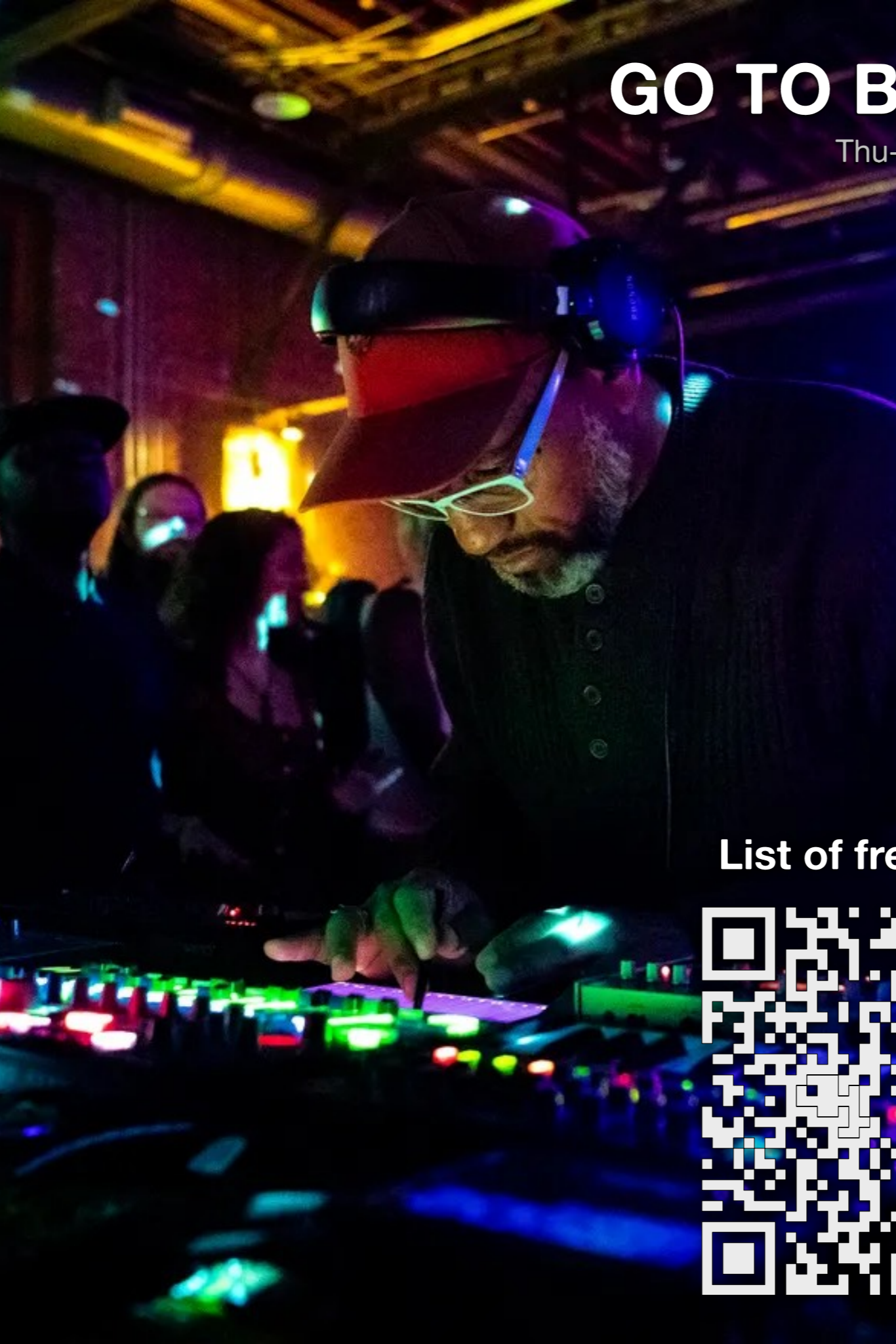


HOW TO POSTER SESSION

PROF L LEE

GO TO BIG EARS

Thu-Sun



List of free shows



Disclaimers

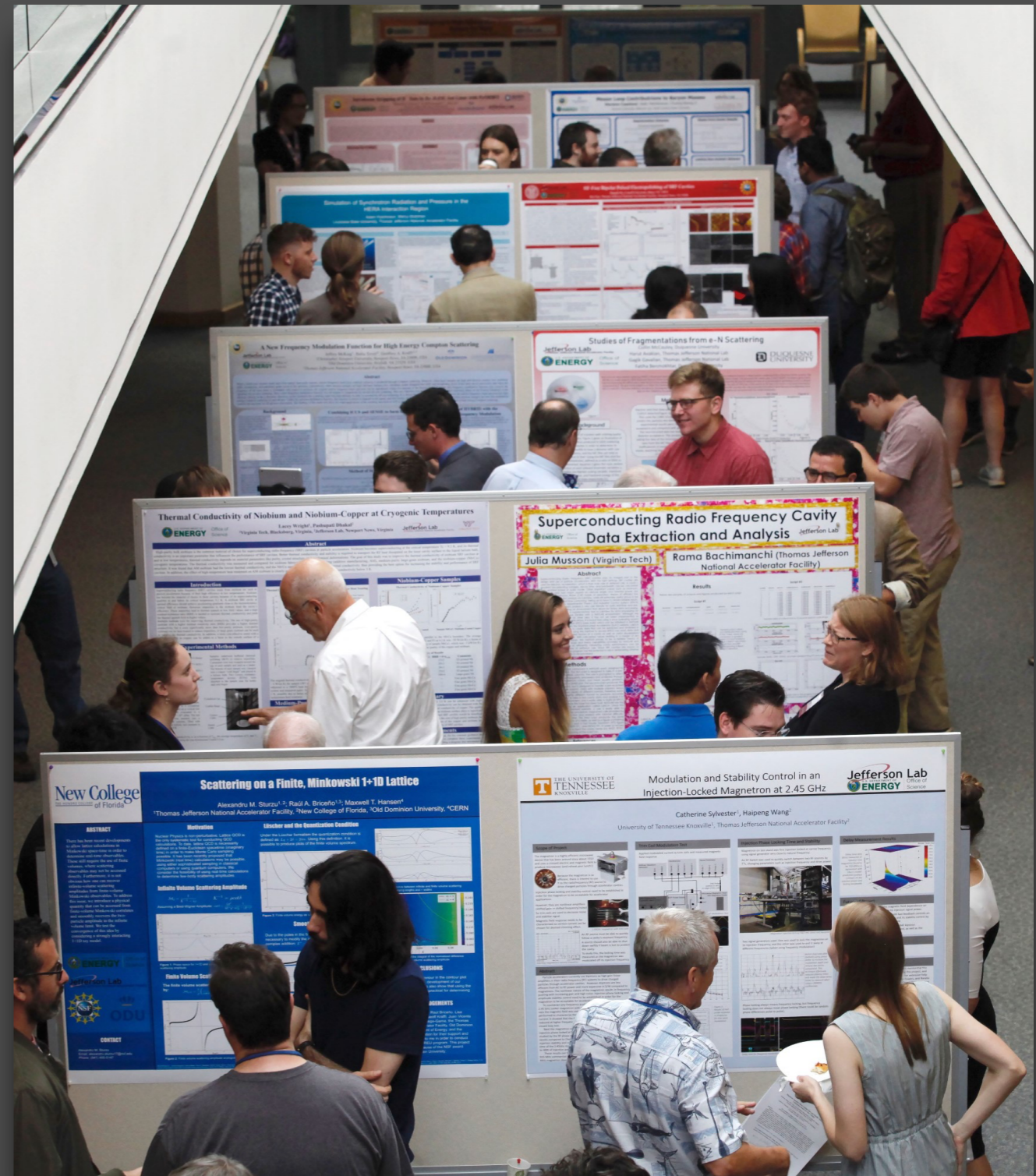
- Only discussing **posters presentations** today
 - (Many of the ideas do translate to other kinds of presentations)
- I've been a judge at many poster sessions, designed poster sessions, given posters, advised many students on posters, etc
- Will be a set of **personal opinions** on how to effectively present a poster
 - Can't guarantee your advisor will agree with me (unless they're me), but I find these tips and framings to useful
- All of these "rules" can be broken if it serves your communication mission

Presenting Your Work

- **Convey information**
 - Effective communication is everything
- **Don't bore your audience**
 - Know your audience! If you're presenting at a subfield conference, that's really different from EUReCA
 - Present things in an **engaging, relatable** way

What is a poster session?

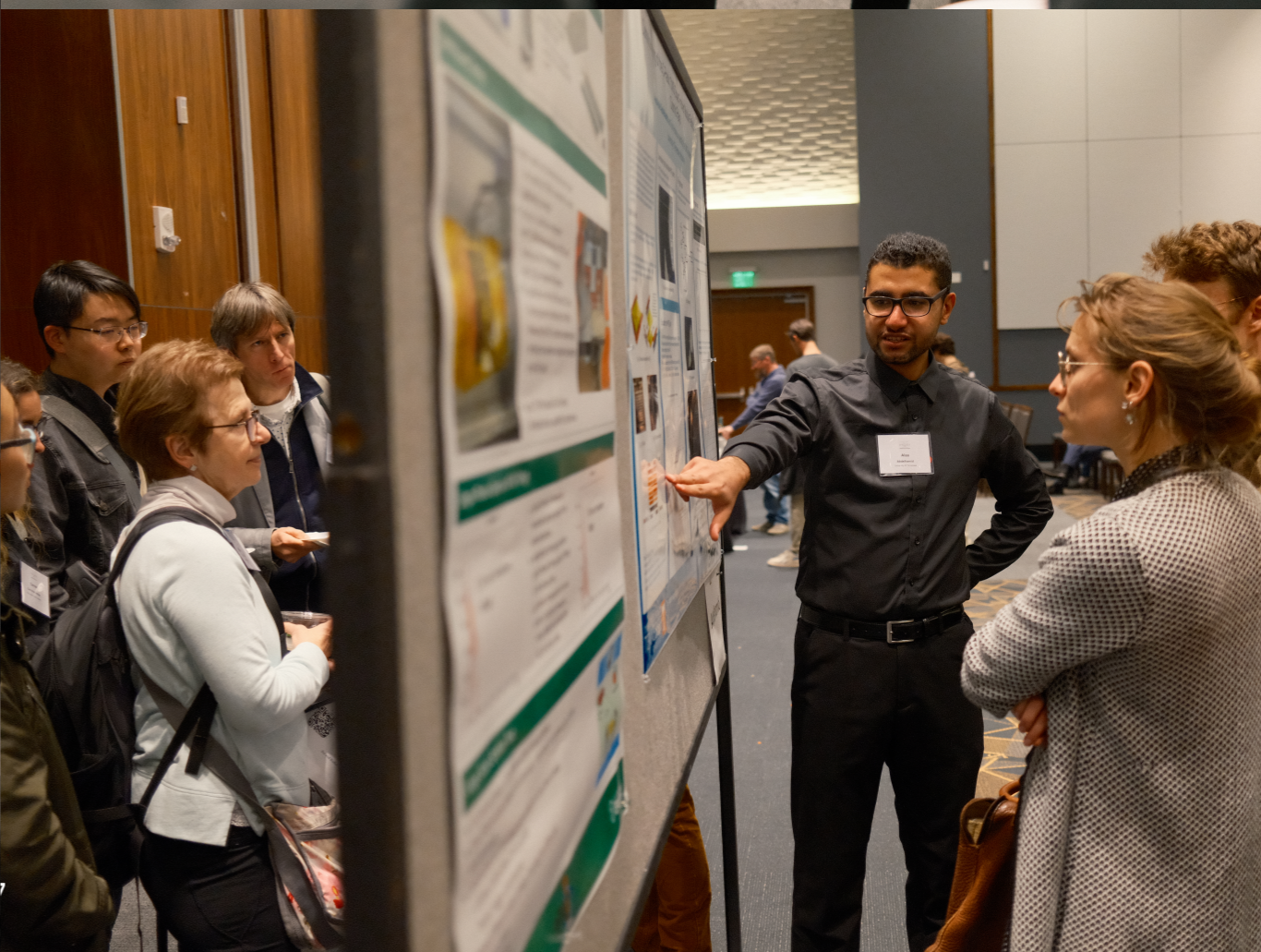
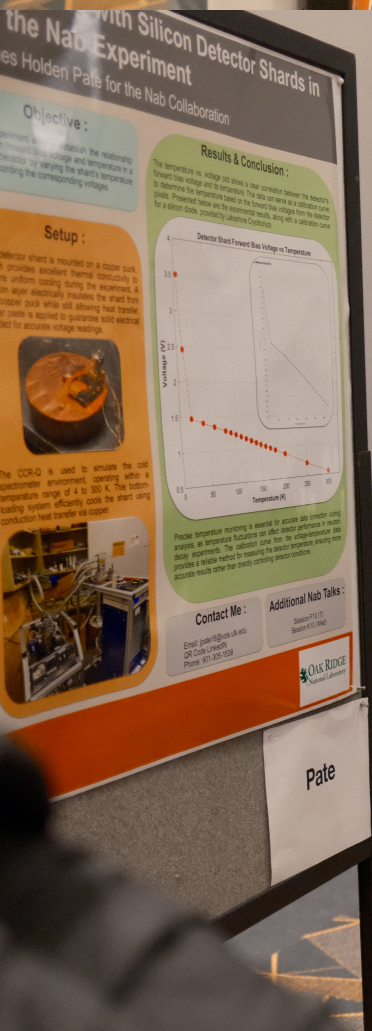
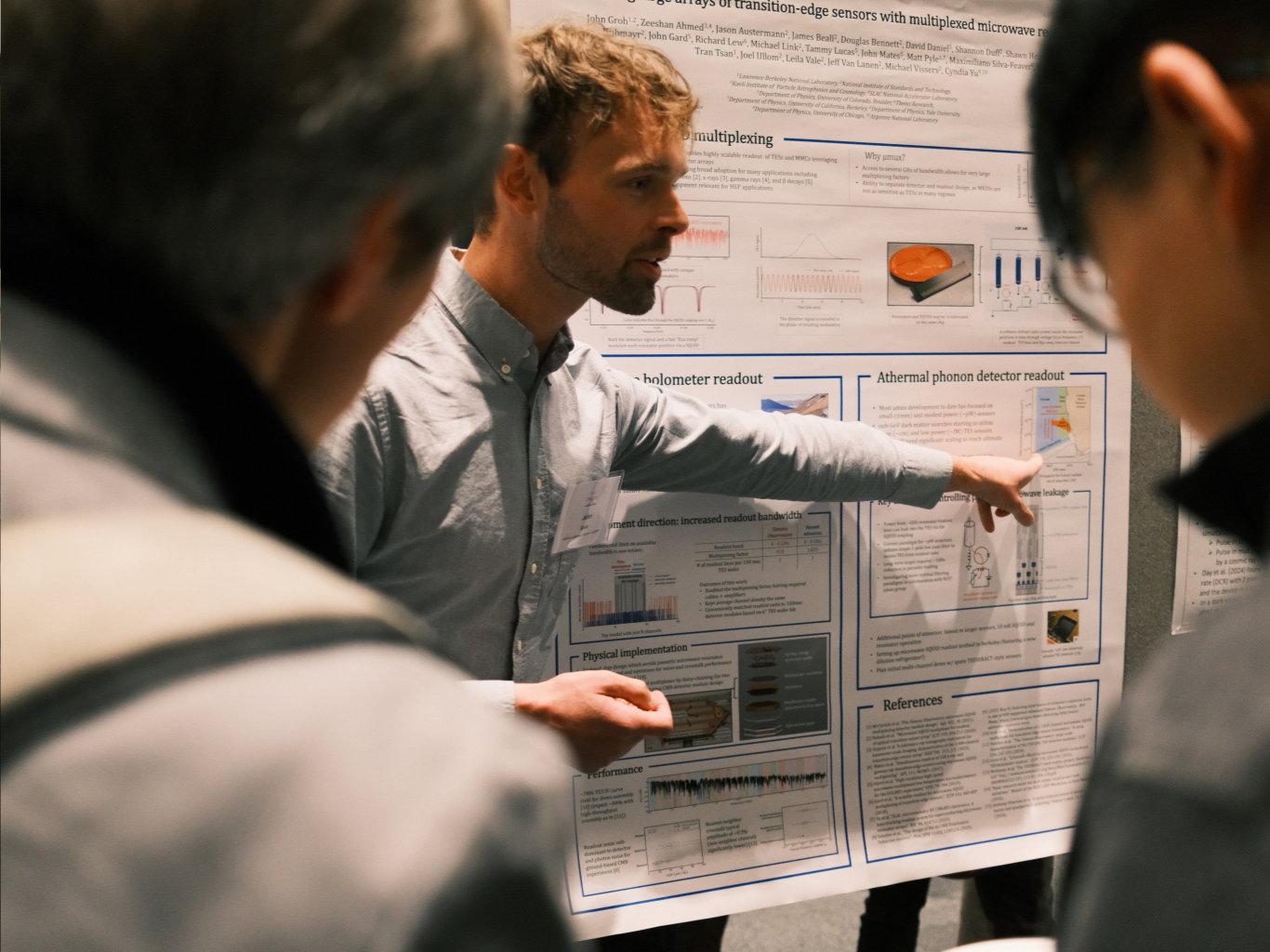
- Presenters print out a poster on their research topic
- They stand in front of the poster
- People walk around, look at the posters, talk to the presenters who give a **quick spiel**
- In physics, these are often aimed at students. In other fields and some physics subfields, they are the main form of presentation.



What is a poster session?

- Often during a conference, or can be a standalone event (e.g. EURēCA)
- There will be **judges** walking around, talking to each presenter
 - Taking notes on the presentation & poster for **awards/prizes/\$\$\$**
- At a conference, these are usually **boozy** events
 - Gets people talking and interacting
 - If legal, it's ok for you to have, but keep it to 1. You're there as a professional physicist presenting your work.

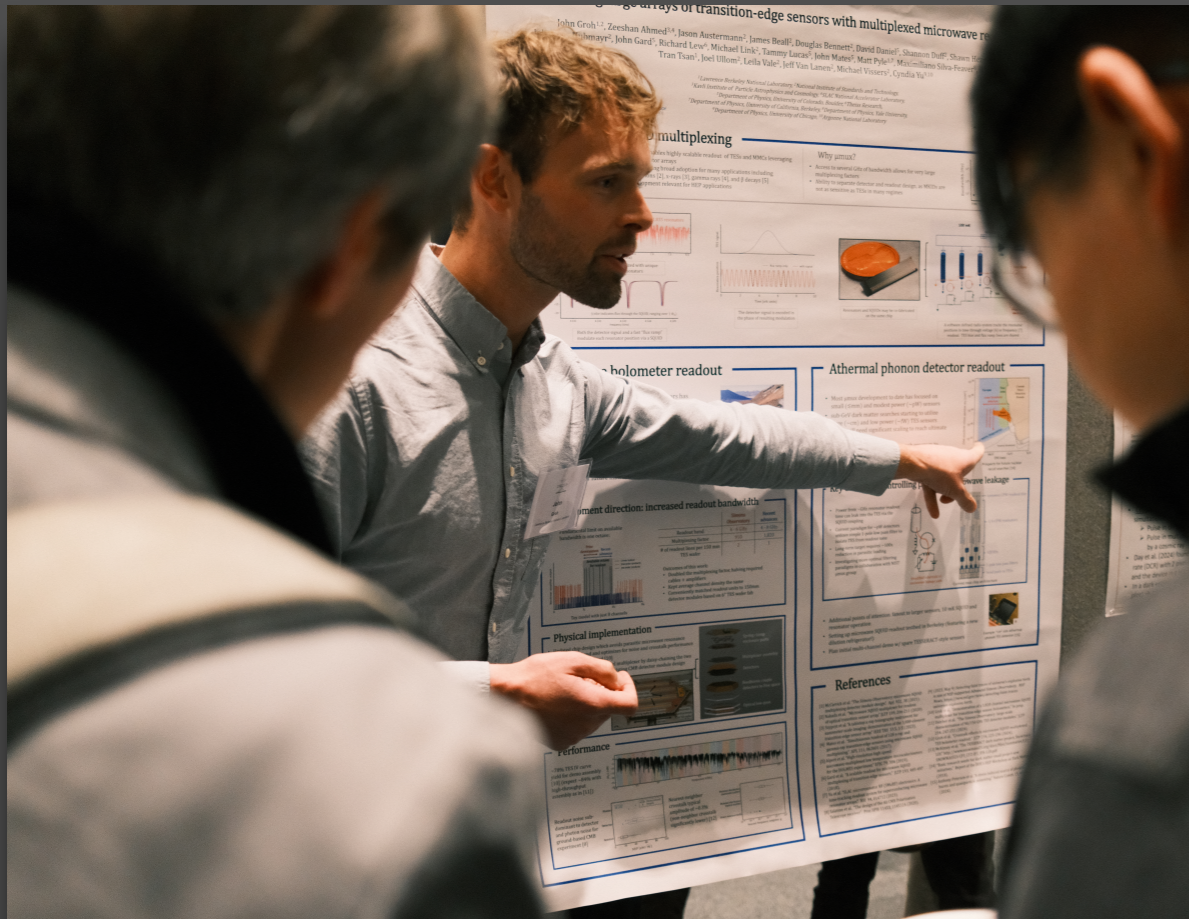




How to present your work...

- Two main parts of a poster presentation
 1. You **present** your work. All about your delivery followup conversations.
 2. **Visual aid** (your actual poster)

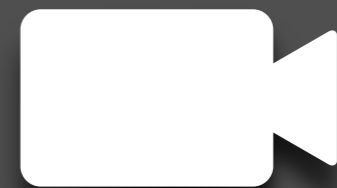
How to present your work... verbally



- Prepare an elevator pitch of your work
 - <5 min
- Know your audience
 - K.I.S.S.
 - People are not impressed with big words or jargon. They're impressed by your demonstration of your in-depth knowledge and curiosity.
- Make it a conversation with the person/crowd
 - Don't stare at your poster! **Talk to the people.**
 - **Face your body to the audience**
 - **Be proactive. "Hi! Can I tell you about my poster?"**

How to present your work... verbally

- Must be engaging. Be animated.
 - If you're bored, they'll be bored.
 - Consider the rhythm and melody of your delivery.
- **Practice & record yourself.** It's super awkward, but do it.
 - You'll hate it. Nobody likes the sound of their own voice.
 - **But there is no better way for you to experience your presentation from the other side!**
- Practice by explaining to your peers. **Practice aloud.** You want vocal muscle memory of what it's like to say the particular technical phrases, etc.



