# Machine Learning in Computer Graphics/Animation

Noah Hendrickson

May 7, 2023

**Abstract**

Machine Learning can apply to many fields of computer science. In this literature review, various papers on machine learning and its application to computer graphics and animation are discussed and evaluated on their contributions to the field as well as what applications they could possibly lead to. Graphics and A discussed include super resolution, ray tracing, path planning, simulations, character animation and more.

## 1 Graphics, Animation, and Machine Learning

Computer Science is a wildly diverse field with everything from low level work on Operating and Distributed Systems, to high level front end website development, to visualization of data, to making machines themselves learn like humans do. The two fields that I want to explore the overlap between in this literature review are Machine Learning (ML) and Computer Graphics/Animation.

### 1.1 Computer Graphics

Computer Graphics deals with visualization of objects on computers, utilizing things like transformations, vertices, textures, and various other concepts in order to render realistic looking models and images to the computer screen and manipulate them in many ways. What sort of ways can we render images? Is it better to attempt to simulate how real light works like in ray tracing or is losing the quality in favor of speed more beneficial like with scan conversion? How do we make images look as good as possible on a screen and how can we apply those images to arbitrary models? How does lighting work in a simulated scene? These questions are the sort that computer graphics attempts to provide solutions to. Graphics is responsible for many things such as NVIDIA's Real Time Ray Tracing technology, all video games that you might play on a console, the development of the Graphics Processing Unit, and many many other technologies.

### 1.2 Computer Animation

Computer Animation takes those things that have been rendered to the screen and attempts to move them in realistic looking ways using various algorithms and math techniques. How to simulate realistic water? What about realistic fire or smoke? How do we make a character in a game or movie move like a real human? How do those characters that we create know where to move to? Computer animation seeks realism and these are all questions that must be answered in order to provide that realism on a computer. It's used in many settings but most prominently in animated movies made by companies like Pixar or Disney, and, like graphics, the games that are played by millions.

### 1.3 Machine Learning

Machine Learning takes the idea of humans learning and growing and attempts to apply that to computers. It combines the processing power of a computer and the ability to ingest new information and make better decisions based on that data in order to solve a multitude of problems that humans alone would not be able to do at all or at least in a timely manner. Statistics, linear algebra, optimization, and various other techniques are all combined in order to solve problems from classification, to regression, to clustering, to even image or text generation. ML is responsible for some of the most eye catching and controversial advances in technology in the past few decades. It is used in fields such as biology/medicine to predict trends in diseases or determine medical history. Its used in transportation for routing and making self driving cars a reality. Machine Learning is used to translate languages quickly and even go so far as to create chat-bots that can have convincing conversation. Its hard to go even a couple days without hearing something about GPT-4 and how it may take jobs because of its capabilities or image generation models being used to deep-fake popular people. Machine Learning in the past few years has truly come to the forefront of technology advancement.

### 1.4 Intersections

One might think that, when hearing the descriptions of these fields, that they have nothing to do with each other? How does something like machine learning apply to graphics and animation? That is exactly what I want to explore with this literature review. The field of machine learning is constantly expanding, and that necessarily means expanding to find solutions to problems in other fields of computer science.

In the fields of graphics and animation, new advancements and new solutions to problems are always looking to be found. What is the best way to simulate the realistic movement of a human or a humans hair? What about simulating realistic fluid? How do we quickly illuminate a scene or generate new textures for our scene objects so that the terrain doesn't look samey and repetitive? Machine Learning can answer some of these, and more, questions using the vast wealth of techniques that have been developed through the decades.

In this literature review, I will examine various papers covering areas of graphics and animation such as fluid dynamics, animating realistic motion, global illumination, and more. These papers will all solve problems by utilizing machine learning techniques such as neural networks, support vector machines, K-nearest-neighbors, and many, many others. By the end of this paper, I hope that you, the reader, will have gained a deeper understanding as to how machine learning can be applied to graphics and animation and just how powerful of a tool it is in those contexts.

## 2 Graphics Research

Graphics research with machine learning is pioneered heavily by GPU companies such as NVIDIA with their DLSS (deep learning super sampling) and RTX (real time ray tracing) technologies and AMD. Even Intel and Microsoft have, or will have, their foot in the door of machine learning applied to graphics. Other applications include texture generation, image denoising, image and video generation, and more. In order to really explore how machine learning is used to solve these problems, we must go to the experts and examine the internals of these technologies as well as the technical papers that describe them..

### 2.1 Super Resolution

Super Resolution is a technique to up-scale images into a higher quality through the use of various techniques, including deep learning. [35] This technology has a lot of applications, but one major application is gaming where high frame rates and fast computation is very important, meaning running at a lower resolution for faster computation and up-scaling to a higher resolution for better visuals is incredibly inticing. This process may seem quite a bit like upscaling, and some may wonder why they're differentiated. The process of up-scaling utilizes algorithms such as nearest neighbors search, various forms of interpolation, and spline fitting, whereas SR uses more complicated algorithms that will be discussed. The main difference though is that SR makes the assumption that there are multiple images contributing to the high resolution version of a low resolution image, i.e. motion from adjacent frames in a video, whereas up-scaling does not make this assumption. Up-scaling also loses many of the higher frequency features of the image, whereas SR does not.

When it comes to devising a method to produce high resolution (HR) versions of low resolution (LR) videos, there are a few factors that must be taken into consideration. Motion between frames must be estimated in order to determine relationships between frame movement, motion blur must be taken into consideration so as to keep realism, and the HR frame image must be obtained from the LR frame image. [14] There are a plethora of ways that this can be achieved. Some methods involve pre-computing motion data, also called registration, then obtaining the HR images using that data later on. [4] Other methods employ regularized reconstruction by modelling errors in the registration process as gaussian noise and utilizing deconvolution. [18] These techniques tend to involve many unknown parameters which need to be painstakingly tuned, however, still other methods can solve these issues. In the paper titled ”Variational Bayesian Super Resolution”, the authors implement a SR method involving Variational Bayesian Analysis to provide uncertainties during the restoration process and lead to more robust HR images and prevent more errors in addition to adaptive motion estimation to produce even more accurate motion estimates than previous methods. [1]

