
EPiC Series in Computing

Volume 77, 2021, Pages 123–132

Proceedings of ISCA 30th International Conference on Software Engineering and Data Engineering


# Using Virtual Reality and Machine Learning Techniques to Visualize the Human Spine

Lin Hall[1], Ping Wang[1], Grayson Blankenship[1], Emmanuel Zenil Lopez[1], Chris Castro[1], Zhen Zhu[2], and Rui Wu[1]

[1] Department of Computer Science, East Carolina University, Greenville, North Carolina, U.S.A
halll05@students.ecu.edu; wangp19@students.ecu.edu; blankenshipg17@students.ecu.edu; zenillopeze15@students.ecu.edu; castroch16@students.ecu.edu; wur18@ecu.edu

[2] Department of Engineering, East Carolina University, Greenville, North Carolina, U.S.A
zhuz@ecu.edu

**Abstract**

Machine learning technique usage has exploded in recent years, as has the utilization of virtual reality techniques. One area that these tools can be utilized is the practice of medicine. In this research, we propose a framework to visualize the position and rotation of human spines based on machine learning predictions. This framework approach is significant due to the importance of medical visualizations and organ tracking, with uses ranging from education of medical students, to surgical uses. Subsequently, using machine learning techniques with virtual reality offers real-time medical visualizations which is significant for surgery. According to our experiment results, our proposed framework can accurately predict position and rotation data.

## 1 Introduction

Machine learning is important in pattern recognition, and emulates human intelligence by learning from the surrounding environment[7]. In this paper, we will predict the position of an HTC Vive Pro Tracker using machine learning based on the position of three other trackers. This research can be used in surgical medicine to visualize and locate important physiological markers along the spine during surgery.

In medicine, machine learning is useful for developing models on clinical datasets [3], as well as clinical medicine and diagnosis[14]. This usefulness extends to predictive analysis as well. Machine learning prediction has been utilized in medical research areas including: cancer [4, 13], stroke [12], and dementia[1]. Two of the main types of machine learning models are supervised (including data labels) and unsupervised (data labels are not included). These model types can be used to implement a time series analysis [24].

Machine learning visualization can be extended to virtual reality. One of the main uses of machine learning in virtual reality is data visualization[6]. For machine learning, data visualization is important, particularly in medicine, for knowledge extraction[19]. Also, virtual reality can utilize machine learning to visualize numerical simulations in three dimensions. As a result,

machine learning can facilitate the spatial understanding of complex data[6]. Machine learning and virtual reality user can work simultaneously to work on specific data oriented tasks. This technique is referred to as human in the loop, and is implemented to reduce the variability in machine learning model predictions [23]. Our research, however, will focus on utilizing machine learning techniques to visualize tracking machine learning model results.

Visualizing data in virtual reality is measured differently than visualizing ML success related to a machine learning model. Virtual reality allows the complete immersion into the data [6]. Our work is an example of visualizing machine learning model results for HTC Vive Pro Tracker locations in a virtual reality environment. The purpose of this research is to accurately predict tracker locations using Unity, machine learning, and the HTC Vive Pro virtual reality technology.

The use of virtual reality in medicine is a relatively new concept. In the past, VR has been used as an educational tool in medicine, developed through the use of expert domain analysis [18]. However, in terms of usage in a surgical scenario yielded mixed results [10]. Further, Hettig et al. based their visualizations on preoperative imaging of the patient. In contrast, a study in 2021 regarding the use of augmented and virtual reality in spinal surgery found that for pedicle screw placement along the spine had a deviation of less than 2° deviation with 97% accuracy [8]. Even still, this study was dependent on visualizations of a static spine. Shifts in the patient were detrimental to pedicle screw placement. However, a static visualization during surgery is not likely. In the case of pedicle screws, each screw placement can cause shifts along the spine [8]. Our research, however, uses machine learning to account for this spinal shift.