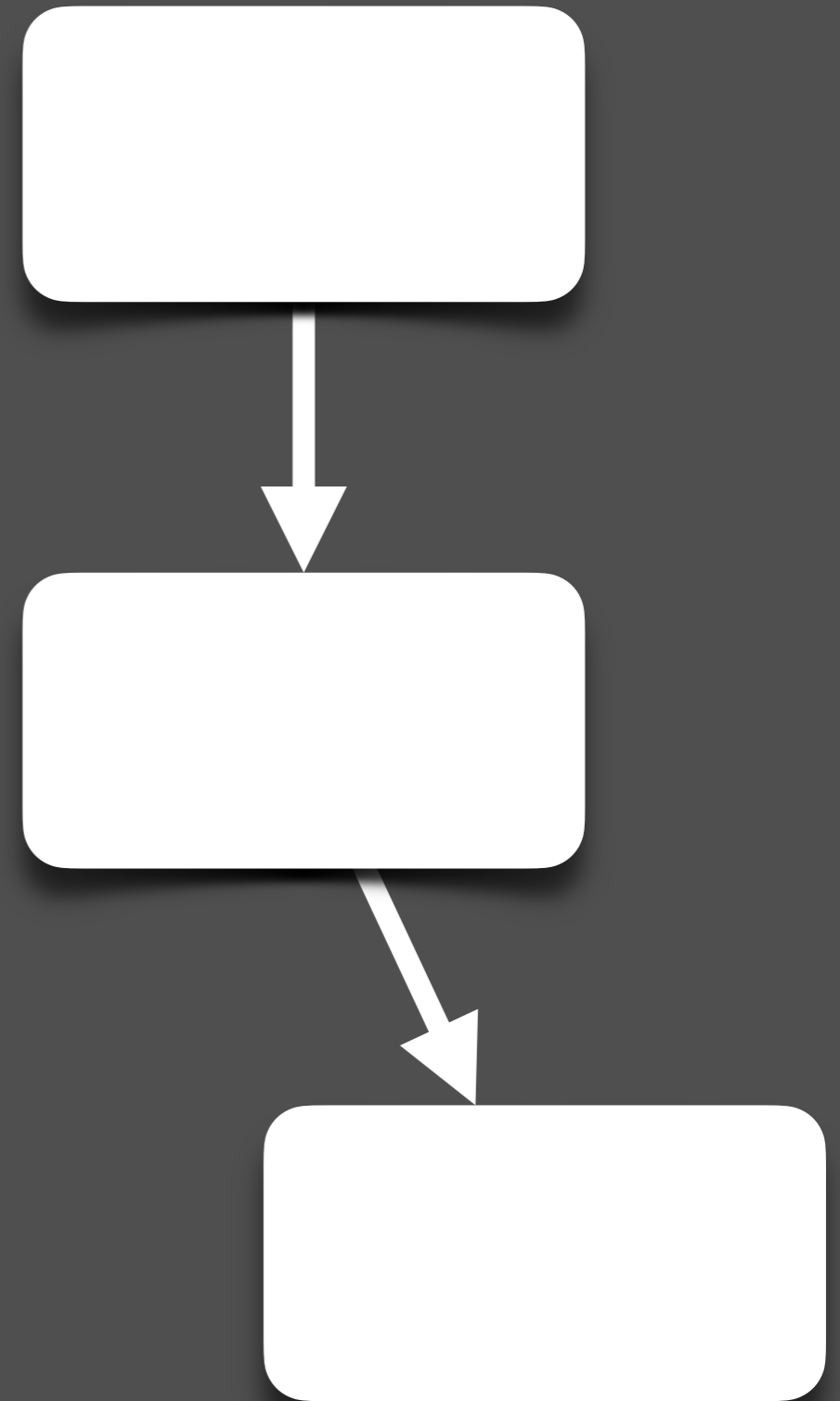
How to present your work... verbally

- Don't get lost in details. **Keep it pithy. Keep it fun.**
- This is a **high level overview** of your work
- If they look like they're trying to get away, be graceful about it
 - Finish your thought, thank them, and "let me know if you have any other questions!"
- **You don't have to go through your whole poster!**
 - Just point out your favorite highlights
 - If there's an interesting story to tell about your work, do it



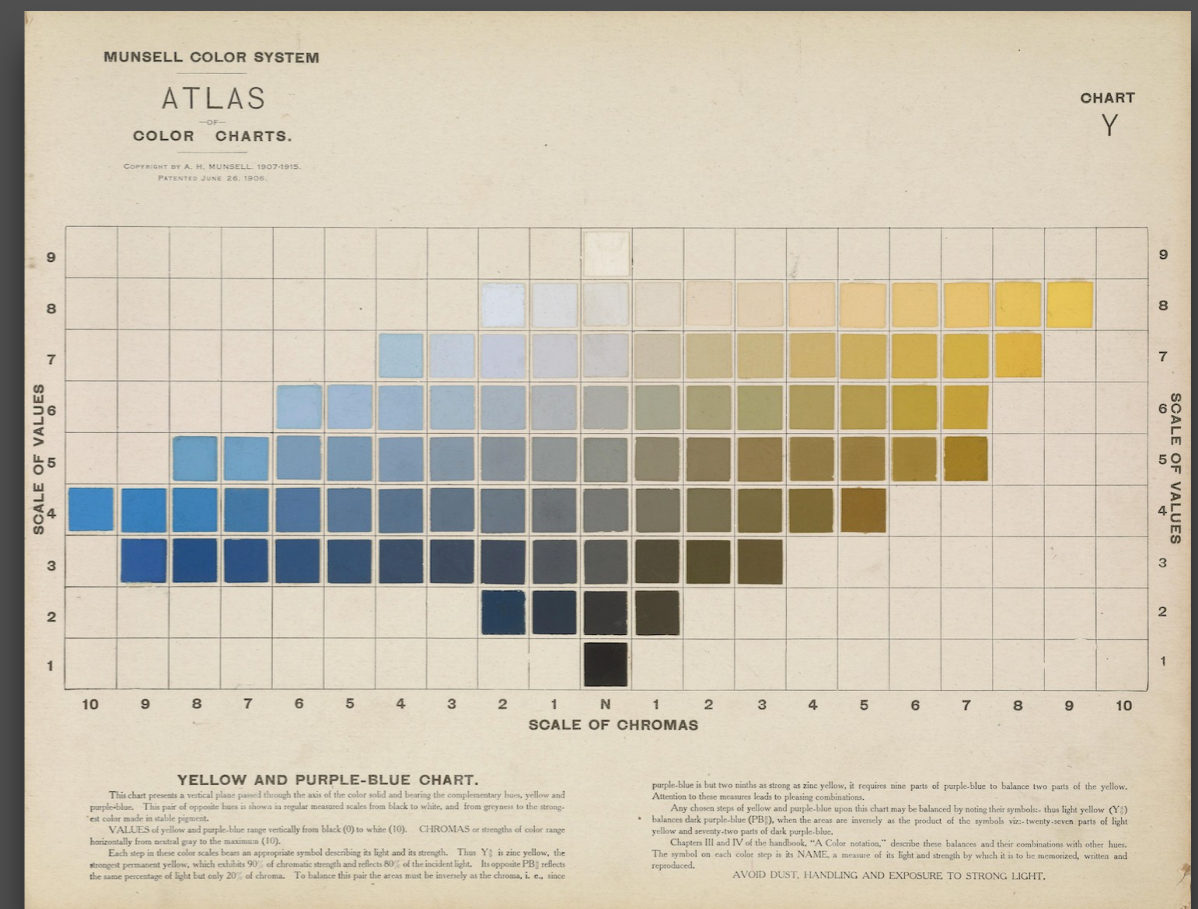
How to present your work... visually

- It's a **visual medium**. Be visual. Use flow charts. Use diagrams. Use plots.
- Looking for a **captivating visual design**
 - Minimum text. This is a visual aid.
 - Should be read from top-right to bottom-left
- Unless required, don't use the conference template. **Make the poster uniquely yours.**
- **We'll go through some examples in a bit**



Advanced Tips: Reaching the Next Level

- Learn some **basics of graphic design**
 - [\[LinkedIn Learning Course on Graphic Design Basics\]](#)
- Define a **consistent color language**
- Mind your **fonts**. The free fonts from google fonts are great.
 - Messy: Have a million fonts
 - Basic: Have one carefully chosen font that gives the character you want to convey
 - Advanced: Pairing typefaces



PRIMARY

Montserrat

Aa Bb Cc Dd Ee Ff Gg Hh Ii
Jj Kk Ll Mm Nn Oo Pp Qq
Rr Ss Tt Uu Vv Ww Xx Yy Zz

SECONDARY

Lato

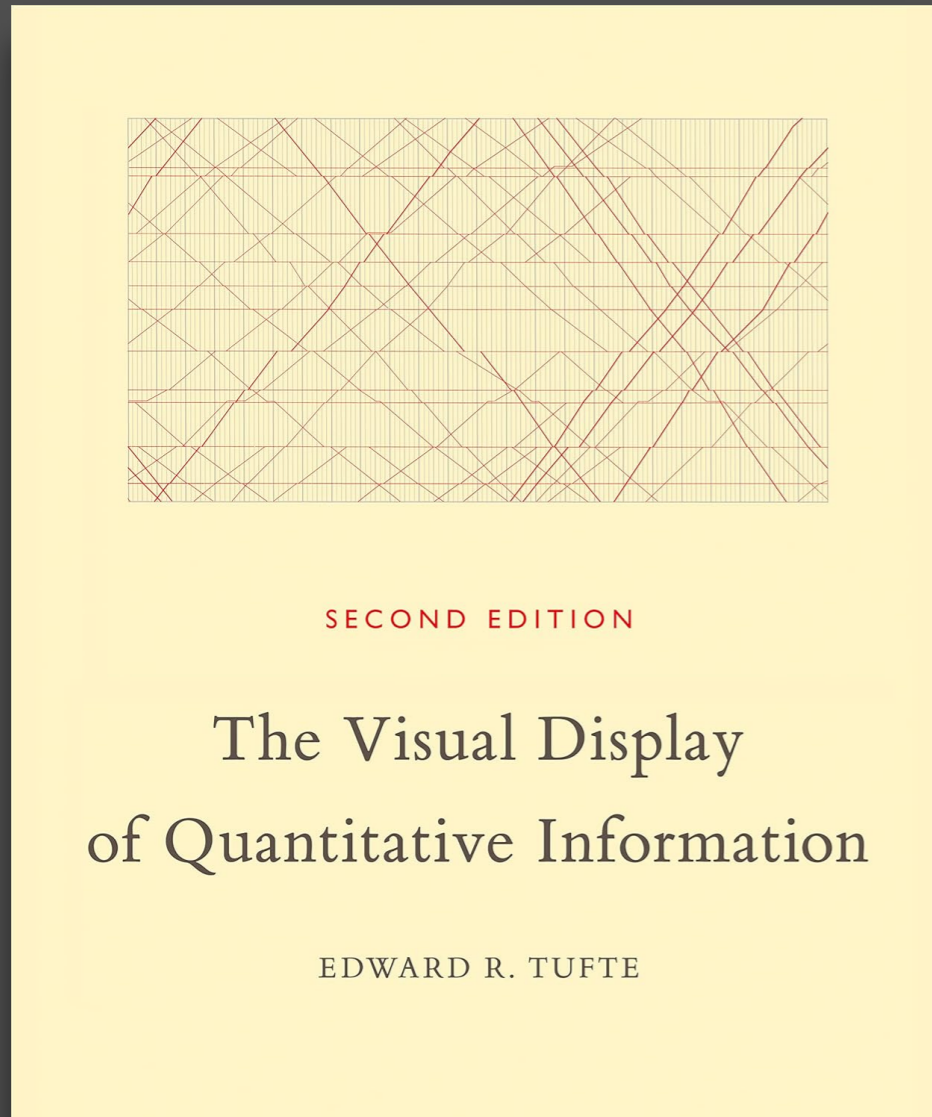
Aa Bb Cc Dd Ee Ff Gg Hh Ii
Jj Kk Ll Mm Nn Oo Pp Qq Rr
Ss Tt Uu Vv Ww Xx Yy Zz

APPLICATION

Lorem ipsum dolor sit

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

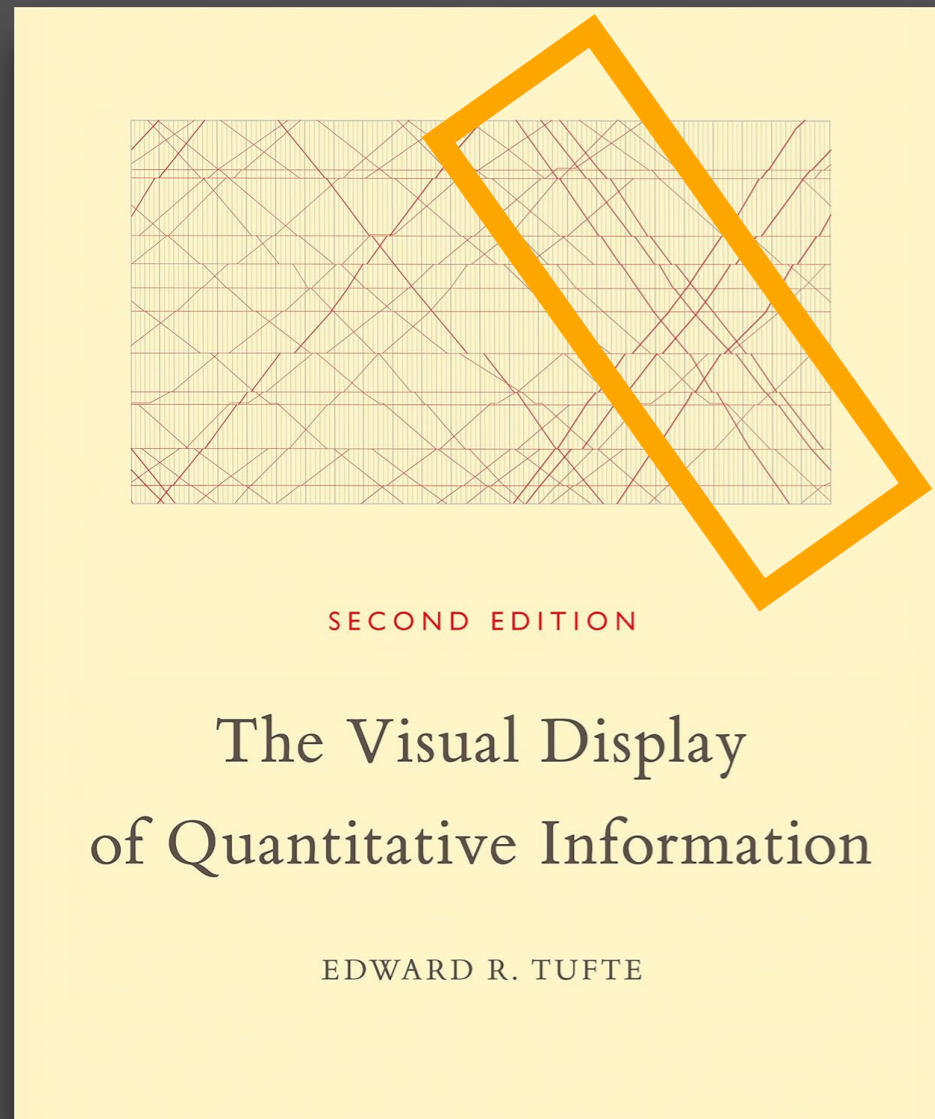
Advanced Tips: Reaching the Next Level



Highly recommended book!

- Guide the eyes of the viewer. Use arrows. **Bold stuff.** Highlight regions of plots.
- Play with **depth**
 - There is a z-axis of your poster!
 - Tool for emphasis. Important things placed “in front”

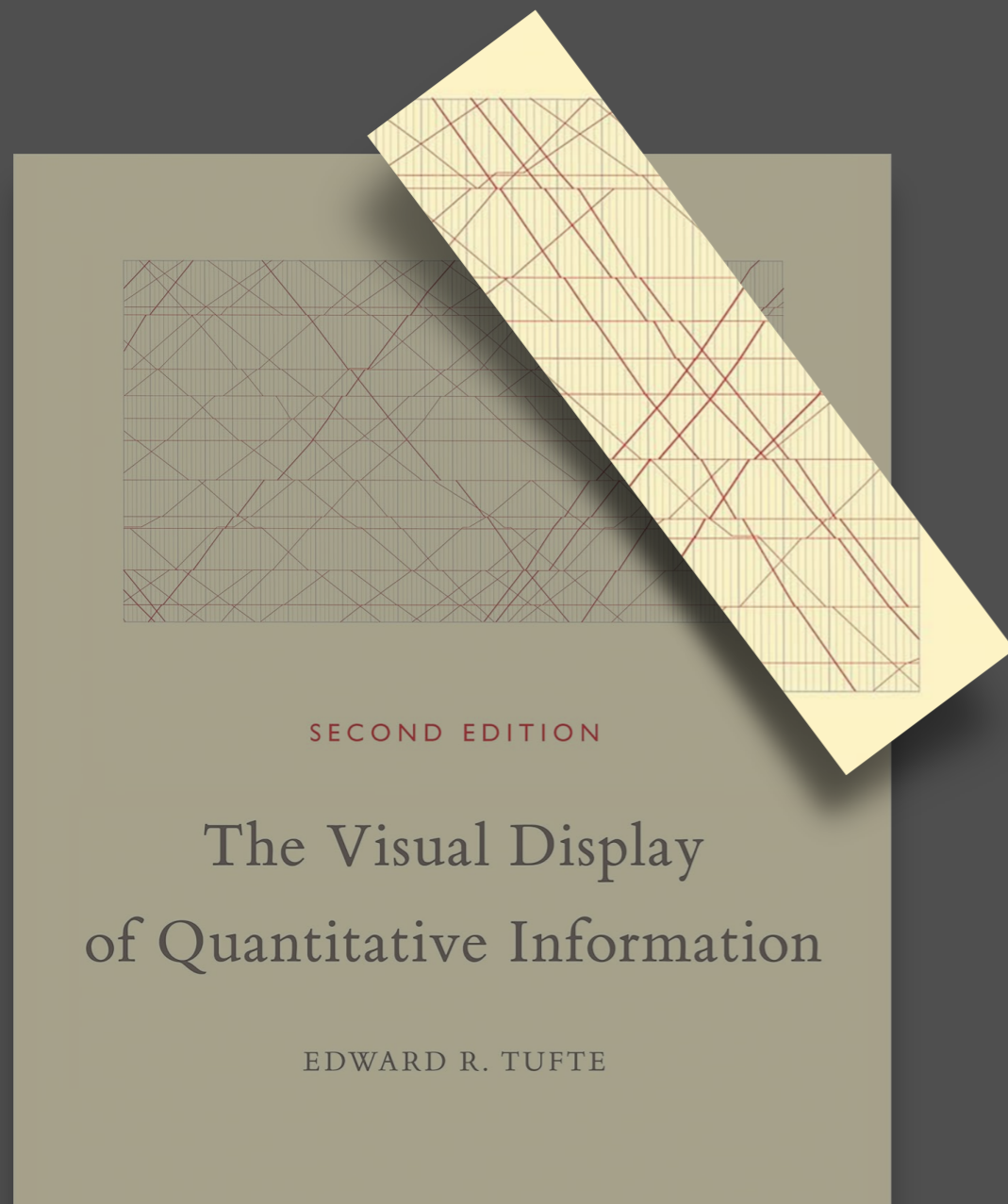
Advanced Tips: Reaching the Next Level



Highly recommended book!

- Guide the eyes of the viewer. Use arrows. **Bold stuff.** Highlight regions of plots.
- Play with **depth**
 - There is a z-axis of your poster!
 - Tool for emphasis. Important things placed “in front”

Advanced Tips: Reaching the Next Level

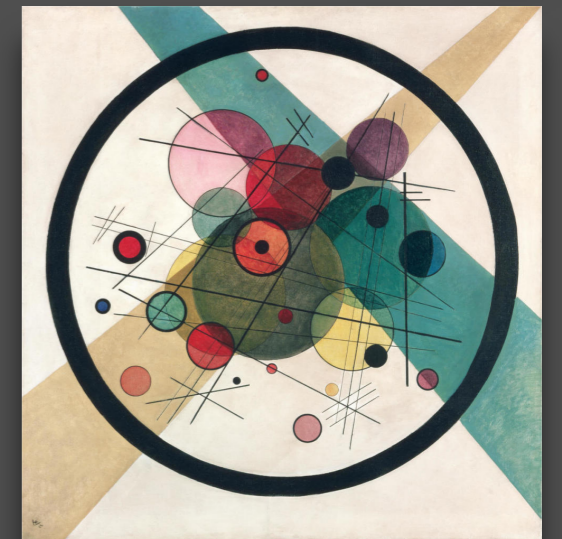


Highly recommended book!

- Guide the eyes of the viewer. Use arrows. **Bold stuff.** Highlight regions of plots.
- Play with **depth**
 - There is a z-axis of your poster!
 - Tool for emphasis. Important things placed “in front”

Advanced Tips: Reaching the Next Level

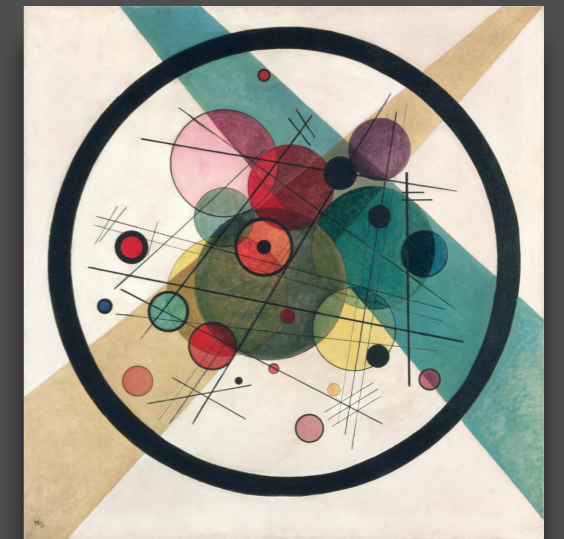
- Mind your **spacing**! Designs with cramped text are stressful to look at. Just because it “fits” doesn’t mean it fits. This is one of the biggest problems I see in posters (and in slides for presentations). Ideas need room to **breathe**!
- Make sure stuff is aligned! Learn how to use the align and distribute functions of your apps.
- Design for multiple viewing distances
 - Make it appealing when up close
 - But also: Stand out in crowd
 - If I walk in and scan room, why should I go visit your poster?
- If it makes sense to have a **prop**, do it!
- The poster is your **canvas**!
 - Use it to its fullest and make it yours
 - Give your poster space and context
 - **“Paint” beyond** the edge of the paper



Use symmetry. Shapes and sizes should be **motivated**.

Advanced Tips: Reaching the Next Level

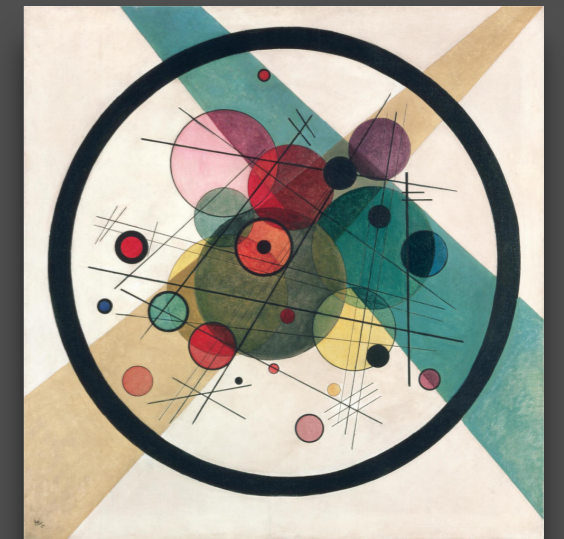
- Mind your **spacing**! Designs with cramped text are stressful to look at. Just because it “fits” doesn’t mean it fits. This is one of the biggest problems I see in posters (and in slides for presentations). Ideas need room to **breathe**!
- Make sure stuff is aligned! Learn how to use the align and distribute functions of your apps.
- Design for multiple viewing distances
 - Make it appealing when up close
 - But also: Stand out in crowd
 - If I walk in and scan room, why should I go visit your poster?
- If it makes sense to have a **prop**, do it!
- The poster is your **canvas**!
 - Use it to its fullest and make it yours
 - Give your poster space and context
 - **“Paint” beyond** the edge of the paper



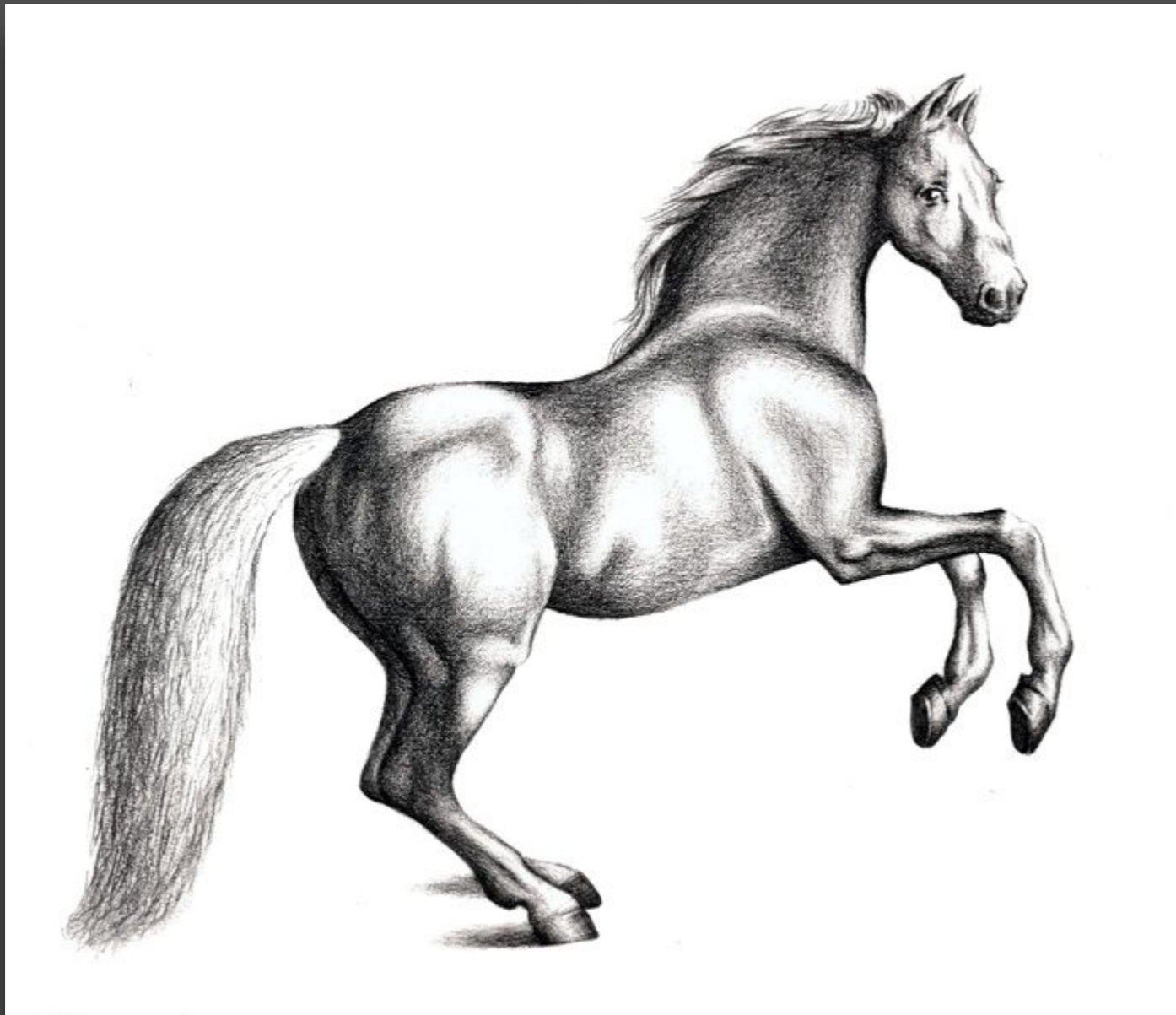
Use symmetry. Shapes and sizes should be **motivated**.

Advanced Tips: Reaching the Next Level

- Mind your **spacing**! Designs with cramped text are stressful to look at. Just because it “fits” doesn’t mean it fits. This is one of the biggest problems I see in posters (and in slides for presentations). Ideas need room to **breathe**!
- Make sure stuff is aligned! Learn how to use the align and distribute functions of your apps.
- Design for multiple viewing distances
 - Make it appealing when up close
 - But also: Stand out in crowd
 - If I walk in and scan room, why should I go visit your poster?
- If it makes sense to have a **prop**, do it!
- The poster is your **canvas**!
 - Use it to its fullest and make it yours
 - Give your poster space and context
 - **“Paint” beyond** the edge of the paper



Use symmetry. Shapes and sizes should be **motivated**.



Don't just place your
content on a page...

Give it visual context.
Make it a window into a space.

