teapot = http://graphics.cs.williams.edu/data/meshes.xml#2

The following meshes were taken from the SHREC 2011 shape retrieval contest dataset found at:
http://www.itl.nist.gov/iad/vug/sharp/contest/2011/NonRigid/data.html

falling
ant

The following mesh was taken from Keenan's 3D model repository which can be found at: http://www.cs.cmu.edu/~kmcrane/Projects/ModelRepository/

Spot

The following meshes were taken from the AIM@SHAPE-VISIONAIR shape repository which can be found at: http://visionair.ge.imati.cnr.it/ontologies/shapes

bulldog model is provided courtesy of VCG-ISTI
frog model is provided courtesy of Frank_terHaar
kitten model is provided courtesy of Frank_terHaar
Red circular box model is provided courtesy of INRIA
chair model is provided courtesy of IMATI
cup model is provided courtesy of MPII
sheep model is provided courtesy of Frank_terHaar
Ramesses model is provided courtesy of IMATI
gargoyle model is provided courtesy of VCG-ISTI
raptor model is provided courtesy of INRIA
grog model is provided courtesy of VCG-ISTI

References

Itti, L., 2007. Visual Salience. *Scholarpedia.*

Itti, L. & Koch, C., 2000. A saliency-based search mechanism for overt and covert shifts of visual attention. *Elsevier.*

Judd, T., Krista, E., Frédo, D. & Antonio, T., 2009. Learning to Predict Where Humans Look. *International Conference on Computer Vision.*

Lee, C. H., Varshney, A. & Jacobs, D. W., 2005. Mesh Saliency. *SIGGRAPH.*

Mirela, B.-C. & Gotsman, C., 2008. Characterizing Shape Using Conformal Factors. *Eurographics Workshop on 3D Object Retrieval.*

Peyre, G., 2007. *Toolbox Graph MathWorks File Exchange.* [Online]
Available at: https://uk.mathworks.com/matlabcentral/fileexchange/5355-toolbox-graph
[Accessed 23 June 2017].

Pickup, D., Sun, X., Rosin, P. L. & Martin, R. R., 2015. Euclidean-distance-based canonical form for non-rigid 3D shape retrieval. *Elsevier.*

Song, R., Liu, Y., Martin, R. R. & Rosin, P. L., 2014. Mesh Saliency via Spectral Processing. *ACM Transactions on Graphics.*

Wang, Z., Bovik, A. C., Sheikh, H. R. & Simoncelli, E. P., 2004. *Image Quality Assessment: From Error Visibility to Structural Similarity.* s.l.:IEEE transactions on image processing.