While these techniques work, some of the most effective techniques of implementing SR involve the use of deep learning, and especially convolutional neural networks, generative adversarial networks, and transformers. Convolutional Neural Networks, CNNs, were one of the first deep learning techniques utilized for the SR task. One of the earlier uses of a CNN for the task of super resolution is presented in ”Video Super-Resolution with Convolutional Neural Networks.” [14] In the paper, the authors present three different CNN architectures which are differently modified versions of a reference SR CNN. The reference architecture that the network is based off consists of 3 layers and an output, with the variations between the three networks coming from the times in which the neighboring frames are incorporated into the network. The three different times where neighboring frames can be concatenated are before layer 1, between layer 1 and layer 2, and between layer 2 and layer 3. The authors also implement two other techniques in Filter Symmetry Enforcement (FSE) and Adaptive Motion Compensation. The former enforces weights in the convolutional layers due to properties surrounding those weights being the same for inputs which allows backpropogation to filters to occur at the same time. The latter serves to reduce the influence of neighboring frames by ”controlling the convex combination between the reference and the neighboring frame at each pixel location.” [14] The authors tested their network by measuring the peak signal to noise ratio (PSNR) and structural index similarity (SSIM) and found that in almost every case it outperformed any of the previously mentioned techniques. Only in the case of a 4x scale factor for the HR image did the CNN fall behind the Bayesian technique, even though on average the CNN still beat out the Bayesian technique handily. Due to the rapid innovations in the field of ML, even more techniques have been found since that paper. GANs have recently come into favor for generation of images, and, naturally, they work pretty well on generating HR versions of LR images as well.

Generative Adversarial Networks are a type of network that focus on generating new things by perturbing deeply encoded feature vectors to produce a new thing when decoded. GANs employ a generator-discriminator architecture in order to train, where the generator generates a sample point and the discriminator attempts to discern whether the sample, and other real samples passed through, are ”real” or ”fake.” This serves an adversarial purpose and allows the generator to train to get better at making real looking images. This applies to SR in that a LR image can be encoded into deep features and a GAN can decode it into a HR image. In the paper titled ”Spatially Adaptive Losses for Video Super-Resolution With GANs”, the authors implement a ResNet SR generator and a convolutional discriminator. [34] In addition to the normal adversarial min-max problem, the authors also propose a spatially adaptive loss that applies to both pixel and deep feature space which controls the amount that certain areas contribute to the loss depending on spatial activity. High spatial activity areas, like edges, would contribute more than low spatial activity areas. This model outperforms the previous state of the art GAN models in each category (PSNR, SSIM, PerceptDist) at all scale factors (2x, 3x, 4x). The issue with GANs is that convergence is slow and the discriminator can easily outpace the generator to a point where the generator does not learn anything.

Even newer ML technology continues to provide insight into better SR methods. Recent transformer architectures, such as the spatio-temporal architecture that is presented in ”Video Super-Resolution Transformer”, can achieve either just as good or better PSNR and SSIM scores than the current state of the art, but at the cost of significantly more parameters. [28] That last point gets at a big part of a successful super resolution method: its important that it does the job quickly **and** accurately. A transformer architecture with 43.8M parameters may score well, but if it takes longer than 10ms at inference time, it already will not be fast enough to be deployed in a real time environment. This can be seen by looking at what is used in production. NVIDIA, with its insanely powerful GPUs, has Deep Learning Super Sampling 3 (DLSS 3), which is a convolutional SR method utilizing optical flow accelerators to capture motion and speed of pixels. [20] If even NVIDIA is using a method that was first introduced more than 7 years ago, its clear that speed is king in this area.

## 2.2 Realistic Rendering and Illumination

Rendering in computer graphics is the process of computing an image or video that may or may not be realistic from a set of 2d or 3d points in space or set of equations. The two main kinds of rendering that are done are scan conversion and ray tracing. Scan conversion is the typical OpenGL, all triangles, depth buffer, shader, vertex, etc... process that you might learn about in an intro graphics class. The user specifies vertices in space that define triangles that are moved around in the computer world space using transformations. Viewing of these shapes is done through perspective projections and specialized camera coordinate systems. Lighting, shadows, reflectance, and transparency, while possible to do, do not look very realistic through this means of rendering. Everything is represented with vectors and matrices. GPUs have been honed to be as fast as possible on this type of rendering. Special hardware has even been made to do the kinds of math operations involved with transformations and clipping.

On the flip side, ray tracing, while not being a new technology, has certainly come into the mainstream view as of recent years with the advent of NVIDIA's RTX GPU line, allowing ray tracing to be done in real time. Ray Tracing is the process of rendering realistic looking scenes through the use of "rays." The basic idea of a ray is that it is a line that goes from the eye of the camera in a direction out into the scene. You can think of it sort of as a light ray, but in the opposite direction. Because the ray is just a line, you can do all sort of intersection tests with it and objects in the scene to be able to determine where and what the ray hits as it goes into the scene. Whatever the ray hits transfers some of the color of the object to the ray and the final color of the ray is used to color the pixel it was shot through. There are two key difference between ray tracing and scan conversion. The first is that ray tracing can make incredibly realistic looking scenes. Bouncing rays for reflections is as easy as a simple equation, transparency is almost equally as easy but with a few more quirks, realistic lighting can be implemented using equations like the Bi-Directional Reflectance Function or Radiosity in order to render things such as light passing through transparent objects, total internal reflection, subsurface-scattering, and numerous other phenomena found in real life that scan-conversion rendering could never hope to capture. However, this all comes at the price of the second key difference: ray tracing, compared to scan conversion, is painfully slow. All of the intersection testing and reflecting and light calculations build up to produce a program that takes a ton of CPU/GPU power to run, and, unlike scan conversion, there is not widespread access to specialized hardware to do the kind of computations needed. While there is the obvious solution of just creating hardware that is simply fast enough to do it, that is not always feasible. Thus, machine learning can be used to accelerate these computations in order to render realistic looking scenes in a fraction of the time.

One of the major bottlenecks in ray tracing and rendering realistic scenes is the aspect of global illumination. Light goes everywhere, so in order to get a realistic approximation of it, graphics programmers must get crafty. In the past methods such as Bi-Directional Path Tracing [17] or Metropolis Light Transport [32] have been used in addition to Radiosity and Radiance Caching, both of which are "biased" methods, meaning they produce an almost realistic looking image at less computation cost. None of these methods have been based on machine learning techniques, however.