And finally this one

**YOU WILL READ
THIS FIRST!**

And then you'll read this

Then this one

And finally this one

**YOU WILL READ
THIS FIRST!**

And then you'll read this

Then this one

Visual **hierarchy** is key to
conducting the viewer's attention

“Do’s and Don’ts” Using

Example Posters



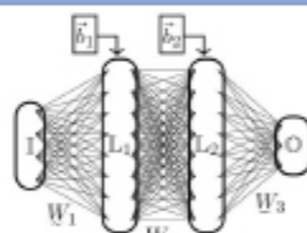
Physics-informed, Interpretable Machine Learning

Midshipman 2/C Nourachi
Professor Kevin McIlhany, Physics Department



Applying Machine Learning to Physics

- In traditional computing/algorithms, problem-solvers write a ruleset which generates a desired output from an input. In a machine learning (ML) approach, we want to construct an object that will find the rules for us which generate the desired output from a given input.
- There are many different forms of ML, including Artificial Neural Networks (ANN), Genetic Algorithms, Statistical Learning, Bayesian Inference, Gaussian Processes,
- Of the different forms of ML, Neural Networks show the most promise of being "interpretable" because they can generate structures which closely mirror traditional approaches. Comparing the ANN-generated architecture to the known traditionally-generated architecture allows us to understand what relationships the ANN is learning from the data it is being fed.
- Neural Networks can be used to extract information about physical systems by the following methods:
 - Differential Step Method
 - Evolves a system from state $\psi(t)$ to state $\psi(t+\Delta t)$
 - End-to-End Method
 - Evolves a system from an initial state $\psi(0)$ to a final state $\psi(\Delta t)$
 - Encoder-Decoder Method
 - Compress a state ψ to a smaller set of variables describing some aspect of the system and then decode back to the input state
 - Ex) encode a set of points on a circle to just the radius and center position, then expand it back out to points on the circle



$$\begin{aligned}\bar{L}_1 &= f_1(W_1 \cdot \bar{I}_j + \bar{b}_1) \\ \bar{L}_2 &= f_2(W_2 \cdot \bar{L}_1 + \bar{b}_2) \\ \bar{L}_3 &= f_3(W_3 \cdot \bar{L}_2 + \bar{b}_3) \\ &\vdots \\ \bar{O}_j &= f_n(W_n \cdot \bar{L}_{n-1} + \bar{b}_n)\end{aligned}$$

Images obtained from tech note - 01

Neural Net Design

- Layers** are mappings from an input vector to an output vector. The individual components of the output vector are referred to as nodes. A neural network can be composed of a single layer, but linking layers together allows for more complex mappings. Layer size is determined by the number of nodes that comprise the layer.
- Weights and biases** are the functional components of a layer which act on the input vector to produce an output for the next layer. Every node of a layer has a weighted connection to every node of the layer in front of it. Mathematically, this takes the form of a matrix of weights that are applied to the input vector, resulting in a linear transformation. Biases are applied to each node of the network to create thresholding behavior.
- Transfer/Activation Functions**, shown to the right, are wrapper functions that each node is passed through to add nonlinearity to the net. Some are referred to as squashing functions because they compress all nodes to a value between a small range (e.g. tanh compresses output between -1 and 1)
- A **Loss/Cost/Energy Function** quantifies how much error exists between the output of the neural net and the desired output. It takes various forms based on the type of learning that is imposed on the net (e.g. supervised vs. unsupervised). Commonly used in supervised learning is the mean squared error function, where O refers to an output vector of the net and T refers to the target vector that was desired:

$$MSE = \frac{1}{n} \sum_{i=1}^n |O_i - T_i|^2$$

Learning in a ANN equates to minimizing the loss function by tweaking all of the weights within the layers.

- For our purposes, the loss function is of particular interest because physical rules can be demanded of the ANN by addition of a meaningful **regularization term**. For example, we can demand that a quantity within the data be conserved and allow the neural net find the best mapping which does not violate that condition.
- A **Training/Optimization function** is the mathematical method by which the loss function is minimized. There are many to choose from, such as random changes of weights or a more methodical gradient descent algorithm.

Example Activation Functions

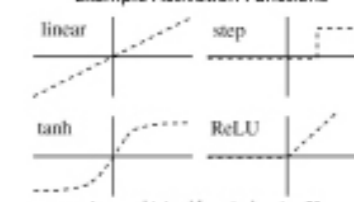
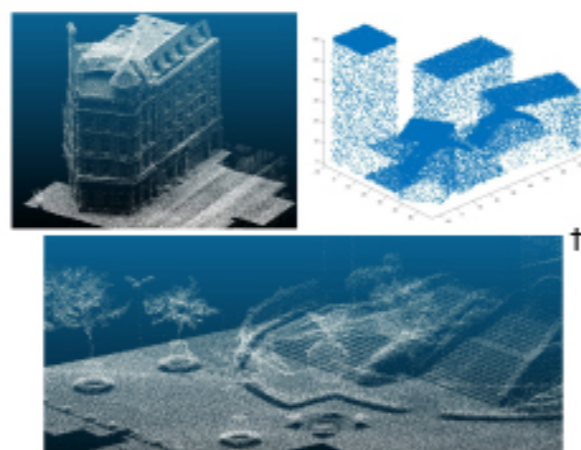


Image obtained from tech note - 01

Processing LiDAR Data of Urban Infrastructure

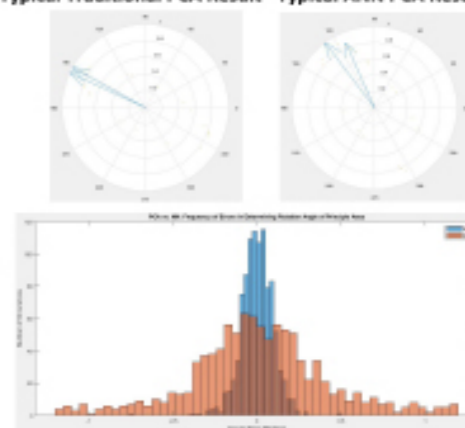
- Neural Networks are useful for their ability find relationships within large amounts of data and reduce the dimensionality of that data down to only what the data scientist is interested in. In other words, NNs can compress loads of uninterpretable data into small chunks of information that are useful.
- By flying over a city with LiDAR equipment, a point-cloud can be created which has thousands of points showing where light reflected off of the surface of some object, such as a building face or bench. We want to be able to take this data set of x, y, z coordinates and cluster/map them to the geometric features that they describe.
- For example, we want to take the thousands of points that lie on the face of one of the buildings and first separate them from the millions of other points in the dataset, then we want those points to be mapped to a finite plane that describes that side of the building. Repeating this process over the whole dataset will allow us to construct a usable 3D model of the entire city.
- LiDAR point cloud data of urban infrastructure is one such case where incredibly large datasets need to be reduced to much lower dimensionality while still maintaining all the information contained within the data. For this reason, NNs have a lot of potential in solving this problem.



Network-based PCA Approach

- Principle Component Analysis (PCA)** is a method of finding vectors in a data set which have maximum variance (i.e. the vectors along which the data is most dispersed).
- With regards to ellipses, performing PCA yields the directions of both the semi-major and semi-minor axes. In solving this problem, we are interested in the principle axes of ellipsoids.
- We have determined that ANNs are capable of performing PCA, although generally with less accuracy for isolated structures. The data shown at the right shows that the error distribution is wider for ANN-PCA. This is also illustrated by the typical results when applying PCA to ellipses; the two vectors in each image show the deviation of the calculated principle axes from the true principle axes.
- That being said, ANNs have a distinct advantage over traditional PCA computation which is important in processing the LiDAR data: NNs can discriminate between separate structures within given data while traditional PCA cannot.
- Generally, our intended method for processing the LiDAR data with an ANN is by:
 - 1) Cluster the point cloud data into small pancake-like ellipsoidal collections of points
 - 2) Use principal component analysis to find the unit surface normal vectors of all the ellipsoids
 - 3) Cluster surfaces with similar surface normal
 - 4) Average the surface normal vectors within a cluster to get a larger surface with one normal
 - 5) Repeat 3 and 4 until all complete surfaces have been extracted from the point cloud.

Typical Traditional PCA Result Typical ANN PCA Result



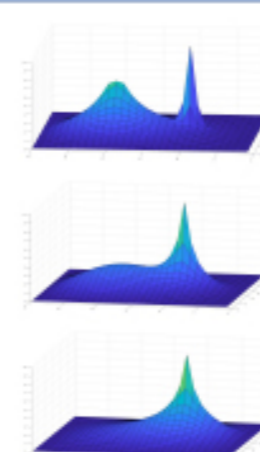
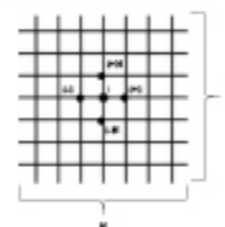
Architecture Matching of (In)Homogeneous Diffusion Solutions

- The homogeneous diffusion equation is given by: $\frac{\partial u}{\partial t} = \kappa \nabla^2 u(r, t)$
- Where κ is a factor representing the ease of diffusion at every point in space and u is the scalar function which describes the magnitude of some physical quantity at every point in space and time (i.e. temperature). This simplified equation represents the special case where κ is constant over whatever domain is being considered.
- The most common way to solve this equation is by using separation of variables and solving for the eigenvalues and eigenmodes that span the entire domain. Unfortunately, it is computationally expensive to solve in this way when the boundary conditions change with time, as the eigenmodes must be recomputed over the entirety of the new domain at every timestep.
- For changing boundary condition situations, we can resort to an approximate solution which acts on a discrete domain called **cellular automata**. While not exact, this approach is computationally inexpensive when handling changing boundary conditions, and it yields a solution that has close resemblance to neural network architecture.
- This similarity allows us to match the architecture of a neural network to the cellular automata solution. From there, we can potentially train the neural network to find a better approximation that has equal computational cost. In the inhomogeneous case, we can potentially use the network to extract the **K-landscape** from data describing the time evolution of a diffusive system. This has applications in manufacturing and materials science for its ability to reveal imperfections in metals based on how heat diffuses through them.

Cellular Automata Approach

- Performing separation of variables yields 2 equations which describe the spatial and time evolution of the system separately.

$$\nabla^2 \psi(r) = -\frac{\lambda}{\kappa} \psi(r) \quad \frac{\partial \phi}{\partial t} = -\lambda \phi(t)$$
- Suppose the state ψ exists in a discrete, 2D grid space where each grid point is separated by a small distance h .
 - The $N \times N$ grid can be represented by an $N^2 \times 1$ vector if serialized by the index i .
- In this representation, the values, λ , are indexed by i , allowing us to evolve the system on site-by-site basis, solely based upon nearest neighbor interactions.
- We can approximate ∇^2 to 1st order as the discrete Laplacian
- While only an approximate solution, this method can easily handle changing boundary conditions because at each timestep, system evolution is computed by local interactions rather than over the entire domain.



Cellular Automata → ANN

- From the cellular automata approach, we obtain a matrix operator which acts on an input state to evolve it to the next state.
- The cellular automata operator's architecture closely resembles that of a neural network. We can directly import the operator matrix as the weight matrices of a neural network!
- We can add some random variation to the weights during training and hopefully steer the training toward a higher order, nonlinear solution. If we can accomplish this, it will yield a robust solution that is also relatively computationally inexpensive.
- NN for Extracting K-Landscape**
- In the inhomogeneous case, κ can take a different value at every point in space, and the systems evolution is dictated by:

$$\frac{\partial u}{\partial t} = \nabla \cdot (\kappa(r) \nabla u(r, t)) + \kappa(r) \nabla^2 u(r, t)$$
- While much harder to reach, we have the cellular automata solution for the inhomogeneous case and can perform similar architecture matching to an ANN.
- Alternatively, using an encoder-decoder architecture, we predict that an ANN can encode the state of the system to a space which describes the κ -landscape. It should then be able to continue mapping to either the initial state or the state of the system at the next timestep.



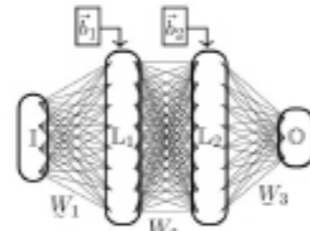
Physics-informed, Interpretable Machine Learning

Midshipman 2/C Nourachi
Professor Kevin McIlhany, Physics Department



Applying Machine Learning to Physics

- In traditional computing/algorithms, problem-solvers **write a ruleset** which generates a desired output from an input. In a machine learning (ML) approach, we want to construct an object that will **find the rules for us** which generate the desired output from a given input.
- There are many different forms of ML, including **Artificial Neural Networks (ANN)**, Genetic Algorithms, Statistical Learning, Bayesian Inference, Gaussian Processes,
- Of the different forms of ML, Neural Networks show the most promise of being "interpretable" because they can generate structures which closely mirror traditional approaches. Comparing the ANN-generated architecture to the known traditionally-generated architecture allows us to understand what relationships the ANN is learning from the data it is being fed.



$$\begin{aligned}\bar{L}_1 &= f_1(W_1 \cdot \bar{I}_j + \bar{b}_1) \\ \bar{L}_2 &= f_2(W_2 \cdot \bar{L}_1 + \bar{b}_2) \\ \bar{L}_3 &= f_3(W_3 \cdot \bar{L}_2 + \bar{b}_3) \\ &\vdots \\ \bar{O}_j &= f_n(W_n \cdot \bar{L}_{n-1} + \bar{b}_n)\end{aligned}$$

Images obtained from tech note - 01

Neural Net Design

- Layers** are mappings from an input vector to an output vector. The individual components of the output vector are referred to as **nodes**. A neural network can be composed of a single layer, but linking layers together allows for more complex mappings. Layer size is determined by the number of nodes that comprise the layer.
- Weights and biases** are the functional components of a layer which act on the input vector to produce an output for the next layer. Every node of a layer has a weighted connection to every node of the layer in front of it. Mathematically, this takes the form of a matrix of weights that are applied to the input vector, resulting in a linear transformation. Biases are applied to each node of the network to create thresholding behavior.
- Transfer/Activation Functions**, shown to the right, are wrapper functions that each node is passed through to add nonlinearity to the net. Some are referred to as **squashing functions** because they compress all nodes to a value between a small range (e.g. tanh compresses output between -1 and 1).
- A **Loss/Cost/Energy Function** quantifies how much error exists between the output of the neural net and the desired output. It takes various forms based on the type of learning that is imposed on the net (e.g. supervised vs. unsupervised). Commonly used in supervised learning is the mean squared error function, where \bar{O} refers to an output vector of the net and \bar{T} refers to the target vector that was desired:

$$MSE = \frac{1}{n} \sum_{i=1}^n (\bar{O}_i - \bar{T}_i)^2$$

Learning in a ANN equates to minimizing the loss function by tweaking all of the weights within the layers.

- For our purposes, the loss function is of particular interest because physical rules can be demanded of the ANN by addition of a meaningful **regularization term**. For example, we can demand that a quantity within the data be conserved and allow the neural net find the best mapping which does not violate that condition.
- A **Training/Optimization function** is the mathematical method by which the loss function is minimized. There are many to choose from, such as random changes of weights or a more methodical gradient descent algorithm.

Example Activation Functions

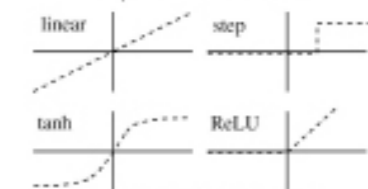
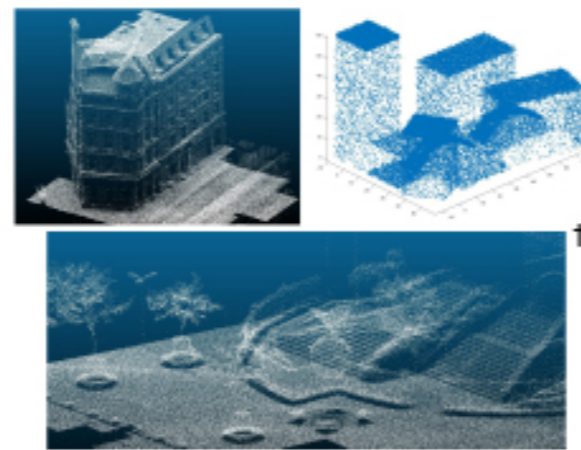


Image obtained from tech note - 01

Way too much text/detail

Processing LiDAR Data of Urban Infrastructure

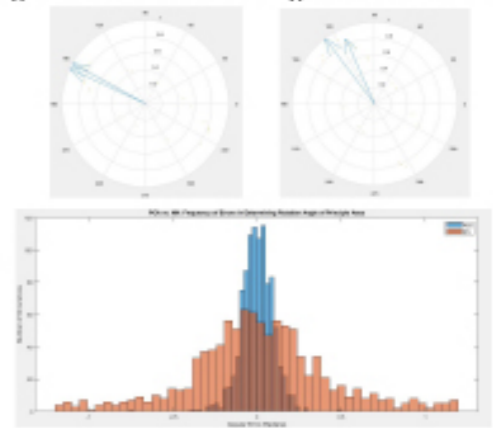
- Neural Networks are useful for their ability find relationships within large amounts of data and reduce the dimensionality of that data down to only what the data scientist is interested in. In other words, NNs can compress loads of uninterpretable data into small chunks of information that are useful.
- By flying over a city with LiDAR equipment, a point-cloud can be created which has thousands of points showing where light reflected off of the surface of some object, such as a building face or bench. We want to be able to take this data set of **x, y, z coordinates** and **cluster/map** them to the geometric features that they describe.
- For example, we want to take the thousands of points that lie on the face of one of the buildings and first separate them from the millions of other points in the dataset, then we want those points to be mapped to a finite plane that describes that side of the building. Repeating this process over the whole dataset will allow us to construct a usable 3D model of the entire city.
- LiDAR point cloud data of urban infrastructure is one such case where incredibly large datasets need to be reduced to much lower dimensionality while still maintaining all the information contained within the data. For this reason, NNs have a lot of potential in solving this problem.



Network-based PCA Approach

- Principle Component Analysis (PCA)** is a method of finding vectors in a data set which have maximum variance (i.e. the vectors along which the data is most dispersed).
- With regards to ellipses, performing PCA yields the directions of both the semi-major and semi-minor axes. In solving this problem, we are interested in the principle axes of ellipsoids.
- We have determined that ANNs are capable of performing PCA, although generally with less accuracy for isolated structures. The data shown at the right shows that the error distribution is wider for ANN-PCA. This is also illustrated by the typical results when applying PCA to ellipses; the two vectors in each image show the deviation of the calculated principle axes from the true principle axes.
- That being said, ANNs have a distinct advantage over traditional PCA computation which is important in processing the LiDAR data: NNs can discriminate between separate structures within given data while traditional PCA cannot.
- Generally, our intended method for processing the LiDAR data with an ANN is by:
 - 1) Cluster the point cloud data into small pancake-like ellipsoidal collections of points
 - 2) Use principal component analysis to find the unit surface normal vectors of all the ellipsoids
 - 3) Cluster surfaces with similar surface normal
 - 4) Average the surface normal vectors within a cluster to get a larger surface with one normal
 - 5) Repeat 3 and 4 until all complete surfaces have been extracted from the point cloud.

Typical Traditional PCA Result Typical ANN PCA Result

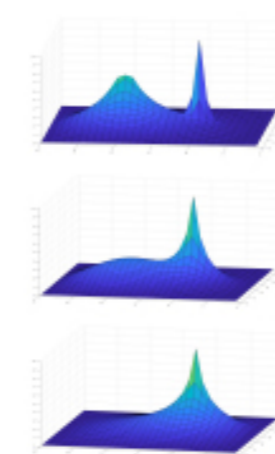
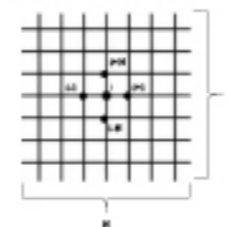


Architecture Matching of (In)Homogeneous Diffusion Solutions

- The homogeneous diffusion equation is given by: $\frac{\partial u}{\partial t} = \kappa \nabla^2 u(r, t)$
- Where κ is a factor representing the ease of diffusion at every point in space and u is the scalar function which describes the magnitude of some physical quantity at every point in space and time (i.e. temperature). This simplified equation represents the special case where κ is constant over whatever domain is being considered.
- The most common way to solve this equation is by using separation of variables and solving for the eigenvalues and eigenmodes that span the entire domain. Unfortunately, it is computationally expensive to solve in this way when the boundary conditions change with time, as the eigenmodes must be recomputed over the entirety of the new domain at every timestep.
- For changing boundary condition situations, we can resort to an approximate solution which acts on a discrete domain called **cellular automata**. While not exact, this approach is computationally inexpensive when handling changing boundary conditions, and it yields a solution that has close resemblance to neural network architecture.
- This similarity allows us to match the architecture of a neural network to the cellular automata solution. From there, we can potentially **train the neural network to find a better approximation** that has equal computational cost. In the inhomogeneous case, we can potentially use the network to extract the **K-landscape** from data describing the time evolution of a diffusive system. This has applications in manufacturing and materials science for its ability to reveal imperfections in metals based on how heat diffuses through them.

Cellular Automata Approach

- Performing separation of variables yields 2 equations which describe the spatial and time evolution of the system separately.
- $\nabla^2 \psi(r) = -\frac{\lambda}{\kappa} \psi(r) \quad \frac{\partial \phi}{\partial t} = -\lambda \phi(t)$
- Suppose the state ψ exists in a discrete, 2D grid space where each grid point is separated by a small distance h .
 - The $N \times N$ grid can be represented by an $N^2 \times 1$ vector if serialized by the index i .
- In this representation, the values, λ , are indexed by i , allowing us to evolve the system on site-by-site basis, solely based upon nearest neighbor interactions.
- We can approximate ∇^2 to 1st order as the discrete Laplacian
- While only an approximate solution, this method can easily handle changing boundary conditions because at each timestep, system evolution is computed by local interactions rather than over the entire domain.



Cellular Automata → ANN

- From the cellular automata approach, we obtain a matrix operator which acts on an input state to evolve it to the next state.
- The cellular automata operator's architecture closely resembles that of a neural network. We can directly import the operator matrix as the weight matrices of a neural network!
- We can add some random variation to the weights during training and hopefully steer the training toward a higher order, nonlinear solution. If we can accomplish this, it will yield a robust solution that is also relatively computationally inexpensive.
- NN for Extracting K-Landscape**
- In the inhomogeneous case, κ can take a different value at every point in space, and the systems evolution is dictated by:

$$\frac{\partial u}{\partial t} = \nabla \cdot (\kappa(r) \nabla u(r, t)) + \lambda(r) \nabla^2 u(r, t)$$
- While much harder to reach, we have the cellular automata solution for the inhomogeneous case and can perform similar architecture matching to an ANN.
- Alternatively, using an encoder-decoder architecture, we predict that an ANN can encode the state of the system to a space which describes the K-landscape. It should then be able to continue mapping to either the initial state or the state of the system at the next timestep.



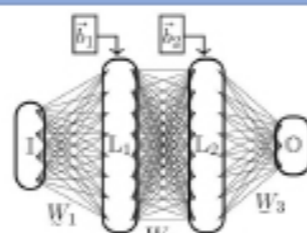
Physics-informed, Interpretable Machine Learning

Midshipman 2/C Nourachi
Professor Kevin McIlhany, Physics Department



Applying Machine Learning to Physics

- In traditional computing/algorithms, problem-solvers **write a ruleset** which generates a desired output from an input. In a machine learning (ML) approach, we want to construct an object that will **find the rules for us** which generate the desired output from a given input.
- There are many different forms of ML, including **Artificial Neural Networks (ANN)**, Genetic Algorithms, Statistical Learning, Bayesian Inference, Gaussian Processes,
- Of the different forms of ML, Neural Networks show the most promise of being "interpretable" because they can generate structures which closely mirror traditional approaches. Comparing the ANN-generated architecture to the known traditionally-generated architecture allows us to understand what relationships the ANN is learning from the data it is being fed.



$$\begin{aligned}\bar{L}_1 &= f_1(W_1 \cdot \bar{I}_j + \bar{b}_1) \\ \bar{L}_2 &= f_2(W_2 \cdot \bar{L}_1 + \bar{b}_2) \\ \bar{L}_3 &= f_3(W_3 \cdot \bar{L}_2 + \bar{b}_3) \\ &\vdots \\ \bar{O}_j &= f_n(W_n \cdot \bar{L}_{n-1} + \bar{b}_n)\end{aligned}$$

Images obtained from tech note - 01

Neural Net Design

- Layers** are mappings from an input vector to an output vector. The individual components of the output vector are referred to as nodes. A neural network can be composed of a single layer, but linking layers together allows for more complex mappings. Layer size is determined by the number of nodes that comprise the layer.
- Weights and biases** are the functional components of a layer which act on the input vector to produce an output for the next layer. Every node of a layer has a weighted connection to every node of the layer in front of it. Mathematically, this takes the form of a matrix of weights that are applied to the input vector, resulting in a linear transformation. Biases are applied to each node of the network to create thresholding behavior.
- Transfer/Activation Functions**, shown to the right, are wrapper functions that each node is passed through to add nonlinearity to the net. Some are referred to as squashing functions because they compress all nodes to a value between a small range (e.g. tanh compresses output between -1 and 1).
- A **Loss/Cost/Energy Function** quantifies how much error exists between the output of the neural net and the desired output. It takes various forms based on the type of learning that is imposed on the net (e.g. supervised vs. unsupervised). Commonly used in supervised learning is the mean squared error function, where \bar{O} refers to an output vector of the net and \bar{T} refers to the target vector that was desired:

$$MSE = \frac{1}{n} \sum_{i=1}^n (\bar{O}_i - \bar{T}_i)^2$$

Learning in a ANN equates to minimizing the loss function by tweaking all of the weights within the layers.

- For our purposes, the loss function is of particular interest because physical rules can be demanded of the ANN by addition of a meaningful **regularization term**. For example, we can demand that a quantity within the data be conserved and allow the neural net find the best mapping which does not violate that condition.
- A **Training/Optimization function** is the mathematical method by which the loss function is minimized. There are many to choose from, such as random changes of weights or a more methodical gradient descent algorithm.