Spinal visualizations using technology are improving [21]. However, most current visualization processes of the human spine involve mathematical formulas and radiological imaging [11]. Further, machine learning and other innovative techniques have imaged the human spine with a success rate rivaling that of manual imaging [9].

Utilizing the three dimensional immersion of a VR environment, we will compare machine learning model accuracy measures ($R^2$, $rmse$, etc.) with a visual test related to tracker location. In other words, we will work to compare the model scores and the visual representation of the tracker location predictions from the machine learning model. Further, the usage of the three HTC Vive Pro Trackers will serve as a rigid-body emulation of the human spine. Using a time series regression model, we will use tracker location data to accurately predict the location of the "fourth" tracker (which is predicted as part of the time series regression), which should match up with our "first" tracker. Each tracker represents a position along the human spine. Further, the position of each tracker is visualized using the Unity Virtual Reality Environment.

In this study, we created a Unity virtual reality app to visualize the predicted tracker location. The trackers were placed on a HTC Vive Pro racket. To train the machine learning model, movement data was collected using the racket for the X, Y, Z, pitch, roll, and yaw direction information. Next, the training data was preprocessed, and fed into the time series regression model. Once the data was trained, ML accuracy measures were used to test the quality of the data collection. Then, a client-server (in the Flask programming library) was built to run the machine learning models, and output predicted tracker locations. Subsequently, this output data was fed into the Unity application. Further, to locate any data outliers, an extreme event split was implemented. Subsequently, the Unity application scene would display the three trackers, and the predicted fourth tracker. Being able to accurately predict the fourth tracker would be integral to the safety of the patient during spinal surgery.

The main contribution of this paper is to provide a framework approach to using virtual reality and machine learning techniques to track and visualize the position of HTC Vive Pro

Trackers. This framework includes predicting time series data, which emulates various positions along the human spine, to which we use a gradient boosting regressor model to predict a specific location. Further, to improve the efficiency of the predicted spinal position we mitigate data outliers by utilizing the extreme event split technique. Finally, this research will utilize virtual reality to visualize the human spine by utilizing the machine learning techniques mentioned above. The framework approach provided in this research can be expanded to future medical visualizations.

In section 2 of this paper, previous related work to time series regression analysis and comparable research to our study is reviewed. In section 3, the application architecture is presented. This section includes a process workflow from data collection to virtual reality visualization of the predicted tracker location. Also, the methodology behind the HTC Vive Pro will be will be provided. Section 4 presents an in-depth assessment of the machine learning tracker models. Data collection and the extreme event split will be shown as well. Next, section 5 will provide results for our tracker testing. Conclusions regarding our research will be provided in section 6. Finally, section 7 will present acknowledgements for this paper.

## 2 Related Work

Rapid advances in technology and medical device development in the 21st century are bringing about a new era of medicine, contributing to healthier and more productive lives. As technology and patient complexity continues to increase, demands for novel approaches to ensure competency have arisen [16]. Virtual reality simulator becomes a powerful tool for surgical trainees to repeatedly practice without potential harm to patients and animals.

The efforts of Bissonnette to distinguish surgical training levels using virtual reality simulator and machine learning methods indicate the power of virtual reality and machine learning on surgical training and evaluation. The authors divided spine surgeons, spine fellows, orthopaedic and neurosurgery residents, and medical students from 4 Canadian universities into 2 groups– senior and junior, according to their training level. 22 participants are senior and 19 of them are junior. All the participants were asked to perform a spinal surgery, in a virtual reality environment. The virtual hemilaminectomy required participants to remove the L3 lamina with a simulated burr in their dominant hand while controlling bleeding with a simulated suction instrument in their nondominant hand [2]. Participants were required to remove the L3 lamina in 5 minutes, without damaging surrounding tissues. Their position, angle, force application of the simulated burr and suction instruments, and removed tissue volumes during the procedure were recorded at 20 ms intervals. These data is collected as metrics for training machine learning algorithms. Five classification algorithms were applied: support vector classifier, K-Nearest Neighbors classifier, Linear Discriminant analysis, Naive Bayes classifier, and decision tree classifier. The accuracy was assessed through leave-one-out cross-validation and support vector classifier achieved accuracy score of 97.6%.