Before getting into deep learning and more advanced machine learning techniques for global illumination, I want to examine a paper from 2013 titled "Global Illumination with radiance regression functions." [23] In a scene with a moving light source, the radiance at a point can be determined utilizing various information about the point such as the surface normal, BRDF for the point, viewing direction, incoming radiance from viewing direction, and a few other factors. Reflected radiance at that point can be split into two components: a local illumination component and an indirect illumination component. While the local component is easy to calculate, its the indirect component that requires much computation. In order to efficiently approximate this value, the authors implement a 2 hidden layer neural network that takes in the position, a point light's position, viewing direction, the surface normal at the position, and reflectance parameters and outputs the reflected radiance at that point. This results in a non-linear mapping between the values at the intersection point and the reflected radiance, leading to a good approximation. In order to handle scenes with lots of objects, space partitioning is also implemented to speed up computation by fitting a seperate model to each of the space partitions. While this method produces good results, it relies on much pre-computed data for the scene as well as too large of inputs to the network, leading to slow training time.

As far as I can tell, the first introduction of deep learning into the problem is a paper titled "Neural Network Ambient Occlusion" in which the authors present a Neural Network based method to calculate ambient occlusion. [8] Ambient occlusion is a part of the global illumination that is especially computationally expensive, so being able to quickly approximate it with a Neural Network is an important speedup. The model is trained with depth and normal buffers for a pixel and the ground truth occlusion mapping for that pixel in order to produce an approximate occlusion mapping. This process is done offline, however, can be dropped in in place of other ambient occlusion algorithms with noticeable performance and accuracy improvement. The resulting network shows noticeable performance and accuracy improvement over the, at the time, current state of the art screen space ambien occlusion algorithms, being able to sample more in a shorter span of time. While ambient occlusion is important, its not everything to global illumination.

In recent years, as mentioned in the previous section, generative adversarial networks have come to be useful in many fields, and global illumination is no different. The first paper to introduce GANs to the world of global illumination is titled ”Deep Illumination: Approximating Dynamic Global Illumination with Generative Adversarial Networks.” [31] In the paper, the authors introduce a novel method to utilize various information buffers created in the process of the ray tracing as inputs to the GAN which then would output the image to be rendered from those information buffers. In diffuse scenes, this model produces impressive results, even being able to produce the lighting results in real time with dynamic camera and object positions. Though, despite being a good approximation of realistic lighting, it only accounts for diffuse materials, meaning transparency is a no-go. Additionally, it wasn't perfect, and occasionally introduced artifacts or illuminated incorrectly when adding new objects to the scene.

In order to solve the issue of lack of scene generality and inability to render transparent objects, the paper titled ”Deep Radiance Caching: Convolutional Autoencoders Deeper in Ray Tracing” introduces a method for deep radiance caching. [11] Normal radiance caching ”focuses on accelerating rendering of glossy materials, by caching an optimized representation of the radiance received on a surface.” In order to extend this to machine learning in order to solve the diffuse-only and lack-of-generality limitations, the authors of the paper combine radiance caching with a convolutional autoencoder to predict a radiance map at the first intersection point in the scene using depth and normal map information. That high quality radiance map is then used to approximate the indirect light from any bounces beyond the first. This method differs from the previously mentioned GAN method [31] in that it does find the first intersection in the scene for each pixel, rather than not casting any rays, however, this difference leads to numerous benefits including no offline pre-training, easy generalization to any scene, a much wider variety of materials able to be rendered, and accelerated rendering speed.

All of the discussed methods so far have been focused around rendering via ray tracing and solving the slow global illumination process. In a bit of a divergence from that while still keeping the general idea, I want to examine the paper titled ”NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis.” [21] Ray tracing can create incredibly realistic images, videos, and scenes through intense calculation, but in order to actually create those realistic scenes, the creator of the scene must painstakingly describe the locations of every triangle, every circle, every bit of the scene in addition to every lighting coefficient, reflectance and refractance property, light source, and anything in between. In the paper, the authors solve this problem of having to create the scene manually by devising a network that can take images of a real scene and realistically view the scene from any angle, despite no actual modelling of the scene. In order to achieve this, a scene is represented as a 5D continuous function consisting of the radiance emitted in each direction at each point. This representation is approximated using a multi-layer perceptron which takes in an initial position and viewing direction and outputs color of the ray and the volume density, the ray in this case being represented by the position and view direction being marched through the scene. Many of these rays are marched through the scene during training time. The volume density represents the probability that the ray terminates at location x. An additional positional encoding, like those used in transformer architectures but with no notion of order, was also added to the MLP due to poor performance at representing high frequency variation in color and geometry. The resulting model had higher PSNR and SSIM scores than any of the other state of the art models that were tested against that also employ other deep learning techniques such as other kinds of MLPs and convolutional models. A really interesting extension of this technology is presented in the paper titled ”Unconstrained Scene Generation with Locally Conditioned Radiance Fields” in which the authors utilizes GANs again to generate new scene geometry with a dynamically moving camera, showing that the applications of this technology are real and very interesting. [3]

## 2.3 Graphics Papers Analysis

The papers discussed here are certainly not all of the research regarding machine learning in the field of computer graphics. Rendering, Illumination, and Super Resolution are just three topics in a vast field. Other topics that involve machine learning in graphics that I won't go in depth on include things like using neural networks for light scattering in hair [36], relighting of scenes from sparse samples [30], texture synthesis [5] and semantic image manipulation [30], and much much more. If I wanted to cover everything, I think that I would run out of both time and page space, however, these give a good representation of the current trends of the field as well as important developments.

When it comes to super resolution research, each paper discussed here did an excellent job of summarizing the current state of the art, while also contributing something novel to the field. Whether it was testing various CNNs, using GANs, or coming up with a novel transformer, each paper layed out how their contribution improved on current technology in a clear and understandable way. The only thing that I wish the papers discussed more was the implications on speed at inference time. It is super important for these models to be fast, as was discussed, however, few of them mentioned it in a meaningful capacity. This, I think, was a detriment to the ideas that the papers presented, as it leaves out a huge area of application for these models in real time applications. It is clear that the research is far from done. The papers discussed do not go back very far in time and already there have been tons of developments. With the advent of transformers and their ability to get incredibly detailed representations of images, its possible that the future of super resolution is heading the way of attention. This would only be the case, though, either if computation power increases dramatically, or something is developed that increases the speed or decreases the size of these models. Most of the transformer papers that I found dealt with single image super resolution, so I think that it would be interesting to see more papers explore the multi-image super resolution using transformers so as to get the results of using a transformer while also reducing the cost at inference time. In addition to transformers, I think that it would be interesting to see more papers explore super resolution using diffusion models. Diffusion models have come to be incredibly powerful tools in generating images from text prompts, which GANs were previously used for as well, so I could see them quite naturally being extended to fit this niche as well.

