VIRTUAL ASSET PROTECTION USING VIEW-BASED SIMILARITY MATCHING

WILLIAM RAFFE

Supervisors: JIANKUN HU AND FABIO ZAMBETTA

Consultant: KAI XI

Honours Thesis

School of Computer Science and Information Technology
RMIT University
Melbourne, AUSTRALIA

October, 2009

Abstract

The copyright protection of virtual assets in massively multiplayer online games that utilise user generated content is an underappreciated research field. Current processes are manual, requiring users and business who have designed a virtual asset to lodge complaints or initiate legal action to protect their copyright. We believe that this system should be automated in order to promote population growth and creative business ideas in the virtual world. In this paper, we specifically examine the asset type of 3D models. To achieve the protection of these assets, we propose that when a new model is introduced to a system, it is checked for similarity against all other existing models in the system, reporting a copyright infringement if the similarity is too high. We approach the issue of judging similarity between 3D models by first examing the field of content driven data retrieval systems. We use a View Based method, gathering 2D projections so that we can work with pixel data rather than 3D vertex coordinate data. This allows us to investigate the use of advanced techniques from the fields of biometrics and computer vision. Primarily, we use the Correlation Filtering and Hausdorff Distance algorithms to detect similarity between two 2D projections. These algorithms perform poorly in our early experiments so we introduce a clustering structure which improves both efficiency and effectiveness. Our final system produces promising results and provides a genuine acceptance rate (GAR) of 88%. However, this is an initial step in a large body of research. Our system is not at a point where it can be automated and efficiently executed in a game environement and we finalise this paper by discussing future work towards progressing this research.

Contents

1 Introduction

In recent years, interactive media has become a multi-billion dollar industry (Hu and Zambetta, 2008), with global video game sales rivalling other top contending media such as movies and music. A subcategory of these games is one that allows the game to be played over the Internet. Multiplayer online games (MOGs) present a virtual realm in which players can play together, compete against each other, or simply provide a new means for friends to communicate and express themselves all through the use of Internet based technology.

Massively multiplayer online games (MMOGs) are yet an even stricter classification. MMOGs are typically categorised as having an extensive population of players and having at least some small form of persistence. This persistence is usually manifested in the form of each player having an avatar, or in other words, a character that represents them in the virtual world. This avatar will usually have a consistent name and appearance even if the player is to logout of the game and log back in. However, persistence in these virtual worlds has extended to allow for players' avatars to have belongings and characteristics such as virtual items, currency, real-estate, professions, rank, skills, and abilities. These various persistent attributes, including the avatar itself, are all grouped under the title of virtual assets. However, persistence does not only apply to individual virtual assets, it also refers to the state of the virtual world, with more and more games and interactive online environments having consistent politics, economies, and landscape that change gradually over time just as they would in the real world. It should also be noted here that while this paper uses the term MMOG, these virtual worlds are not always used in the classical sense of a 'game'; some of these virtual worlds are purely designed for social interaction, much like social networking websites, and they have no intended objectives or rewards.

MMOGs are becoming increasingly popular and more and more software developers are investing in creating their own virtual worlds. It is not uncommon to see online communities with populations in the thousands, sometimes even hundreds of thousands, subscribed to a single virtual world at any one time. This raises a few opportunities and challenges. With such large populations, there exist business potentials that as of yet have been largely neglected. At the same time, there exists increasing security concerns. The real world value of virtual assets can be surprisingly high and so measures must be taken to safeguard them from theft and other malicious acts

In our research we focus on protecting the asset type of 3D models that may be created by a user and shared through a user content driven multiplayer game. These 3D models make up the majority of the graphical display that is presented to players. We hope that when a new 3D object is introduced to the virtual world, it can be checked against all other existing 3D models in the game; if the new model is too similar to an existing one and cannot be considered unique, then it's Instantiation is denied or reported for inspection by an administrator.

We borrow a few ideas from the related field of research of content based data retrieval systems, such as search engines that allow for media to be submitted rather then a text based descriptions. In order to do this, we must be able to compare two 3D models and get a similarity measurement. There are many approaches to this but we have chosen to use 2D projections of the models, effectively allowing us to work with pixel information rather then working directly with 3D coordinate data. This allows us to utilise techniques such as correlation filtering, which has been popularised in biometric authentication, and Hausdorff distance, which has applications in object detection in computer vision.

1.1 Background of Massively Multiplayer Online Games (MMOG)

Real-money trade (RMT) of virtual assets is a growing topic of importance. Analysis of the value of these assets is increasing as individuals, industry, and research groups begin to identify business potentials within the field but also recognize the economic impact of RMT on the real and virtual worlds. However, the topic is still highly controversial with RMT being considered by some players as cheating in certain games.

RMT trade is already an existing market with external websites such as eBay.com and PlayerAuctions.com facilitating the sale of virtual currency, items, high ranking avatars, and services promising to improve an avatar within a certain time frame. PlayerAuctions.com provide direct purchases and sales of virtual currency, thus offering a superficial exchange rate between virtual currencies and the American dollar. These exchange rates, however, may lead to a practice that Hu and Zambetta (2008) term as 'gold farming'. Gold farming involves users playing a game merely to gain virtual currency and other assets that they can sell later on. This process runs the risk of turning a game into more of a business rather than an environment for entertainment and social interaction, leading some users to complete repetitious tasks within the game to build their assets faster rather then interacting with the rest of the community as the game developers may have intended.

As controversial as RMT may be in some games, virtual worlds do exist where it is actively encouraged as a way of growing the community and supporting creative business potential. Second Life, developed by Linden Labs, is one such game. Rymaszewski et al. (2007) provides a comprehensive look at Second Life, how to use it, the culture, the changing environment, the history, and the economics. Second Life is not so much a game as it is a virtual world that mimics many of the natural and legal systems of the real world. There are no objectives in Second Life but it israther just a place for social interaction. What makes an MMOG different from social networking websites though is the ability to be represented by a 3D avatar and own virtual assets such as clothing, land, architecture, and other miscellaneous objects. What separates this Second Life from most MMOGs though is that these assets, with the exception of land ownership, are created and traded by the citizens of the virtual world, not the developer of the software, Linden Labs. With a population in the millions at it's peak, Second Life is the most popular user content driven MMOG to date.

There have been a few significant events in Second Life's policy since it's release in early 2003. Originally, participation in the Second Life virtual world relied on a monthly membership system much like most other MMOGs. In mid 2003, the developer, Linden Labs, introduced a tax to curb the rate of objects being instantiated in the world in an attempt to alleviate over activity on the servers. However, the tax was widely protested by users as discouraging creative involvement in the world and on November 14th 2003 the tax system, along with the mandatory monthly subscription system, were abandoned. Instead, Linden Labs began charging for land ownership, growing a virtual real-estate industry and relaxing restraints on users who did not want to pay for the use of the software. This date also marks a signiticant change in Second Life's ownership laws as intellectual property rights were now recognised for object designs that users created. This meant that 3D objects, 2D textures, audio, and programming scripts could now be created, copyrighted, and sold in the virtual world. As we have seen, the advent of RMT allows for businesses in Second Life to make real world profits and in 2006 at least one user had a virtual empire worth US$1million[1], while by 2009 at least a dozen users were reported as having reached the same achievement[2].

RMT also introduces many security concerns. Lehdonvirta (2005) acknowledges as virtual as-

[1] http://www.businessweek.com/the_thread/techbeat/archives/2006/11/second_lifes_fi.html

[2] http://nwn.blogs.com/nwn/2009/03/million.html

sets in a game begin to gain real world value, less reputable parties will attempt to find exploits or security loop holes in order to gain assets without earning them through typical gameplay or trade. Muttik (2008) also provides statistical evidence to show that the password stealing Trojans targeted at popular MMOGs are a prominent security concern and that the number of these Trojan instances is fast approaching those used to steal password information about bank accounts. This is most likely because, while the real world value of virtual assets is increasing, security measures are not advancing at a proportional rate and thus attempting to steal passwords to a game could prove more profitable and less risky. Hu and Zambetta (2008) provides a framework into the classification of security concerns in MMOGs and Muttik (2008) gives some straightforward measures that developers can take to increase the security of their systems. However, some security weaknesses in games will require extensive research by academics and advancements by industry in order to rectify the issues.

1.2 Research Problem

While there are many different types of cheating and attacks that plague MMOGs, we focus on those that rob users and businesses of their intellectual property in games like Second Life. We believe that if we can demonstrate secure methods for protecting virtual assets then we will encourage creative communities in these types of games to grow and give developers new possibilities in exploring user content driven games.

In this paper we focus on the protection of 3D models. For example, in Second Life, 3D models can be created by users and uploaded to the server so that they can be displayed in the game world. There is a known attack on this method that is called CopyBot[3], a piece of software that intercepts the communication of 3D mesh data. This is hard to protect against because when a 3D model is placed in the virtual world, all client machines that can view the object need to know how to render the object onto a computer monitor. Even if this transfer of data were to be encrypted, at some stage the 3D mesh data needs to be translated into a form that is readable by a computer graphics card and at this point it can be copied. Thus, a 3D model can be copied without it's original creator knowing and then re-uploaded to the server and redistributed under a different name, amounting to the theft of product blueprints in the virtual world.

While the use of CopyBot is prohibited under Linden Labs' Terms of Service Agreement[4], there exist no automated mechanisms to protect assets. A complaint can be filed with Linden Labs to ban the account of the perpetrator but because accounts are now freely available in Second Life, the attacker can simply launch the exact same attack again. These assets are also protected by the Digital Millennium Copyright Act in the United States and if a user or business feels that they have suffered economic loss from the actions of their attacker, they can file a lawsuit. However, this can be financially costly and some attackers may live in other nations around the world, a situation that complicates the enforcement of the DMCA law. Regardless, we propose that it is always better to automatically prevent a situation from occurring than to require asset owners to manually manage their copyright and require lengthy and expensive legal procedures.

1.3 Research Objectives

For the protection of virtual assets that are in the form of 3D models, we propose a system to detect whether a newly uploaded object is a copy of an existing object on the server. It is also not enough to

[3]`http://wiki.secondlife.com/wiki/Help:CopyBot`

[4]`https://blogs.secondlife.com/community/features/blog/2006/11/15/use-of-copybot-and-similar-tools-a-tos-violation`

check for exact copies, as this approach would be easily circumvented by modifying the 3D model's appearance slightly. Thus, our goal for this research can be summarised as:

- We seek to prevent the registration of 3D models that are copies of, or too similar to, an existing 3D model in an expanding dataset

To accomplish this goal, we primarily investigate the use similarity comparisons between 3D models. Given a new model to be registered, we compare it with models within the existing dataset to find any that may be similar.

In attempting to achieve this objective, we narrow our scope to the research that we set out to complete this year. We propose and answer the following research questions:

- Are the Correlation Filtering and Hausdorff Distance algorithms suitably effective at calculating similarity between 3D models? How do these two algorithms compare to each other in terms of effectiveness?

- Are the Correlation Filter and Hausdorff Distance algorithms efficient enough to be used in a real-time games environment? How do these algorithms compare to each other in terms of efficiency?

- Can we create a structure to improve the effectiveness and efficiency of these algorithms?

- Is it possible for the structure that we introduce to degrade the effectiveness of these algorithms in some instances?

Formal definitions of effectiveness and efficiency are given in Section 4.2 where we discuss the metrics used throughout our experiments. Determining whether or not these algorithms are effective or efficient enough can be quite hard to measure and can be subjective, depending on the requirements of the user and the 3D models that form the dataset. Therefore we use human intuition to judge the results of effectiveness and common sense to judge efficiency. For a frame of reference on how we judge these, view Section 3.1.

We believe that if we are able to answer these questions that we will be making contributions towards:

- Increasing the security of massively multiplayer online games. Specifically, the protection of virtual asset copyrights over user generated content.

- Expanding the knowledge base of content driven information retrieval in 3D model databases.

- Evaluating the Correlation Filtering and Hausdorff Distance algorithms in a unique setting and providing a structure to increase their effectiveness and efficiency.

We borrow a few ideas from the related field of research of content based data retrieval systems, such as search engines that allow for media to be submitted rather then a text based descriptions. In order to do this, we must be able to compare two 3D models and get a similarity measurement. There are many approaches to this but we have chosen to use 2D projections of the models, effectively allowing us to work with pixel information rather then working directly with 3D coordinate data. This allows us to utilise techniques such as correlation filtering, which has been popularised in biometric authentication, and Hausdorff distance, which has applications in object detection in computer vision.

2 Related Research

The field of research that addresses the objectives in Section 1.3 is very narrow and specific to a games environment. Therefore , we have not been able to find any related research that is specific to this topic. Instead, we will expand the scope of our literature survey to encompass matching 3D objects in a content driven retrieval system. These systems can be thought of as a searching algorithm for a database of the 3D models that, instead of taking a text based description of what the user wishes to find, takes a 3D model as the query itself. Text driven queries may be effective for searching text documents, however using them to search for media will require the object to have metadata, such as a label or tag. Labelling objects serves as a poor method of describing media such as images, video, audio, and 3D models because it can be very subjective and non-encompassing. For example, the way one user describes a 3D model may not be the same way another user would describe it. Content driven retrieval systems are powerful because they can objectively classify a media type and give an exact similarity value between query and database objects.

Tangelder and Veltkamp (2008) provides a comprehensive taxonomy of methods used to compute the similarity between two 3D models, which can be used in a 3D model content driven retrieval system. In this section we give a brief summary of different categories of approaches as well as specific methods that we have briefly explored throughout the planning of our own techniques.