Regression algorithms can also be combined with virtual reality technique in surgical field. Dubin implemented machine learning algorithms to develop regression models and predict Global Evaluative Assessment of Robotic Skills(GEARS) score using virtual reality simulator[5]. GEARS is a validated surgical proficiency testing tool, which is widely used in training programs. 74 participants were required to perform a basic VR exercise(Ring and Rail1) and a complex VR exercise(Suture Sponge1) on two simulators–dV-Trainer(dVT) and da Vinci Skills Simulator(dVSS). The simulator give scores of each exercise for each participant. And the recorded video was sent to human evaluators for review using the GEARS tool. Linear regression models were generated for each exercise on each simulator to predict GEARS score based

on simulator score.

Although combining virtual reality technique with machine learning algorithms, both of the two research works used simple machine learning algorithms and focus on the performance of the simulators while we applied complex machine learning algorithms for predictive modeling and visualized the results with Unity—a virtual reality visualization tool. Bissonnette et al built classification model using Support Vector Classifier to distinguish senior level and junior level of surgeons. Dubin built simple linear regression to predict the GEAR score of medical students. Instead of simple machine learning models, we built several machine learning models to predict six dependent variables–three positional variables and three rotational variables and compare the performance of the predicting models to determine the most proper model for each of the six dependent variables. Besides, the related research work built models on data of small size. Bissonnette has forty one participants for model building and Dubin built linear regression model on 74 participants while we collected more than five thousand observations with HTC Vivo Pro Racket to predict position data consisting of six components.

We applied extreme event split algorithm to split the time series dataset into normal dataset and extreme dataset. Extreme value is defined as an observation with value at the boundaries of the domain. Anomaly detection in time series has attracted considerable attention due to its importance in many real-world application including intrusion detection, energy management and finance [17]. Most anomaly detection methods require manually set thresholds or assumptions on the distribution of data. Isolated forest algorithm is one of the commonly used extreme value detecting algorithms. The term isolation in this case means "separating an instance from rest of instances". The two processing stages of isolation forest include training stage and testing stage. The training stage builds isolation trees using subsamples of training set. The testing stage passes instances in testing set to obtain anomaly score for each instance. In the training stage, isolation trees are constructed by recursively partitioning a subsample $X'$ until all instances are isolated. Each isolation tree is constructed using a sub-sample $X'$ randomly selected without replacement from X[22]. The normal points tend to be isolated at the deeper end of the tree, whereas anomalies are closer to the tree root, due to their singularity nature. The shorter the average path length, the higher the chances to be anomalies[22]. In testing stage, outliers are identified and labeled based on anomaly score of each instance. 95% quantile method is used to determine the threshold of extreme value and the instances with anomaly score greater than the threshold are classified as outliers. The extreme value machine (EVM) introduced in 2018, has become an important tool in multivariate statistics and machine learning in the past few years. Generalized Pareto distribution(GPD) classifier is an alternative approach of EVM. It requires the generalized Pareto distribution assumption from extreme value theory. The idea of the EVM is to approximate the distribution of the margin distance of each point in each class using extreme value theory. A new point is then classified as normal if it is inside the margin of some point in the training set with high probability[20].We applied the DSPOT algorithm of splitting the data into normal events and extreme events. Then we built machine learning models on normal dataset and extreme dataset separately and compare the results with that of model built on dataset without extreme event splitting.