Example Activation Functions

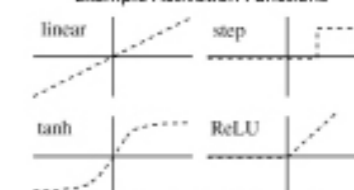
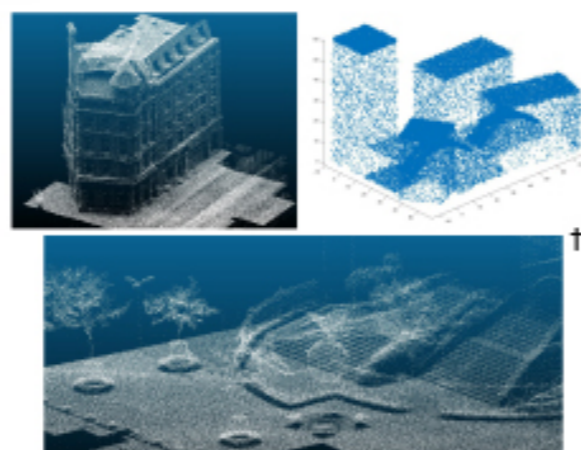


Image obtained from tech note - 01

Way too much text/detail

Processing LiDAR Data of Urban Infrastructure

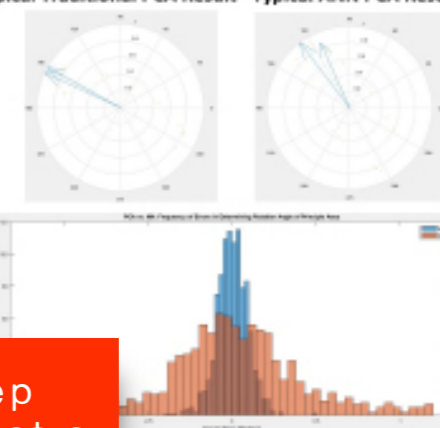
- Neural Networks are useful for their ability find relationships within large amounts of data and reduce the dimensionality of that data down to only what the data scientist is interested in. In other words, NNs can compress loads of uninterpretable data into small chunks of information that are useful.
- By flying over a city with LiDAR equipment, a point-cloud can be created which has thousands of points showing where light reflected off of the surface of some object, such as a building face or bench. We want to be able to take this data set of x, y, z coordinates and cluster/map them to the geometric features that they describe.
- For example, we want to take the thousands of points that lie on the face of one of the buildings and first separate them from the millions of other points in the dataset, then we want those points to be mapped to a finite plane that describes that side of the building. Repeating this process over the whole dataset will allow us to construct a usable 3D model of the entire city.
- LiDAR point cloud data of urban infrastructure is one such case where incredibly large datasets need to be reduced to much lower dimensionality while still maintaining all the information contained within the data. For this reason, NNs have a lot of potential in solving this problem.



Network-based PCA Approach

- Principle Component Analysis (PCA)** is a method of finding vectors in a data set which have maximum variance (i.e. the vectors along which the data is most dispersed).
- With regards to ellipses, performing PCA yields the directions of both the semi-major and semi-minor axes. In solving this problem, we are interested in the principle axes of ellipsoids.
- We have determined that ANNs are capable of performing PCA, although generally with less accuracy for isolated structures. The data shown at the right shows that the error distribution is wider for ANN-PCA. This is also illustrated by the typical results when applying PCA to ellipses; the two vectors in each image show the deviation of the calculated principle axes from the true principle axes.
- That being said, ANNs have a distinct advantage over traditional PCA computation which is important in processing the LiDAR data: NNs can discriminate between separate structures within given data while traditional PCA cannot.
- Generally, our intended method for processing the LiDAR data with an ANN is by:
 - 1) Cluster the points
 - 2) Use principal component analysis
 - 3) Cluster surfaces
 - 4) Average the results
 - 5) Repeat 3 and 4

Typical Traditional PCA Result Typical ANN PCA Result



Too many topics! Keep focused on one project at a time. Maybe two.

Architecture Matching of (In)Homogeneous Diffusion Solutions

- The homogeneous diffusion equation is given by: $\frac{\partial u}{\partial t} = \kappa \nabla^2 u(r, t)$
- Where κ is a factor representing the ease of diffusion at every point in space and u is the scalar function which describes the magnitude of some physical quantity at every point in space and time (i.e. temperature). This simplified equation represents the special case where κ is constant over whatever domain is being considered.
- The most common way to solve this equation is by using separation of variables and solving for the eigenvalues and eigenmodes that span the entire domain. Unfortunately, it is computationally expensive to solve in this way when the boundary conditions change with time, as the eigenmodes must be recomputed over the entirety of the new domain at every timestep.
- For changing boundary condition situations, we can resort to an approximate solution which acts on a discrete domain called **cellular automata**. While not exact, this approach is computationally inexpensive when handling changing boundary conditions, and it yields a solution that has close resemblance to neural network architecture.
- This similarity allows us to match the architecture of a neural network to the cellular automata solution. From there, we can potentially **train the neural network to find a better approximation** that has equal computational cost. In the inhomogeneous case, we can potentially use the network to extract the **K-landscape** from data describing the time evolution of a diffusive system. This has applications in manufacturing and materials science for its ability to reveal imperfections in metals based on how heat diffuses through them.

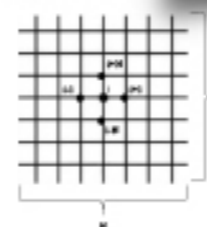
Cellular Automata Approach

- Performing separation of variables yields 2 equations which describe the spatial and time evolution of the system separately.

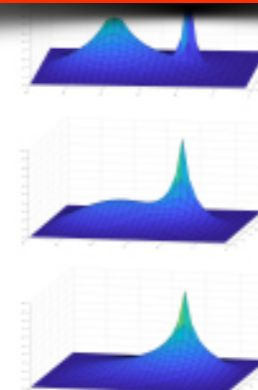
$$\nabla^2 \psi(r) = -\frac{\lambda}{\kappa} \psi(r) \quad \frac{\partial \phi}{\partial t} = -\lambda \phi(t)$$

- Suppose the state ψ exists in a discrete, 2D grid space where each grid point is separated by a small distance δ .
 - The $N \times N$ grid can be represented by an $N^2 \times 1$ vector if serialized by the index i .
- In this representation, the values, λ , are indexed by i , allowing us to evolve the system on site-by-site basis, solely based upon nearest neighbor interactions.
- We can approximate ∇^2 to 1st order as the discrete Laplacian

- While only an approximate solution, this method can easily handle changing boundary conditions because at each timestep, system evolution is computed by local interactions rather than over the entire domain.



- based on the initial conditions, we can then solve for λ_i and construct a matrix operator which evolves the state in discrete timesteps, as is shown to the right.



- The cellular automata operator's architecture closely resembles that of a neural network. We can directly import the operator matrix as the weight matrices of a neural network!
- We can add some random variation to the weights during training and hopefully steer the training toward a higher order, nonlinear solution. If we can accomplish this, it will yield a robust solution that is also relatively computationally inexpensive.
- NN for Extracting K-Landscape**
- In the inhomogeneous case, κ can take a different value at every point in space, and the systems evolution is dictated by:

$$\frac{\partial u}{\partial t} = \nabla \cdot (\kappa(r) \nabla u(r, t)) + \lambda(r) \nabla^2 u(r, t)$$
- While much harder to reach, we have the cellular automata solution for the inhomogeneous case and can perform similar architecture matching to an ANN.
- Alternatively, using an encoder-decoder architecture, we predict that an ANN can encode the state of the system to a space which describes the K-landscape. It should then be able to continue mapping to either the initial state or the state of the system at the next timestep.



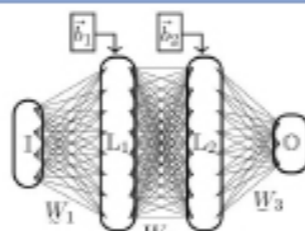
Physics-informed, Interpretable Machine Learning

Midshipman 2/C Nourachi
Professor Kevin McIlhany, Physics Department



Applying Machine Learning to Physics

- In traditional computing/algorithms, problem-solvers **write a ruleset** which generates a desired output from an input. In a machine learning (ML) approach, we want to construct an object that will **find the rules for us** which generate the desired output from a given input.
- There are many different forms of ML, including **Artificial Neural Networks (ANN)**, Genetic Algorithms, Statistical Learning, Bayesian Inference, Gaussian Processes,
- Of the different forms of ML, Neural Networks show the most promise of being "interpretable" because they can generate structures which closely mirror traditional approaches. Comparing the ANN-generated architecture to the known traditionally-generated architecture allows us to understand what relationships the ANN is learning from the data it is being fed.



$$\begin{aligned}\bar{L}_1 &= f_1(W_1 \cdot \bar{I}_j + \bar{b}_1) \\ \bar{L}_2 &= f_2(W_2 \cdot \bar{L}_1 + \bar{b}_2) \\ \bar{L}_3 &= f_3(W_3 \cdot \bar{L}_2 + \bar{b}_3) \\ &\vdots \\ \bar{O}_j &= f_n(W_n \cdot \bar{L}_{n-1} + \bar{b}_n)\end{aligned}$$

Images obtained from tech note - 01

Neural Net Design

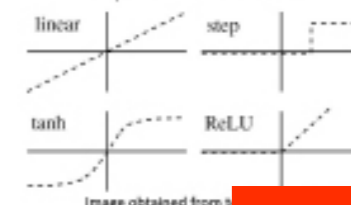
- Layers** are mappings from an input vector to an output vector. The individual components of the output vector are referred to as nodes. A neural network can be composed of a single layer, but linking layers together allows for more complex mappings. Layer size is determined by the number of nodes that comprise the layer.
- Weights and biases** are the functional components of a layer which act on the input vector to produce an output for the next layer. Every node of a layer has a weighted connection to every node of the layer in front of it. Mathematically, this takes the form of a matrix of weights that are applied to the input vector, resulting in a linear transformation. Biases are applied to each node of the network to create thresholding behavior.
- Transfer/Activation Functions**, shown to the right, are wrapper functions that each node is passed through to add nonlinearity to the net. Some are referred to as squashing functions because they compress all nodes to a value between a small range (e.g. tanh compresses output between -1 and 1).
- A **Loss/Cost/Energy Function** quantifies how much error exists between the output of the neural net and the desired output. It takes various forms based on the type of learning that is imposed on the net (e.g. supervised vs. unsupervised). Commonly used in supervised learning is the mean squared error function, where \bar{O} refers to an output vector of the net and \bar{T} refers to the target vector that was desired:

$$MSE = \frac{1}{n} \sum_{i=1}^n (\bar{O}_i - \bar{T}_i)^2$$

Learning in a ANN equates to minimizing the loss function by tweaking all of the weights within the layers.

- For our purposes, the loss function is of particular interest because physical rules can be demanded of the ANN by addition of a meaningful **regularization term**. For example, we can demand that a quantity within the data be conserved and allow the neural net find the best mapping which does not violate that condition.
- A **Training/Optimization function** is the mathematical method by which the loss function is minimized. There are many to choose from, such as random changes of weights or a more methodical gradient descent algorithm.

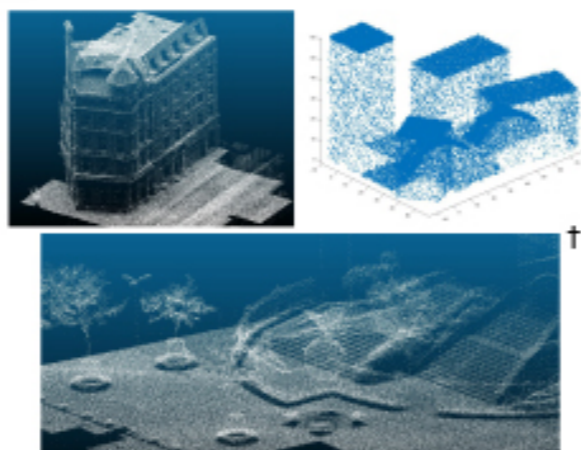
Example Activation Functions



Way too much text/detail

Processing LiDAR Data of Urban Infrastructure

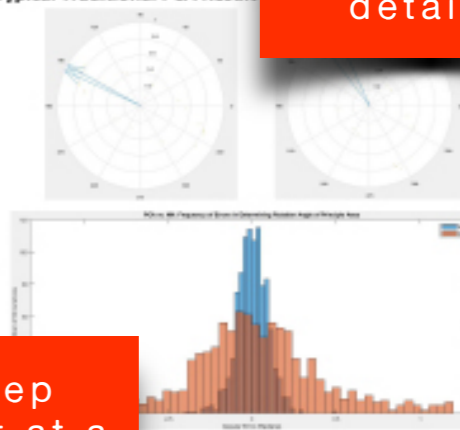
- Neural Networks are useful for their ability find relationships within large amounts of data and reduce the dimensionality of that data down to only what the data scientist is interested in. In other words, NNs can compress loads of uninterpretable data into small chunks of information that are useful.
- By flying over a city with LiDAR equipment, a point-cloud can be created which has thousands of points showing where light reflected off of the surface of some object, such as a building face or bench. We want to be able to take this data set of x, y, z coordinates and cluster/map them to the geometric features that they describe.
- For example, we want to take the thousands of points that lie on the face of one of the buildings and first separate them from the millions of other points in the dataset, then we want those points to be mapped to a finite plane that describes that side of the building. Repeating this process over the whole dataset will allow us to construct a usable 3D model of the entire city.
- LiDAR point cloud data of urban infrastructure is one such case where incredibly large datasets need to be reduced to much lower dimensionality while still maintaining all the information contained within the data. For this reason, NNs have a lot of potential in solving this problem.



Network-based PCA Approach

- Principle Component Analysis (PCA)** is a method of finding vectors in a data set which have maximum variance (i.e. the vectors along which the data is most dispersed).
- With regards to ellipses, performing PCA yields the directions of both the semi-major and semi-minor axes. In solving this problem, we are interested in the principle axes of ellipsoids.
- We have determined that ANNs are capable of performing PCA, although generally with less accuracy for isolated structures. The data shown at the right shows that the error distribution is wider for ANN-PCA. This is also illustrated by the typical results when applying PCA to ellipses; the two vectors in each image show the deviation of the calculated principle axes from the true principle axes.
- That being said, ANNs have a distinct advantage over traditional PCA computation which is important in processing the LiDAR data: NNs can discriminate between separate structures within given data while traditional PCA cannot.
- Generally, our intended method for processing the LiDAR data with an ANN is by:
 - 1) Cluster the points
 - 2) Use principal component analysis
 - 3) Cluster surfaces
 - 4) Average the results
 - 5) Repeat 3 and 4

Typical Traditional PCA Result



Too many topics! Keep focused on one project at a time. Maybe two.

Architecture Matching of (In)Homogeneous Diffusion Solutions

- The homogeneous diffusion equation is given by: $\frac{\partial u}{\partial t} = \kappa \nabla^2 u(r, t)$
- Where κ is a factor representing the ease of diffusion at every point in space and u is the scalar function which describes the magnitude of some physical quantity at every point in space and time (i.e. temperature). This simplified equation represents the special case where κ is constant over whatever domain is being considered.
- The most common way to solve this equation is by using separation of variables and solving for the eigenvalues and eigenmodes that span the entire domain. Unfortunately, it is computationally expensive to solve in this way when the boundary conditions change with time, as the eigenmodes must be recomputed over the entirety of the new domain at every timestep.
- For changing boundary condition situations, we can resort to an approximate solution which acts on a discrete domain called **cellular automata**. While not exact, this approach is computationally inexpensive when handling changing boundary conditions, and it yields a solution that has close resemblance to neural network architecture.
- This similarity allows us to match the architecture of a neural network to the cellular automata solution. From there, we can potentially train the neural network to find a better approximation that has equal computational cost. In the inhomogeneous case, we can potentially use the network to extract the κ landscape from data describing the time evolution of a diffusive system. This has applications in manufacturing and materials science for its ability to reveal imperfections in metals based on how heat diffuses through them.

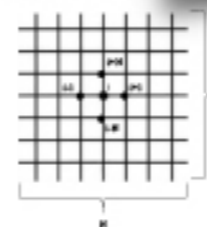
Cellular Automata Approach

- Performing separation of variables yields 2 equations which describe the spatial and time evolution of the system separately.

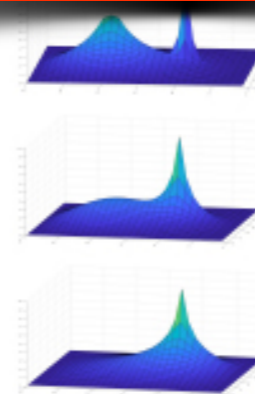
$$\nabla^2 \psi(r) = -\frac{\lambda}{\kappa} \psi(r) \quad \frac{\partial \phi}{\partial t} = -\lambda \phi(t)$$

- Suppose the state ψ exists in a discrete, 2D grid space where each grid point is separated by a small distance δ .
 - The $N \times N$ grid can be represented by an $N^2 \times 1$ vector if serialized by the index i .
- In this representation, the values, λ , are indexed by i , allowing us to evolve the system on site-by-site basis, solely based upon nearest neighbor interactions.
- We can approximate ∇^2 to 1st order as the discrete Laplacian

- While only an approximate solution, this method can easily handle changing boundary conditions because at each timestep, system evolution is computed by local interactions rather than over the entire domain.



- based on the initial conditions, we can then solve for λ_i and construct a matrix operator which evolves the state in discrete timesteps, as is shown to the right.



- The cellular automata operator's architecture closely resembles that of a neural network. We can directly import the operator matrix as the weight matrices of a neural network!

- We can add some random variation to the weights during training and hopefully steer the training toward a higher order, nonlinear solution. If we can accomplish this, it will yield a robust solution that is also relatively computationally inexpensive.

NN for Extracting K-Landscape

- In the inhomogeneous case, κ can take a different value at every point in space, and the systems evolution is dictated by: $\frac{\partial u}{\partial t} = \nabla \cdot (\kappa(r) \nabla u(r, t)) + \kappa(r) \nabla^2 u(r, t)$
- While much harder to reach, we have the cellular automata solution for the inhomogeneous case and can perform similar architecture matching to an ANN.
- Alternatively, using an encoder-decoder architecture, we predict that an ANN can encode the state of the system to a space which describes the κ -landscape. It should then be able to continue mapping to either the initial state or the state of the system at the next timestep.



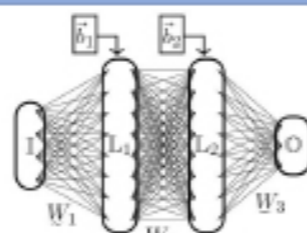
Physics-informed, Interpretable Machine Learning

Midshipman 2/C Nourachi
Professor Kevin McIlhany, Physics Department



Applying Machine Learning to Physics

- In traditional computing/algorithms, problem-solvers write a ruleset which generates a desired output from an input. In a machine learning (ML) approach, we want to construct an object that will find the rules for us which generate the desired output from a given input.
- There are many different forms of ML, including Artificial Neural Networks (ANN), Genetic Algorithms, Statistical Learning, Bayesian Inference, Gaussian Processes,
- Of the different forms of ML, Neural Networks show the most promise of being "interpretable" because they can generate structures which closely mirror traditional approaches. Comparing the ANN-generated architecture to the known traditionally-generated architecture allows us to understand what relationships the ANN is learning from the data it is being fed.



$$\begin{aligned}\bar{L}_1 &= f_1(W_1 \cdot \bar{I}_1 + \bar{b}_1) \\ \bar{L}_2 &= f_2(W_2 \cdot \bar{L}_1 + \bar{b}_2) \\ \bar{L}_3 &= f_3(W_3 \cdot \bar{L}_2 + \bar{b}_3) \\ &\vdots \\ \bar{O}_j &= f_n(W_n \cdot \bar{L}_{n-1} + \bar{b}_n)\end{aligned}$$

Images obtained from tech note - 01

Neural Net Design

- Layers** are mappings from an input vector to an output vector. The individual components of the output vector are referred to as nodes. A neural network can be composed of a single layer, but linking layers together allows for more complex mappings. Layer size is determined by the number of nodes that comprise the layer.
- Weights and biases** are the functional components of a layer which act on the input vector to produce an output for the next layer. Every node of a layer has a weighted connection to every node of the layer in front of it. Mathematically, this takes the form of a matrix of weights that are applied to the input vector, resulting in a linear transformation. Biases are applied to each node of the network to create thresholding behavior.
- Transfer/Activation Functions**, shown to the right, are wrapper functions that each node is passed through to add nonlinearity to the net. Some are referred to as squashing functions because they compress all nodes to a value between a small range (e.g. tanh compresses output between -1 and 1).
- A **Loss/Cost/Energy Function** quantifies how much error exists between the output of the neural net and the desired output. It takes various forms based on the type of learning that is imposed on the net (e.g. supervised vs. unsupervised). Commonly used in supervised learning is the mean squared error function, where \bar{O} refers to an output vector of the net and \bar{T} refers to the target vector that was desired:
$$MSE = \frac{1}{n} \sum_{i=1}^n (\bar{O}_i - \bar{T}_i)^2$$

Learning in a ANN equates to minimizing the loss function by tweaking all of the weights within the layers.

- For our purposes, the loss function is of particular interest because physical rules can be demanded of the ANN by addition of a meaningful **regularization term**. For example, we can demand that a quantity within the data be conserved and allow the neural net find the best mapping which does not violate that condition.
- A **Training/Optimization function** is the mathematical method by which the loss function is minimized. There are many to choose from, such as random changes of weights or a more methodical gradient descent algorithm.

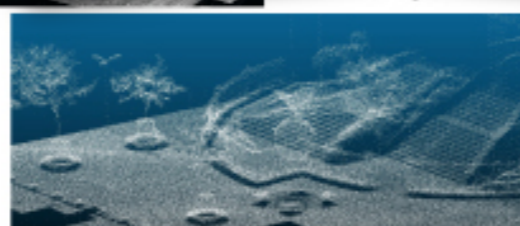
Example Activation Functions



Way too much text/detail

Processing LiDAR Data of Urban Infrastructure

- Neural Networks are useful for their ability find relationships within large amounts of data and reduce the dimensionality of that data down to only what the data scientist is interested in. In other words, NNs can compress loads of uninterpretable data into small chunks of information that are useful.
- By flying over a city with LiDAR equipment, a point-cloud can be created which has thousands of points showing where light reflected off of the surface of some object, such as a building face or bench. We want to be able to take this data set of x, y, z coordinates and cluster/map them to the geometric features that they describe.
- For example, we want to take the thousands of points that lie on the face of one of the buildings and first separate them from the millions of other points in the dataset, then we want those points to be mapped to a finite plane that describes that side of the building. Repeating this process over the whole dataset will allow us to construct a usable 3D model of the entire city.
- LiDAR point cloud data of urban infrastructure is one such case where incredibly large datasets need to be reduced to much lower dimensionality while still maintaining all the information contained within the data. For this reason, NNs have a lot of potential in solving this problem.

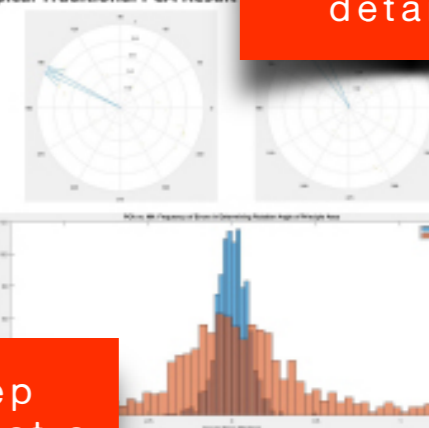


Is a full math-y definition of a generic neural network architecture helping anyone understand?

Principal Component Analysis (PCA) Approach

- Principal Component Analysis (PCA)** is a method of finding vectors in a data set which have maximum variance along which the data is most dispersed.
- Performing PCA yields the directions of both the semi-major and semi-minor axes. In this case, we are interested in the principle axes of ellipsoids.
- ANNs are capable of performing PCA, although generally with less accuracy. The data shown at the right shows that the error distribution is skewed. The two vectors in each image show the deviation of the calculated principle axes from the true principle axes.
- That being said, ANNs have a distinct advantage over traditional PCA computation which is important in processing the LiDAR data: NNs can discriminate between separate structures within given data while traditional PCA cannot.
- Generally, our intended method for processing the LiDAR data with an ANN is to:
 - 1) Cluster the points
 - 2) Use principal component analysis
 - 3) Cluster surface
 - 4) Average the results
 - 5) Repeat 3 and 4

Typical Traditional PCA Results



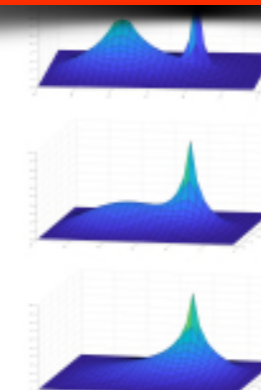
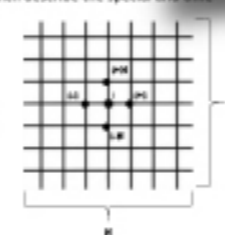
Too many topics! Keep focused on one project at a time. Maybe two.

Architecture Matching of (In)Homogeneous Diffusion Solutions

- The homogeneous diffusion equation is given by: $\frac{\partial u}{\partial t} = \kappa \nabla^2 u(r, t)$
- Where κ is a factor representing the ease of diffusion at every point in space and u is the scalar function which describes the magnitude of some physical quantity at every point in space and time [i.e. temperature]. This simplified equation represents the special case where κ is constant over whatever domain is being considered.
- The most common way to solve this equation is by using separation of variables and solving for the eigenvalues and eigenmodes that span the entire domain. Unfortunately, it is computationally expensive to solve in this way when the boundary conditions change with time, as the eigenmodes must be recomputed over the entirety of the new domain at every timestep.
- For changing boundary condition situations, we can resort to an approximate solution which acts on a discrete domain called **cellular automata**. While not exact, this approach is computationally inexpensive when handling changing boundary conditions, and it yields a solution that has close resemblance to neural network architecture.
- This similarity allows us to match the architecture of a neural network to the cellular automata solution. From there, we can potentially train the neural network to find a better approximation that has equal computational cost. In the inhomogeneous case, we can potentially use the network to extract the κ landscape from data describing the time evolution of a diffusive system. This has applications in manufacturing and materials science for its ability to reveal imperfections in metals based on how heat diffuses through them.

Cellular Automata Approach

- Performing separation of variables yields 2 equations which describe the spatial and time evolution of the system separately.
$$\nabla^2 \psi(r) = -\frac{\lambda}{\kappa} \psi(r) \quad \frac{\partial \phi}{\partial t} = -\lambda \phi(t)$$
- Suppose the state ψ exists in a discrete, 2D grid space where each grid point is separated by a small distance δ .
 The $N \times N$ grid can be represented by an $N^2 \times 1$ vector if serialized by the index i .
 In this representation, the values, λ_i , are indexed by i , allowing us to evolve the system on site-by-site basis, solely based upon nearest neighbor interactions.
- We can approximate ∇^2 to 1st order as the discrete Laplacian
- While only an approximate solution, this method can easily handle changing boundary conditions because at each timestep, system evolution is computed by local interactions rather than over the entire domain.
- based on the initial conditions, we can then solve for λ_i and construct a matrix operator which evolves the state in discrete timesteps, as is shown to the right.