As for the rendering and illumination papers, they are very much in the same boat of being comprehensive on previous state of the art, and how the papers themselves iterate, improve, and advance that state of the art. Each of the papers takes into consideration important factors about the mediums that they are applying the methods to. Inference speed is considered as its a core importance in the ray tracing process. Accuracy is always considered as there's no point in using different methods if they are just completely worse. I noticed, especially with these papers, that they built off of each other in a very natural way. Earlier papers would come up with a solution to some illumination issue, but that solution would come with its downsides. Then the papers from a few years later would employ new ML techniques and solve those downsides while keeping the good performance. So on and so forth. The thing that I wish these papers did more of was explore how applicable these models could be to allow real time ray tracing in things like games. Most of them did mention the improved performance and that they could be used in real time, however, none really gave examples of them being used in real time. This could be a medium issue, as video is hard to convey through a PDF, but I think that it would have been interesting to see the actual fruits of their labor, and how it compares to non-approximation methods. Additionally, it would be interesting for some of these papers to go more in depth on descriptions of previous state of the art methods. Many of the papers, when mentioning the previous state of the art methods, would just say the acronyms and cite the paper they were presented in, where I think describing, at least at a high level, the method and how it worked would give the reader a better and more thorough understanding of what was being improved and how. I think this also applies to the results sections. Many of the papers used metrics to evaluate their methods, but did not thoroughly explain why those metrics were used and what they were measuring. In a different direction, I think it would be interesting for more investigation to be done on the application of, again, diffusion models to this process. If GANs can be applied to solve some of the problems presented, I think that it would again be pretty straight forward to also apply diffusion based models. Whether or not this would work, I do not know, but it would be interesting nonetheless.

Overall, this process of learning more about the state of the art in machine learning applied to graphics research has made me all the more interested in pursuing further applications, however, there still remains a large field to explore in that of computer animation.

## 3 Animation Research

Animation research with machine learning is a very wide and deep subject. Applications of machine learning to animation include path planning, realistic physical simulations such as cloth or liquid, crowd simulations, realistic animation of human or animal meshes, animation of still images, realistic facial expressions for applications such as VR, realistic behavior of agents in a scene, and many more. Because there are many different topics, most of which don't really have much in relation to each other thematically, this section will be structured slightly different to the previous. Rather than analyzing many papers on one or two topics, one or two papers will be discussed for multiple topics.

### 3.1 Navigation: Path Planning and Crowds

Many non-machine learning based approaches to plan paths include probabilistic road-maps, Voronoi diagrams, rapidly exploring random trees, visibility graphs, and other techniques. While the non-ML based path planning algorithms work, very often they lack the natural look that we might expect from a real agent navigating a scene. The paper titled "Path Planning using Neural A* Search" presents a novel data-driven search method for path planning problems. [37] The A* algorithm is a heuristic-driven search algorithm that is often implemented in search-based planning, as it is guaranteed to find a solution path if one exists, however, these search-based planning methods are often not as efficient as others. In order to remedy this, the authors devise a Neural A* in which the typical A* algorithm is converted to be differentiable in order to propagate loss and learn. The model implements a convolutional autoencoder that translates a problem instance into a guidance map which imposes a guidance cost to each node in the space. A "differentiable A* module", which represents variables in the traditional A* algorithm as matrices in order to allow for matrix operations, is then executed on the guidance map in order to determine a path. Inside the actual module, various matrix operations, convolutions, and cost calculations are done in order to determine this path. The loss for the network is the mean L1 loss between the ground truth path map and the map of all the nodes searched by the Neural A*. The results of the testing done show that the Neural A* method worked well in various environments but had limitations such as assuming grid world environments and that each node had unit cost. It sometimes outperformed previous data-driven models, but faltered on larger, more complicated spaces.

The previous paper explored planning for a single agent, however, in order to achieve crowd simulation, many paths must be planned at once, and interaction must be handled. Some methods utilize physics properties to simulate crowd behavior, however, these struggle to capture complexity of human behavior. Other methods use deep learning in order to learn the interactions, however, these struggle to generalize beyond the data provided. To implement a better method, the authors of "Physics-infused Machine Learning for Crowd Simulation" combine physics and deep learning to produce a Physics-Infused Machine Learning framework for crowd simulation. [38] The model introduced in the paper combines a physical model and a neural network model. The physical model theoretically could be represented as any physical model for modelling agent interaction, however, the authors chose the most widely used model, the social forces model [6], to demonstrate their method. The neural network portion required a bit more thought and care, as learning small interactions between pedestrians, generalizing to different scenarios, and generalizing to longer time periods is a very difficult task. In order to solve this, the authors introduced three methods: a graph network based crowd simulator (GCS), student-teacher co-forcing training (STCT), and collision avoidance learner (CAL). The graph based crowd simulator is based on the idea that crowds are similar to particle systems and, because GNs have been shown to work well on those, they might work on this problem as well. The STCT portion is introduced to allow for more generalization in the model and employs techniques found in NLP problems to train with multiple-step roll-outs and add a reverse long-term discounted factor to "make the model focus on long-term accuracy and deal with error accumulation to get robust predictions." The CAL accounts for the previous two methods not being able to see very small collision avoidance moves by adding loss terms. All three of these methods together allow for the model to outperform the state of the art data-driven models by a wide margin, and even inch ahead of some of the most widely used physical-driven models.

Simulation of crowds is all well and good, but its important to have a way to analyze how good a simulation is doing at capturing a realistic crowd. The authors of "A Data-Driven Framework for Visual Crowd Analysis," one of which is one of the best professors I've had, propose a machine learning based method to do just this. [2] The authors make use of techniques such as support vector machines, k-nearest neighbors, localized p-value estimation, and Pareto Depth Analysis [10], the first three mainly for comparison, allowing for unsupervised anomaly detection under multiple evaluation criteria in real time. Various comparison metrics are used between agents, however, SVM, k-NN, and k-LPE all fail to capture anomalies when multiple criteria are applied. With Pareto Depth Analysis, along with dimensionality reduction in order to allow real time, applied, anomalies are easily detected under a variety of simulation techniques and criteria. The method is not without limitations, however, such as requiring testing and training data to be very similar and requiring external representative data. Additionally, the method assumes the training data, consisting of real life crowd data, is of normal movement, allowing for the unsupervised nature of the method.