# 3 Application Architecture

To visualize the position of the HTC Vive Pro trackers, three components are needed: the hardware (HTC Vive Pro trackers and racket), the backend software (Flask server, machine learning techniques), and the frontend software (SteamVR and Unity). Figure 1 illustrates the visualization process for predicting the tracker positions. In this section, we illustrate how each

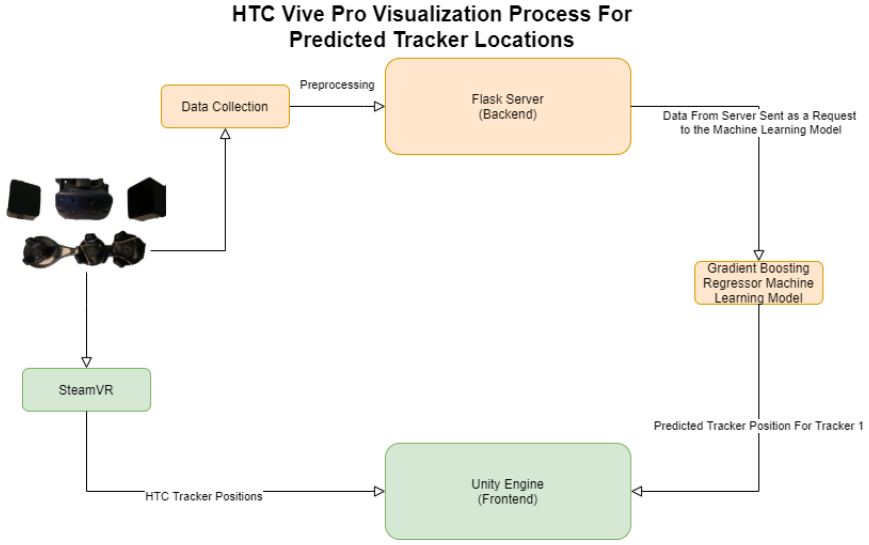


Figure 1: A flowchart illustrating the process in how data collection occurs. First, the data is collected from the HTC Vive Pro Setup. Next, the data is preprocessed and sent to the Flask server for the real-time tracker location prediction. Each HTC Vive Pro Tracker is attached to a HTC Vive Pro Racket, which is considered a rigid-body. Finally, using SteamVR and Unity, the tracker locations are visualized.

of these components are connected.

The data, once collected, is presented to a machine learning model being run from a Flask server. This server outputs the position of the predicted tracker. The output from the server is dependent on the Gradient Boosting Regressor machine learning algorithm. This predicted tracker position should match the prediction of tracker 1. Once the machine learning model predicts the output of the tracker position, the model is ready to be visualized using the Unity virtual reality engine, in conjunction with the HTC Vive Pro VR System.

Once a model prediction is generated for the predicted HTC Vive Pro Tracker, the position can be visualized in Unity. Using the HTC Vive Pro, Unity is able to show how close the predicted tracker is compared to tracker 1. For example, if the model prediction is completely correct, no difference should be seen between the prediction and tracker 1. Further, the predicted tracker should change positions along with any changes to the other trackers on the HTC Vive Pro Racket. Figure 1 illustrates a flow chart for this process.

# 4 Tracking Models

Machine learning involves a set of algorithms that learn from an environment. Further, this environment typically comes in the form of data. Therefore, machine learning is heavily dependent on quality data. The data used in machine learning aids the algorithms (models) in recognizing patterns in the data. As a result, machine learning models can be used to predict future trends in the collected data.

With that said, data collection is integral to a successful machine learning algorithm, such as gradient boosting regressor shown in Figure 1. For this research, linear tracker position is

predicted using HTC Vive Pro Trackers to simulate human spines. These trackers have a 270 degree field of view, which allow for data collection in virtually every direction. To emulate a linear tracker position, three trackers are used. These trackers are manually attached to a HTC Vive Pro Wireless Racket. Each HTC Vive Pro Tracker has a unique name. A unique identifier for each tracker is integral for the overall prediction. To summarize, there are three physical HTC Vive Pro Trackers for this research. Therefore, the fourth predicted tracker should match up with one of the physical trackers. In this case, the predicted tracker will match the position of the first tracker (Tracker 1).