- The cellular automata operator's architecture closely resembles that of a neural network. We can directly import the operator matrix as the weight matrices of a neural network!
- We can add some random variation to the weights during training and hopefully steer the training toward a higher order, nonlinear solution. If we can accomplish this, it will yield a robust solution that is also relatively computationally inexpensive.
- NN for Extracting K-Landscape**
- In the inhomogeneous case, κ can take a different value at every point in space, and the systems evolution is dictated by:
$$\frac{\partial u}{\partial t} = \nabla \cdot (\kappa(r) \nabla u(r, t)) + \lambda(r) \nabla^2 u(r, t)$$
- While much harder to reach, we have the cellular automata solution for the inhomogeneous case and can perform similar architecture matching to an ANN.
- Alternatively, using an encoder-decoder architecture, we predict that an ANN can encode the state of the system to a space which describes the κ -landscape. It should then be able to continue mapping to either the initial state or the state of the system at the next timestep.

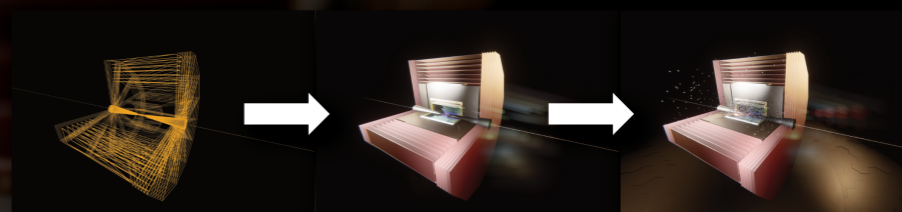
Cellular Automata → ANN

MUON DETECTOR VISUALIZATION IN UNREAL ENGINE

Aims

Compelling visuals are central to communicating with collaborators, the public, and for our own understanding. One of our goals is to create beautiful displays of muon collider elements: the detector, the collisions, and the machine.

We wanted to interface our muon collider and HEP tools with **state of the art 3D rendering engines** such as Blender and Unreal Engine 5 using industry standard formats. This gives us the ability to build high quality visuals with the latest techniques built by the \$200B video game industry, with built-in rendering and camera effects, such as depth of field, ambient occlusion, chromatic aberration, etc, in a highly performant application. **This gives a uniquely immersive event display framework that has utility for our research and capturing the public's imagination.**



Detector Geometry

The existing software framework defines a detector geometry in DD4hep's "compact" XML format which is converted to a ROOT/GEANT4-compatible format for the simulation workflow. We've built a new workflow that converts this same geometry description into an **industry standard 3D model GLTF format** that can then be edited in Blender as needed.

These 3D models are then imported into **Unreal Engine** for high quality rendering and interactive application development.

The highly intuitive representation of the detector has already led to **multiple bug fixes** in the geometry and allows for a **concrete sense of scale** for the detector that numbers in an XML file were unable to convey.



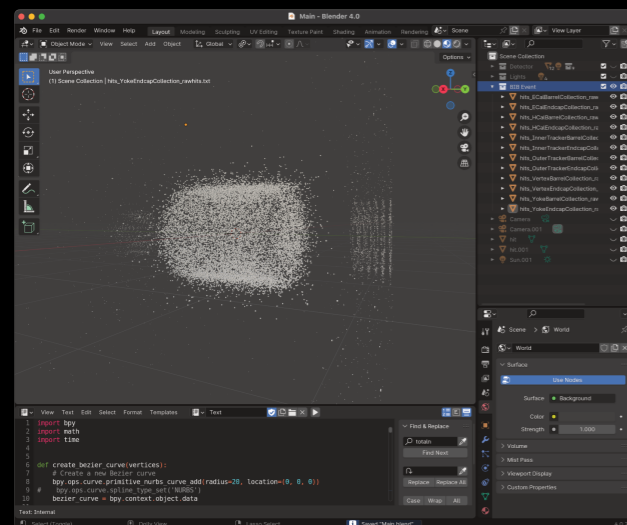
Detector Geometry in dd compact XML Format

Convert to GDML using dd tools

Convert to industry standard GLTF w/ Custom Scripts using pyG4ometry

Geometry Editing in Blender (Open Source 3D Modeling Package)

Import Edited GLTF into Unreal Engine For Final Lighting, Rendering, and Interactivity



Muon Collider software stack can write out event info in EDM4HEP format

Use uproot-based scripts to extract event data

Use Blender Python bindings to create 3d objects representing detector or reconstructed objects, output to GLTF

Import GLTF into Unreal Engine For Final Lighting, Rendering, and Interactivity

Event Data Conversion

Muon Collider software stack supports outputting to the **ROOT-readable EDM4HEP format**. We read in these files using uproot to process collision event information. The locations of hits are turned into 3D mesh objects using Blender's built-in python interface. These meshes can be then **exported into GLTF** for importing into Unreal Engine.

This process is currently manual and improvements are necessary, especially to let this framework scale to more complicated events. Building out an **configurable, automated tool to turn EDM4HEP files into GLTF** meshes remains an open task and will eventually allow for an Unreal Engine application that has the ability to import EDM4HEP files and directly render events for the user.

Current Output

Setup can create **rendered stills** to visualize the detector, detector signals, or physics objects. Stills are useful for outreach, talks, posters, etc and can be customized to individual messages.

The **3D models** are even supported in tools like Keynote for engaging animations.

Custom executable contains an **interactive 3D interface** into the virtual detector. Currently implemented as a playable third person game, which has the added benefit of giving a human character for scale.



Recently featured on cover of *Science*, March 20, 2024

Open Tasks

- Polish up current application with **menus and UI**
- Build EDM4hep-reading capabilities into application for **automated event mesh construction**
- Implement **collision animations**
- **Geometry conversion** workflow improvements
- Design **interactive game**
- **VR/AR/XR development** for Quest 3 or Apple Vision Pro

} Initial conversations with Drexel's Entrepreneurial Game Studio

New collaborations very welcome!
Opportunities for work with digital media programs, etc, and to build out a unique outreach program

LAWRENCE LEE, CHARLES BELL

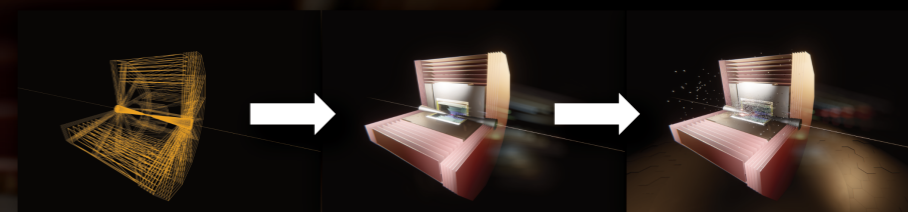
MUON DETECTOR VISUALIZATION IN UNREAL ENGINE

THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

Aim
Note the
generous
spacing

Communicating with collaborators, the public, and for our own understanding. One of our goals is to create a visualization of the detector elements: the detector, the collisions, and the machine.

We wanted to create a visualization of the detector and HEP tools with **state of the art 3D rendering engines** such as Blender and Unreal Engine 5 using industry standard formats. This gives us the ability to build high quality visuals with the latest techniques built by the \$200B video game industry, with built-in rendering and camera effects, such as depth of field, ambient occlusion, chromatic aberration, etc, in a highly performant application. **This gives a uniquely immersive event display framework that has utility for our research and capturing the public's imagination.**



Detector Geometry

The existing software framework defines a detector geometry in DD4hep's "compact" XML format which is converted to a ROOT/GEANT4-compatible format for the simulation workflow. We've built a new workflow that converts this same geometry description into an **industry standard 3D model GLTF format** that can then be edited in Blender as needed.

These 3D models are then imported into **Unreal Engine** for high quality rendering and interactive application development.

The highly intuitive representation of the detector has already led to **multiple bug fixes** in the geometry and allows for a **concrete sense of scale** for the detector that numbers in an XML file were unable to convey.

DD4hep

pyG4ometry

Blender

UNREAL ENGINE

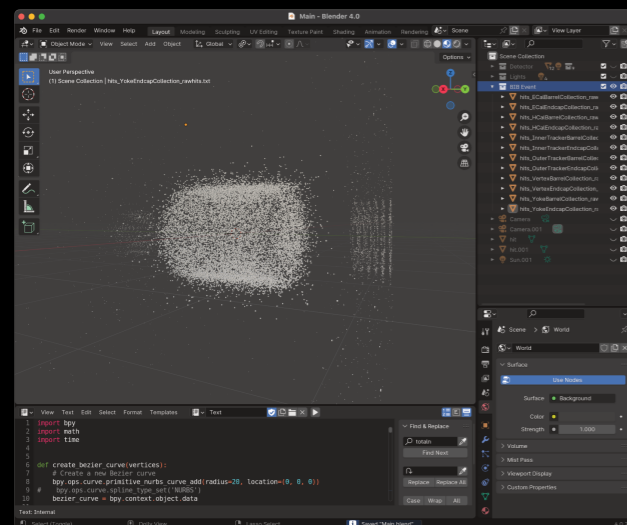
Detector Geometry in dd compact XML Format

Convert to GDML using dd tools

Convert to industry standard GLTF w/ Custom Scripts using pyG4ometry

Geometry Editing in Blender (Open Source 3D Modeling Package)

Import Edited GLTF into Unreal Engine For Final Lighting, Rendering, and Interactivity



HEP 4

uproot

Blender

UNREAL ENGINE

Muon Collider software stack can write out event info in EDM4HEP format

Use uproot-based scripts to extract event data

Use Blender Python bindings to create 3d objects representing detector or reconstructed objects, output to GLTF

Import GLTF into Unreal Engine For Final Lighting, Rendering, and Interactivity

Event Data Conversion

Muon Collider software stack supports outputting to the **ROOT-readable EDM4HEP format**. We read in these files using uproot to process collision event information. The locations of hits are turned into 3D mesh objects using Blender's built-in python interface. These meshes can be then **exported into GLTF** for importing into Unreal Engine.

This process is currently manual and improvements are necessary, especially to let this framework scale to more complicated events. Building out an **configurable, automated tool to turn EDM4HEP files into GLTF** meshes remains an open task and will eventually allow for an Unreal Engine application that has the ability to import EDM4HEP files and directly render events for the user.

Current Output

Setup can create **rendered stills** to visualize the detector, detector signals, or physics objects. Stills are useful for outreach, talks, posters, etc and can be customized to individual messages.

The **3D models** are even supported in tools like Keynote for engaging animations.

Custom executable contains an **interactive 3D interface** into the virtual detector. Currently implemented as a playable third person game, which has the added benefit of giving a human character for scale.



Recently featured on cover of Science, March 20, 2024

Open Tasks

- Polish up current application with **menus and UI**
- Build EDM4hep-reading capabilities into application for **automated event mesh construction**
- Implement **collision animations**
- **Geometry conversion** workflow improvements
- Design **interactive game**
- **VR/AR/XR development** for Quest 3 or Apple Vision Pro

} Initial conversations with Drexel's Entrepreneurial Game Studio

New collaborations very welcome!
Opportunities for work with digital media programs, etc, and to build out a unique outreach program

This work has been supported by the Department of Energy, Office of Science, under Grant No. DE-SC0023321 and the National Science Foundation, under Award No. 2235028.

FERMILAB-POSTER-24-0228-V, Aug 2024

LAWRENCE LEE, CHARLES BELL

MUON DETECTOR VISUALIZATION IN UNREAL ENGINE

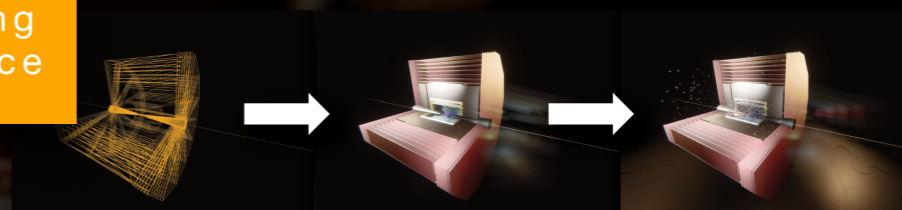
THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

Aim
Note the
generous
spacing

Communicating with collaborators, the public, and for our own understanding. One of our goals is to create a visualization of the detector elements: the detector, the collisions, and the machine.

We want to use the state of the art 3D rendering engines such as Blender and Unreal Engine 5 using industry standard formats. This gives us the ability to build high quality visuals with the latest techniques built by the \$200B video game industry, with built-in rendering and camera effects, such as depth of field, ambient occlusion, chromatic aberration, etc, in a highly performant application. This gives a uniquely immersive event display framework that has utility for our research and capturing the public's imagination.

Unique looking
from a distance



Detector Geometry

The existing software framework defines a detector geometry in DD4hep's "compact" XML format which is converted to a ROOT/GEANT4-compatible format for the simulation workflow. We've built a new workflow that converts this same geometry description into an **industry standard 3D model GLTF format** that can then be edited in Blender as needed.

These 3D models are then imported into **Unreal Engine** for high quality rendering and interactive application development.

The highly intuitive representation of the detector has already led to **multiple bug fixes** in the geometry and allows for a **concrete sense of scale** for the detector that numbers in an XML file were unable to convey.

DD4hep

pyG4ometry

Blender

UNREAL ENGINE

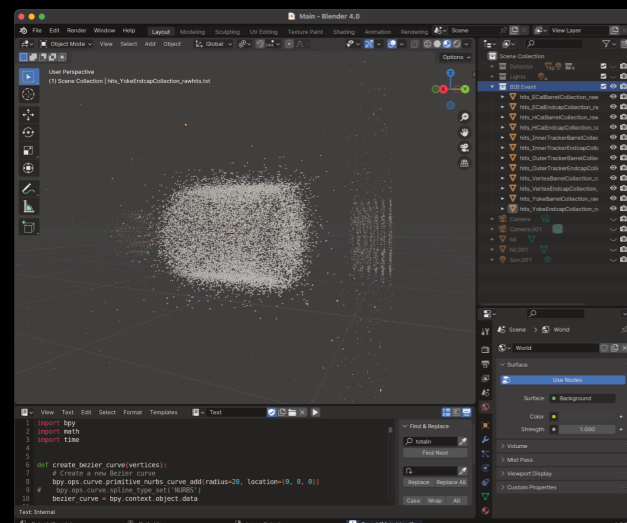
Detector Geometry in dd compact XML Format

Convert to GDML using dd tools

Convert to industry standard GLTF w/ Custom Scripts using pyG4ometry

Geometry Editing in Blender (Open Source 3D Modeling Package)

Import Edited GLTF into Unreal Engine For Final Lighting, Rendering, and Interactivity



Event Data Conversion

Muon Collider software stack supports outputting to the **ROOT-readable EDM4HEP format**. We read in these files using uproot to process collision event information. The locations of hits are turned into 3D mesh objects using Blender's built-in python interface. These meshes can be then **exported into GLTF** for importing into Unreal Engine.

This process is currently manual and improvements are necessary, especially to let this framework scale to more complicated events. Building out an **configurable, automated tool to turn EDM4HEP files into GLTF** meshes remains an open task and will eventually allow for an Unreal Engine application that has the ability to import EDM4HEP files and directly render events for the user.

HEP 4

uproot

Blender

UNREAL ENGINE

Muon Collider software stack can write out event info in EDM4HEP format

Use uproot-based scripts to extract event data

Use Blender Python bindings to create 3d objects representing detector or reconstructed objects, output to GLTF

Import GLTF into Unreal Engine For Final Lighting, Rendering, and Interactivity

Current Output

Setup can create **rendered stills** to visualize the detector, detector signals, or physics objects. Still are useful for outreach, talks, posters, etc and can be customized to individual messages.

The **3D models** are even supported in tools like Keynote for engaging animations.

Custom executable contains an **interactive 3D interface** into the virtual detector. Currently implemented as a playable third person game, which has the added benefit of giving a human character for scale.



Recently featured on cover of Science, March 20, 2024

Open Tasks

- Polish up current application with **menus and UI**
- Build EDM4hep-reading capabilities into application for **automated event mesh construction**
- Implement **collision animations**
- **Geometry conversion** workflow improvements
- Design **interactive game**
- **VR/AR/XR development** for Quest 3 or Apple Vision Pro

} Initial conversations with Drexel's Entrepreneurial Game Studio

New collaborations very welcome!
Opportunities for work with digital media programs, etc, and to build out a unique outreach program

This work has been supported by the Department of Energy, Office of Science, under Grant No. DE-SC0023321 and the National Science Foundation, under Award No. 2235028.

FERMILAB-POSTER-24-0228-V, Aug 2024

LAWRENCE LEE, CHARLES BELL

MUON DETECTOR VISUALIZATION IN UNREAL ENGINE

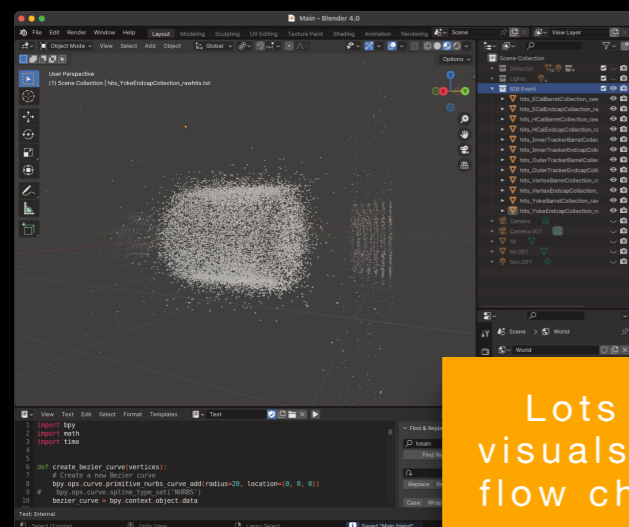
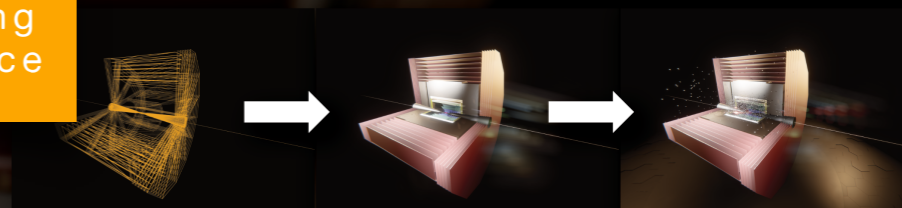
THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

Aim
Note the
generous
spacing

Communicating with collaborators, the public, and for our own understanding. One of our goals is to create a visualization of the detector elements: the detector, the collisions, and the machine.

We want to create a visualization of the detector and HEP tools with **state of the art 3D rendering engines** such as Blender and Unreal Engine 5 using industry standard formats. This gives us the ability to build high quality visuals with the latest techniques built by the \$200B video game industry, with built-in rendering and camera effects, such as depth of field, ambient occlusion, chromatic aberration, etc, in a highly performant application. **This gives a uniquely immersive event display framework that has utility for our research and capturing the public's imagination.**

Unique looking
from a distance



Lots of
visuals and
flow charts

Detector Geometry

The existing software framework defines a detector geometry in DD4hep's "compact" XML format which is converted to a ROOT/GEANT4-compatible format for the simulation workflow. We've built a new workflow that converts this same geometry description into an **industry standard 3D model GLTF format** that can then be edited in Blender as needed.

These 3D models are then imported into **Unreal Engine** for high quality rendering and interactive application development.

The highly intuitive representation of the detector has already led to **multiple bug fixes** in the geometry and allows for a **concrete sense of scale** for the detector that numbers in an XML file were unable to convey.

DD4hep

pyG4ometry

Blender

UNREAL ENGINE

Detector Geometry in dd compact XML Format

Convert to GDML using dd tools

Convert to industry standard GLTF w/ Custom Scripts using pyG4ometry

Geometry Editing in Blender (Open Source 3D Modeling Package)

Import Edited GLTF into Unreal Engine For Final Lighting, Rendering, and Interactivity

HEP 4

uproot

Blender

UNREAL ENGINE

Muon Collider software stack can write out event info in EDM4HEP format

Use uproot-based scripts to extract event data

Use Blender Python bindings to create 3d objects representing detector or reconstructed objects, output to GLTF

Import GLTF into Unreal Engine For Final Lighting, Rendering, and Interactivity

Event Data Conversion

Muon Collider software stack supports outputting to the **ROOT-readable EDM4HEP format**. We read in these files using uproot to process collision event information. The locations of hits are turned into 3D mesh objects using Blender's built-in python interface. These meshes can be then **exported into GLTF** for importing into Unreal Engine.

This process is currently manual and improvements are necessary, especially to let this framework scale to more complicated events. Building out an **configurable, automated tool to turn EDM4HEP files into GLTF** meshes remains an open task and will eventually allow for an Unreal Engine application that has the ability to import EDM4HEP files and directly render events for the user.

Current Output

Setup can create **rendered stills** to visualize the detector, detector signals, or physics objects. Still are useful for outreach, talks, posters, etc and can be customized to individual messages.

The **3D models** are even supported in tools like Keynote for engaging animations.

Custom executable contains an **interactive 3D interface** into the virtual detector. Currently implemented as a playable third person game, which has the added benefit of giving a human character for scale.



Recently featured on cover of
Science, March 20, 2024

Open Tasks

- Polish up current application with **menus and UI**
- Build EDM4hep-reading capabilities into application for **automated event mesh construction**
- Implement **collision animations**
- **Geometry conversion** workflow improvements
- Design **interactive game**
- **VR/AR/XR development** for Quest 3 or Apple Vision Pro

} Initial conversations with
Drexel's Entrepreneurial
Game Studio

New collaborations very welcome!
Opportunities for work with digital media programs, etc,
and to build out a unique outreach program

This work has been supported by the Department of Energy, Office of Science, under Grant No. DE-SC0023321 and the National Science Foundation, under Award No. 2235028.

FERMILAB-POSTER-24-0228-V, Aug 2024

LAWRENCE LEE, CHARLES BELL

MUON DETECTOR VISUALIZATION IN UNREAL ENGINE

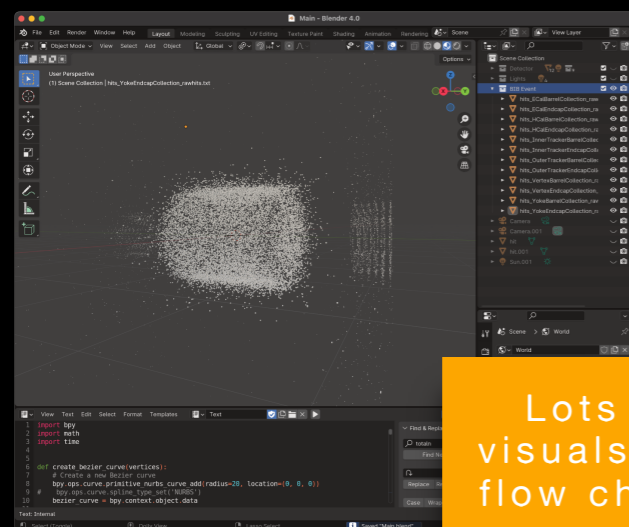
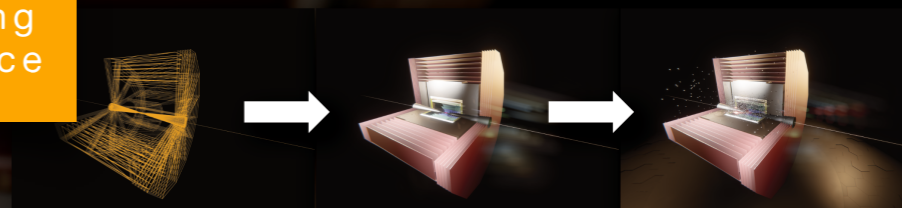
THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

Aim
Note the
generous
spacing

Communicating with collaborators, the public, and for our own understanding. One of our goals is to create a visualization of the detector elements: the detector, the collisions, and the machine.

We want to create a visualization of the detector and HEP tools with **state of the art 3D rendering engines** such as Blender and Unreal Engine 5 using industry standard formats. This gives us the ability to build high quality visuals with the latest techniques built by the \$200B video game industry, with built-in rendering and camera effects, such as depth of field, ambient occlusion, chromatic aberration, etc, in a highly performant application. **This gives a uniquely immersive event display framework that has utility for our research and capturing the public's imagination.**

Unique looking
from a distance



Lots of
visuals and
flow charts

Detector Geometry

The existing software framework defines a detector geometry in DD4hep's "compact" XML format which is converted to a ROOT/GEANT4-compatible format for the simulation workflow. We've built a new workflow that converts this same geometry description into an **industry standard 3D model GLTF format** that can then be edited in Blender as needed.

These 3D models are then imported into **Unreal Engine** for high quality rendering and interactive application development.

The highly intuitive representation of the detector has already led to **multiple bug fixes** in the geometry and allows for a **concrete sense of scale** for the detector that numbers in an XML file were unable to convey.

DD4hep

pyG4ometry

Blender

UNREAL ENGINE

Detector Geometry in dd compact XML Format

Convert to GDML using dd tools

Convert to industry standard GLTF w/ Custom Scripts using pyG4ometry

Geometry Editing in Blender (Open Source 3D Modeling Package)

Import Edited GLTF into Unreal Engine For Final Lighting, Rendering, and Interactivity

HEP 4

uproot

Blender

UNREAL ENGINE

Muon Collider software stack can write out event info in EDM4HEP format

Use uproot-based scripts to extract event data

Use Blender Python bindings to create 3d objects representing detector or reconstructed objects, output to GLTF

Import GLTF into Unreal Engine For Final Lighting, Rendering, and Interactivity

Event Data Conversion

Muon Collider software stack supports outputting to the **ROOT-readable EDM4HEP format**. We read in these files using uproot to process collision event information. The locations of hits are turned into 3D mesh objects using Blender's built-in python interface. These meshes can be then **exported into GLTF** for importing into Unreal Engine.

This process is currently manual and improvements are necessary, especially to let this framework scale to more complicated events. Building out an **configurable, automated tool to turn EDM4HEP files into GLTF** meshes remains an open task and will eventually allow for an Unreal Engine application that has the ability to import EDM4HEP files and directly render events for the user.