2.1 Feature Based

Feature based similarity matching relies on using properties of the 3D models structure to create a numeric or text based key. The features of the model used to calculate the key can be based on volume, geometry of the surface, bounding boxes, edge paths, or even the Fourier Transform of vertex data, a technique we later explore in section 2.5 for the use on pixel data with the Correlation Filtering algorithm. Once the key has been derived, we can use popular, and efficient text and numeric based searching algorithms to detect similarities between models.

2.1.1 Shape Distributions

An example of a Feature Based content based 3D model retrieval algorithms is using Shape Distributions, proposed by Osada et al. (2002). This technique involves using geometric properties to form a probability distribution. These geometric properties are formed by a shape function, such as randomly sampling the distance between any two vertices in the mesh. For low sample rates, this method can be extremely efficient in building the probability distribution. Increasing the number of samples may increase the accuracy with complex meshes but will no doubt require a greater amount of time. However, the theory is that once a probability distribution or a histogram is created as the signature for the model, then all similarity comparisons will be done on these signatures and take a fraction of the time.

The results provided by Osada et al. (2002), however, do show room for improvements when displaying ranked results of a query. Some queries provide results that the human mind would not logically link together. For example, using the 3D model of a couch as a query returns the model of a car, with the system returning a dissimilarity value that indicates high similarity between the two models.

2.2 Graph Based

Another popular approach to matching 3D meshes works by generating a graph structure that describes the geometry of a 3D model. One of the most common uses of these graphs is to map out a skeleton for the model. The skeleton will, of course, have much fewer vertices and edges then the 3D model itself and so calculating similarity between two graphs should be a simpler process. However, Tangelder and Veltkamp (2008) argues that these types of methods can still result in NP-complete complexity if existing graph comparison metrics are used. The Reeb Graph approach appears to be the most popular in this category and we briefly describe an example of it below.

2.2.1 Reeb Graph

The paper presented by Tung and Schmitt (2004) proposes an algorithm whereby the creation of the skeleton is done through the use of one of many functions which dictates how many graph nodes are placed in specific parts of the model. These functions can be based on height, distance from the centre of mass of the model, or geodesic distance. For example, if distance from the centre of mass is chosen as the function, then the further away from the centre of mass, the more nodes are used to describe the graph. Functions such as Geodesic distance works well because it typically depicts limbs with enough nodes to make the skeletal graph accurate and flexible.

These skeletons are then used as a feature which describes the model. From this point on, all similarity comparisons are done on the skeletons rather than the models themselves. This should also have the benefit of making comparisons much easier and faster as the skeleton is made up by a unidirectional graph with an extremely small amount of nodes compared to the number of vertices present in the 3D model.

This appears to be a strong and efficient system. However, for our purposes, we feel that only comparing the skeleton of a model may not be enough to judge the uniqueness of a model. For example, many skeletons for bipedal 3D models, will be very similar, however the difference between them lies in the detail of the shape of their body mass and clothing, something that we feel a skeletal method may very well over look.

2.3 View Based

The final category that Tangelder and Veltkamp (2008) describes is called Geometry Based. However, we feel that many of the methods in this category could also be placed in Feature Based classification. Thus, we focus on one of it's sub-categories, View Based similarity matching.

Utilising mesh data can have benefits. For meshes with a low vertex count it can have better efficieny than a global comparison method on a 2D projection as the number of vertices may be less than the number of pixels in the projection. Also, there is potential for the mesh to be more accurately matched as all vertices may be considered and none are lost to occlusion, which will occur in a 2D projection unless multiple projections are taken from numerous angles. However, many of the example algorithms described in this literature survey do not directly make use of all vertex data, choosing instead to compile a compact signature of sorts from some feature of the vertex data. Many of these approaches are described as having the downfall that these signatures are not able to adequately describe complex or high detail meshes.

View Based techniques are unique in that they don't use structural data about the 3D model but instead gather a 2D projection of the model. This has many benefits of it's own. Firstly, in contrast to the earlier statement, when considering complex 3D models, processing 2D pixel information is usually a lot faster than utilising entire sets of vertex data due to the overwhelming number of vertices.

This is especially true if simple, low cost image formats, such as bitmap, are used and if the resolution of the image is reduced. Secondly, image processing is a very large field of research. If a View Based method is used, it is possible to utilise and build upon an already large body of research in the fields of biometrics, computer vision, pattern recognition, and of course image driven content retrieval systems.

For our own research, we have chosen to explore the use of techniques that have been used in biometrics. Thus, our approach can be classified as a View Based 3d model content driven retrieval system. We begin in Sections 2.5 and 2.6 where we describe the processes of Correlation Filtering and Hausdorff Distance, both of which are pixel oriented pattern recognition approaches that have been tried, tested, and improved in the biometrics research field.

2.4 Watermarking

Watermarking is a possible alternative to shape matching when it comes to protecting virtual assets and 3D media in general. Using watermarking, there would be no need to do similarity searching between objects but instead requireing the watermark to be extracted from the rest of the 3D mesh data and then, using a nave approach, would require one-to-one matching against every other 3D mesh watermark within a given database.

2.4.1 Visible Watermarking

This involves placing a visible trademarked symbol somewhere on the media so that when it is viewed it is clearly identifiable who this media item belongs to. For 3D meshes, this approach isnt practical for numerous reasons. Firstly, if the watermark is too obviously visible, it will distract from the quality of the 3D model. Secondly, it would be quite difficult to make a watermark that is built into the surface mesh of the 3D model. It is more likely that the watermark would be embedded into the texture, which is a 2D image overlaid onto the 3D model, usually to give the model added detail and colour. In this case, it is quite easy to remove the texture from a 3D model and replace it with a new one, effectively throwing away the watermark. The final disadvantage is that automating the process to discover a visible watermark may still require computer vision techniques such as pattern and shape matching discussed throughout this paper.

2.4.2 Invisible Watermarking

Invisible watermarks involve encoding a data structure into the medias data. The data structure should ideally uniquely identify the media, not be able to be separated from the rest of the media's data, and should not change the visual appearance of the media. Watermarking is a well researched topic in 2D media such as images and video and has also been researched for use with 3D mesh data, though the latter is relatively small field thus far.

Approaches such as the one suggested in Ohbuchi et al. (2002) claim that they can work regardless of any linear transforms carried out on the model and that it will even work in the presence of subtle vertex coordinate noise or mesh simplification, which stabilises it against minor vertex position changes. However, it is not clear whether these approaches will handle meshes that have, for example, a slight addition or subtraction from the mesh surface such as an extra appendage. Another downside to the technique suggested in this paper is that by adding the watermark it is possible to distort the mesh and deteriorate the overall visual quality of the 3D model, which can be seen in the results provided by Ohbuchi et al. (2002). Also from these same results, we see that unfortunately the encoding and decoding process of these watermarks can be very time inefficient for 3D meshes with a large number of vertices, which will cause scalability issues. However, considering that it is reported that these watermarks are retained despite mesh simplification, it may be possible to reduce the number of vertices in a mesh in order to improve efficiency.

In theory, the watermarking approaches described by Ohbuchi et al. (2002) can actually be used to create a similarity measurement that resembles the one we are trying to achieve in our research. This watermarking approach is primarily used to discover if a 3D mesh has been modified. The results at the end of the paper show that the algorithm is able to determine how many vertices have been modified from the watermarked object. With further investigation and development it may be possible to use these same techniques to discover how many vertices differ between two 3D meshes and report it as a similarity approximation.

In summary, we do not discount the use of watermarking, or other approaches, as viable techniques to protect virtual assets. However, we are forced to choose a focus for our research and this requires us to forego exploration of watermarking as well as Feature Based and Graph Based similarity matching of 3D models at this point in time.

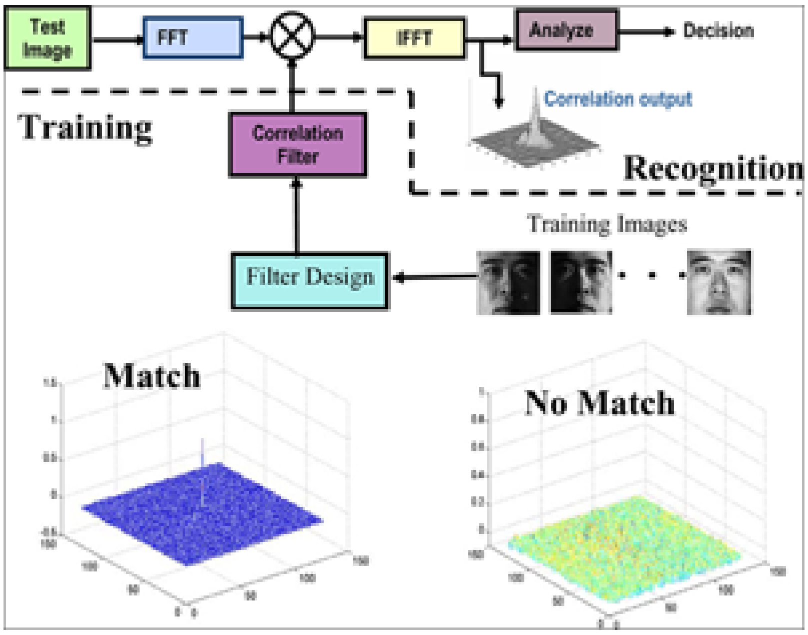

Figure 1: Process of training and using a Correlation Filter. Image provided by Kumar et al. (2006)

2.5 Correlation Filter Background

Using Correlation Filters in image processing is a an active research field and one that benefits from on going contributions. This technique for signal processing through the spatial frequency domain has been investigated since the mid 1960's as a means of pattern recognition and has recently gained popularity as a tool for biometric authentication and identification with authors such as Kumar et al. (2006) making significant contributions. Authentication is a one-to-one matching process that can be used in login systems while identification works with one-to-many searches that, given a biometric sample, will attempt to identify to whom the sample belongs, within a given database of pre-gathered biometrics. Thus, the system we propose in this paper is designed as an identification system; given a 3D model, identify any similar 3D models in a dataset.

Figure 1 shows the general process that a Correlation Filter algorithm undertakes in a biometrics application. The training stage can use the Fourier Transform of one or multiple sample biometric images of a single user in order to create the filter. In this case, it a face recognition system so all the training images are of one persons face. In some Correlation Filter algorithms, we can provide multiple training images with varying facial expressions, ambient lighting, position, and sometimes even slight rotations, thus causing our resulting filter to be more robust to these situational effects. During the testing stage, the Fourier Transform of the test image is correlated in the frequency domain against the designed filter. The result is put through an Inverse (Fast) Fourier Transform and from this point we are able to analyse the energy of the correlation process to make a decision as to whether or not the test image is similar to the training image. Figure 1 also shows the energy output samples of a matching authentic user and an imposter. If a match is to occur, we expect to see a sharp peak in our output energy. In Section 3.4 we will explain further how we analyse this energy.

The paper presented by Kumar et al. (2006) introduces us to some of the basic Correlation Filter implementations, equations behind them, and ongoing advancements in the field. One of the original Correlation Filters that was first designed in the 1960's was the Matched Filter (MF) (Turin, 1960), which had been planned as an optical filter to be implemented in hardware. This filter design is characterised as a pattern recognition system that only uses one training image per filter and is able to handle noise and slight translations of the pattern but performes poorly with rotation, scaling, and distortions of the pattern (North, 1963). A step beyond this is the Synthetic Discriminant Function (SDF) Filter (Hester and Casasent, 1980) that allows for multiple training images to be used in the creation of a filter, allowing the Correlation system to be more encompassing of variations in sample quality and environment. The process described in Figure 1 more accurately represents a SDF filter than an MF. A more refined version of a SDF filter is a Minimum Average Correlation Energy (MACE) filter (Mahalanobis et al., 1987), which produces a more defined peak in authentic cases and is more robust against translation of the object within the image space. There are more advanced filters than MACE, however we choose to use this filter in our system due to it's trade off between simplicity of implementation and it's boasted effectiveness.

2.6 Hausdorff Distance Background

Hausdorff Distance is a shape discovery and matching algorithm commonly used in computer vision. At it's core, Hausdorff Distance is used to determine the distance between two polygons on a coordinate plane. Figure 2 shows the process of calculating Hausdorff Distance between two shapes on the same 2D plane. First, the distances between a point of A and each point of B is calculated and the shortest distance is chosen, discarding the rest. This process is then repeated for all points of A. Once all these minimum distances are found, we then select the longest distance out of this set. This is the distance between polygon A and polygon B and can be represented by the equation:

$$\begin{aligned} h(A,B) &= \max_{\forall a \in A}[\min_{\forall b \in B}(\|a-b\|)] \\ & \qquad where\ \|a-b\|\ indicates\ the\ distance\ between\ two\ points\ a\ and\ b \end{aligned}$$

This is not our final Hausdorff Distance however because $h(A,B)$ is not equal to $h(B,A)$. In Figure 2, the red line represents the distance $h(A,B)$ but the distance $h(B,A)$ will most likely be from the point at the right angle of the triangle to the point of the square in the middle of the image and therefore will have a different distance value. Thus, the Hausdorff Distance is traditionally taken as the maximum of these two distances in order to represent the furthest distance between these two polygons. We can modify the above equation to represent this:

$$\begin{aligned} H(A,B) &= \max(h(A,B), h(B,A)) \\ & \quad where\ h(A,B) = \max_{\forall a \in A}[\min_{\forall b \in B}(\|a-b\|)] \end{aligned}$$

When used in image processing, we can think of the process as overlaying the two shapes on top of one another and then calculating the distance between bright pixels on one shape and bright pixels on another. For our purposes, this means first getting images of the two models involved in the calculation. These images should be the same resolution and the 3D models should be centred within the image space. We then calculate the edges of the model in the image by using an image detection algorithm such as the Sobel method (Heath et al., 1998). An example of this can be seen in Figure 5. This is done so that we do not need to consider every effective/bright pixel in the image, rather just the pixels that define the edge of the model. We then calculate pixel distance from each pixel of the edge of one model to each edge pixel of another and follow then Hausdorff Distance algorithm described above. Thus, if the two images are exactly the same, then the edge points of the two images will be in the same pixel positions, resulting in distances of zero for every point and therefore a final Hausdorff Distance value of zero. Therefore, we actually classify Hausdorff Distance as a disimilarity measure because as the distance value increases the two images are considered less and less similar.