As mentioned earlier, the three trackers are placed on an HTC Vive Pro Wireless Racket. The significance of the racket is that each tracker is placed in a straight line in three equidistant positions along the device. Therefore, for the purpose of this research, the racket emulates a human spine. Subsequently, the trackers are individual locations along the spine.

To ensure the data is collected from each tracker properly, these sensors must be synced to the Steam virtual reality software, and the HTC Vive Pro Lighthouses (Base Stations). These wireless lighthouses are responsible for determining the position of the trackers and the headset during the VR experience. Typically, these lighthouses are placed approximately six feet apart. In parallel, the HTC Vive Pro Headset must be active at all times for data collection to occur. Failure to do so will result in the data being inconsistent, or even uncollected.

Tracker Data is collected regarding the coordinate positions positions related to the virtual reality environment created using the Unity Game Engine. In the Unity environment, the X and Z axes make up the cartesian X and Y directions. Further, the Y axis in Unity is vertical, making up the cartesian Z direction. These directions are considered positions for the purpose of this research. Further, rotational data is also collected in the pitch, roll, and yaw directions, which are considered orientations for this research.

As stated earlier, machine learning is heavily dependent on data collection. This process is necessary to properly learn from the virtual reality environment, and to accurately predict tracker location. The data is collected into six individual files, one for each position and orientation. The collected data includes random movements for each position/orientation type. The random movements ensure that as many possible positions of the trackers is accounted for. The overall goal of the data collection, as well as the random movements, is to ensure the algorithm can adequately learn the position of the trackers when trained.

To collect the data, a Steam virtual reality scene is initialized in order to create the virtual reality environment. Using this environment, the tracker positions are output as raw data. To obtain the tracker positions, Unity provides an API to calculate the location of the sensors. The position is dependent on the location of the tracker relative to the lighthouse base stations. Before collecting any data, the point of origin must be determined. This is important because the tracker locations are determined based on their distance from this origin point. Further, when activating the Steam VR environment, it is imperative to have the HTC Vive Pro Racket at the point of origin in order to ensure accurate data collection.

While the Steam VR environment scene is active, the output of data occurs continuously until the scene is stopped. While the Steam VR scene is active, the HTC Vive Pro Racket (with each tracker attached) is shifted in the desired direction for different positions. For example, if data collection along the Unity X axis is collected, then the racket must be moved in the X direction. These movements occur at various positions. Each position is repeated for approximately 15-20 seconds. Subsequently, a dataset is created for each of the six linear or rotation directions in which data is collected.

The collected data is then preprocessed to remove incomplete samples and null values. Subsequently, the data is transformed into a comma-separated value (csv) format. As mentioned

before, there is a dataset for each direction in whic data is collected. Consequently, there will be six different datasets. Further, for the data collected on the linear axes, the csv files are cleaned to remove unnecessary directional information. For instance, if data is collected for the X position, the Y and Z positional information is removed from this particular dataset. As a result, the data is less noisy, and the focus of the data can be on one direction at a time. Further, this organization of data results in streamlined troubleshooting, as well as the ease of collecting new data if necessary.

Once the data is preprocessed and cleaned, it is tested for quality. Here, simple linear regression is used from the scikit-learn machine learning library [15]. To ensure the accuracy of the data collection for each directional file, the $R^2$ value is calculated. If the dataset was incomplete, or held any null or unaccepted data types, a modeling error is returned. Further, if the $R^2$ value was low, the data was required to be recollected.

### 4.1 Extreme Event Split

We applied Drift Streaming Peaks-Over-Threshold(DSPOT) to detect extreme events of the time series data and split the dateset into normal dataset and extreme dataset. As stated earlier, isolation forest extreme value detector and GPD classifier rely on either manually set thresholds or assumptions on the distribution of data. By using the DSPOT approach, We do not assume the distribution of the value but rely on extreme value theory to estimate accurately low probability areas and then discriminate outliers[17].