Hierarchy of scale
→ Hierarchy of
attention

Open Tasks

- Polish up current application with **menus and UI**
- Build EDM4hep-reading capabilities into application for **automated event mesh construction**
- Implement **collision animations**
- **Geometry conversion** workflow improvements
- Design **interactive game**
- **VR/AR/XR development** for Quest 3 or Apple Vision Pro

} Initial conversations with Drexel's Entrepreneurial Game Studio

Current Output

Setup can create **rendered stills** to visualize the detector, detector signals, or physics objects. Still are useful for outreach, talks, posters, etc and can be customized to individual messages.

The **3D models** are even supported in tools like Keynote for engaging animations.

Custom executable contains an **interactive 3D interface** into the virtual detector. Currently implemented as a playable third person game, which has the added benefit of giving a human character for scale.



Recently featured on cover of Science, March 20, 2024

New collaborations very welcome!
Opportunities for work with digital medial programs, etc, and to build out a unique outreach program

This work has been supported by the Department of Energy, Office of Science, under Grant No. DE-SC0023321 and the National Science Foundation, under Award No. 2235028.

FERMILAB-POSTER-24-0228-V, Aug 2024

LAWRENCE LEE, CHARLES BELL

MUON DETECTOR VISUALIZATION IN UNREAL ENGINE

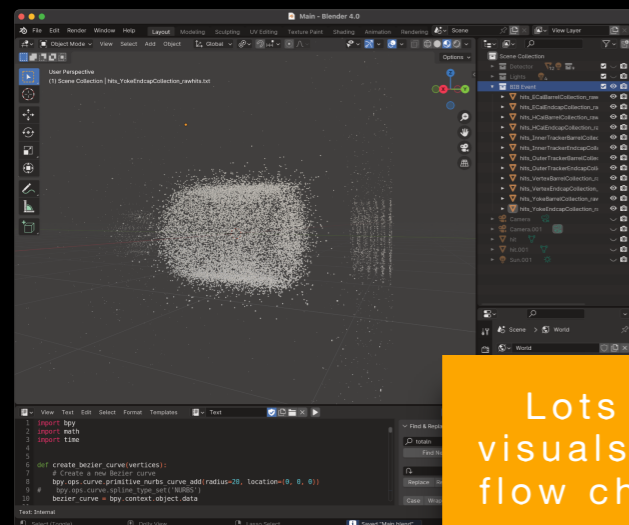
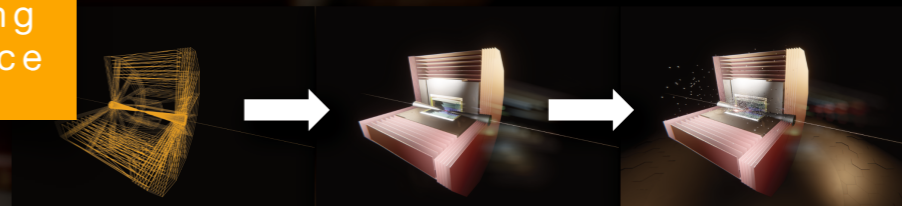
THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

Aim
Note the
generous
spacing

Communicating with collaborators, the public, and for our own understanding. One of our goals is to create a visualization of the detector elements: the detector, the collisions, and the machine.

We want to create a visualization of the detector and HEP tools with **state of the art 3D rendering engines** such as Blender and Unreal Engine 5 using industry standard formats. This gives us the ability to build high quality visuals with the latest techniques built by the \$200B video game industry, with built-in rendering and camera effects, such as depth of field, ambient occlusion, chromatic aberration, etc, in a highly performant application. **This gives a uniquely immersive event display framework that has utility for our research and capturing the public's imagination.**

Unique looking
from a distance



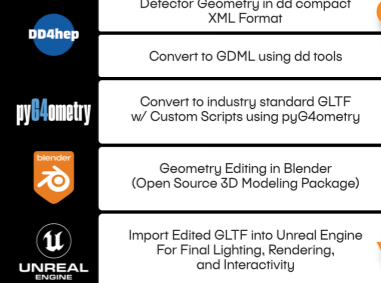
Lots of
visuals and
flow charts

Detector Geometry

The existing software framework defines a detector geometry in DD4hep's "compact" XML format which is converted to a ROOT/GEANT4-compatible format for the simulation workflow. We've built a new workflow that converts this same geometry description into an **industry standard 3D model GLTF format** that can then be edited in Blender as needed.

These 3D models are then imported into **Unreal Engine** for high quality rendering and interactive application development.

The highly intuitive representation of the detector has already led to **multiple bug fixes** in the geometry and allows for a **concrete sense of scale** for the detector that numbers in an XML file were unable to convey.



Event Data Conversion

Muon Collider software stack supports outputting to the **ROOT-readable EDM4HEP format**. We read in these files using uproot to process collision event information. The locations of hits are turned into 3D mesh objects using Blender's built-in python interface. These meshes can be then **exported into GLTF** for importing into Unreal Engine.

This process is currently manual and improvements are necessary, especially to let this framework scale to more complicated events. Building out an **configurable, automated tool to turn EDM4HEP files into GLTF** meshes remains an open task and will eventually allow for an Unreal Engine application that has the ability to import EDM4HEP files and directly render events for the user.

Hierarchy of scale
→ Hierarchy of
attention

Open Tasks

- Polish up current application with **menus and UI**
- Build EDM4hep-reading capabilities into application for **automated event mesh construction**
- Implement **collision animations**
- **Geometry conversion** workflow improvements
- Design **interactive game**
- **VR/AR/XR development** for Quest 3 or Apple Vision Pro

} Initial conversations with
Drexel's Entrepreneurial
Game Studio

Current Output

Setup can create **rendered stills** to visualize the detector, detector signals, or physics objects. Stills are useful for outreach, talks, posters, etc and can be customized to individual messages.

The **3D models** are even supported in tools like Keynote for engaging animations.

Custom executable contains an **interactive 3D interface** into the virtual detector. Currently implemented as a playable third person game, which has the added benefit of giving a human character for scale.



Recently featured on cover of
Science, March 20, 2024

Be clear about
your takeaways

New collaborations very welcome!

Opportunities for work with digital media programs, etc,
and to build out a unique outreach program

This work has been supported by the Department of Energy, Office of Science, under Grant No. DE-SC0023321 and the National Science Foundation, under Award No. 2235028.

FERMILAB-POSTER-24-0228-V, Aug 2024

LAWRENCE LEE, CHARLES BELL

MUON DETECTOR VISUALIZATION IN UNREAL ENGINE

THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

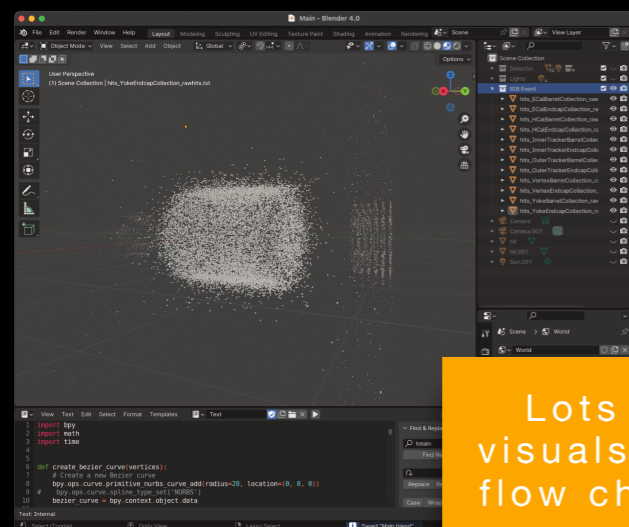
Aim
Compete
Note the
generous
spacing

Communicating with collaborators, the public, and for our own understanding. One of our goals is to create a visualization of the detector, the collisions, and the machine.

We want to communicate with collaborators and HEP tools with **state of the art 3D rendering engines** such as Blender and Unreal Engine 5 using industry standard formats. This gives us the ability to build high quality visuals with the latest techniques built by the \$200B video game industry, with built-in rendering and camera effects, such as depth of field, ambient occlusion, chromatic aberration, etc, in a highly performant application. **This gives a uniquely immersive event display framework that has utility for our research and capturing the public's imagination.**

Unique looking
from a distance

Some folks think you
shouldn't have an image
in the BG. Do whatever
serves communication
best.



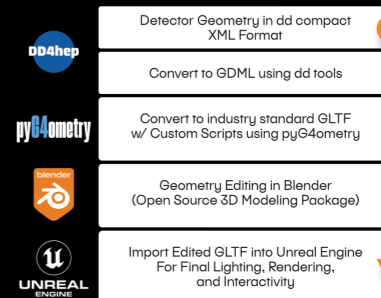
Lots of
visuals and
flow charts

Detector Geometry

The existing software framework defines a detector geometry in DD4hep's "compact" XML format which is converted to a ROOT/GEANT4-compatible format for the simulation workflow. We've built a new workflow that converts this same geometry description into an **industry standard 3D model GLTF format** that can then be edited in Blender as needed.

These 3D models are then imported into **Unreal Engine** for high quality rendering and interactive application development.

The highly intuitive representation of the detector has already led to **multiple bug fixes** in the geometry and allows for a **concrete sense of scale** for the detector that numbers in an XML file were unable to convey.



Event Data Conversion

Muon Collider software stack supports outputting to the **ROOT-readable EDM4HEP format**. We read in these files using uproot to process collision event information. The locations of hits are turned into 3D mesh objects using Blender's built-in python interface. These meshes can be then **exported into GLTF** for importing into Unreal Engine.

This process is currently manual and improvements are necessary, especially to let this framework scale to more complicated events. Building out an **configurable, automated tool to turn EDM4HEP files into GLTF** meshes remains an open task and will eventually allow for an Unreal Engine application that has the ability to import EDM4HEP files and directly render events for the user.

Hierarchy of scale
→ Hierarchy of
attention

Open Tasks

- Polish up current application with **menus and UI**
- Build EDM4hep-reading capabilities into application for **automated event mesh construction**
- Implement **collision animations**
- **Geometry conversion** workflow improvements
- Design **interactive game**
- **VR/AR/XR development** for Quest 3 or Apple Vision Pro

} Initial conversations with
Drexel's Entrepreneurial
Game Studio

Current Output

Setup can create **rendered stills** to visualize the detector, detector signals, or physics objects. Still are useful for outreach, talks, posters, etc and can be customized to individual messages.

The **3D models** are even supported in tools like Keynote for engaging animations.

Custom executable contains an **interactive 3D interface** into the virtual detector. Currently implemented as a playable third person game, which has the added benefit of giving a human character for scale.



Recently featured on cover of
Science, March 20, 2024

Be clear about
your takeaways

New collaborations very welcome!

Opportunities for work with digital media programs, etc,
and to build out a unique outreach program

This work has been supported by the Department of Energy, Office of Science, under Grant No. DE-SC0023321 and the National Science Foundation, under Award No. 2235028.

FERMILAB-POSTER-24-0228-V, Aug 2024

A Reinterpretation of an LHC Search for Displaced Vertices and Muons in RPV SUSY Models

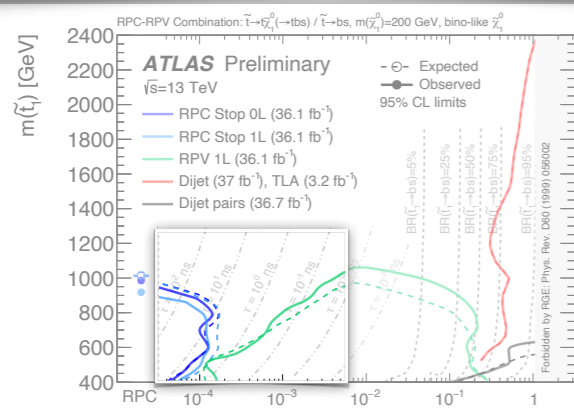
Taylor Sussmane ¹, Lawrence Lee ¹, Karri Folan DiPetrillo ²

¹ University of Tennessee, Knoxville TN

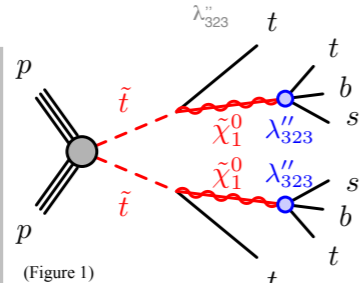
² Fermilab, Batavia IL

Motivation

- Supersymmetry (SUSY) is an extension of the Standard Model of Particle Physics (SM) that would solve some of the problems with the SM, by postulating a fermion/boson symmetry.
- Supersymmetry predicts a heavier partner (sparticle) for each of the SM particles.
- RPV SUSY models violate R-parity conservation, meaning that a SUSY particle could decay into SM particles.

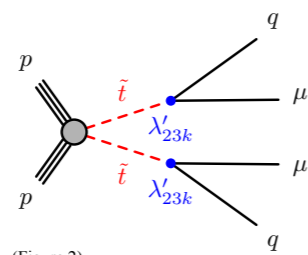
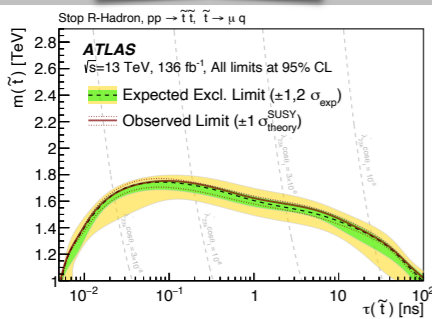


- Previous ATLAS analyses have obtained limits on the RPV SUSY model to the right [1], but there is a noticeable gap in the 10ps to 10ns proper lifetime range that should be explored with a search for long lived particles (LLP).
- We want to see if we have sensitivity to that range using a previous ATLAS search that requires a displaced vertex and muon.



(Figure 1)

ATLAS DV+Mu analysis

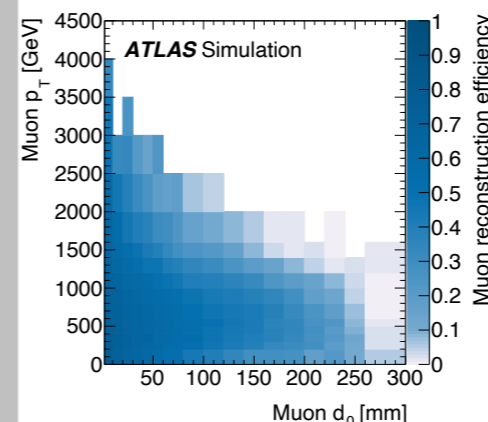
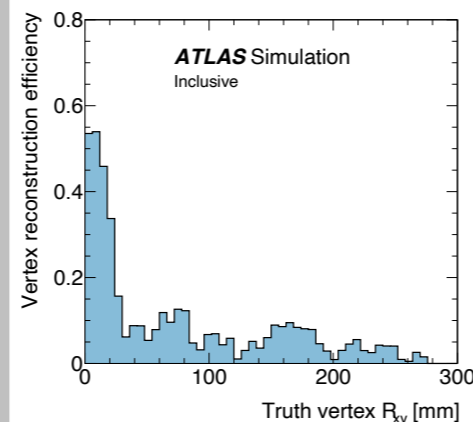


(Figure 2)

The ATLAS DV+Mu analysis [2] searched for the process above. It requires events with a displaced vertex and muon. The diagram in Figure 1 can give a displaced vertex and muon, through the decay of displaced top quarks. Luckily, that search had reinterpretation information that could be used to look for any process giving the same signature, like the one we're interested in!

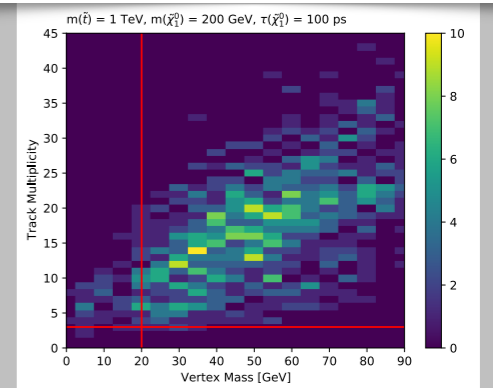
Reinterpretation Information

The ATLAS DV+Mu search provided generic reinterpretation information to be used for other models that have a displaced vertex and muon. That information was used on a grid of Monte Carlo simulation samples with various parameters for the target signal model. The material defined event and object acceptances, which would cut any events that do not meet certain truth criteria. For each event that passed the acceptance, efficiencies could be calculated. After the efficiency is calculated, a random number is thrown and the event may be rejected.



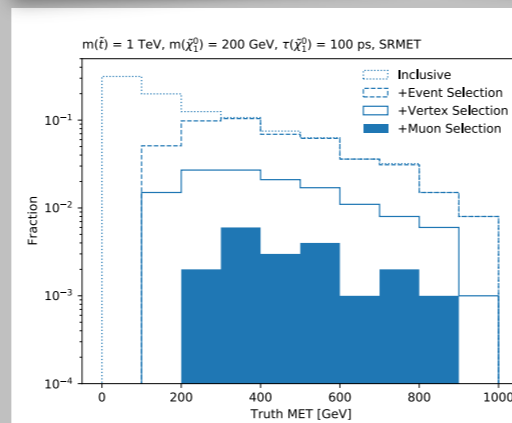
These plots give an example of how the efficiency for each cut was calculated. These histograms are from the reinterpretation information of the DV+Mu search. The application of these efficiencies is supposed to emulate the detection of signal in an LHC collision.

This plot shows the distribution of track multiplicities and vertex masses for a signal model. The signal region (SR) is the region in the top right. As shown, most of the signal resides in the SR, so loosening the cuts would not significantly change the results because the search had very little background.



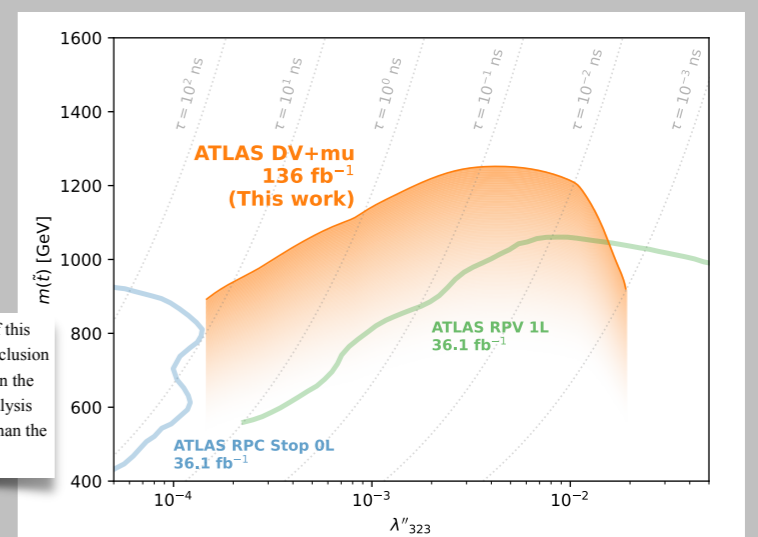
Results and Conclusions

After all the cuts, the number of surviving events is normalized based on luminosity and cross section, which is a function of stop mass. If the normalized number of surviving events is larger than the published BSM event yields (3 events), then those parameters (stop mass/neutralino lifetime) are excluded. Repeating this process over the entire grid of simulations, the exclusion contour can be traced out in the parameter space, as seen below.



This plot shows the MET distribution of the events that survive the various cuts made.

This plot shows the final results of this reinterpretation analysis. The new exclusion contour fills in all of the gap seen in the ATLAS Preliminary. This LLP analysis excludes about 200-350 GeV more than the ATLAS RPV 1L analysis.



References

- [1] ATLAS Collaboration, *Reinterpretation of searches for supersymmetry in models with variable R-parity-violating coupling strength and long-lived R-hadrons*, (2018) ATLAS-CONF-2018-003
- [2] ATLAS Collaboration, *Search for long-lived, massive particles in events with a displaced vertex and a muon with large impact parameter in pp collisions at sqrt(s)=13 TeV with the ATLAS detector*, Phys. Rev. D 102, 032006 (2020), arXiv:2003.11956



A Reinterpretation of an LHC Search for Displaced Vertices and Muons in RPV SUSY Models

Taylor Sussmane ¹, Lawrence Lee ¹, Karri Folan DiPetrillo ²

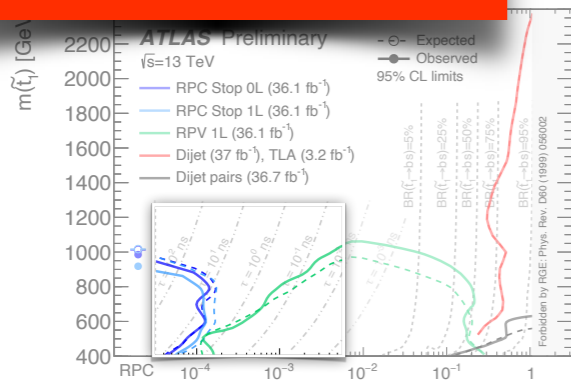
¹ University of Tennessee, Knoxville TN

² Fermilab, Batavia IL

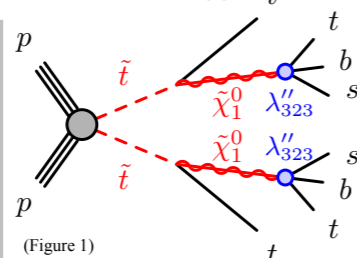
Motivation

- Supersymmetry (SUSY) is an extension of the Standard Model of Particle Physics (SM) that would solve some of the problems with the SM, by postulating a fermion/boson symmetry.
- Supersymmetry predicts a heavier partner (sparticle) for each of the SM particles.
- RPV SUSY models violate R-parity conservation, meaning that a SUSY particle could decay into SM

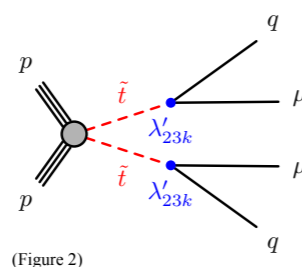
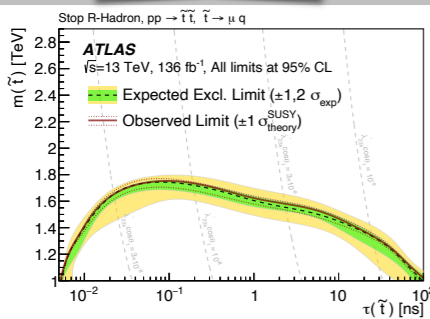
(Some cramped text)



- Previous ATLAS analyses have obtained limits on the RPV SUSY model to the right [1], but there is a noticeable gap in the 10ps to 10ns proper lifetime range that should be explored with a search for long lived particles (LLP).
- We want to see if we have sensitivity to that range using a previous ATLAS search that requires a displaced vertex and muon.



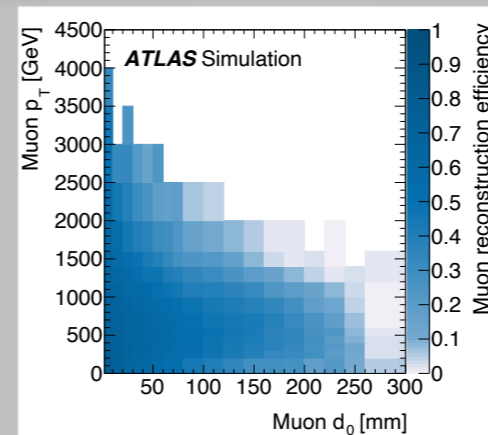
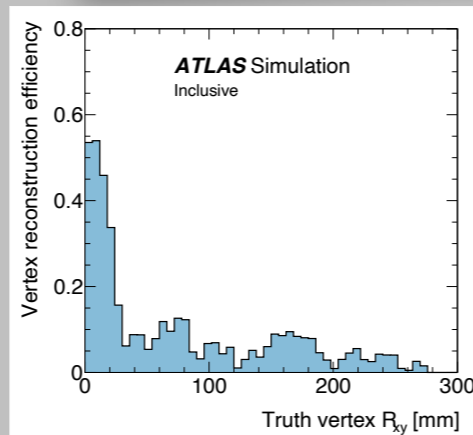
ATLAS DV+Mu analysis



The ATLAS DV+Mu analysis [2] searched for the process above. It requires events with a displaced vertex and muon. The diagram in Figure 1 can give a displaced vertex and muon, through the decay of displaced top quarks. Luckily, that search had reinterpretation information that could be used to look for any process giving the same signature, like the one we're interested in!

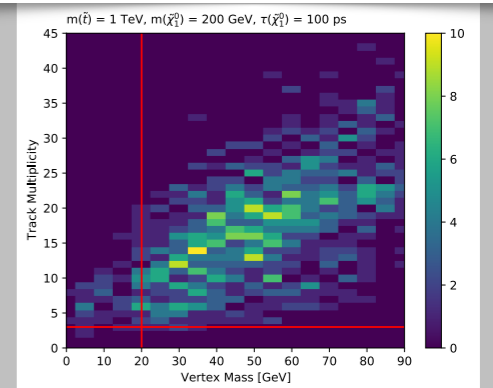
Reinterpretation Information

The ATLAS DV+Mu search provided generic reinterpretation information to be used for other models that have a displaced vertex and muon. That information was used on a grid of Monte Carlo simulation samples with various parameters for the target signal model. The material defined event and object acceptances, which would cut any events that do not meet certain truth criteria. For each event that passed the acceptance, efficiencies could be calculated. After the efficiency is calculated, a random number is thrown and the event may be rejected.



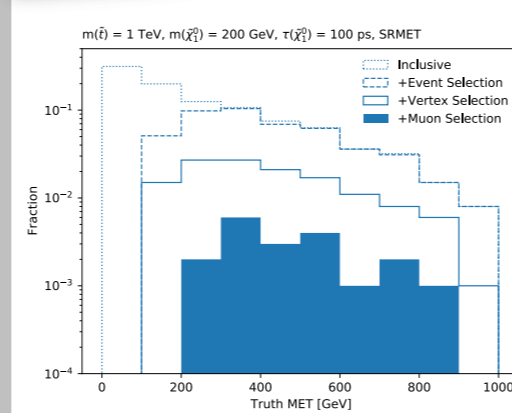
These plots give an example of how the efficiency for each cut was calculated. These histograms are from the reinterpretation information of the DV+Mu search. The application of these efficiencies is supposed to emulate the detection of signal in an LHC collision.

This plot shows the distribution of track multiplicities and vertex masses for a signal model. The signal region (SR) is the region in the top right. As shown, most of the signal resides in the SR, so loosening the cuts would not significantly change the results because the search had very little background.