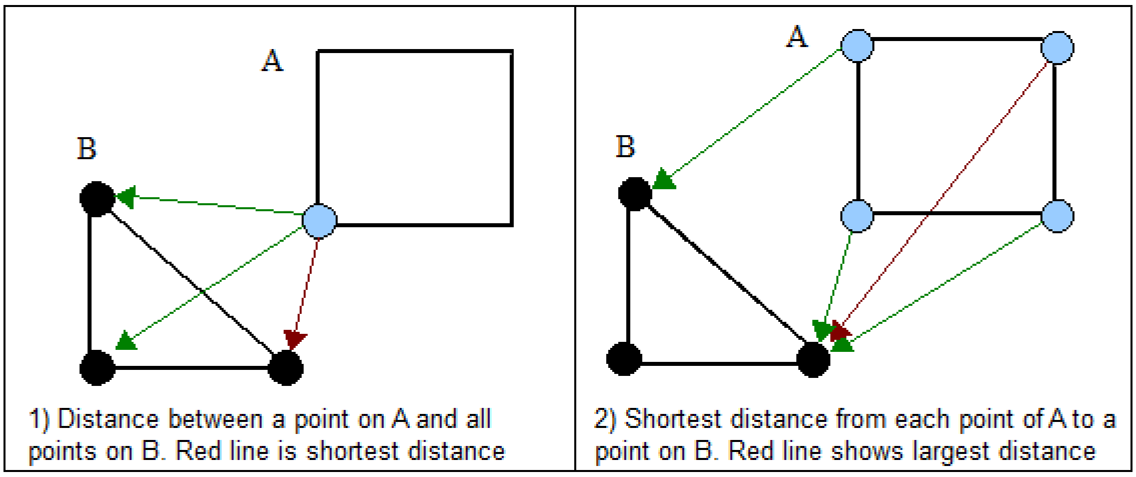
A
B
1) Distance between a point on A and all points on B. Red line is shortest distance
A
B
2) Shortest distance from each point of A to a point on B. Red line shows largest distance

Figure 2: The Hausdorff Distance algorithm on primative shapes with points at their corners

The Sobel edge detection method is one of the original edge detection algorithms and often acts as a benchmark for new approaches (Heath et al., 1998). Algorithms do exist that are more advanced, such as the Canny approach (Canny, 1986), which handles images with noise. However, with the noiseless images that we use, Canny results in more edges being identified and thus a more detailed edge image. This may prove useful in some cases, however we use the Sobel algorithm because more simplistic edge results in fewer points which in turn results in less computation time.

Approaches also exist that gather Hausdorff Distance by utilising line segments (Gao and Leung, 2002) rather than individual points. This can increase time efficiency because some shapes, such as those in Figure 2 will have very few lines but many pixel points that define those lines. For our experiments throughout this paper though, we use the basic point based Hausdorff Distance for simplicity.

3 Algorithm Details

The structure of our final system is as follows:

- Build Time
  - Get 2D projections of all 3D models in dataset
  - Downsample dataset images
  - Divide dataset into effective pixel Bands
  - Divide Bands into Sub-clusters through the use of Hausdorff Distance representatives
- Search/Run Time
  - Get 2D projection of search model
  - Downsample search image
  - Select effective pixel Band(s) that the search image falls into
  - Select Sub-cluster(s) that has the best Hausdorff Distance value(s) between the search image and the Sub-cluster's representative
  - Calculate Correlation value or Hausdorff Distance between the search image and every dataset image in the Sub-cluster(s) that was selected. Rank the results

We have two major stages, build time and search time. Build time is done offline and therefore we are not too concerned with the efficiency of the calculations carried out during this stage. Search time is considered as being online and may be executed while a user is playing the game, therefore we not only look for combined effectiveness with the build time algorithm but also efficiency during this stage.

3.1 Definitions

Here we define a few terms that are used regularly throughout this paper. Comparing two 3D models can be quite subjective, different games and different uses of our proposed algorithm will require varying levels of similarity to be distinguished. Figure 3 attempts to give a frame of reference for which we are working in.

Different: The two images are clearly of different models. A good visual guide for this is to first check whether the two models are roughly of the same primitive shape of not. For example, if both models are not roughly rectangular, triangular, circular, and so on, then the models are different. If they are the same shape, we can then check if they are oriented in the same plane. If not, then they should be considered different. For example, a rectangle in the y-axis is different from a rectangle in the x-axis. We can also check limbs and appendages using the same type of rules. Finally, it may also be useful to visually check if the models are of the same category of objects. For example, both models are of aircraft, insects, humans, road vehicles, or so forth.

Similar: For our purposes and with the dataset that we use for our tests, two models are similar if they are roughly the same shape, size, orientation, and position in their 2D projections and have the same number of limbs. The two models can however differ in fine details. To check if two models are similar visually, we can use the same rules outlined above for two different models. In general, we classify two images as being similar if, on first glance, they appear to be the same but with closer inspection we can see the difference in details. As mentioned before, deciding the boundary between

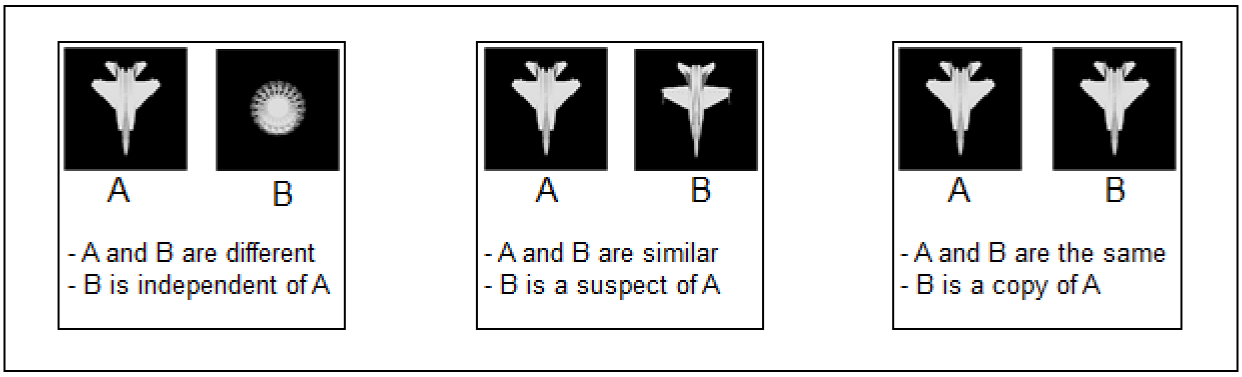

Figure 3: Visual examples of similarity definitions

'different' and 'similar' can vary depending on the requirements of the algorithm. For example, in a video game with many bipedal character models, two models should be considered similar if they are both of male humans wearing roughly the same outfit but with a few differences in details; therefore a male human model and an ogre model should be classified as being different. However, with a content based retrieval system and the dataset that we have available to use for our tests, it may suffice for both models to simply both be bipedal.

Same: If two models are the same, then one is a copy of the other. These models can be considered the same if there is no immediately obvious differences between them.

Search Image: We define a search image here because the use of this term can become somewhat ambiguous when used across multiple algorithms such as Correlation Filtering and Hausdorff Distance. In the case of a content based retrieval system, the search image would be our query image. That is, we wish to find models/images that are similar to the search image. In the case of a Correlation Filter, the search image would be used as the training image during the filter design stage.

Dataset Images: Dataset images are all other images in a database that will be compared to the search image. That is, we are looking for models from the list of dataset image that are similar to the search image.

3.2 Getting a 2D Projection

As mentioned before, the algorithm outlined in this paper is a view-based approach. This means that we must first get a 2D projection of our model. To do this, we simply prepared the the 3D model to be displayed on screen using it's default orientation and normalised the scaling and translations. Then, instead of displaying the model on screen, we capture the buffer information and write it into an image format such as the bitmap file format. This was done for all search and dataset models and all images have a resolution of 256x256. This part of our approach was not optimised and therefore any time-based performance issues listed throughout this paper do not take this step into account. Instead, it can be seen as a preparation step for our primary algorithm's execution.

3.2.1 Normalising Scale and Translation

The process of normalising a 3D model's scale is done so that all objects appear to be roughly the same size, while at the same time keeping their original proportions. This process is common in computer graphics and an example of this process is depicted in Figure 4. This particular approach usually involves creating a bounding box in 3D coordinate space and translating the object such that it's centre is in the middle of the box. The 3D model data is then analysed to find the largest or smallest vertex value[5]. This vertex is then considered as the furthest extremity of the model and so we scale the model proportionally, enlarging it or shrinking it, until this vertex is just at the edge of the bounding box. With this furthest point at the edge of the bounding box, it is assumed that all other vertices of the 3D model lie within the box as well.

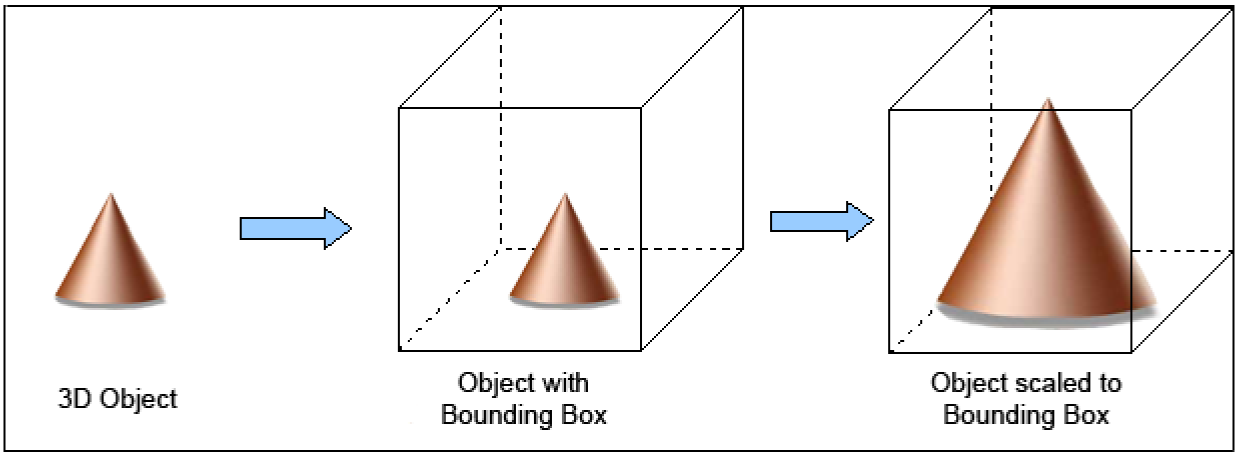

Figure 4: Normalising 3D Model Scale

3.2.2 No Normalising of Rotation

For the purpose of searching for copies or suspects of a 3D model in a game, we do not bother with attempting to normalise rotation. This is because all models should be in the same orientation frame to begin with. Each game will have its own processes for reading 3D mesh files, placing the 3D object in the virtual world, and displaying it on a computer monitor. While these rules may differ between games, they should be standardised to apply to all 3D objects in a single game. Therefore, if an attacker copies a 3D model and attempts to fool our algorithm by simply rotating the models vertex coordinates, the model will not be displayed correctly in the virtual world. For example, if a 3D chair model is copied, rotated, and then uploaded into back into the game, it may appear to be defying gravity, lying on its back, or facing a wall.

While rotation does not affect our algorithm for it's use in games, it may affect a content based image retrieval system. While this is not the focus of our research, if the need to normalise rotation arises there are existing studies into this process (Petrou and Bosdogianni, 1999). However, these methods can lack the effectiveness of the translation and scaling normalisation approach and many similarity algorithms designed for content based 3D model retrieval systems use rotation invarient measures instead, such as those presented by Kazhdan et al. (2003).

3.3 Pixel Matching

Pixel Matching is a nave approach to comparing two images. It simply involves counting how many pixels the two images have in common. Thus, for images with 256x256 pixels, comparing two images

[5]A vertex that is furthest away from the point of origin in 3D space, either in the positive or negative direction

that are exact copies of one another will result in a value of 65536 and comparing an image with it's inverse will give a value of 0.

This approach is nave because unlike Correlation Filtering or Hausdorff Distance it is very dependent on pixel position and value. For example, if an image of an insect is modified so that its legs are all shifted only a few pixels from each other, then the pixel match count between these two images may be very low. Correlation Filtering and Hausdorff Distance, however, ideally ignore this slight change is translation and can correctly match the two insects as having the same shape and detail. We do provide experiments and results for it however, because it acts as a simple benchmark for efficiency and effectiveness.