## 3.2 Physics: Fluids, Moving Images, and Cloth

Computational fluid dynamics, the study of simulating fluid flow such as air, water, and fire, is a large point of research in both computer animation as well as mechanical and aerospace engineering. Most non-ML techniques to simulate these phenomena include utilization of approximations of the incompressible Navier-Stokes equation which relates momentum, pressure, temperature, and density of a fluid to approximate flow. In this section, I will cover two papers that propose different Neural Network based strategies to simulate these fluids. The first is titled ”Machine learning–accelerated computational fluid dynamics” [16], in which the authors utilize a convolutional neural network, and the second is titled ”A new fluid flow approximation method using a vision transformer and a U-shaped convolutional neural network” [13], in which the authors combine a vision transformer and convolutional neural network.

While both papers attempt to solve the same problem of making approximation of the Navier-stokes equation more efficient, they take a fundamentally different approach to it. The CNN paper takes the approach of rather than purely applying a ML solution, instead using ML to accelerate current iterative solvers. The ViT-UNet paper takes the opposite approach and purely applies the ML solution instead of augmenting iterative solvers. The iterative solver being augmented in the CNN paper is one that ”employs finite volume method on a regular staggered mesh, with first-order explicit time stepping.” [16] A convolutional network then, at each time step, generates a latent vector for each grid location in order for the iterative solver to account for local structure. This method of solving allows for learned interpolation, in which volume averages are able to be interpolated in order to calculate convective flux with first order accuracy, or learned correction, in which a residual correction to the navier-stokes equation is modelled which is simpler to implement, more flexible, and has fewer inductive biases, but is less interpretable. The downside to this method of approximating fluid flow is that conditions for the initial flow velocity must be fixed. Additionally, due to the nature of CNNs, they are not able to capture global context to the flow of the fluid easily. In order to solve these issues, the ViT-UNet paper employs a visual-transformer to capture that global context. Input images are converted into feature maps in which local features are extracted in addition to embedding of velocity conditions and geometric and condition features. Positional embeddings are used to identify positions and self-attention is calculated on all to get global context. The decoding portion then upsamples these features to obtain the steady-state flow. Total variation loss, representing natural flow regarding vertical and horizontal directions, is used. When it comes to the CNN paper's results, they are quite impressive. The authors say a roughly 40x to 80x speedup in computation speed while maintaining similar accuracy to baseline methods, but still has the downsides mentioned before. In a similar vein, the ViT-UNet paper sees results that exceed 1000x speedup over baseline methods, with very little accuracy loss. The authors also see very good generalization of their model when working with unseen initial velocity vectors. The downside that both methods share is that neither tackles the problem of 3D fluid dynamics, both being only confined to 2 dimensions.

In a similar vein to fluid dynamics, but used for a different task, the paper titled ”Animating Pictures with Eulerian Motion Fields” handles simulation of fluid flow in a more abstract way. [9] Rather than attempting to simulate fluid flow via Navier-stokes, the authors utilize a generative encoder-decoder architecture that approximates eulerian integration to turn a still image of fluid into a looping video of flowing fluid. An image-to-image translation network generates a motion field that can be reused for the whole video. At each frame of the video, a displacement field is generated using Eulerian integration and the motion field generated previously. A novel technique called symmetric splatting is then used to warp the deep features using the displacement map. Symmetric splatting solves issues native to this method in which portions of the image are left empty due to movement of the pixels by making use of the property of the method that allows generation of displacement fields in the past as well to fill in those empty spaces. Collisions of pixels are handled by a modified pixel-softmax with a learned component. To address the looping of the video, the pixel-softmax is modified further such that the past displacement map contributes all the pixel values at the start and the ratio transitions to the future displacement field contributing towards the end, guaranteeing the looping. This method produces relatively realistic results, however, they do sometimes look odd. Additionally, much like the previous two papers, it was not generalized into 3 dimensions.

Related to integration, but not really to fluid simulation, is simulation of cloth. In basic methods of cloth simulation, integrations methods such as Eulerian Integration, Heun's method, or 4th order Runge-Kutta can be used to simulate semi-realistic movements. Going beyond integration, various methods of optimization and solutions of large linear systems can be done to make even more convincing cloth simulations. In the paper titled ”Swish: Neural Network Cloth Simulation on Madden NFL 21”, researchers from Electronic Arts introduce a PCA and Neural Network method of simulating tight clothing for their game, Madden NFL 21. [19] Swish uses a PCA-based representation of the cloths shape to train a stateless pose-to-cloth shape regressor. The shape vector for the cloth is then inferred for each frame using stacked PCA weights for the normal maps and mesh shape. This method is very fast, being able to run in real time, but does not handle dynamics, a significant issue.

## 3.3 Motion: Humans, Animals, and Motion Phases

Movement and environment interaction for characters is a key part of computer animation, and often one of the hardest to do realistically. Many tactics such as inverse kinematics, key-framing, splining, etc... attempt this, but few can get *really* realistic movement.

A series of papers, presented in SIGGRAPH each year since 2017, seek to address the problem of realistic character animation through the use of Deep Learning, specifically the use of motion phases and neural networks trained around those. The earliest paper, titled ”Phase-Functioned Neural Networks for Character Control”, is the introduction to the Phase Functional Neural Network. [7] The PFNN is a network based around a periodic function of the phase, which the paper calls the phase function. In order for the network to determine the movement of the character, it requires the current phase, the user control parameters, the previous character state, and the environment parameters. The network then computes the updated state and phase of the character for the current frame in addition to a trajectory prediction and food contact points for inverse kinematics post-processing. The network is a mere three layers and the phase function used for the network, while theoretically able to be anything, was chosen to be a cubic Catmull-Rom spline. This method of movement generation works incredibly well, being able to generate very realistic movement while remaining compact with fast enough inference that it can be done in real time, easily beating out the current state of the art. It can even handle uneven terrain, so long as that is included in training!

The previously described paper, while revolutionary for bipedal character animation, was not enough to simulate quadruped motion. The second paper, titled Mode-Adaptive Neural Networks for Quadruped Motion Control” and presented in SIGGRAPH 2018, solves this issue. [39] Quadruped motion is much more complicated than bipedal motion, due to the wide variety in gaits, and data is very hard to come by. The task is so difficult, in fact, that there was no previous work in data-driven, realistic quadruped motion before this paper. In order for the authors to simulate quadruped motion, a new architecture, called Mode-Adaptive Neural Networks, was needed. This network architecture makes use of a technique called ”Mixture of Experts” [12] in which ”a number of experts are used to cope with inputs in different regions.” In the case of the MANN presented here, the mixture of experts refers to the use of a gating network that determines weights for the motion network through blending of the expert weights which are trained on particular types of movements. The motion network takes in the previous character state and user controls in order to determine the current state of the character. The motion of the quadruped character was classified into six motion types–locomotion, sitting, standing, idling, lying, and jumping–and four locomotion modes–walk, pace, trot, and canter. The final model easily outperformed all other baseline models and got very close to the ground truth values in all the testing. Having used the software myself as well, I can definitely say that the movement is incredibly realistic. The downside to this model, as compared to the PFNN, is that the model cannot handle terrain outside of flat ground, however, some IK tricks can be used to simulate it, but it is not nearly as realistic as the walk cycle.