### 4.2 Real-time Prediction

Figure 1 illustrates the visualization process for the predicted tracker locations. The hardware seen in this figure illustrates the virtual reality hardware setup. Further, on the HTC Vive Pro racket, the predicted tracker 1 is in the middle, and is adjacent on either side by tracker 2 (right) and tracker 3 (left). Next, the preprocessed data is sent to a Flask Server. On this server, a gradient boosting regressor machine learning model predicts the location of tracker 1. This predicted location is distributed in real-time from the Flask Server. In other words, the predicted location coordinates are calculated continuously as long as the server is active. The information is then sent to the Unity application in the form of coordinates. These coordinates are processed through the Unity API and are visualized. Figure 2 shows the visualization of the trackers, including the predicted tracker.

## 5 Results

### 5.1 Gradient Boosting Regressor

Initial results of the gradient boosting regressor model show a high accuracy with low root mean square error for each position (X, Y, Z) and rotation (pitch, roll, yaw). For this model, the accuracy is high due to the linearity, as well as continuity of the data. Table 1 shows the accuracy of the predicted tracker locations for each position and rotation in relation to the actual tracker coordinates. As a whole, the GBR performed well. However, the pitch rotation RMSE was much higher for the real time prediction. Conversely, the extreme event split performed particularly well for this rotation, indicating that data contained several outliers. This is important since the data collection is performed by manual simultaneous movements of the trackers. Therefore, it is possible not all of the movements are consistent.

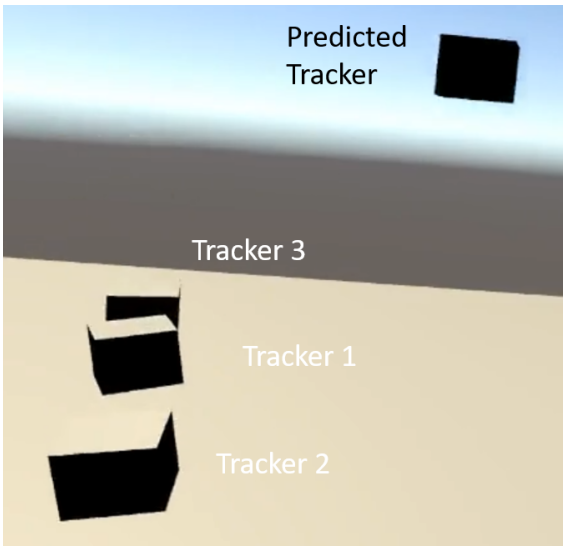


Figure 2: An illustration of the virtual reality visualization of the HTC Vive Pro Wireless Racket setup with the attached trackers. The three trackers represent positions along the human spine. The middle object, tracker 1, is the tracker to be predicted. Trackers 2 and 3 represent other locations along the human spine. These trackers are used to help predict the location of tracker 1. When predicted correctly, the visualized tracker should overlay tracker 1.

| Method | Metric | Position | | | | | |
|---|---|---|---|---|---|---|---|
| | | X | Y | Z | Pitch | Roll | Yaw |
| GBR (Base Data) | RMSE | 0.0003 | 0.0012 | 0.0014 | 0.0719 | 0.2587 | 0.2587 |
| GBR (Real Time) | RMSE | 0.1298 | 0.1358 | 0.0481 | 0.7249 | 0.0186 | 0.648 |
| GBR (Extreme Event Split - Base) | RMSE | 0.1021 | 0.0017 | 0.0135 | 0.0058 | 0.2905 | 0.2941 |
| GBR (Extreme Event Split - Real Time) | RMSE | 0.1048 | 0.0662 | 0.0202 | 0.0929 | 0.0229 | 0.1169 |

Table 1: This table displays the model results of the base gradient boosting regressor model, real-time model prediction, and the extreme-event split on the input data. Root Mean Square Error was calculated for each type of model run. The GBR model performed well for each position and rotation, with the exception of pitch, where the extreme event split had the best results.