3.4 Correlation Filter

In section 2.5 we briefly discussed the process of training and testing with a Correlation Filter. We saw that once the test image and the filter were correlated, an Inverse Fast Fourier Transform is carried out on the result to give an output of energy strength in a complex array the with the same size as the image resolution. For this process we used the code that was developed by Y. Wang and Kumar (2007). To evaluate this energy output swe use the Peak Correlation Energy (PCE) function:

$$\begin{aligned} pce &= \frac{peak}{average} \\ & \quad where\ for\ a\ matrix\ of\ energy\ values\ k, \\ & \quad peak = \max(K),\ and\ average = \frac{\sum K}{|K|} \end{aligned}$$

As can be guessed, in the equation above, the value of peak is the highest energy value in the set while average is the mean of all the energy values in the set. If a distinct peak is seen within the energy results then it indicates a high similarity between the two images. In the function above, if we have a high peak value and the surrounding energy is low, then a high PCE value will be returned. If two images are exact duplicates of each other, then the peak energy will equal to the size of the image and the energy values through out the rest of the matrix will be zero. Thus, for an duplicate images of a 256x256 resolution, the PCE value will equal 65536.

We also explored the use of another popular energy analysis function know as Peak Side-lobe Ratio (PSR) (Kumar et al., 2006) which is defined by the function:

$$psr = \frac{peak - average}{std}$$

Peak here is calculated the same as in the PCE function. However, the average and the standard deviation (std) are not gathered from the entire energy sample but instead only from a region of interest surround the peak, known as the side-lobe, but excluding a smaller centre region around the peak. In Kumar et al. (2006) the region of interest is 20x20 matrix elements in size with the centre exclusion zone being of size 5x5. Thus, only 375 energy values are considered when calculating the average and standard deviation. This approach is designed to make use of a property of the MACE filter that produces minimal side-lobe energy in images that are similar. However, for our purposes we found PSR to be much less consistent and stable then that of PCE and that the execution time required for the PCE calculation was significantly smaller. Thus, all results shown in this paper are gathered through the use of the PCE measure.

We should also note that in our experiments, we use our search image as the training image for the filter design. Then, during search time, we calculate the dataset images against the filter, one at a time. We chose to do it this way so that time was not wasted creating or loading filters for each dataset item, instead, just creating one filter for the search image at the beginning of our algorithm.

3.5 Hausdorff Distance

Before we analyse the results of our initial Hausdorff Distance, we would like to highlight some changes we made from the Hausdorff Distance algorithm outlined in section 2.6. Many implementations of Hausdorff Distance use the function described in 2.6. However, throughout our experiments we have found that this gives a biased result. The above equation looks for the maximum distance between the two images. Thus, if A is close to B but B is far from A, then the Hausdorff Distance between images A and B will indicate that they are of very little similarity. For reasons as to how this phenomenon can occur, see section 2.6.

We decided instead to investigate another implementations of Hausdorff Distance that considered the average distance, such as those presented by Shapiro and Blaschko (2004). This slightly different equation substitutes the maximum calculations for average calculations instead:

$$\begin{aligned} H(A,B) &= \frac{h(A,B)+h(B,A)}{2} \\ & where\ h(A,B) = \frac{\sum_{\forall a \in A}[\min_{\forall b \in B}(\|a-b\|)]}{|A|} \\ & where\ \|a-b\|\ indicates\ the\ distance\ between\ two\ points\ a\ and\ b \end{aligned}$$

However, we found that this equation also gave undesirable results. The reason behind this is that, for our purposes, using average in $h(A,B)$ does not accurately represent distance between two sets of points. This is because it is possible for two images or sets of points to be almost be exactly the same except for an extra, distinct appendage, perhaps an extra leg added onto a bipedal character model, for example. The average distance will give more weight to the majority of the sets of points being exactly the same and practically ignore the extra appendage, thus classifying the two sets of points as very similar. This may be desirable in some applications but for our purposes we want to consider these two sets of points as being different.

We eventually decided to utilise beneficial aspects of both of the above interpretations of Hausdorff Distance. The equation we utilise for all of Hausdorff Distance experiments hence forth is:

$$\begin{aligned} H(A,B) &= \frac{h(A,B)+h(B,A)}{2} \\ & where\ h(A,B) = \max_{\forall a \in A}[\min_{\forall b \in B}(\|a-b\|)] \end{aligned}$$

We have found that this provides a good balance between the two previous equations. It lowers the chance of one direction of distance calculations taking precedence over the other and also allows us to keep the key factor that will prevent major appendages from not carrying enough weight in the distance calculations of one direction[6].

[6]The distance from A to B is one direction and the distance from B to A is another

3.6 Bands

The first improvement we made to the system was to cluster images of 3D models based on effective pixels. We call these clusters 'Bands' and they are used to group models with similar proportions. This is a logical step that utilises the efficiency and rough accuracy of a Pixel Matching type approach. Thus, we do not disregard the benefits of Pixel Matching but rather modify the approach slightly in order to utilise it as an initial filter before applying the Hausdorff Distance and Correlation Filter algorithms.

3.6.1 Effective Pixels

As the result figures throughout this paper show, all of the images taken of the models use a black background. Therefore, effective pixels of an image are any pixels that are not black. The bitmap image file format, like many other image formats, provides a matrix of pixel information, with one element in the matrix per pixel. For our tests, we use grey scale bitmap which allows 8 bits per pixel. That is, each matrix element can be between 0 and 255, pure black and pure white respectively. Thus, in the case of the bitmap image file format, an effective pixel is a any pixel that has a value greater then zero.

For counting effective pixels, we simply use a binary check. If a pixel has a value greater then zero, then we add one to a counter. Thus, for a 256x256 image, an all white image will have a value of 65536 and an all black image will have a value of zero.

We still need to perform a more advanced similarity comparison by using a Correlation Filter or Hausdorff Distance because an effective pixel measure does not give any indication to the shape or detail of the image. For example, if our search image that is used in the ranking figures throughout this section was rotated around the z-axis it would become a horizontal rectangular shape rather then a vertical one, which, under the definitions stated in 3.1 would classify the two images as different, even though they would have the same number of effective pixels.

3.6.2 Creating the Bands

When creating and searching the Bands we use the equation:

$$BandRange = P \pm (P * \Delta)$$

where P is the number of effective pixels that was responsible for creating the Band and Δ is a percentage value that determines the range of the Band. When building the Bands from a given dataset, we randomly choose a first image, count it's effective pixels, and then create a new Band. When we add a second image, we again count it's effective pixels but this time we see if it lies within in the range $[P_1 \pm (P_1 * \Delta)]$ where P_1 is the number of effective pixels of the first image. If the second image does lie within this range, then it is added to this Band, otherwise a new Band is created. These steps are repeated, comparing the third image to the Bands created by the first and possibly second image.

An alternative approach would be to use fixed size Bands. However, we have decided not to use this option as it presents a problem with boarder cases. For example, if the Bands are in ranges of 1000 and an image has an effective pixel count of 999, it may have more in common with the images in the Band above rather then the Band it is ultimately placed in. The equation that we have chosen to use allows for simple overlapping of the Bands, which is discussed in more detail in Section 3.8.

When utilising the Bands during our similarity search we follow the same equation as above; gathering the effective pixel count of the search image and evaluating it against that of the Band ranges. If a corresponding Band is identified, then we continue on to use Correlation Filtering or Hausdorff Distance calculations to get similarity values between the search image and all images within that Band.

The Bands that were used for the following Banding and Downsampling results were created and searched using a $\Delta = 10\%$.

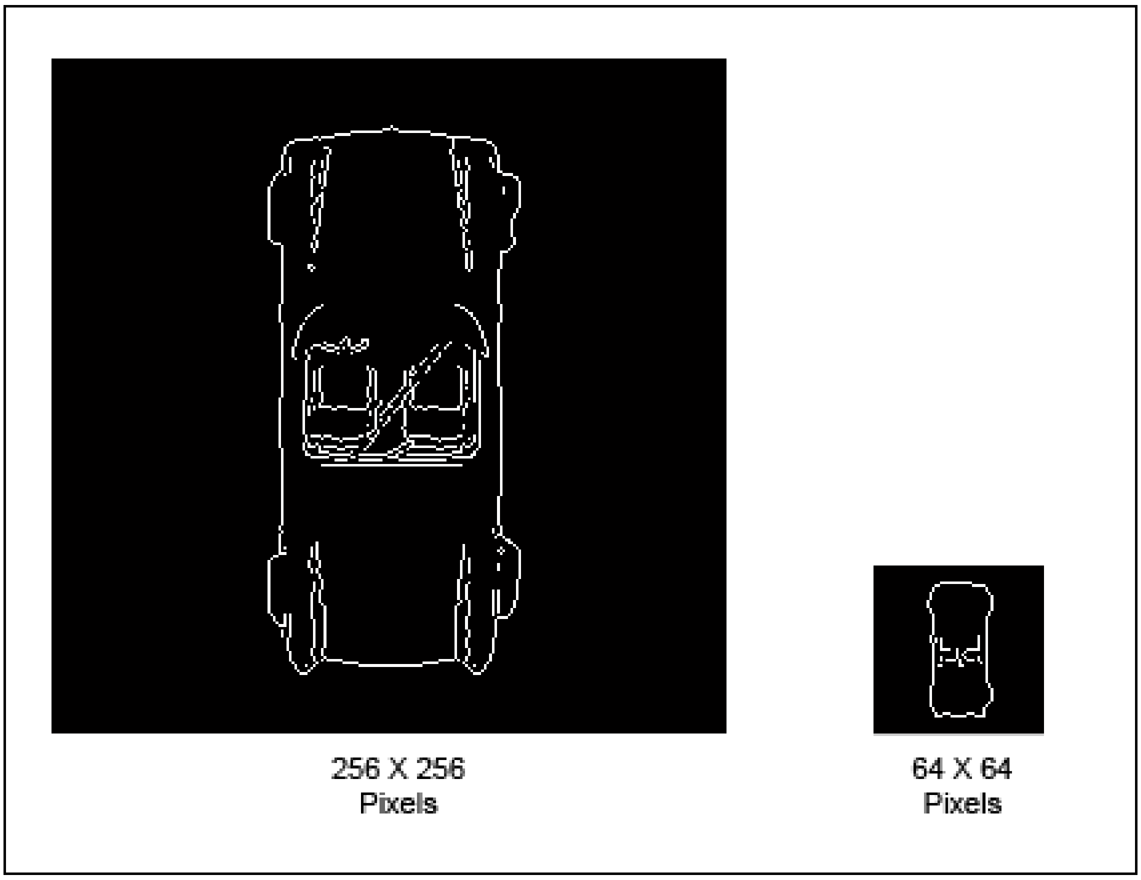

Figure 5: The edges of a model before and after the image has been downsampled

3.7 Downsampling

Downsampling is a process of lowering the resolution of an image. The algorithm we used is as follows: for a sample rate of n, we keep every n^{th} pixel, starting with the first element of the pixel matrix, and dispose of all pixels in between. We first do this along the columns of the pixel matrix and then along it's rows. In Figure 5 we use a sample rate of $(n = 4)$, and thus $[(256 \div 4) \times (256 \div 4) = 64 \times 64]$. The images in Figure 5 also show the results of an edge calculation during the Hausdorff Distance algorithm for both an original sized image and the it's downsampled counterpart.

All of our Downsampling experiments are done using the Banding configuration. For consistency, we reconstructed the Bands with the downsampled images and also downsampled the images before calculating the correct Band for a search image. We have found that the downsampling method used here is quite stable at reproducing the image at a lower resultion but it is still possible for a white pixel to be removed and represented by a black pixel that was in an n^{th} position in the pixel matrix. Therefore, reconstructing the Bands with the downsampled images can lead to different distributions of the models. This also means that images with very fine dots or lines in them may result in just a plain black image after being downsampled. Thus, this approach works better for models that present a larger surface area for it's solo 2D projection.

3.8 Sub-clusters

Here we introduce a second layer of clustering that should offload some similarity calculations to build time rather then requiring all images in a Band to be considered during search time.

At this time we also modify our Banding algorithm to use $\Delta = 20\%$ instead of the 10% that was used in previous test. Also, we now check for overlapping. That is, if a dataset image can fall into the effective pixel region of two Bands then it is placed in both and if a search image could belong in more then one Band, then both Bands are searched. Our Sub-clustering approach below allows us to search both Bands without incuring much more of a execution time penalty.

3.8.1 Creating the Sub-clusters

To create the Sub-clusters we use Hausdorff Distance during build time, immediately after we have completed the construction of the Bands. The first step in creating the Sub-cluster is to randomly chose an image in the current Band, this will be the representative of our first Sub-cluster in this Band. We then use this representative as a search image of sorts, comparing it to every other image within the Band. Any image that is less than σ distance to the representative is grouped into this initial Sub-cluster. The image that has the furthest distance from our first representative is chosen as our second representative and the process is repeated. Again, the image that is furthest away from our second representative is chosen to be our third. We should point out that a representative cannot exist in any other cluster. For example, for a set of distance values X, a new representative will be at position $\max(X)$. If the corresponding image has already been placed in a Sub-cluster then we use the image at position $[\max(X) - n]$, increasing the value of n until we find the index of an image that has not already been placed in a Sub-cluster or until the distance value at position $[\max(X) - n]$ is less than σ, indicating that all images in this Band have been placed in a Sub-cluster.

While an image cannot become a representative if it has already been placed in a Sub-cluster, the image can exist as a member of multiple Sub-cluster. It can even be a representative of one Sub-cluster and a member of another, if and only if, it was chosen as a representative before it became a member of another Sub-cluster. This ensures that a Sub-cluster properly encapsulate all images that are similar to it's representatives.

We have mentioned above that a value σ is used during the construction process. This value is a maximum Hausdorff Distance value, beyond which we judge two images to be different and thus should not co-exist in the same Sub-cluster. This value will differ depending on the dataset used and the granularity required of the searching algorithm. To determine this, we must determine to what extent our two models have the same features and details to be considered similar by our system. We calculated our σ value by modifying fifty images from our dataset in a way that we deemed the resulting images to be similar to, but not a duplicate of, their originals. We then calculated the Hausdorff Distance between each of the modified images and their paired original images and took the highest distance as our σ value. The resulting value that we utilised caused a Sub-clusters to contain models that are roughly the same shape and orientation but that allowed for varying detail and minor appendages. If we wanted a stricter similarity measure, we would reduce σ value towards zero.