Walking is an important part to character animation, however, it is not everything. In order to make a character move in a realistic manner, yes, they must walk realistically, but they must also realistically interact with the environment and other agents. This was covered a bit in the 2017 paper with handling uneven terrain, however, there are many more cases such as climbing through windows, picking up objects, maneuvering around other agents, and dynamic arm/leg movements that were not covered. The papers titled "Neural State Machine for Character-Scene Interactions" [27], "Local Motion Phases for Learning Multi-Contact Character Movements" [25], and "Neural Animation Layering for Synthesizing Martial Arts Movements" [26], presented in SIGGRAPH 2019, 2020, and 2021 respectively, solve these sorts of interactions and more. The 2019 paper deals with interaction between the character and the environment, such as sitting in a chair, climbing through a window, or picking up a box. It utilizes much of the same ideas found in the previously described papers such as expert weights to encode various different positions, phase functions for smooth movement to the goal position, and a motion-prediction-plus-gating-network architecture, albeit with different inputs and outputs specific to the task. The 2020 paper deals more with character-character interaction, such as two characters playing basketball, dribbling, blocking, and the like. Like the previous two papers, this one implements the experts technique with the motion-prediction-gating-network architecture, but with a couple important differences. The main difference is the introduction of local motion phases, which allow independent control of various limbs of the character leading to the ability to do something like dribbling a basketball or reaching out to block. The key to introducing these local motion phases is introducing new inputs to the network such as conditioning features, contact information, opponent information, and the local motion phases themselves. Although the example given in the paper is specific to the action of playing basketball, it is generalizable to any action with the proper training data. Once again, the results of this technology are incredibly realistic and also incredibly cool! The method can even be generalized to handle the problem of quadruped locomotion by computing the local motion phase of every foot by its contact to the ground, greatly improving over the previously described method for quadruped movement. [39] The 2021 paper takes this idea of limb control even further by introducing a motion generating scheme in which the motion generation process is decoupled from the control process, allowing for novel animations to be created easily and quickly. The motion generator network, again, uses the mixture of experts scheme and takes the joint trajectories as well as the pose data to generate new pose data for the next frame. Control modules are introduced in order to more finely control the movements that the model can learn. These are general and can theoretically be anything, but for the sake of the paper, the authors introduced it using boxing movements such as idling, locomotion, attacking, targeting, and more. A control interface is used in order to combine these control modules into a control sequence which is used to generate the movement utilizing the motion generator network. The results of this paper are, to no surprise, very good! The authors demonstrate their network's ability to expertly handle situations such as additive motion layering, where movements are necessary to be done under different conditions at different times, transition motion synthesis, where transitions between two different motions is smooth and natural, signature movements, and character interactions, all still in real time. Downsides to the method include conceptually unrealistic movements producing unrealistic motion, not perfect recreation of desired motions, and the fact that the user still has to be the one to create the motion to be peformed before it can be performed.

The final paper, titled "DeepPhase: Periodic Autoencoders for Learning Motion Phase Manifolds" and presented in SIGGRAPH 2022, is a divergence from the norm of the last few papers. The idea of phase functions still pops up, however, the architecture around it is changed. [24] The key difference for this architecture is that the effects of multiple periodic motions between local parts of the character on each other are considered, rather than considering them separately. A convolutional autoencoder is utilized in order to convert the movements into latent space vectors. Each channel of the latent space is enforced to be in the form of a periodic function in order to learn a phase variable for each channel. Phase manifolds are generated in order to greatly smooth motions within the same class by shifting the periodic autoencoder along motion curves. In order to control motion, a framework similar to the 2020 paper is used but with the inputs being the phase vectors of the phase manifold instead of velocities or contact local phases. The periodic autoencoder, as compared to the previous iterations of the phase function neural network, is able to express much more realistic looking movements of various limbs. Examples given in the paper include intricate dances, waving of arms while running, and a dogs tail wagging while walking, along with a representation of the phase manifolds for each of these actions. The method produces more, and more realistic, join movements than any of the previously discussed methods of character animation, and can still be applied to quadrupeds as well as interactions with world objects. Limitations include ambiguitiy about which motion to generate and not being good at dancing to arbitrary music, though, not many of us are good at that either, so I think it gets a pass!

## 3.4 Animation Papers Analysis

Like the papers discussing graphics research, these papers presented here do not even barely scratch the surface of machine learning research applied to graphics. Other topics that weren't covered but are absolutely important in the field include reinforcement learning applied to path planning [22], other forms of video synthesis [33], facial expression or mouth movement synthesis [15] [29], and much much more. Regarding the papers discussed here, however, we can see that machine learning has a ton to offer the field of computer animation.

For the navigation papers discussed, they all covered their topics in an in depth manner, clearly conveying to the reader what was required to know about the topic. Each discussed previous work in the field and what they contributed that was novel. It is clear, however, that there is still much research to be done in the field of applying machine learning to path planning tasks. None of these methods mentioned anything about reinforcement learning based solutions to the problem which, while not the type of learning they were implementing, would have been nice to have as a baseline comparison to see how the two styles contrasted. It was interesting to see that lack of use of convolutional nets in the crowd path planning section and perhaps discussion around their possible uses would have enriched those papers. Additionally, despite it being relatively new, I think in the future I would like to see exploration between the periodic networks discussed in the character animation section and the crowd simulation. Perhaps there's a connection there that could lead to improved results.

For the fluid, image, and cloth animation papers, despite being quite different topics, they fed into each other nicely and covered quite a few of the same topics. It was interesting to see how applicable convolutional networks and transformers were to the problem of fluid flow and approximating the Navier-stokes equation. When reading those papers, much was discussed that went slightly over my head due to lack of experience in the specific field of fluid approximation. Perhaps it would have benefitted the papers to explain a bit better the methods behind what they were doing in order to give the people reading it a bit of a better idea even if they didn't have too much experience. The image animation using eulerian integration paper, while not producing *amazing* results, showed something incredible by making a still image move. I would, however, had like to seen a bit more exploration with utilizing the method in 3 dimensions. The same criticism applies to the fluid dynamics papers. 3D fluid simulation is a huge topic and more exploration of that would have been really interesting to see. As for the cloth simulation paper, it was a bit bare-bones in describing the process. It would have been beneficial to the paper to describe a bit more in detail the previous state of the art, what the paper was iterating on, and go a bit more into detail about the method they were implementing. Additionally, a bit more discussion about the lack of dynamic movement and how that could be remedied would have been nice.