## 5.2 Real-time Prediction

Table 1 displays the model results of the gradient boosting regressor (GBR) model run using the base data, the GBR results for the real-time prediction that is visualized using the Unity VR platform, and finally, the extreme event split (using GBR) of the base input data and extreme event split. The extreme event split on the real time data had lower RMSE than the base data for all of the values except for the roll data. Note the particularly large decrease in RMSE for the pitch rotation data from base data to extreme event split. Overall, we believe this is likely due to the increased number of extreme events during the real time data collection. In other words, the extreme event split performs well at dissecting the outliers in position and rotation of the trackers. While these values are adequate for the real-time prediction, there is still quite a bit of deviance between the predicted tracker and tracker 1 in the visualization seen in figure

2. An example of the real-time visualization can be seen **here**. This visualization illustrates the relationship of the trackers and the predicted coordinates received from the Flask server. The reason for the deviance between the predicted tracker and tracker 1 is an issue with the sampling from tracker 1. The data collection frequency using the Unity API is not consistent. The frequency issue increased the difficulty to train an accurate gradient boosting regressor model on the Flask server. In the future, we plan to solve the problem with a resampling preprocessing step. Resampling the data ensures that the model is being trained at an equal interval in relation to the data collection from the Unity API.

# 6 Conclusion

In Table 1, we have shown the results of the predicted tracker location in relation to tracker 1 for this research. As a result, the gradient boosing regressor model has proven useful for the machine learning application of this research, as well as the extreme-event split. With that said, some improvements on the virtual reality visualization are needed. Sampling intervals are detrimental to the visualization in that the model is not able to predict the correct location due to the difference in the positions related to the timestamps of the collected data. Therefore, the model is able to predict the past data accurately as seen in Table 1, while the real-time prediction is inaccurate for the virtual reality visualization.

Future research goals include data resampling to provide accurate sampling timestamps for the real-time prediction. Also, prediction for a non-rigid body is significant for an actual human body.

# 7 Acknowledgements

This work is supported in part by the National Science Foundation IUSE/PFE:RED award #1730568.

# References

[1] Gopi Battineni, Nalini Chintalapudi, and Francesco Amenta. Machine learning in medicine: Performance calculation of dementia prediction by support vector machines (svm). *Informatics in Medicine Unlocked*, 16:100200, 2019.

[2] Vincent Bissonnette, Nykan Mirchi, Nicole Ledwos, Ghusn Alsidieri, Alexander Winkler-Schwartz, and Rolando F Del Maestro. Artificial intelligence distinguishes surgical training levels in a virtual reality spinal task. *The Journal of bone and joint surgery. American volume*, 101(23):e127, 2019.

[3] Deo Rahul C. Machine learning in medicine. *Circulation*, 132(20):1920–1930, Nov 2015.

[4] Joseph A. Cruz and David S. Wishart. Applications of machine learning in cancer prediction and prognosis. *Cancer Informatics*, 2:117693510600200030, Jan 2006.

[5] Ariel Kate Dubin, Danielle Julian, Alyssa Tanaka, Patricia Mattingly, and Roger Smith. A model for predicting the gears score from virtual reality surgical simulator metrics. *Surgical endoscopy*, 32(8):3576–3581, 2018.

[6] Mohamed El Beheiry, Sébastien Doutreligne, Clément Caporal, Cécilia Ostertag, Maxime Dahan, and Jean-Baptiste Masson. Virtual reality: Beyond visualization. *Journal of Molecular Biology*, 431(7):1315–1321, 2019.

[7] Issam El Naqa and Martin J. Murphy. *What Is Machine Learning?*, page 3–11. Springer International Publishing, 2015.