3.8.2 Searching the Sub-clusters

To utilise this structure, we first find the Band(s) that the search image belongs to. We then calculate the Hausdorff Distance between the search image and the representatives of each of the Sub-clusters. The clusters with lowest Hausdorff Distance values in each Band are then examined, calculating Hausdorff Distance or Correlation with all of the images in the chosen Sub-cluster(s).

If we want to relax restraints on Sub-cluster selection, we can introduce a variation variable that dictates how far a representative's distance value must be from the best ranked representative's distance value in order for it's Sub-cluster to be considered. For example, if Sub-cluster A has a Hausdorff Distance of 6 between it's representative and a search image, B has a value of 7, C has a value of 11, and our variation value is 2, then all of the images in Sub-clusters A and B will be examined for similarity to our search image but C will not. For all of our Sub-cluster experiments, we use a variation value of zero. This means that if two or more representatives have exactly the same Hausdorff Distance to the search image, then all of these Sub-clusters are examined, otherwise they are discarded. Taking into account that our Hausdorff Distance values are calculated in floating point precision, this occurrence is rare but still possible due to the relatively limited range of Hausdorff values, usually between 0 and 150.

4 Experiments and Results

For most of our experiments we gather results for the two image processing algorithms, Hausdorff Distance and Correlation Filtering, separately while we modify other variables and attempt to improve the performance of both algorithms. This allows us to compare the effectiveness and efficiency of the two pixel based image analysis techniques.

4.1 Dataset and Testing Environment

We use the dataset that was created by Chen et al. (2003) for the use of benchmarking content based 3D model retrieval systems. We use the dataset that has 1833 models in it, though due to errors in gaining 2D projections of the models we utilise only 1829 of these. The dataset is freely available online[7]. It should be noted that not all models of the same type are oriented in the same plane and so this dataset is more suitable to represent a content based 3D model retrieval system rather then a game. For a clearer explanation of why this is, see Section 3.2.2. All 2D projections of the models were gathered using the OpenGL graphics API along with the Corona image capturing library[8] for C++. All images have a resolution of 256x256.

All tests were carried out on a Intel Core2 2.40GHz CPU. For our implementation the CPU is the major hardware component that determines execution times. To implement our the algorithm in this paper we have used MATLAB version 7.5.0 (R2007b)[9]. MATLAB provides a script-like language that allows for easy implementation of mathematical based image processing algorthms. MATLAB also provides the tools necessary to profile the execution time of our implementation.

4.2 Metrics

Throughout our experiments we use three metrics to evaluate the effectiveness of our algorithms; these are ranking, Genuine Acceptance Rate (GAR) and time. When we use the terms *performance* and *effectiveness*, we are usually relating to ranking or Genuine Acceptance Rate (GAR).

Ranked Results: Ranking appears most often as our measure of effectiveness and is what most of our figures depict. These results are gathered by sorting dataset images in descending order, with the images that are judged most similar to our search image by the Hausdorff and Correlation algorithms being returned at top positions. However, ranking is somewhat of a subject measure. While the rank order is firmly reliant on the algorithms used, the only benchmark we have for this measure is our own opinion. For example, if a 3D model is ranked highly in similarity, we must use a human opinion to judge whether the rank is appropriate and whether other models should be placed higher then what the system has achieved. This human opinion may vary from person to person and this is why we have outlined our interpretation of similarity in Section 3.1.

Genuine Acceptance Rate (GAR): GAR gives us a good understanding of whether or not our structures are working as intended. GAR is often used as a performance metric for biometrics and other security approaches. In the biometrics field, it represents the rate of a genuine being recognised and properly authenticated. If a set of images all come from the same person or object, then these models are considered intraclass, where as, if the images are of different users or objects then they are interclass relative to each other. In order to gather GAR in a biometrics system, multiple instances of the biometric are gathered of a single genuine user and tested. For example, a facial recognition

[7]`http://3D.csie.ntu.edu.tw/~dynamic/`
[8]`http://corona.sourceforge.net/`
[9]`http://www.mathworks.com`

system will gather multiple images of a user's face with different positioning, facial expressions, and under different environmental conditions such as variable lighting. If all of these test images are properly authenticated, then the GAR for that one user is 100%. Conversely, the False Rejection Rate (FRR), the rate that a genuine user fails to be authenticated properly, is $(FRR = 1 - GAR)$.

For our purposes this definition needs to be changed slightly. Due to the nature of the 3D models in games, when gathering sample images of the models we are able to control all positional and environmental settings, such as lighting, translation, rotation, and scaling. For details about this, see Section 3.2.1. Therefore, all images of a single model will always be the same. For our purposes there are no intraclass instances, all models are considered interclass relative to each other. Hence, we change GAR to refer to the rate at which, given a slightly modified model, our system is able to find the original dataset model that it is based upon. It should be noted that this metric is primarily used to test the performance of our clustering methods rather then the effectiveness of Correlation Filters or Hausdorff Distance. Also, in biometrics, it is understood that achieving an error rate of zero may never be possible. Therefore, GAR will never be one hundred percent. However, research can aim to improve biometrics systems to a point where the error rate is acceptable. This same deduction can be made to protecting virtual assets through similarity comparisons and so our expectations are that GAR can reach over 90%.

There is another metric commonly used in biometrics research, known as Genuine Rejection Rate (GRR) and it's inverse, False Acceptance Rate (FAR). GRR is the rate at which an imposter is correctly denied access by the authentication system. For our purposes this would require creating search images that have no similar counterparts within the entire dataset. This is a difficult and time consuming task that we deemed not feasible at this time, thus we do not concern ourselves with GRR.

Time: This is a self explanatory metric. We refer to this metric as *efficiency* from here on. We use seconds and milliseconds to describe it. As the Correlation Filter and Hausdorff Distance algorithms are the most computationally intensive aspects of our system, time can also be related to how many 3D models are tested against a search image.

4.3 Initial Experiments

Our initial experiments were to purely test the effectiveness of the image comparison techniques that we would continue to use throughout this research: Correlation Filters and Hausdorff Distance. There is no clustering or other improvements, simply one search image tested against 1829 dataset images at a 256x256 resolution.

Pixel Matching Results

We have chosen to show the results of the Pixel Matching experiment in Figure 6 because it acts as a good benchmark for the search image that we have used for all of the subsequent experiments. We can see that a lot of the images returned in the ranked results are of roughly the same shape at first glance. This is what we try to achieve with the more advanced algorithms. However, the models in ranked results 6, 8, 9, and 10 do not quite match what most human opinions would consider similar to the search model. They are of the same orientation and height but some of them have much different shapes while others have noticeably different widths. Thus, through out our experiments on the more advanced algorithms we hope to achieve effectiveness better than the Pixel Matching results.

It should also be noted that Pixel Matching is extremely efficient as it is simply 256x256 binary comparisons, which subsequently uses far less calculations then those needed for Correlation Filtering and Hausdorff Distance.

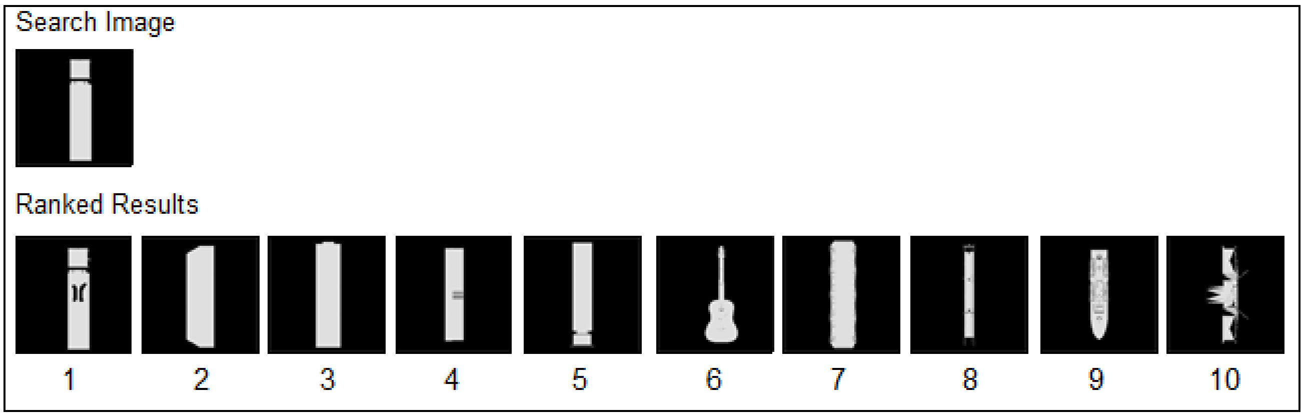

Figure 6: Results of Pixel Matching on the entire dataset

Initial Correlation Filter Results

The initial results of the Correlation Filter, shown in Figure 7, are not promising. Many of the objects are of completely differing shapes and sizes. Most of the anomalies seem to occur when an image has very few bright pixels or are saturated. After the anomalies though, further down the ranking, the performance begins to stabilise, showing definitive patterns in size and orientation of the models. Thus, our first improvement to our system, described in Section 3.6, involves attempting to reduce the possibility of these anomalies arising.

While these initial results may be discouraging, we decide to continue exploring the Correlation Filter option due to it's efficiency. Venkataramani and Kumar (2003) states that Correlation Filters are computationally expensive and inefficient. However, this observation is made when there is a dataset of millions of images, which, for example, may cover an entire population of a country. For now we are conducting our experiments with a dataset of only a few thousand objects and we do not anticipate a user content driven multiplayer game to extend much further beyond this number of uploaded 3D models in the near future. Thus, for our purposes, Correlation Filtering is very fast and outperforms the other object recognition algorithm we utilise, Hausdorff Distance.

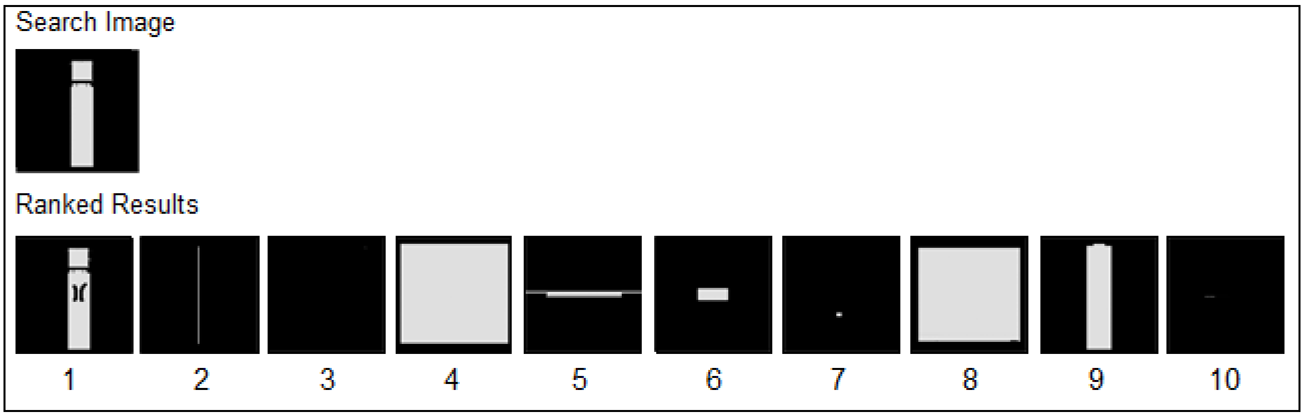

Figure 7: Results of Correlation Filtering on the entire dataset

Initial Hausdorff Distance Results

The results shown in Figure 8 are a lot more promising then the initial results for Correlation Filtering. The overall shape of the ranked result models are more in line with that of the search image though there are still a few discrepancies. Again, ranked result 1 is our modified image that we manually generated from the search image. It's inclusion at the top of the ranks is an encouraging indication. Generally the models seen in these results are more similar in width to the search image then the results for Pixel Matching are, though ranked result 9 of a guitar makes a return, which is a puzzling outcome for both Pixel Matching and Hausdorff Distance.

A clear difference compared to the Pixel Matching results are images such as ranked result 2 in Figure 8. This model is ranked high because it's outer shape is similar to our search image. However, it is clear that the centre of the model is quite different, with many hollow areas that would cause it to be ranked much higher in the Pixel Matching approach. While these may or may not be perfect results as of yet, we see them as positive indications of the potential of Hausdorff Distance and that this algorithm fulfils it's intended purpose as being a shape matching algorithm. However, as Table 1 in Section 4.5 shows, the execution time of Hausdorff Distance on a full dataset of a thousand or more models is clearly unacceptable for the real-time environment of a game.