The last papers on the topic of character animation are perhaps my favorite set of technical papers ever. Not only do they chronologically and topically flow into each other masterfully, they describe the previous state of the art, reasons behind design, and the methods in extreme detail. Providing public code for the papers is just a cherry on top of the already delicious cake. I do wish that, as a whole, the papers explored the variety of characters that could be animated. Most of the examples shown and the code given have examples of human and wolf character models either boxing or playing basketball. While those are impressive as they are and do a good job of showing off the work, I think it would have been beneficial to show even more variety in the characters that can be represented, perhaps even seeing how they work in a realistic game scene. A thing that I would be interested in them exploring is the effect that a transformer might have on periodic autoencoder. Transformers have been shown to be very good at doing the work that CNNs do, and perhaps this case is no different. It might break the compactness or inference speed of the network, but it would be interesting to see nonetheless.

Computer animation is one of the most interesting fields of computer science to me, and I can't wait to see how machine learning continues to push it to the absolute best it can be.

## 4 Conclusion

In conclusion, this literature review covered super resolution and how convolutional neural networks, generative adversarial networks, and transformers are being applied to obtain incredibly accurate and real time upscaled videos. We covered global illumination in ray tracing and how it has evolved from slow but very accurate methods to faster methods carried by neural networks with comparable accuracy. We discussed why path planning is important in crowd simulation and how machine learning can be applied to it to, not only create realistic crowds, but also learn more about and improve current physical methods. We compared techniques to simulate fluid dynamics, compared the pros and cons of each, and explored the relationship between CNNs and transformers applied to the task. we briefly covered integration methods and how tasks that traditionally involve those can be augmented with neural networks to create relatively realistic looking results. Finally, we followed a chain of papers discussing a truly novel solution to the problem of character animation and explored how motion phases can be interpreted by neural networks to control characters in an incredibly realistic way.

The important thing to take away from this is that animation and graphics are always changing, as is every other field of computer science. Regression for global illumination led to the use of neural networks. Convolutional neural networks for fluid flow led to the use of transformers for the same task. Basic neural networks for character animation led into periodic function neural networks which led into convolutional autoencoders for the same purpose. As more machine learning techniques are discovered, more research is sure to come out connecting those new techniques to the state of the art and improving on it.

In the future, I hope to do my own work on things like those discussed here. I hope to be able to contribute something to the field that is meaningful, iterates on the state of the art, and, most importantly, is cool! I also can't wait to see how the field evolves over the course of the next few years. Even just what has been discovered over my college years is incredibly exciting, and the future is sure to be just as, if not more.

## References

[1] S. Derin Babacan, Rafael Molina, and Aggelos K. Katsaggelos. “Variational Bayesian Super Resolution”. In: *IEEE Transactions on Image Processing* 20.4 (2011), pp. 984–999. DOI: 10.1109/TIP.2010.2080278.

[2] Panayiotis Charalambous et al. “A Data-Driven Framework for Visual Crowd Analysis”. In: *Comput. Graph. Forum* 33.7 (Oct. 2014), pp. 41–50. ISSN: 0167-7055. DOI: 10.1111/cgf.12472. URL: https://doi.org/10.1111/cgf.12472.

[3] Terrance DeVries et al. *Unconstrained Scene Generation with Locally Conditioned Radiance Fields*. 2021. arXiv: 2104.00670 [cs.CV].

[4] M. Elad and A. Feuer. “Restoration of a single superresolution image from several blurred, noisy, and undersampled measured images”. In: *IEEE Transactions on Image Processing* 6.12 (1997), pp. 1646–1658. DOI: 10.1109/83.650118.

[5] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. *Texture Synthesis Using Convolutional Neural Networks*. 2015. arXiv: 1505.07376 [cs.CV].

[6] Dirk Helbing and Pé ter Molnár. “Social force model for pedestrian dynamics”. In: *Physical Review E* 51.5 (May 1995), pp. 4282–4286. DOI: 10.1103/physreve.51.4282. URL: https://doi.org/10.1103%2Fphysreve.51.4282.

[7] Daniel Holden, Taku Komura, and Jun Saito. “Phase-Functioned Neural Networks for Character Control”. In: *ACM Trans. Graph.* 36.4 (July 2017). ISSN: 0730-0301. DOI: 10.1145/3072959.3073663. URL: https://doi.org/10.1145/3072959.3073663.

[8] Daniel Holden, Jun Saito, and Taku Komura. “Neural Network Ambient Occlusion”. In: *SIGGRAPH ASIA 2016 Technical Briefs*. SA ’16. Macau: Association for Computing Machinery, 2016. ISBN: 9781450345415. DOI: 10.1145/3005358.3005387. URL: https://doi.org/10.1145/3005358.3005387.

[9] Aleksander Holynski et al. *Animating Pictures with Eulerian Motion Fields*. 2020. arXiv: 2011.15128 [cs.CV].

[10] Ko-Jen Hsiao et al. *Multi-criteria Anomaly Detection using Pareto Depth Analysis*. 2013. arXiv: 1110.3741 [cs.LG].

[11] Giulio Jiang and Bernhard Kainz. *Deep Radiance Caching: Convolutional Autoencoders Deeper in Ray Tracing*. 2020. arXiv: 1910.02480 [cs.GR].

[12] M.I. Jordan and R.A. Jacobs. “Hierarchical mixtures of experts and the EM algorithm”. In: *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*. Vol. 2. 1993, 1339–1344 vol.2. DOI: 10.1109/IJCNN.1993.716791.

[13] Hyoeun Kang et al. “A new fluid flow approximation method using a vision transformer and a U-shaped convolutional neural network”. In: *AIP Advances* 13.2 (Feb. 2023). 025233. ISSN: 2158-3226. DOI: 10.1063/5.0138515. eprint: https://pubs.aip.org/aip/adv/article-pdf/doi/10.1063/5.0138515/16759425/025233_1_online.pdf. URL: https://doi.org/10.1063/5.0138515.

[14] Armin Kappeler et al. “Video Super-Resolution With Convolutional Neural Networks”. In: *IEEE Transactions on Computational Imaging* 2.2 (2016), pp. 109–122. DOI: 10.1109/TCI.2016.2532323.