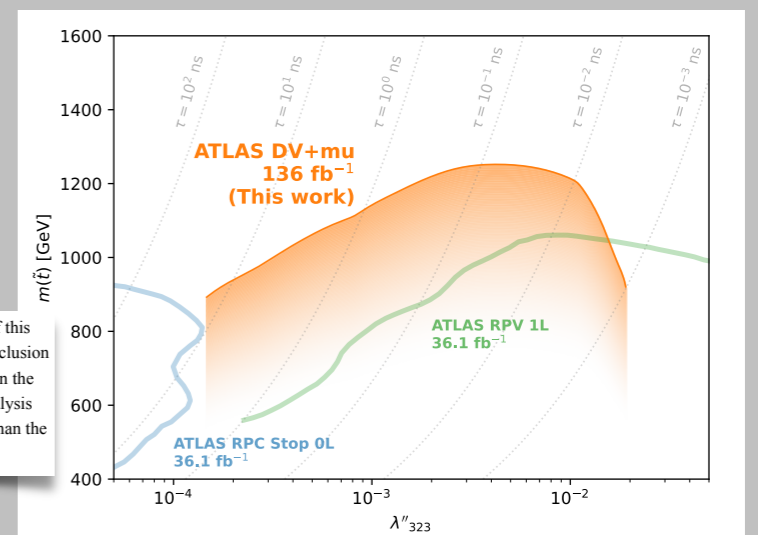
Results and Conclusions

After all the cuts, the number of surviving events is normalized based on luminosity and cross section, which is a function of stop mass. If the normalized number of surviving events is larger than the published BSM event yields (3 events), then those parameters (stop mass/neutralino lifetime) are excluded. Repeating this process over the entire grid of simulations, the exclusion contour can be traced out in the parameter space, as seen below.



This plot shows the MET distribution of the events that survive the various cuts made.

This plot shows the final results of this reinterpretation analysis. The new exclusion contour fills in all of the gap seen in the ATLAS Preliminary. This LLP analysis excludes about 200-350 GeV more than the ATLAS RPV 1L analysis.



References

- [1] ATLAS Collaboration, *Reinterpretation of searches for supersymmetry in models with variable R-parity-violating coupling strength and long-lived R-hadrons*, (2018) ATLAS-CONF-2018-003
- [2] ATLAS Collaboration, *Search for long-lived, massive particles in events with a displaced vertex and a muon with large impact parameter in pp collisions at sqrt{s}=13 TeV with the ATLAS detector*, Phys. Rev. D 102, 032006 (2020), arXiv:2003.11956



A Reinterpretation of an LHC Search for Displaced Vertices and Muons in RPV SUSY Models

Taylor Sussmane¹, Lawrence Lee¹, Karri Folan DiPetrillo²

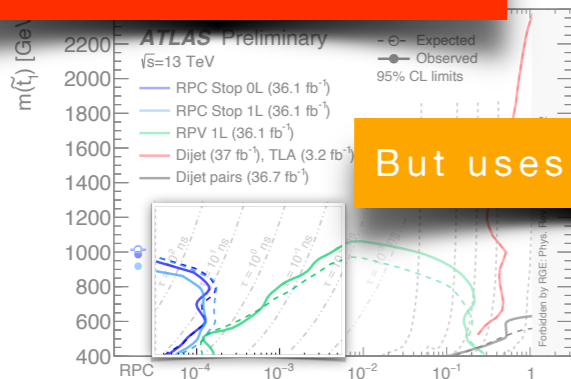
¹ University of Tennessee, Knoxville TN

² Fermilab, Batavia IL

Motivation

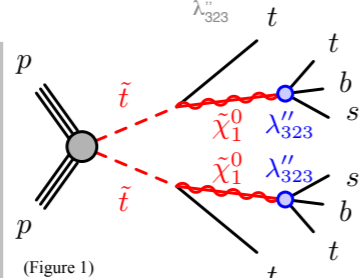
- Supersymmetry (SUSY) is an extension of the Standard Model of Particle Physics (SM) that would solve some of the problems with the SM, by postulating a fermion/boson symmetry.
- Supersymmetry predicts a heavier partner (sparticle) for each of the SM particles.
- RPV SUSY models violate R-parity conservation, meaning that a SUSY particle could decay into SM

(Some cramped text)

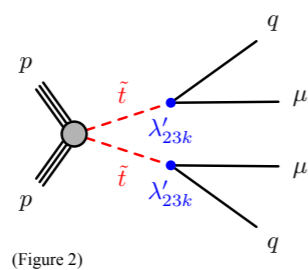
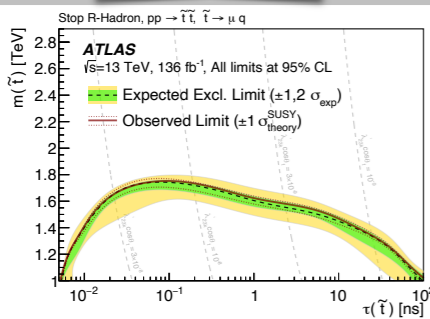


But uses the z-axis!

- Previous ATLAS analyses have obtained limits on the RPV SUSY model to the right [1], but there is a noticeable gap in the 10ps to 10ns proper lifetime range that should be explored with a search for long lived particles (LLP).
- We want to see if we have sensitivity to that range using a previous ATLAS search that requires a displaced vertex and muon.



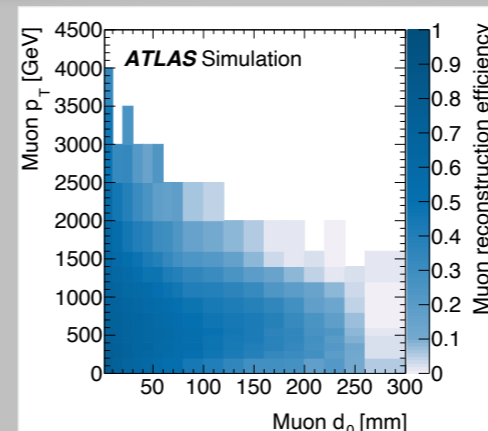
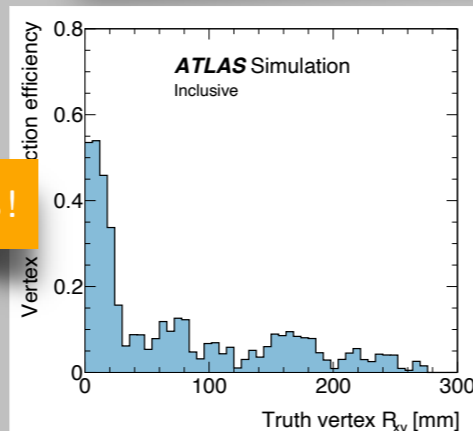
ATLAS DV+Mu analysis



The ATLAS DV+Mu analysis [2] searched for the process above. It requires events with a displaced vertex and muon. The diagram in Figure 1 can give a displaced vertex and muon, through the decay of displaced top quarks. Luckily, that search had reinterpretation information that could be used to look for any process giving the same signature, like the one we're interested in!

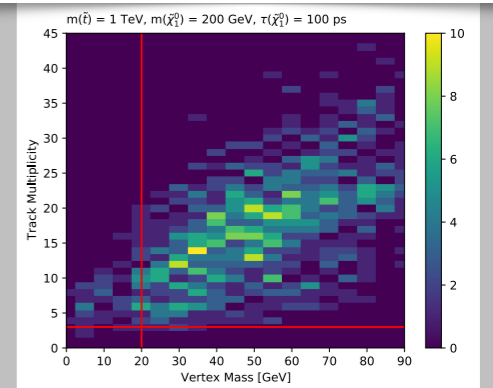
Reinterpretation Information

The ATLAS DV+Mu search provided generic reinterpretation information to be used for other models that have a displaced vertex and muon. That information was used on a grid of Monte Carlo simulation samples with various parameters for the target signal model. The material defined event and object acceptances, which would cut any events that do not meet certain truth criteria. For each event that passed the acceptance, efficiencies could be calculated. After the efficiency is calculated, a random number is thrown and the event may be rejected.



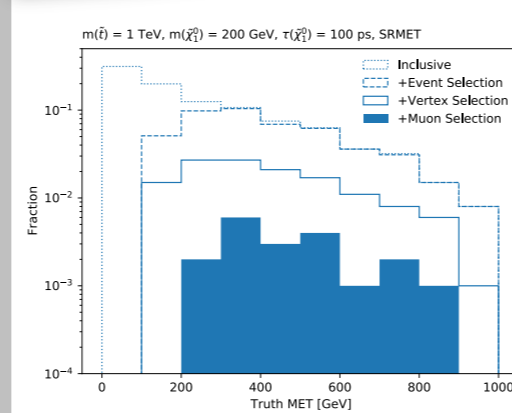
These plots give an example of how the efficiency for each cut was calculated. These histograms are from the reinterpretation information of the DV+Mu search. The application of these efficiencies is supposed to emulate the detection of signal in an LHC collision.

This plot shows the distribution of track multiplicities and vertex masses for a signal model. The signal region (SR) is the region in the top right. As shown, most of the signal resides in the SR, so loosening the cuts would not significantly change the results because the search had very little background.



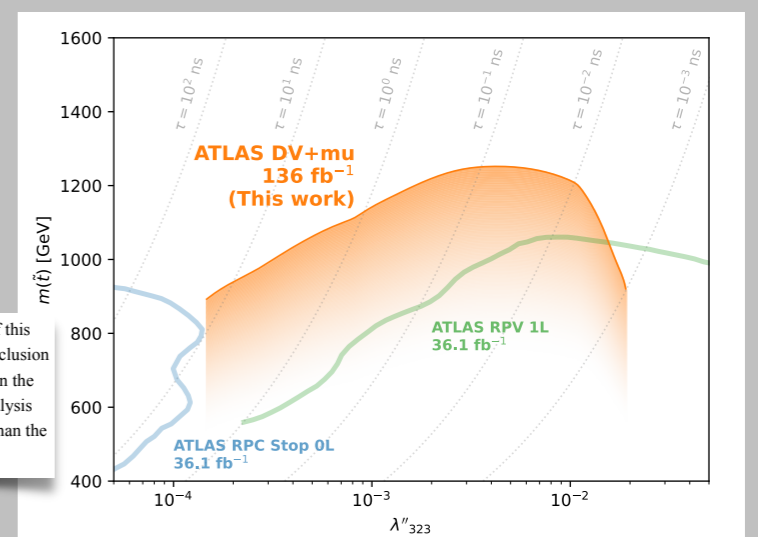
Results and Conclusions

After all the cuts, the number of surviving events is normalized based on luminosity and cross section, which is a function of stop mass. If the normalized number of surviving events is larger than the published BSM event yields (3 events), then those parameters (stop mass/neutralino lifetime) are excluded. Repeating this process over the entire grid of simulations, the exclusion contour can be traced out in the parameter space, as seen below.



This plot shows the MET distribution of the events that survive the various cuts made.

This plot shows the final results of this reinterpretation analysis. The new exclusion contour fills in all of the gap seen in the ATLAS Preliminary. This LLP analysis excludes about 200-350 GeV more than the ATLAS RPV 1L analysis.



References

- [1] ATLAS Collaboration, *Reinterpretation of searches for supersymmetry in models with variable R-parity-violating coupling strength and long-lived R-hadrons*, (2018) ATLAS-CONF-2018-003
- [2] ATLAS Collaboration, *Search for long-lived, massive particles in events with a displaced vertex and a muon with large impact parameter in pp collisions at $\sqrt{s}=13$ TeV with the ATLAS detector*, Phys. Rev. D 102, 032006 (2020), arXiv:2003.11956



A Reinterpretation of an LHC Search for Displaced Vertices and Muons in RPV SUSY Models

Taylor Sussmane ¹, Lawrence Lee ¹, Karri Folan DiPetrillo ²

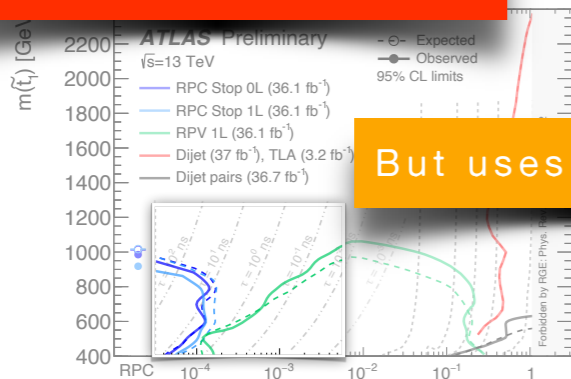
¹ University of Tennessee, Knoxville TN

² Fermilab, Batavia IL

Motivation

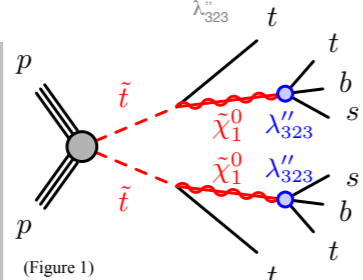
- Supersymmetry (SUSY) is an extension of the Standard Model of Particle Physics (SM) that would solve some of the problems with the SM, by postulating a fermion/boson symmetry.
- Supersymmetry predicts a heavier partner (sparticle) for each of the SM particles.
- RPV SUSY models violate R-parity conservation, meaning that a SUSY particle could decay into SM

(Some cramped text)

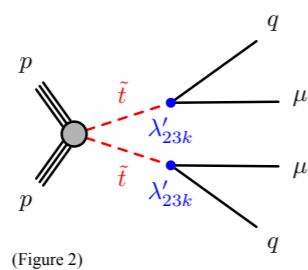
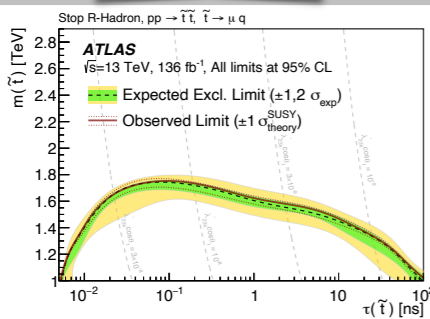


But uses the z-axis!

- Previous ATLAS analyses have obtained limits on the RPV SUSY model to the right [1], but there is a noticeable gap in the 10ps to 10ns proper lifetime range that should be explored with a search for long lived particles (LLP).
- We want to see if we have sensitivity to that range using a previous ATLAS search that requires a displaced vertex and muon.



ATLAS DV+Mu analysis

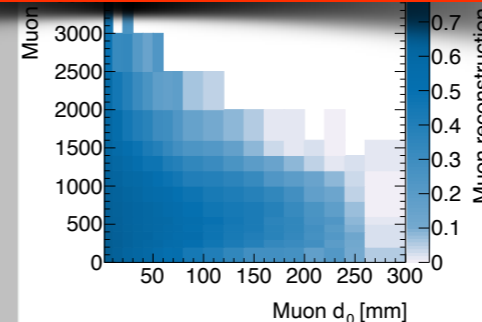
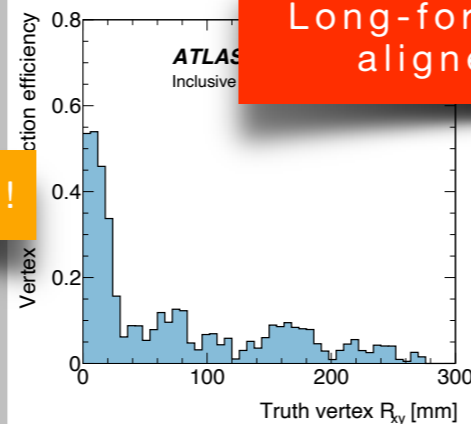


The ATLAS DV+Mu analysis [2] searched for the process above. It requires events with a displaced vertex and muon. The diagram in Figure 1 can give a displaced vertex and muon, through the decay of displaced top quarks. Luckily, that search had reinterpretation information that could be used to look for any process giving the same signature, like the one we're interested in!

Reinterpretation Information

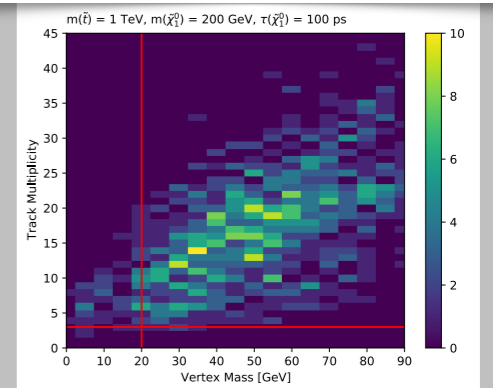
The ATLAS DV+Mu search provided generic reinterpretation information to be used for other models that have a displaced vertex and muon. That information was used on a grid of Monte Carlo simulation samples with various parameters for the target signal model. The material defined event and object acceptances, which would cut any events that do not meet certain truth criteria. For each event that passed the acceptance, efficiencies could be calculated. After the efficiency is calculated, a random number is thrown and the event may be rejected.

Long-form text should be left aligned or justified-left



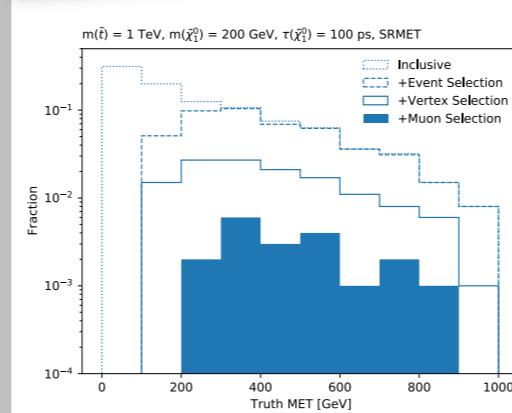
These plots give an example of how the efficiency for each cut was calculated. These histograms are from the reinterpretation information of the DV+Mu search. The application of these efficiencies is supposed to emulate the detection of signal in an LHC collision.

This plot shows the distribution of track multiplicities and vertex masses for a signal model. The signal region (SR) is the region in the top right. As shown, most of the signal resides in the SR, so loosening the cuts would not significantly change the results because the search had very little background.



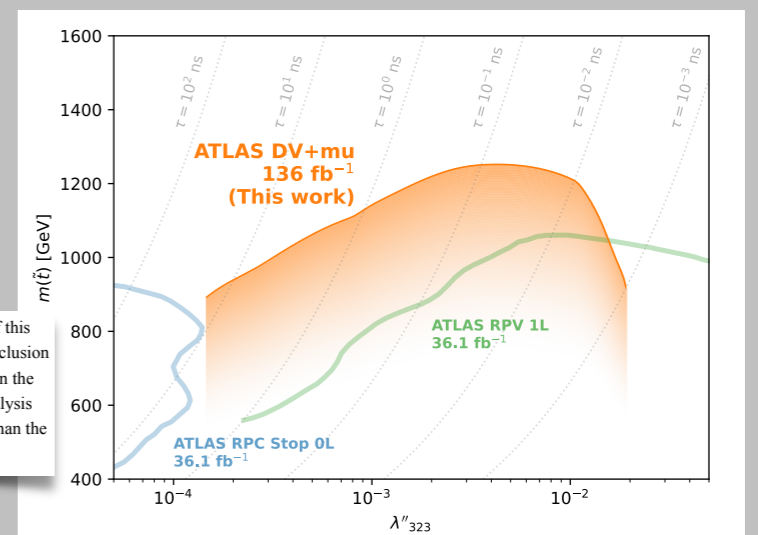
Results and Conclusions

After all the cuts, the number of surviving events is normalized based on luminosity and cross section, which is a function of stop mass. If the normalized number of surviving events is larger than the published BSM event yields (3 events), then those parameters (stop mass/neutralino lifetime) are excluded. Repeating this process over the entire grid of simulations, the exclusion contour can be traced out in the parameter space, as seen below.



This plot shows the MET distribution of the events that survive the various cuts made.

This plot shows the final results of this reinterpretation analysis. The new exclusion contour fills in all of the gap seen in the ATLAS Preliminary. This LLP analysis excludes about 200-350 GeV more than the ATLAS RPV 1L analysis.



References

- [1] ATLAS Collaboration, *Reinterpretation of searches for supersymmetry in models with variable R-parity-violating coupling strength and long-lived R-hadrons*, (2018) ATLAS-CONF-2018-003
- [2] ATLAS Collaboration, *Search for long-lived, massive particles in events with a displaced vertex and a muon with large impact parameter in pp collisions at sqrt(s)=13 TeV with the ATLAS detector*, Phys. Rev. D 102, 032006 (2020), arXiv:2003.11956



A Reinterpretation of an LHC Search for Displaced Vertices and Muons in RPV SUSY Models

Taylor Sussmane¹, Lawrence Lee¹, Karri Folan DiPetrillo²

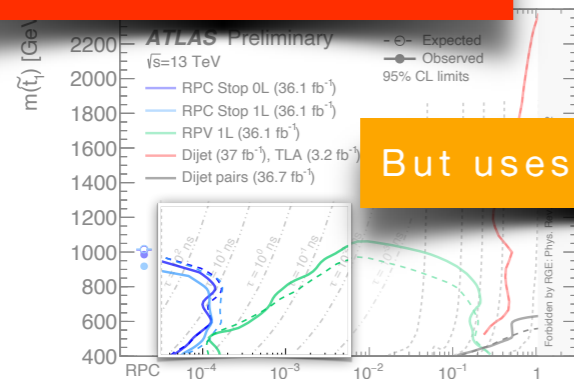
¹ University of Tennessee, Knoxville TN

² Fermilab, Batavia IL

Motivation

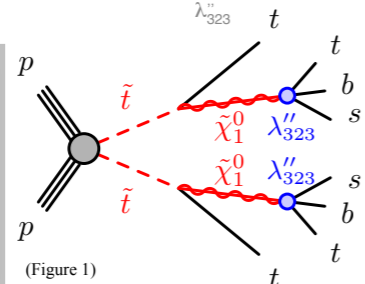
- Supersymmetry (SUSY) is an extension of the Standard Model of Particle Physics (SM) that would solve some of the problems with the SM, by postulating a fermion/boson symmetry.
- Supersymmetry predicts a heavier partner (sparticle) for each of the SM particles.
- RPV SUSY models violate R-parity conservation, meaning that a SUSY particle could decay into SM

(Some cramped text)

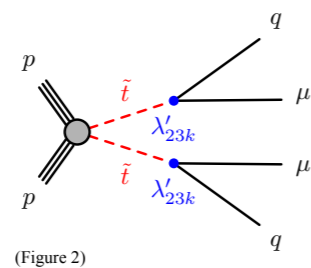
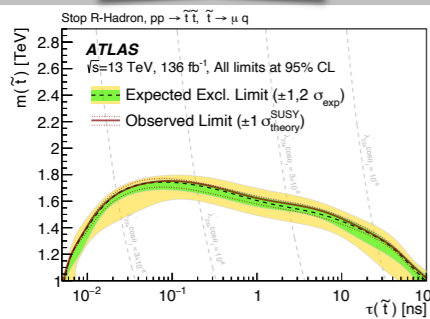


But uses the z-axis!

- Previous ATLAS analyses have obtained limits on the RPV SUSY model to the right [1], but there is a noticeable gap in the 10ps to 10ns proper lifetime range that should be explored with a search for long lived particles (LLP).
- We want to see if we have sensitivity to that range using a previous ATLAS search that requires a displaced vertex and muon.



ATLAS DV+Mu analysis

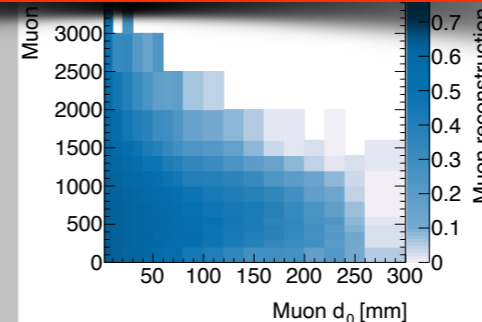
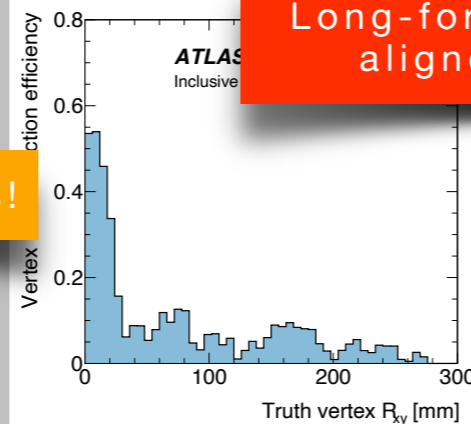


The ATLAS DV+Mu analysis [2] searched for the process above. It requires events with a displaced vertex and muon. The diagram in Figure 1 can give a displaced vertex and muon, through the decay of displaced top quarks. Luckily, that search had reinterpretation information that could be used to look for any process giving the same signature, like the one we're interested in!

Reinterpretation Information

The ATLAS DV+Mu search provided generic reinterpretation information to be used for other models that have a displaced vertex and muon. That information was used on a grid of Monte Carlo simulation samples with various parameters for the target signal model. The material defined event and object acceptances, which would cut any events that do not meet certain truth criteria. For each event that passed the acceptance, efficiencies could be calculated. After the efficiency is calculated, a random number is thrown and the event may be rejected.

Long-form text should be left aligned or justified-left

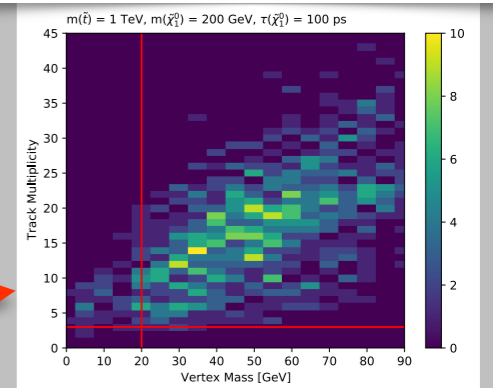


These plots give an example of how to reinterpret information of the emulate

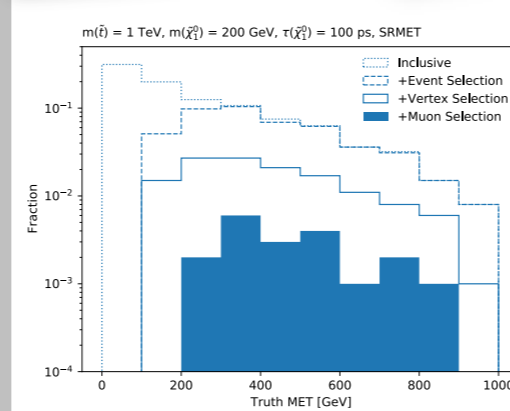
Unmotivated edges should be minimized

→ Give it a border, shadow, or make the BG match to make it disappear

This plot shows the distribution of track multiplicities and vertex masses for a signal model. The signal region (SR) is the region in the top right. As shown, most of the signal resides in the SR, so loosening the cuts would not significantly change the results because the search had very little background.