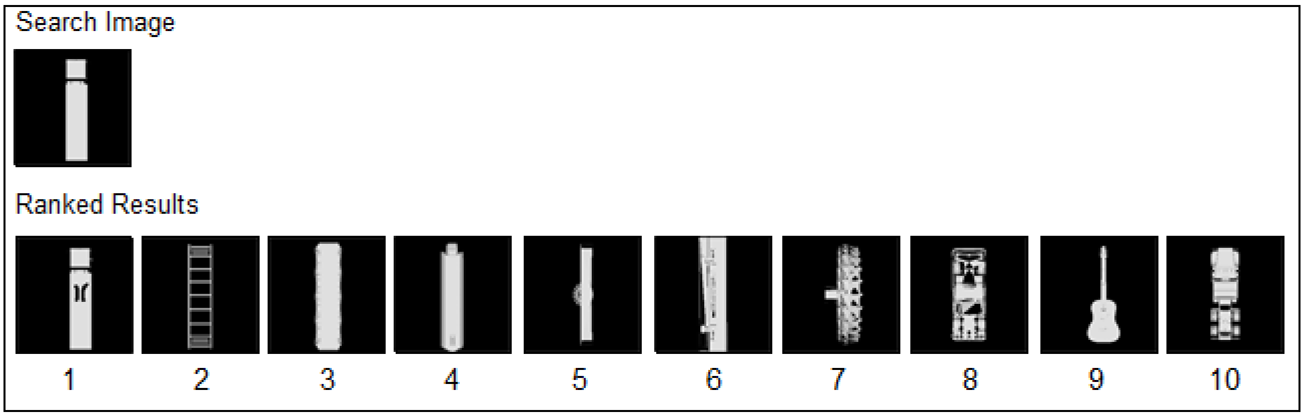

Figure 8: Results of Hausdorff Distance on the entire dataset

4.4 Banding Experiments

As mentioned in 3.2.1, all models are enlarged or minimised to fit a coherent scale. However, this does not mean that the proportions are the same. For example, in Figure 7, ranked results 2 and 4 are both scaled so that they fit into the same sized bounding box; this gives them the same height but does not necessarily mean they will have the same width. Also, because these models are three dimensional, we must also consider the z-axis, which can be imagined as going into and out of the computers monitor. Ranked result 7 in Figure 7 may appear to not be scaled properly but it has in fact been scaled so that vertices on the z-axis are touching the edges of the bounding box. Thus, by clustering images into effective region bands, we perform an initial, efficient, and somewhat obvious check to make sure the models have of roughly the same proportions in their 2D projection, preventing models that are thin in the x and y dimensions from being compared with those that are large in these dimensions.

Bands: Correlation Filter Results

The Banding structure improves the Correlation Filter algorithm by acting as a preliminary filter that gets rid of the anomalies that exist in Figure 7. It is also promising to see that in Figure 9 most of the results have roughly the same physical proportions as our search image. The MACE filter is described by Kumar et al. (2002) for it's shift invariance property that allows it to recognise objects despite translation in the image space. More advanced filter algorithms such as the RI-MACE filter by Ravichandran and Casasent (1994) allow for Correlation Filtering to also be rotation invariant. While rotation invariance is not a defining feature of the MACE filter that we have utilised, it appears that it is still capable of it in some form, or at least more capable then the Hausdorff Distance approach, as many of the results in in Figure 9 are of similar proportions to the search image but of different orientations. This is a property that may be worth exploring for datasets that do not have a fixed orientation requirements. For our purposes in games, however, we view the results with differing orientations as errors and so we can say that while the effectiveness of the Correlation approach has improved, it is still not ideal.

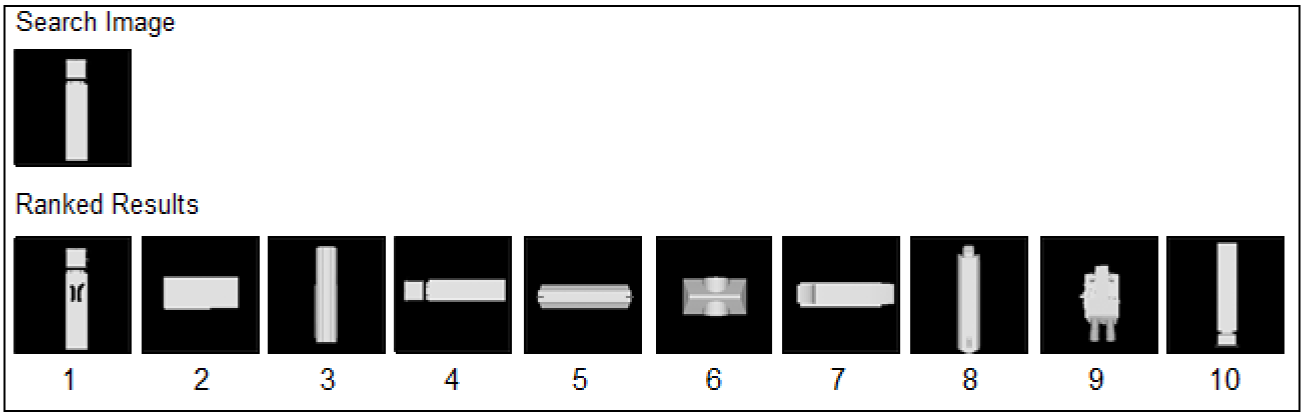

Figure 9: Results of Correlation Filtering after Banding the dataset

Bands: Hausdorff Distance Results

Banding is not as effective for the Hausdorff Distance calculations as it was for the Correlation Filter, however it still provides some benefit, with a few negative results as well. For starters, Figure 10 shows an increase in effectiveness with the removal of ranked results 2, 5, and 9 from Figure 8. These three models have been identified by the Hausdorff algorithm as being similar to our search model but a human perspective will identify them as having different densities, sizes, and general shape respectively. Thus, by Banding images through effective pixels, we can remove these types of anomalies.

Unfortunately, there is one negative case in these results. Ranked result 3 from Figure 8 is not included in Figure 10 even though an we have judged it to be suitably similar. The model depicted in this rank result is in fact slightly bigger, which gives it a larger effective pixel count. Therefore, this is most likely a case of this image being a border case and has been placed in the the next Band. These types of issue should be addressed once we overlap the Bands. However, making this alteration to the process at this stage could cause a search image to be compared against all dataset images in two full Bands, reducing the efficiency and lowering the positive impact of the Bands. We will explore this overlapping option once we have implemented other measures that should increase the efficiency of our overall system.

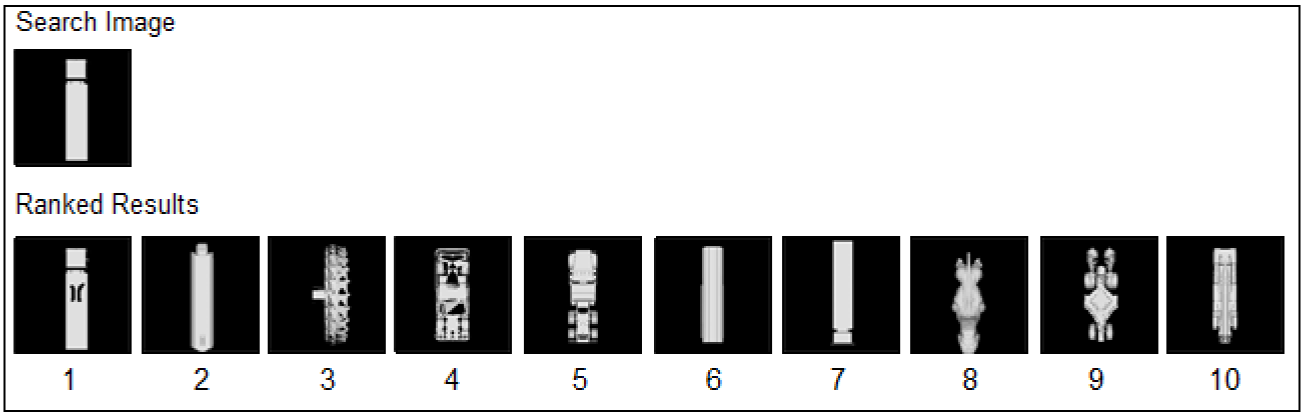

Figure 10: Results of Hausdorff Distance after Banding the dataset

4.5 Downsampling Experiments

For these experiments, we continue to use our Banding algorithm, though the Bands are reconstructed using the downsampled dataset images. All images in these experiments are now downsampled to 64x64 pixels. However, in the following results, the search images and ranked results are shown in their original form, not in their downsampled form.

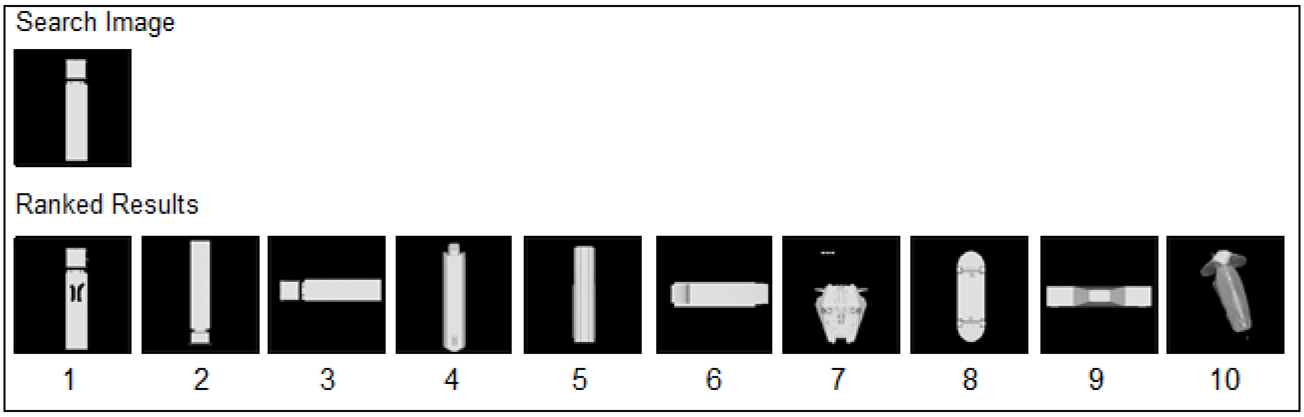

Figure 11: Results of Correlation Filtering after downsampling and Banding the dataset

Downsampling: Correlation Filter Results

We decided to explore downsampling as a way of improving the effectiveness of the Correlation Filter algorithm after reading findings by Venkataramani and Kumar (2003) that stated that a lower sampling rate or resolution could give a more desirable error rate. The reasons Venkataramani and Kumar (2003) have given for this change in error rate is that a lower resolution image can in fact reduce some noise in the image. This could work to our advantage if we examine Figure 7 and see that downsampling could cause the fine lines and dots in ranked results 2, 3, 7, and 10 to be removed, leaving simply a plain black image that should not be matched with our search result.

Considering that our Banding approach already handles these types of issues, however, the improved effectiveness of our Correlation Filter implementation after downsampling is somewhat puzzeling. Our images have no noise and it would seem logical that downsampling should not change the energy of our images by much. Regardless, the images in ranked results 2, 4, and 5 in Figure 5 have been promoted to higher ranks, which is ideal as they match the orientation of our search model better then the other results returned. The reasons behind this will need further investigation.

Downsampling: Hausdorff Distance Results

While we had originally intended to only use the downsampling technique with the Correlation Filter, more considerations and a few tests proved that it could also be used to improve aspects of the Hausdorff Distance calculations. By comparing Figure 12 with Figure 10 we can see that there is also a slight performance improvement. The specific images we should pay attention to are Ranked Results 6 and 7 from Figure 10. An initial glance by a human observer would classify these two models as being more similar to the search image than some of those that are ranked higher then them in that set of results. In Figure 12 these two models are appropriately promoted to ranks 4 and 5 respectively. This is because the downsampling reduces the amount of detail that is considered by the Hausdorff algorithm. Figure 5 shows how the edges of the model are simplified, further numbing the Hausdorff Distance algorithm to detail and noise on the model and instead allowing it to consider shape only. It should be noted however that while downsampling has improved effectiveness with this search image, it may also degrade performance if we do in fact want the calculations to consider the finer details.

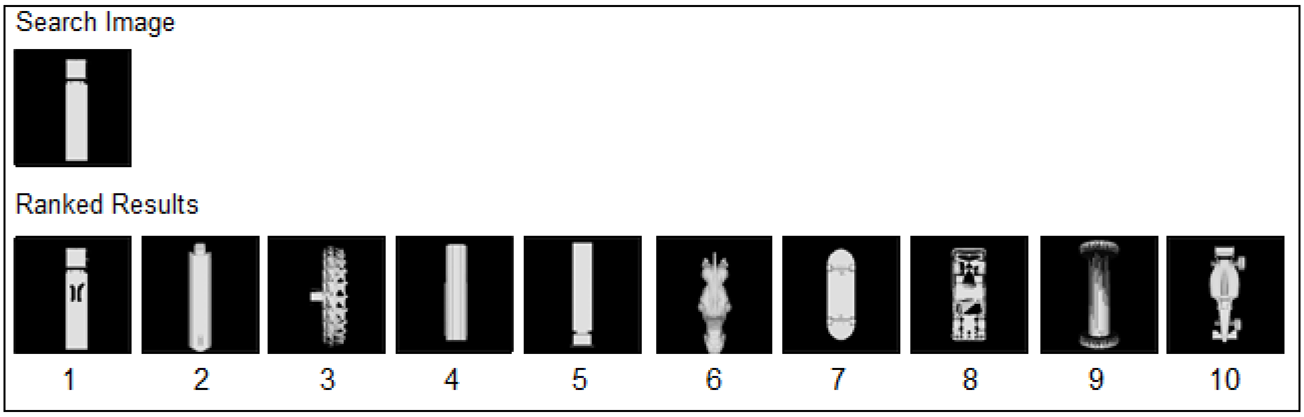

Figure 12: Results of Hausdorff Distance after downsampling and Banding the dataset

While the increase in effectiveness for Hausdorff Distance is debatable, one other benefit of downsampling cannot be argued. Table 1 shows the difference between using standard 256x256 images and ones that are downsampled to 64x64 pixels. Both tests involve comparing one search image to a dataset of one hundred models. The table shows the total time taken for all one hundred dataset images for each of the major functions involved in calculating Hausdorff Distance; the process of extracting the edges of the model in image, getting the points of the edges, and calculating the distance between all of these points and using the logic in Figure 2.6 to determine the final Hausdorff value.