[8] Mitchell S. Fourman, Hamid Ghaednia, Amanda Lans, Sophie Lloyd, Allison Sweeney, Kelsey Detels, Hidde Dijkstra, Jacobien H.F. Oosterhoff, Duncan C. Ramsey, Synho Do, and Joseph H. Schwab. Applications of augmented and virtual reality in spine surgery and education: A review. *Seminars in Spine Surgery*, page 100875, 2021.

[9] Fabio Galbusera, Gloria Casaroli, and Tito Bassani. Artificial intelligence and machine learning in spine research. *JOR SPINE*, 2(1):e1044, 2019.

[10] Julian Hettig, Sandy Engelhardt, Christian Hansen, and Gabriel Mistelbauer. Ar in vr: assessing surgical augmented reality visualizations in a steerable virtual reality environment. *International Journal of Computer Assisted Radiology and Surgery*, 13(11):1717–1725, Nov 2018.

[11] L. Humbert, J.A. De Guise, B. Aubert, B. Godbout, S. Parent, D. Mitton, and W. Skalli. 3d reconstruction of the spine from biplanar x-rays using longitudinal and transversal inferences. *Journal of Biomechanics*, 40:S160, 2007. Program and Abstracts of the XXI Congress, International Society of Biomechanics.

[12] C. Hung, W. Chen, P. Lai, C. Lin, and C. Lee. Comparing deep neural network and other machine learning algorithms for stroke prediction in a large-scale population-based electronic medical claims database. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, page 3110–3113, Jul 2017.

[13] Yixuan Li and Zixuan Chen. Performance evaluation of machine learning methods for breast cancer prediction. *Applied and Computational Mathematics*, 7(4):212, 2018.

[14] Ziad Obermeyer and Ezekiel J. Emanuel. Predicting the future — big data, machine learning, and clinical medicine. *The New England journal of medicine*, 375(13):1216–1219, Sep 2016.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[16] Michael P Rogers, Anthony J DeSantis, Haroon Janjua, Tara M Barry, and Paul C Kuo. The future surgical training paradigm: Virtual reality and machine learning in surgical education. *Surgery*, 2020.

[17] Alban Siffer, Pierre-Alain Fouque, Alexandre Termier, and Christine Largouet. Anomaly detection in streams with extreme value theory. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1067–1075, 2017.

[18] Samuel B. Tomlinson, Benjamin K. Hendricks, and Aaron Cohen-Gadol. Immersive three-dimensional modeling and virtual reality for enhanced visualization of operative neurosurgical anatomy. *World Neurosurgery*, 131:313–320, Nov 2019.

[19] Alfredo Vellido. The importance of interpretability and visualization in machine learning for applications in medicine and health care. *Neural computing and applications*, pages 1–15, 2019.

[20] Edoardo Vignotto and Sebastian Engelke. Extreme value theory for anomaly detection–the gpd classifier. *Extremes*, 23(4):501–520, 2020.

[21] Gheorghe-Daniel Voinea, Silviu Butnariu, and Gheorghe Mogan. Measurement and geometric modelling of human spine posture for medical rehabilitation purposes using a wearable monitoring system based on inertial sensors. *Sensors*, 17(1), 2017.

[22] Luqing Wang, Qinglin Zhao, Si Gao, Wei Zhang, and Li Feng. A new extreme detection method for remote compound extremes in southeast china. *Frontiers in Earth Science*, 9:243, 2021.

[23] Doris Xin, Litian Ma, Jialin Liu, Stephen Macke, Shuchen Song, and Aditya Parameswaran. Accelerating human-in-the-loop machine learning: Challenges and opportunities. In *Proceedings of the Second Workshop on Data Management for End-To-End Machine Learning*, DEEM'18, New York, NY, USA, 2018. Association for Computing Machinery.

[24] Haiqin Yang, Kaizhu Huang, Irwin King, and Michael R. Lyu. Localized support vector regression for time series prediction. *Neurocomputing*, 72(10):2659–2669, 2009. Lattice Computing and Natural Computing (JCIS 2007) / Neural Networks in Intelligent Systems Designn (ISDA 2007).