[15] Tero Karras et al. “Audio-Driven Facial Animation by Joint End-to-End Learning of Pose and Emotion”. In: *ACM Trans. Graph.* 36.4 (July 2017). ISSN: 0730-0301. DOI: 10.1145/3072959.3073658. URL: https://doi.org/10.1145/3072959.3073658.

[16] Dmitrii Kochkov et al. “Machine learning–accelerated computational fluid dynamics”. In: *Proceedings of the National Academy of Sciences* 118.21 (2021), e2101784118. DOI: 10.1073/pnas.2101784118. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.2101784118. URL: https://www.pnas.org/doi/abs/10.1073/pnas.2101784118.

[17] Eric P. Lafortune and Yves D. Willems. “Bi-directional path tracing”. In: *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics ’93)*. Alvor, Portugal, Dec. 1993, pp. 145–153.

[18] Eun Sil Lee and Moon Gi Kang. “Regularized adaptive high-resolution image reconstruction considering inaccurate subpixel registration”. In: *IEEE Transactions on Image Processing* 12.7 (2003), pp. 826–837. DOI: 10.1109/TIP.2003.811488.

[19] Christopher Lewin. “Swish: Neural Network Cloth Simulation on Madden NFL 21”. In: *ACM SIGGRAPH 2021 Talks*. SIGGRAPH ’21. Virtual Event, USA: Association for Computing Machinery, 2021. ISBN: 9781450383738. DOI: 10.1145/3450623.3464665. URL: https://doi.org/10.1145/3450623.3464665.

[20] Henry C Lin and Andrew Burnes. *NVIDIA DLSS 3: AI-Powered Performance Multiplier Boosts Frame Rates By Up To 4X*. Sept. 2022. URL: https://www.nvidia.com/en-us/geforce/news/dlss3-ai-powered-neural-graphics-innovations/.

[21] Ben Mildenhall et al. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. 2020. arXiv: 2003.08934 [cs.CV].

[22] Michael W. Otte. “A Survey of Machine Learning Approaches to Robotic Path-Planning”. In: 2009.

[23] Peiran Ren et al. “Global Illumination with Radiance Regression Functions”. In: *ACM Trans. Graph.* 32.4 (July 2013). ISSN: 0730-0301. DOI: 10.1145/2461912.2462009. URL: https://doi.org/10.1145/2461912.2462009.

[24] Sebastian Starke, Ian Mason, and Taku Komura. “DeepPhase: Periodic Autoencoders for Learning Motion Phase Manifolds”. In: *ACM Trans. Graph.* 41.4 (July 2022). ISSN: 0730-0301. DOI: 10.1145/3528223.3530178. URL: https://doi.org/10.1145/3528223.3530178.

[25] Sebastian Starke et al. “Local Motion Phases for Learning Multi-Contact Character Movements”. In: *ACM Trans. Graph.* 39.4 (Aug. 2020). ISSN: 0730-0301. DOI: 10.1145/3386569.3392450. URL: https://doi.org/10.1145/3386569.3392450.

[26] Sebastian Starke et al. “Neural Animation Layering for Synthesizing Martial Arts Movements”. In: *ACM Trans. Graph.* 40.4 (July 2021). ISSN: 0730-0301. DOI: 10.1145/3450626.3459881. URL: https://doi.org/10.1145/3450626.3459881.

[27] Sebastian Starke et al. “Neural State Machine for Character-Scene Interactions”. In: *ACM Trans. Graph.* 38.6 (Nov. 2019). ISSN: 0730-0301. DOI: 10.1145/3355089.3356505. URL: https://doi.org/10.1145/3355089.3356505.

[28] Ruiqi Tan et al. “Video Super-Resolution with Spatial-Temporal Transformer Encoder”. In: *2022 IEEE International Conference on Multimedia and Expo (ICME)*. 2022, pp. 1–6. DOI: 10.1109/ICME52920.2022.9859774.

[29] Sarah Taylor et al. “A Deep Learning Approach for Generalized Speech Animation”. In: *ACM Trans. Graph.* 36.4 (July 2017). ISSN: 0730-0301. DOI: 10.1145/3072959.3073699. URL: https://doi.org/10.1145/3072959.3073699.

[30] A. Tewari et al. “State of the Art on Neural Rendering”. In: *Computer Graphics Forum* 39.2 (2020), pp. 701–727. DOI: https://doi.org/10.1111/cgf.14022. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14022. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14022.

[31] Manu Mathew Thomas and Angus G. Forbes. *Deep Illumination: Approximating Dynamic Global Illumination with Generative Adversarial Network*. 2018. arXiv: 1710.09834 [cs.GR].

[32] Eric Veach and Leonidas J. Guibas. “Metropolis Light Transport”. In: *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’97. USA: ACM Press/Addison-Wesley Publishing Co., 1997, pp. 65–76. ISBN: 0897918967. DOI: 10.1145/258734.258775. URL: https://doi.org/10.1145/258734.258775.

[33] Ting-Chun Wang et al. *Video-to-Video Synthesis*. 2018. arXiv: 1808.06601 [cs.CV].

[34] Xijun Wang et al. “Spatially Adaptive Losses for Video Super-resolution with GANs”. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 1697–1701. DOI: 10.1109/ICASSP.2019.8682742.

[35] Alexander Watson. *Deep Learning Techniques for Super-Resolution in Video Games*. 2020. arXiv: 2012.09810 [cs.NE].

[36] Ling-Qi Yan et al. “A BSSRDF Model for Efficient Rendering of Fur with Global Illumination”. In: *ACM Trans. Graph.* 36.6 (Nov. 2017). ISSN: 0730-0301. DOI: 10.1145/3130800.3130802. URL: https://doi.org/10.1145/3130800.3130802.

[37] Ryo Yonetani et al. *Path Planning using Neural A* Search*. 2021. arXiv: 2009.07476 [cs.LG].

[38] Guozhen Zhang et al. “Physics-Infused Machine Learning for Crowd Simulation”. In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. KDD ’22. Washington DC, USA: Association for Computing Machinery, 2022, pp. 2439–2449. ISBN: 9781450393850. DOI: 10.1145/3534678.3539440. URL: https://doi.org/10.1145/3534678.3539440.

[39] He Zhang et al. “Mode-Adaptive Neural Networks for Quadruped Motion Control”. In: *ACM Trans. Graph.* 37.4 (July 2018). ISSN: 0730-0301. DOI: 10.1145/3197517.3201366. URL: https://doi.org/10.1145/3197517.3201366.