	Standard	Downsampled
Detect Edge	42.721 sec	3.088 sec
Get Points	14.262 sec	0.857 sec
Compute Dist	132.311 sec	4.740 sec
Total Time	189.294 sec	8.685 sec

Table 1: Execution times for Hausdorff Distance using 1 search image and 100 dataset images

We can see that all of the times required for the downsampled images are a fraction of those required for the standard sized images, especially for computing the distance between points. This is simple to explain; Figure 5 shows that due to the reduced resolution and granularity of the downsampled image the edges that are calculated are much simpler. This simplified edge, combined with the reduced number of pixels for the image, results in less time spent gathering the point set. In turn, a reduced point set implies less point to point distance calculations.

For one search image and a dataset of only one hundred images these differences in time may not seem too significant but if we were to use, for example, one hundred search images against one hundred dataset images, we will require ten thousand calculations. If we divide the total times of both sized images by one hundred we see that on average it takes only 0.086 seconds (86 milliseconds) for downsampled images and 1.892 seconds (1892 milliseconds) per standard size image. If we use these average times, for ten thousand comparisons, it will take roughly fourteen minutes for downsampled images while it will take a staggering five and a quarter hours with standard images, a time that is clearly unacceptable. The difference in time required for large scale dataset or heavy influxes of new models being uploaded to the virtual world will cause differences in user waiting time in the magnitude of hours.

With evidence supporting the benefit of using downsampled images with both Correlation Filters and Hausdorff Distance calculations, we have made it a staple component for our system and thus from this point onwards we complete all our experiments using downsampled images.

4.6 Sub-clustering Experiments

At this point we evaluate not only the effect that Sub-clustering has on the ranked results but also test the entire structure to see how it may hinder the outcome of the ranked results. We test the Banding structure first and then the entire Sub-cluster and Band system together. This is done so that we can judge the effect of each component. We test the GAR of the Bands in this section rather than earlier on because we have now altered our Banding algorithm to allow for more comprehensive overlapping.

General Sub-clustering Results

Here we utilise Genuine Acceptance Rate (GAR) to judge the performance of our complete structure, which includes both Banding and Sub-clustering. To help evaluate this metric, we randomly chose one hundred images from the complete dataset of models and manually modified them in a way that we believed the new image and the original image were similar but not duplicates of each other. To relate this to biometrics, we consider the modified image as a genuine suspect of the original image. We did not test the original image itself because the overlapping nature of our Banding and Sub-clustering algorithms means that we will always be able to find a duplicate model.

Our test was then to simply use each one of these modified images as a search image and test whether or not the name of the original file could be found in our clustering structure with our search algorithm. That is to say, the only Hausdorff calculations we carried out were on the Sub-cluster Representatives and no Hausdorff or Correlation calculations were used on the images within the chosen Sub-clusters. However, the search images are modified such that if we were to do these calculations we would expect to see the original image to appear within the top ten of the ranked results.

Using the above approach we found that our Bands yielded a GAR of 92%. This means that,with overlapping our Bands and with an effective pixel $\Delta = 20\%$, we are able to find 92% of the original images. The 8% of search images that did not succeed are most likely a result of modifying the images to a greater extent then is allowed by our system. The number of effective pixels are too different for our algorithm to classify the two images as being similar. This GAR value could be increased if we modified the 8% of search images that did not succeed such that they more closely resemble their original images or by increasing the effective region Δ value in order to make our Bands larger. However, we chose to keep this value and report it in this paper in order to show that there exist discrepancies between our algorithm and the expectation of human intuition.

The Sub-cluster structure scores a GAR of 88%. This is on top of the GAR of 92% that we received for the Banding structure, meaning that the introduction of the Sub-cluster reduces the GAR by 4%. We believe this to be an acceptable result. It primarily shows that the approach that we use to select Sub-cluster Representatives during both build time and search time is working well.

In terms of efficiency, our finalised Sub-cluster approach computes Correlation Filter and Hausdorff Distance results against select models within two Bands for the search image we have used in all of our experiments. Hausdorff Distance is calculated against all of the Sub-cluster Representatives of both Bands and then proceeds to execute the Correlation Filter and Hausdorff Distance algorithms on all images in one Sub-cluster from each Band. In total, including the Representative calculations and removing repeat calculations for images that may be in both Sub-clusters, we end up comparing our search image with 211 files. To give a comparison, the Bands that are examined in this experiment have roughly 350 images each. Also, the size of the Band that was examined in our Downsampling test, which utilised an effective pixel Δ of 10%, was 187 images. Thus, for roughly 50 more calculations we are able to examine two Bands of much larger sizes and, as the results below show, we are able to get much more desirable ranked results.

Sub-clusters: Correlation Filter Results

Utilising our Sub-clustering approach leads to an interesting point here. In essence, we are combining forms of Hausdorff Distance and Correlation Filtering into one system, using Hausdorff as a type of filter and Correlation as a way of making a final selection or ranking. This is beneficial because we have consistently had trouble with our Correlation Filter highly ranking objects with a different orientation to our search image but now our Sub-clusters tend to group models of similar orientation and basic size proportions. Now, because of this, ranked results 6 and 8 in Figure 13 make a return after being absent since our Pixel Matching benchmark.

There are some disadvantages that have been introduced with our Sub-clustering though. Ranked result 2 in Figure 13 has not appeared in any of our previous results. It has appeared because our search image is a borderline case when selecting a Band to search. The improvements we have made to our Banding algorithm causes both Bands to be searched. Ranked result 2 most likely existed in a higher or lower band that was not examined in previous tests.

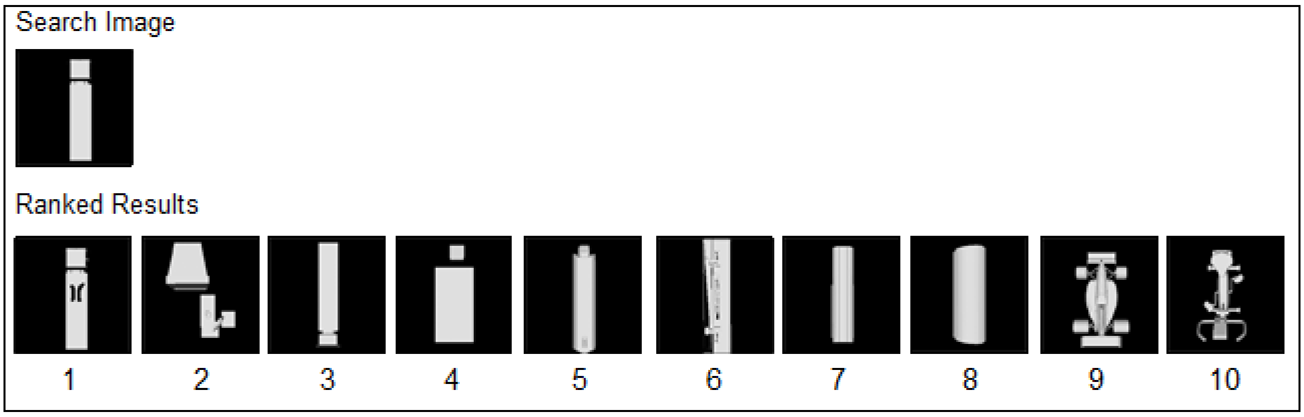

Figure 13: Results of Correlation Filtering after Sub-clustering the dataset

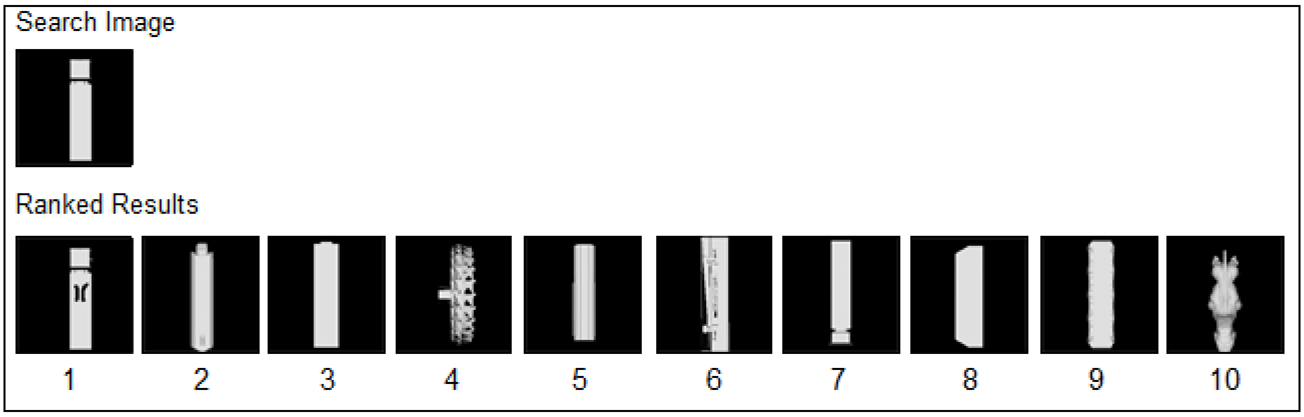

Figure 14: Results of Hausdorff Distance after Sub-clustering the dataset

Sub-clusters: Hausdorff Distance Results

The ranked results for Hausdorff Distance presented in Figure 14 gives strong support to the effectiveness of our complete structure. A quick observation by a human eye would identify ranked result 4, 6, and 10 as not being similar to the search image. This is an improvement though because there are only three models in the top ten that appear to be in error to our desirable outcome, which is less then all other ranked results figures in our previous approaches. Using the Sub-cluster structure also sees the return of models that were present in our Pixel Matching benchmark but not in subsequent tests. However, it also identifies similar models that were not in the Pixel Matching ranked results but that have appeared throughout our Hausdorff Distance tests. Thus, the Sub-cluster structure combines the desired results of both similarity comparison algorithms.

Ranked result 9 has re-entered the top ten after being missing since our initial tests on the entire dataset. It has not been included in our Banding and Downsampling tests because we lacked the ability to handle overlapping Bands in previous iteration and this model had been placed in a separate Band. We notice that it has shifted to a higher rank then in Figure 8 and we deduce that this is due to the effect of downsampling.

4.7 Summary of Results

Throughout our experiments we have seen that Hausdorff Distance has a far more desirable effectiveness then Correlation Filtering. However, there is a trade-off; Correlation Filtering is nearly an instant calculation for a dataset of size we are using, taking only milliseconds, while Hausdorff Distance calculations on a full dataset can take anywhere from a few seconds to a few hours, depending on the number of images and downsample rate. In fact, many of the clustering approaches we have introduced in this paper have shared two goals: to improve the effectiveness of Correlation Filtering while simultaneously increasing the speed of Hausdorff Distance during search time.

The question may arise, should we at this point choose one of the pixel based similarity matching algorithms over the other? The answer is no; both algorithms show promise and both have large avenues of exploration open to us. We have utilised one of the simplest Correlation Filter designs, the MACE filter, and further exploring more recent and advanced filters could prove beneficial to our uses. Similarly, we have used a point based Hausdorff Distance approach in this research, one of the slowest and computationally expensive algorithms of all Hausdorff Distance approaches, and exploring others such as line based could increase efficiency.

It would also be beneficial to look further into combining the two similarity algorithms. Our Sub-clustering technique gave us a brief peak at the possibilities of combining the two approaches by using Hausdorff Distance to find a Sub-cluster and using Correlation Filtering to analyse each dataset image within the Sub-cluster. Considering the cheap execution of our Correlation Filter implementation, adding this algorithm as a secondary measure to any system we create could improve the results for little overhead. For example, if we combine the results of Figure 13 and Figure 14, giving more weight to the Hausdorff Distance results, then ranked results 3, 5, 6, and 7 of Figure 13 will be promoted to the head of the ranking, followed by the rest of results shown in Figure 14.

5 Future Work

There is still a lot of work that can be done in this field. Protecting virtual assets in massively multiplayer games is an under-appreciated field of research. Even more promising is the topic of protecting copyrights over assets such as 3D models, materials, audio, and programming scripts. While there are not many games currently available that allow for user generated content to be hosted on a single set of persistent servers, we believe that if we provide solutions to this copyright situation and other problems, such as efficient uploading of assets to the server, then we will encourage these types of games to be developed and for communities to thrive as users trust the server to uphold their rights.

The algorithms and structures we have explored in this paper are far from perfect but it is an initial step. To improve this type of 3D models copyright protection system we should look deeper into the field of content based model retrieval systems. We have applied one existing category from this research field, view -based model comparison, and combined it with algorithms from the fields of biometrics and computer vision. If we explore other promising categories of content based model retrieval systems, such as those outlined by Tangelder and Veltkamp (2008) and modify them for our specific purpose we could find other effective solutions or avenues to expand upon.

5.1 Future Goals and General Improvements

The ultimate goal would be to make this system fully automated and not require any human interaction. At this point our approach is designed to reduce the search space more then anything else. Our system can return a ranked list of 3D models and allow a user or administrator to decide whether the search model is infringing on the uniqueness of any of the dataset models. Thus, our system resembles a search engine, which allows the user to submit a query in the form of text or content and a list of possible matches are returned; it is then up to the user to decide if any of the items returned match what they desire to find.

We believe though that our purposes allow for much finer granularity and possibly allowing the system to make the decision. That is because while a search engine must take into account the limitations of a humans capacity to describe what they want, checking if a new model is a copy or too similar to an existing model makes use of precise thresholds. If we can make develop accurate comparison algorithms and an efficient clustering approach that does not degrade the effectiveness of the system, then it would be possible to automate the system, effectively turning it into a background rule enforcement process. For this to be integrated into a commercial game though, we must prove that the system has a false acceptance rate and false rejection rate that are acceptable to the players and businesses in the virtual world.

5.2 Multiple Viewpoints

The algorithm we have presented in this paper has room for improvement and further exploration. One of the most obvious of these is to increase the number of projections we utilise per model. Other view based model comparison methods, such as those by Funkhouser et al. (2003), require multiple perspectives to be considered in order to resolve rotation issues. While this is not a concern for us because the use of our algorithm in games allows us to ignore differences in rotation, it would still be useful to gather at least six projections of the model, one from each axis direction. This will benefit us in two ways; firstly, we will be able to consider detail on all sides of the model and secondly it also allows us to better resolve the shape of the model along each axis.

For example, with the search image of a truck that has been used in most of our ranked experiment we do not know the details of the wheel positions, which would be located on the underside of the truck, opposite to our viewing angle. We also do not know the height of the truck; that is to say we do not know how deep the model is along the z-axis.

A nave solution to this may be to have separate Bands and Sub-clusters for each perspective. We would then gather the same perspectives of the search model and run our algorithm for each projection. After we gather values for each separate projection we would need to either fuse the values or create a decision engine to choose which model in the dataset is most similar to the search model from the most number of viewpoints. However, as we have seen, Hausdorff Distance calculations have a poor efficiency, even under the presence of downsampling. Carrying out this process six times for each model would create unwanted overhead, taking us further away from a goal of making this system operate in real-time. Therefore, we require further research into how our system would handle multiple projections per 3D model.

5.3 Improved Clustering

An obvious way that our research can continue to grow is investigate more advanced clustering techniques. The approaches we have explored in this paper are efficient and fairly effective. However, aspects such as overlapping Bands or choosing an appropriate representative for each Sub-cluster are not as refined as they could be. In the future, we will have the option of improving upon our existing clustering approaches and also looking into other clustering approaches.

Approaches have been proposed by Zhang et al. (2007) and in a brief presentation by Liu and Monticino[10]. The method proposed in the latter involves gathering the distance between every element in a dataset and creating a matrix of distances. The pair of objects with the lowest distance from each other are then extracted and merged together. This means they are put into their own cluster. The distance matrix is then recalculated with one element representing the two objects of the cluster. This time, the other objects again have their distance to each other calculated but also have their distance calculated to the cluster set. That is to say, an object is calculated against each of the objects within the cluster and the maximum distance is selected to represent overall distance between that object and the cluster. In our implementation this would mean that we calculate the distance between all images in a Band, select the lowest pair to be placed in a Sub-cluster, then repeat the previous step but now considering the new Sub-cluster as one entity. Liu and Monticino recommend recalculating the distance between the remaining objects in the dataset and the newly formed cluster. However, as Hausdorff Distance is inefficient, this will increase cluster building time by an immense amount. Instead, we propose that it it would be more beneficial to store all distances between objects of the complete dataset and then work with these figures rather then recalculating distances.

The afore mentioned Hausdorff Distance clustering algorithm is the more mature version of what we have done. We would no longer randomly select one image to be the representative and building a Sub-cluster around it, instead the entire Sub-cluster is considered during build time. In theory the clusters should be optimised such that the search image should still only need to be compared to one representative in the Sub-cluster. Thus, this method may increase performance of our overall approach without degrading runtime efficiency. However, what will be affected is build time efficiency because, to construct the initial distance matrix, each model in a dataset or Band will need to be calculated against every other model. For example, if we were not to use Bands, and simply use this clustering algorithm of our entire dataset of 1829 models, a nave approach would yield over

[10]`www.msu.edu/~liurong2/Cluster\%20with\%20Hausdorff.ppt`

three million calculations. If we assume downsampling is used and we know that on average each calculation takes roughly 0.085 seconds then to complete the building of the clusters would take 255,000 seconds, or nearly three full days. While this process is offline and is not required to work at real-time as the search process does, this amount of time is still unreasonable. In conclusion, this clustering method shows promise but we will need to look into improving it's build time efficiency in order for it to be useful in a commercial setting.

5.4 Other Assets

In order to build a complete package that could protect all of a user's intellectual property rights, it would be beneficial for research to begin examining other assets of games like Second Life. Firstly, for materials and textures we should be able to use a lot of the techniques developed in our current line of research. Textures are 2D images that are overlayed on top of 3D models to give them added levels of colour and flat detail. Hence, because textures are simply 2D images, we can continue to use pattern and shape recognition algorithms such as Correlation Filtering and Hausdorff Distance. For the protection of other assets such as audio and programming scripts, there will be much more work required to review existing techniques and build upon them.

5.5 Implement Within Game Architecture

Once we are able to demonstrate a stable asset protection system, it may be beneficial to implement this system into a proper game, or the shell of one at least. There would not need to be any complex game logic or even any gameplay or graphics other then loading the 3D models from our dataset. However, we would need to implement a client-server structure and an all encompassing package to create, upload, and protect 3D models. This may also lead to other fields of research such as the need to efficiently upload models to the server and for the server in turn to efficiently distribute newly added models to all user's in order for them to be viewed in the game world. Though compiling a full implementation such as this seems like a lot of extra work, we believe that it would be useful to be able to demonstrate to industry professionals the possibilities of such a system and the market potential for user content driven online games.

An extension of this idea is that we ideally need to test our implementation with a dataset that more accurately represents that of a games. The dataset we are using is comprehensive but it was designed to be used in content based model retrieval systems and thus has a wide range of models with varying shapes, orientations, and complexity. A game's dataset on the other hand will most likely contain many models with roughly the same shape or that can be placed into a definitive object categories such as human characters, weapons, furniture, architecture, animals, and so forth. Due to the reasons outlined in Section 3.2.2, all images within a category will have a similar orientation. Thus, our system would require less shape matching and more detail detection than it is currently tested for.

6 Conclusion

In this paper we have explored the potential of using the Correlation Filtering and Hausdorff Distance algorithms to protect the copyright of virtual assets. Specifically, the assets that we investigated were 3D models that are used to construct a game's environment. Our approach involved gathering a single 2D projection of each 3D model and applied the afore mentioned algorithms to judge the similarity between model pairs. After our initial experiments highlighted issues with both of these algorithms we set out to create a structure that could benefit both of them. To conclude, we will review the research questions that we asked in Section 1.3.

6.1 The Effectiveness of the Similarity Algorithms

We have mentioned earlier that effectiveness can be a very subjective measure in the context of our field of research. However, we believe that the consensus of all those who observe our results would be that the Hausdorff Distance algorithm continually outperforms the Correlation Filter approach. The ranked results throughout this paper clearly show that the Correlation Filter is returning dataset models that most people's intuition would judge as having very little in common with the search model. The Hausdorff Distance algorithm also has a tendency to rank some models high when a human classification would not do so, which can be viewed as an error or an issue associated with this approach. However, Hausdorff Distance also appropriately places models that we would classify as similar into high ranks. Generally, Hausdorff Distance returns models that are at least roughly of the same shape and orientation as our search model and, for the dataset that we have used, places all of the similar models in a high rank, even if they are not in a correct order or are interleaved with erroneous results. Thus, Hausdorff Distance strikes an acceptable balance between errors and good performance.

6.2 The Efficiency of the Similarity Algorithms

In terms of time based efficiency, the results are quite the opposite way around. Correlation Filtering can be performed on dataset of hundreds or even thousands images in a matter of milliseconds or seconds. This is a time frame that is acceptable for a registration process of a new 3D model or for a content driven information retrieval system and it can be improved to be acceptable for real-time use. On the other hand, the base Hausdorff Distance algorithm, with no modifications or improvements, can take minutes, hours, possibly even days for datasets of the same size. We can say that Hausdorff Distance has good results but unacceptable execution time, while Correlation Filtering is the inverse, providing poor results but having an execution time that is very desirable.

6.3 Our Clustering Structure: Benefits and Weaknesses

Our clustering structure, which includes Banding and the Hausdorff Distance aided Sub-clustering, has proven to be quite capable of improving the short falls of the two pixel based similarity algorithms. Our clustering approach improves the ranking results by adding initial barriers that reduce the erroneous models from being compared to our search image, which benefits both Correlation Filtering and Hausdorff Distance by allowing more appropriate models to move to the head of the ranks. In terms of efficiency, the clustering structure reduces the number of images that are compared to our search image and, along with downsampling of the images, greatly reduces computation time needed. For the size of the datasets that we are dealing with, this primarily benefits Hausdorff Distance and this algorithm is now approaching a point where it can be used during a registration process of a new model in a game or during a query in a content based information retrieval system, though it still needs further refinement.

7 Acknowledgements

Thank you to my primary supervisor, Jiankun Hu, for his ideas, contributions, and expertise that has made this paper possible. I would also like to thank him for continually challanging me and pushing me to succeed. Thank you to Fabio Zambetta for his knowledge of virtual worlds and his original ideas on the protection of virtual assets that forms the base of this paper. I would also like to thank Kai Xi for his expertise on Correlation Filtering, Hausdorff Distance and other pattern recognition algorithms, his ideas have continued to help solve pressing issues with our implementation. Finally I would like to thank my parents, John and Donna Raffe, for their constant support during this year and previous years of study. Thank you also to Donna Raffe for her proof reading efforts.

References

Canny, J. (1986). A computational approach to edge detection. *Readings in Computer Vision: Issues, Problems, Principles and Paradigms*, pages 184–203.

Chen, D., Tian, X., Shen, Y., and Ouhyoung, M. (2003). On visual similarity based 3D model retrieval. In *Computer graphics forum*, volume 22, pages 223–232. Amsterdam: North Holland, 1982-.

Funkhouser, T., Min, P., Kazhdan, M., Chen, J., Halderman, A., Dobkin, D., and Jacobs, D. (2003). A search engine for 3D models. *ACM Transactions on Graphics*, 22(1):83–105.

Gao, Y. and Leung, M. (2002). Line segment Hausdorff distance on face matching. *Pattern Recognition*, 35(2):361–371.

Heath, M., Sarkar, S., Sanocki, T., and Bowyer, K. (1998). Comparison of edge detectors. A methodology and initial study. *Computer Vision and Image Understanding*, 69(1):38–54.

Hester, C. and Casasent, D. (1980). Multivariant technique for multiclass pattern recognition. *Applied Optics*, 19(11):1758–1761.

Hu, J. and Zambetta, F. (2008). Security issues in massive online games. *Security and Communication Networks*, 1(1).

Kazhdan, M., Funkhouser, T., and Rusinkiewicz, S. (2003). Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, page 164. Eurographics Association.

Kumar, B., Savvides, M., Venkataramani, K., and Xie, C. (2002). Spatial frequency domain image processing for biometric recognition. In *Proc. IEEE Int. Conf. Image Processing*, pages 53–56. Citeseer.

Kumar, B., Savvides, M., and Xie, C. (2006). Correlation pattern recognition for face recognition. *Proceedings of the IEEE*, 94(11):1963–1976.

Lehdonvirta, V. (2005). Real-money trade of virtual assets: New strategies for virtual world operators.

Mahalanobis, A., Kumar, B., and Casasent, D. (1987). Minimum Average Correlation Energy Filters. *Applied Optics*, 26:3633–3640.

Muttik, I. (2008). Securing Virtual Worlds Against Real Attacks.

North, D. (1963). An analysis of the factors which determine signal/noise discrimination in pulsed-carrier systems. *Proceedings of the IEEE*, 51(7):1016–1027.

Ohbuchi, R., Mukaiyama, A., and Takahashi, S. (2002). A frequency-domain approach to watermarking 3D shapes. In *Computer Graphics Forum*, volume 21, pages 373–382. Citeseer.

Osada, R., Funkhouser, T., Chazelle, B., and Dobkin, D. (2002). Shape distributions. *ACM Transactions on Graphics (TOG)*, 21(4):807–832.

Petrou, M. and Bosdogianni, P. (1999). *Image processing: the fundamentals*. John Wiley & Sons, Inc. New York, NY, USA.

Ravichandran, G. and Casasent, D. (1994). Advanced in-plane rotation-invariant correlation filters. *IEEE Transactions on pattern analysis and machine intelligence*, 16(4):415–420.

Rymaszewski, M., Au, W., Wallace, M., and Winters, C. (2007). *Second life: The official guide*. Sybex.

Shapiro, M. and Blaschko, M. (2004). On Hausdorff Distance Measures. Technical report, Technical Report UM-CS-2004-071, Department of Computer Science, University of Massachusetts Amherst.

Tangelder, J. and Veltkamp, R. (2008). A survey of content based 3D shape retrieval methods. *Multimedia Tools and Applications*, 39(3):441–471.

Tung, T. and Schmitt, F. (2004). Augmented reeb graphs for content-based retrieval of 3d mesh models. In *Shape Modeling Applications, 2004. Proceedings*, pages 157–166. Citeseer.

Turin, G. (1960). An introduction to matched filters. *IEEE Transactions on Information Theory*, 6(3):311–329.

Venkataramani, K. and Kumar, B. (2003). Fingerprint verification using correlation filters. *Lecture Notes in Computer Science*, pages 886–894.

Y. Wang, J. Hu, K. X. and Kumar, B. (2007). Investigating correlation-based fingerprint authentication schemes for mobile devices using the J2ME technology. *IEEE Workshop on Automatic Identification Advanced Technologies (AutoID).*

Zhang, K., Orgun, M., and Zhang, K. (2007). A prediction-based visual approach for cluster exploration and cluster validation by HOVˆ 3. *Lecture Notes in Computer Science*, 4702:336.