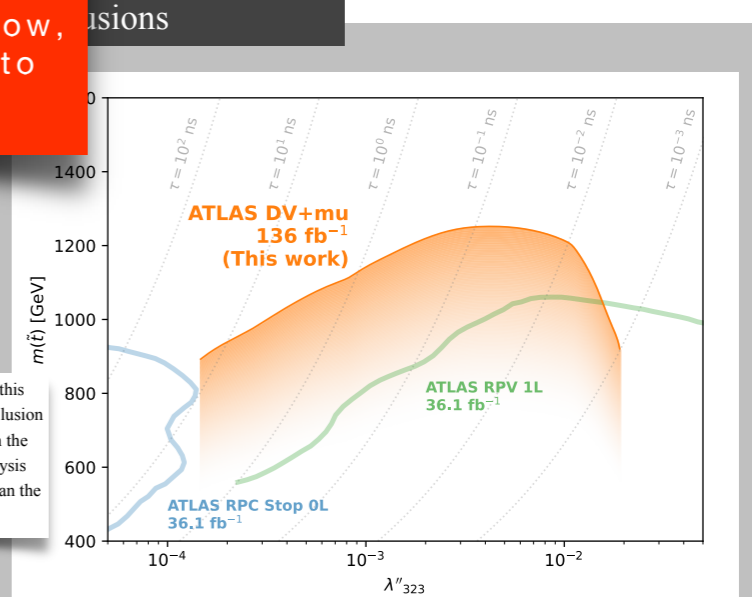


After all the cuts, the number of surviving events is a function of stop mass. If the normalized number of yields (3 events), then those parameters (stop mass/tau) are excluded. In the entire grid of simulations, the exclusion contour can be traced out in the parameter space as seen below.



This plot shows the MET distribution of the events that survive the various cuts made.

This plot shows the final results of this reinterpretation analysis. The new exclusion contour fills in all of the gap seen in the ATLAS Preliminary. This LLP analysis excludes about 200-350 GeV more than the ATLAS RPV 1L analysis.



References

- [1] ATLAS Collaboration, *Reinterpretation of searches for supersymmetry in models with variable R-parity-violating coupling strength and long-lived R-hadrons*, (2018) ATLAS-CONF-2018-003
- [2] ATLAS Collaboration, *Search for long-lived, massive particles in events with a displaced vertex and a muon with large impact parameter in pp collisions at sqrt(s)=13 TeV with the ATLAS detector*, Phys. Rev. D 102, 032006 (2020), arXiv:2003.11956



A Reinterpretation of an LHC Search for Displaced Vertices and Muons in RPV SUSY Models

Taylor Sussmane¹, Lawrence Lee¹, Karri Folan DiPetrillo²

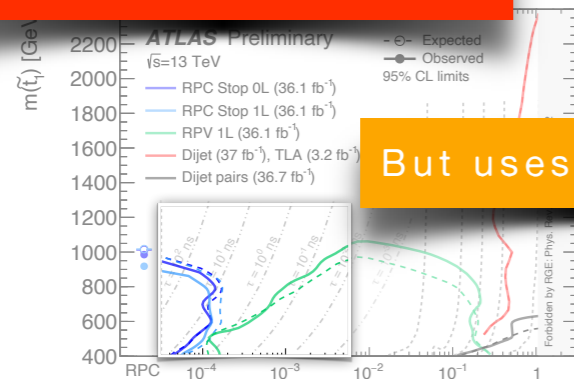
¹ University of Tennessee, Knoxville TN

² Fermilab, Batavia IL

Motivation

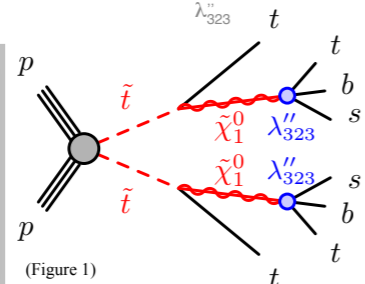
- Supersymmetry (SUSY) is an extension of the Standard Model of Particle Physics (SM) that would solve some of the problems with the SM, by postulating a fermion/boson symmetry.
- Supersymmetry predicts a heavier partner (sparticle) for each of the SM particles.
- RPV SUSY models violate R-parity conservation, meaning that a SUSY particle could decay into SM

(Some cramped text)

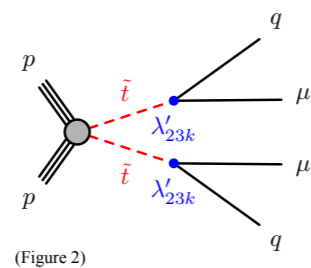
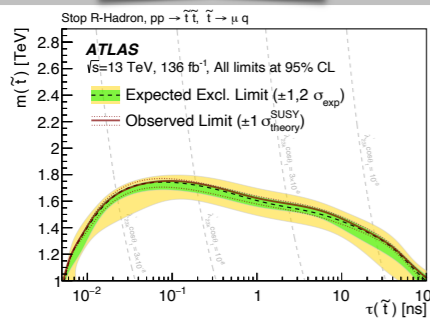


But uses the z-axis!

- Previous ATLAS analyses have obtained limits on the RPV SUSY model to the right [1], but there is a noticeable gap in the 10ps to 10ns proper lifetime range that should be explored with a search for long lived particles (LLP).
- We want to see if we have sensitivity to that range using a previous ATLAS search that requires a displaced vertex and muon.



ATLAS DV+Mu analysis

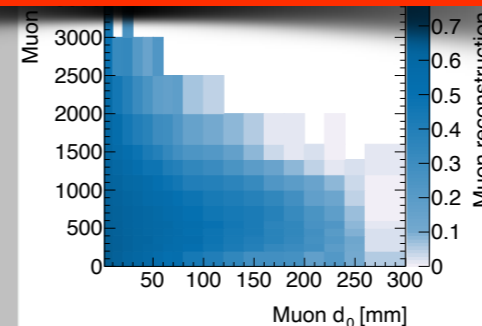
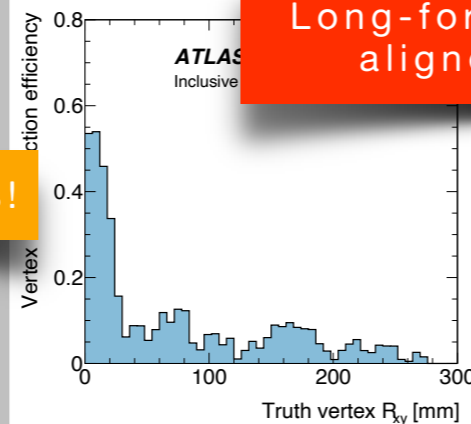


The ATLAS DV+Mu analysis [2] searched for the process above. It requires events with a displaced vertex and muon. The diagram in Figure 1 can give a displaced vertex and muon, through the decay of displaced top quarks. Luckily, that search had reinterpretation information that could be used to look for any process giving the same signature, like the one we're interested in!

Reinterpretation Information

The ATLAS DV+Mu search provided generic reinterpretation information to be used for other models that have a displaced vertex and muon. That information was used on a grid of Monte Carlo simulation samples with various parameters for the target signal model. The material defined event and object acceptances, which would cut any events that do not meet certain truth criteria. For each event that passed the acceptance, efficiencies could be calculated. After the efficiency is calculated, a random number is thrown and the event may be rejected.

Long-form text should be left aligned or justified-left

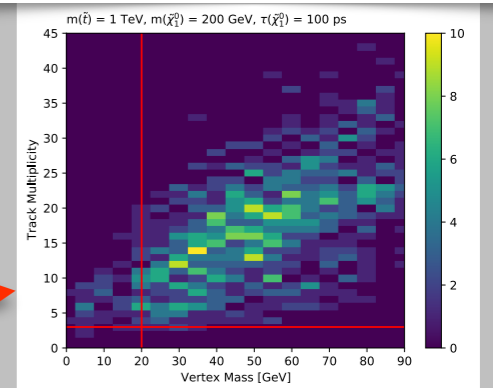


These plots give an example of how to reinterpret information of the emulate

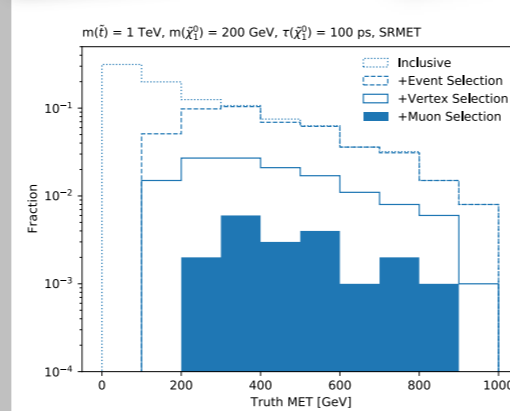
Unmotivated edges should be minimized

→ Give it a border, shadow, or make the BG match to make it disappear

This plot shows the distribution of track multiplicities and vertex masses for a signal model. The signal region (SR) is the region in the top right. As shown, most of the signal resides in the SR, so loosening the cuts would not significantly change the results because the search had very little background.



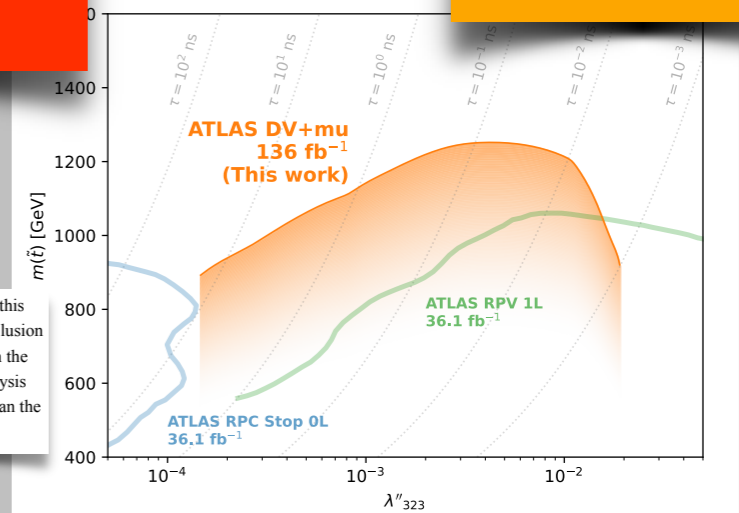
After all the cuts, the number of surviving events is a function of stop mass. If the normalized number of yields (3 events), then those parameters (stop mass/tau) in the entire grid of simulations, the exclusion contour can be traced out in the parameter space as seen below.



This plot shows the MET distribution of the events that survive the various cuts made.

This plot shows the final results of this reinterpretation analysis. The new exclusion contour fills in all of the gap seen in the ATLAS Preliminary. This LLP analysis excludes about 200-350 GeV more than the ATLAS RPV 1L analysis.

Nicely visual!



References

- [1] ATLAS Collaboration, *Reinterpretation of searches for supersymmetry in models with variable R-parity-violating coupling strength and long-lived R-hadrons*, (2018) ATLAS-CONF-2018-003
- [2] ATLAS Collaboration, *Search for long-lived, massive particles in events with a displaced vertex and a muon with large impact parameter in pp collisions at sqrt(s)=13 TeV with the ATLAS detector*, Phys. Rev. D 102, 032006 (2020), arXiv:2003.11956



A Reinterpretation of an LHC Search for Displaced Vertices and Muons in RPV SUSY Models

Taylor Sussmane¹, Lawrence Lee¹, Karri Folan DiPetrillo²

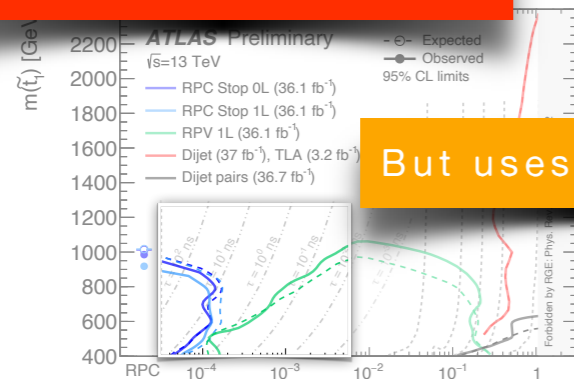
¹ University of Tennessee, Knoxville TN

² Fermilab, Batavia IL

Motivation

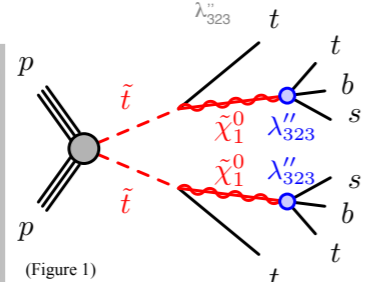
- Supersymmetry (SUSY) is an extension of the Standard Model of Particle Physics (SM) that would solve some of the problems with the SM, by postulating a fermion/boson symmetry.
- Supersymmetry predicts a heavier partner (sparticle) for each of the SM particles.
- RPV SUSY models violate R-parity conservation, meaning that a SUSY particle could decay into SM

(Some cramped text)

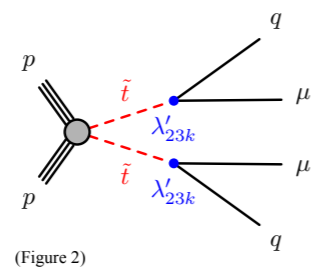
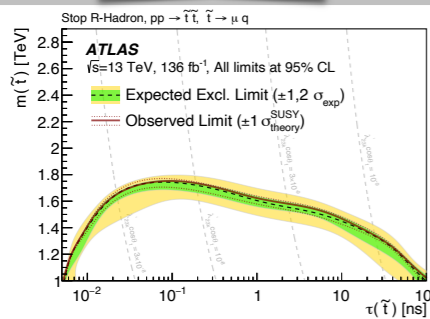


But uses the z-axis!

- Previous ATLAS analyses have obtained limits on the RPV SUSY model to the right [1], but there is a noticeable gap in the 10ps to 10ns proper lifetime range that should be explored with a search for long lived particles (LLP).
- We want to see if we have sensitivity to that range using a previous ATLAS search that requires a displaced vertex and muon.



ATLAS DV+Mu analysis

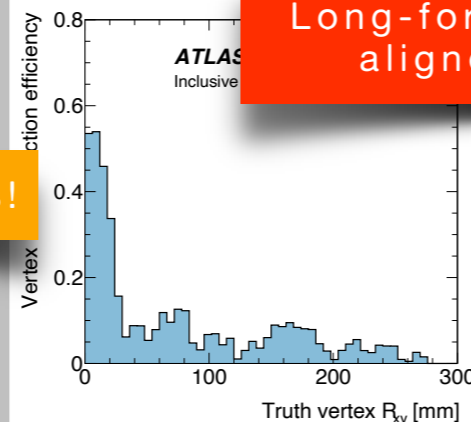


The ATLAS DV+Mu analysis [2] searched for the process above. It requires events with a displaced vertex and muon. The diagram in Figure 1 can give a displaced vertex and muon, through the decay of displaced top quarks. Luckily, that search had reinterpretation information that could be used to look for any process giving the same signature, like the one we're interested in!

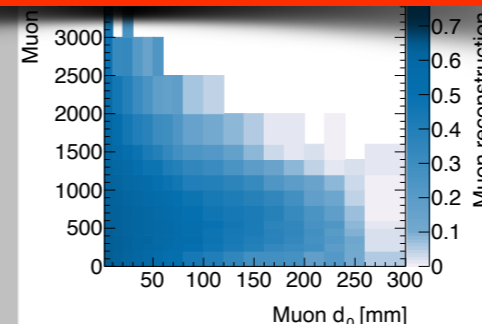
Reinterpretation Information

The ATLAS DV+Mu search provided generic reinterpretation information to be used for other models that have a displaced vertex and muon. That information was used on a grid of Monte Carlo simulation samples with various parameters for the target signal model. The material defined event and object acceptances, which would cut any events that do not meet certain truth criteria. For each event that passed the acceptance, efficiencies could be calculated. After the efficiency is calculated, a random number is thrown and the event may be rejected.

Long-form text should be left aligned or justified-left



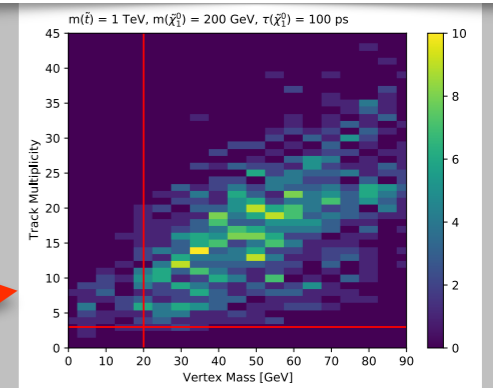
These plots give an example of how to reinterpret information of the emulate



Unmotivated edges should be minimized

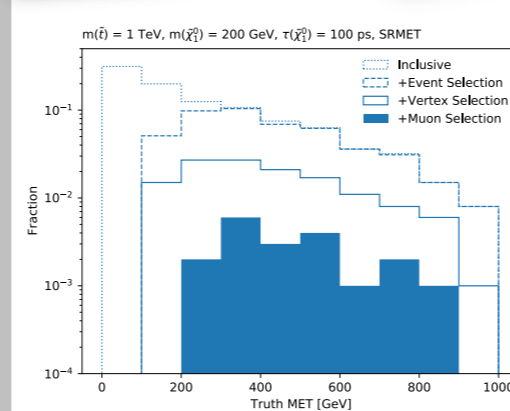
→ Give it a border, shadow, or make the BG match to make it disappear

This plot shows the distribution of track multiplicities and vertex masses for a signal model. The signal region (SR) is the region in the top right. As shown, most of the signal resides in the SR, so loosening the cuts would not significantly change the results because the search had very little background.



Nicely visual!

After all the cuts, the number of surviving events is a function of stop mass. If the normalized number of events is less than 3 (yields 3 events), then those parameters (stop mass/lifetime) are excluded. In the entire grid of simulations, the exclusion contour can be traced out in the parameter space as seen below.



This plot shows the MET distribution of the events that survive the various cuts made.

This plot shows the final results of this reinterpretation analysis. The new exclusion contour fills in all of the gap seen in the ATLAS Preliminary. This LLP analysis excludes about 200-350 GeV more than the ATLAS RPV 1L analysis.

Should use UTK logo, not UT system logo

References

- [1] ATLAS Collaboration, *Reinterpretation of searches for supersymmetry in models with variable R-parity-violating coupling strength and long-lived R-hadrons*, (2018) ATLAS-CONF-2018-003
- [2] ATLAS Collaboration, *Search for long-lived, massive particles in events with a displaced vertex and a muon with large impact parameter in pp collisions at $\sqrt{s}=13$ TeV with the ATLAS detector*, Phys. Rev. D 102, 032006 (2020), arXiv:2003.11956



Motivation

At particle colliders, collimated sprays of hadrons known as jets are commonly produced.

QCD confinement forbids free particles from carrying color charge, i.e., jets consist of color-singlet hadrons.

Experiments measure the momenta and properties of these hadrons and cluster the reconstructed objects into jets using various algorithms.

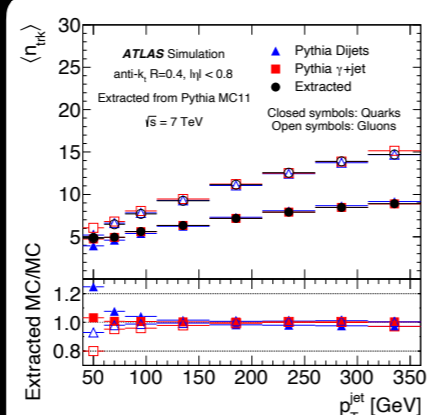
In the jet physics industry, it is common to consider a jet to be defined by its progenitor particle species and its momentum; e.g., a light quark jet of a particular momentum should have a set of properties drawn from the same distributions as every other light quark jet of that momentum scale.

The Problem

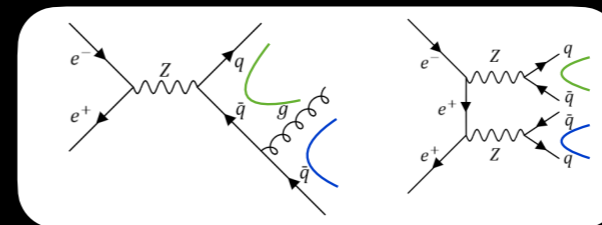
Assumption – Kinematically similar jets of similar origin are all produced from the same underlying physics distributions.

This assumption fails when requiring that observers in all frames must have a Lorentz-consistent view of these jets, e.g., all observers should agree on a jet's particle multiplicity.

There must be a special frame in which a parton's fragmentation occurs. The simplest self-consistent frame would be the rest frame of color-connected particles.



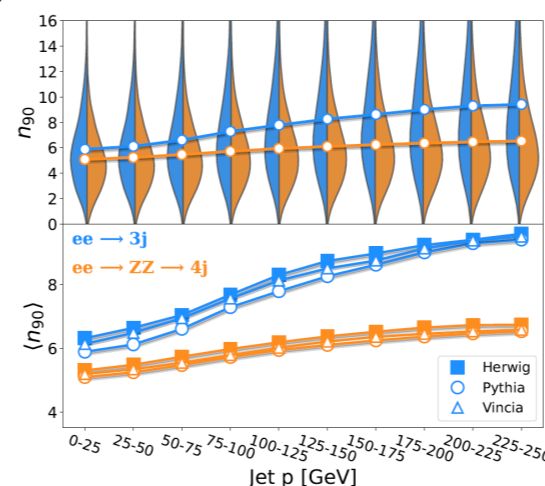
We consider particle multiplicity to be a function of jet momentum.
(Eur. Phys. J. C (2014) 74: 3023)



Feynman diagrams of our processes.
Left: $ee \rightarrow jjj$ Right: $ee \rightarrow ZZ \rightarrow jjjj$

green
anti-green

blue
anti-blue



(Top) Split violin plots showing the normalized n_{90} distributions for two ee collider processes as a function of lab-frame momentum.

(Background)
The collision of two leptons (purple) produces two color singlets, each decaying to a quark pair. The color rest frames are the frames where each of the color-connected $q\bar{q}$ systems are at rest

Analysis

We used MadGraph5 aMC@NLO 3.3.1 to produce 50,000 parton-level events of two processes: $ee \rightarrow jjj$ and $ee \rightarrow ZZ \rightarrow jjjj$, both with a collision energy of $\sqrt{s} = 1$ TeV.

We showered these parton-level events with three models: Pythia 8.306, Vincia, and Herwig 7.2.2, and we clustered the jets with FastJet using Delphes 3.5.1.

In $ee \rightarrow jjj$ (blue), the color rest frame is coincident with the lab frame and particle multiplicity is a function of momentum. However, in $ee \rightarrow ZZ \rightarrow jjjj$ (orange), in which jets obtain large momentum from boosted color rest frames, the dependence is significantly weaker. The mean, $\langle n_{90} \rangle$, is shown as a function of lab-frame jet momentum for different shower models (Bottom). Herwig, Pythia, and Vincia show similar behaviors.

Conclusion

This effect should have significant implications on how jet tagging is done. Jet tagging algorithms try to gain insight into the origin of a jet using the observable properties of the jet shower. In the design, training, calibration, and validation of these taggers, the jet individualism assumption is heavily used.

The training of jet-by-jet taggers should consider the effect of boosted color rest frames, and the language around jet physics should be made more precise.

This effect also represents an under-explored opportunity for discriminating jets from boosted color singlet decays, especially in BSM searches.

Motivation

At particle colliders, collimated sprays of hadrons known as jets are commonly produced.

QCD confinement forbids free particles from carrying color charge, i.e., jets consist of color-singlet hadrons.

Experiments measure the momenta and properties of these hadrons and cluster the reconstructed objects into jets using various algorithms.

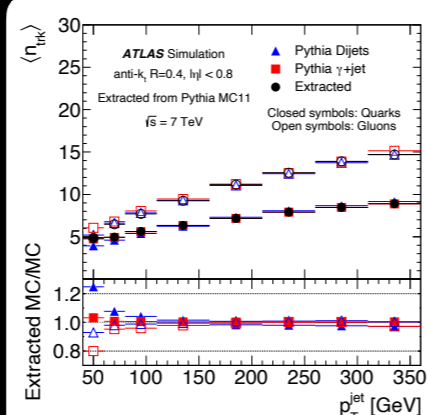
In the jet physics industry, it is common to consider a jet to be defined by its progenitor particle species and its momentum; e.g., a light quark jet of a particular momentum should have a set of properties drawn from the same distributions as every other light quark jet of that momentum scale.

The Problem

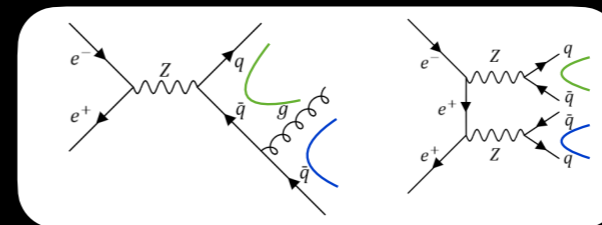
Assumption – Kinematically similar jets of similar origin are all produced from the same underlying physics distributions.

This assumption fails when requiring that observers in all frames must have a Lorentz-consistent view of these jets, e.g., all observers should agree on a jet's particle multiplicity.

There must be a special frame in which a parton's fragmentation occurs. The simplest self-consistent frame would be the rest frame of color-connected particles.



We consider particle multiplicity to be a function of jet momentum.
(Eur. Phys. J. C (2014) 74: 3023)



Feynman diagrams of our processes.
Left: $ee \rightarrow jjj$ Right: $ee \rightarrow ZZ \rightarrow jjjj$

Eye catching from a distance!

Analysis

We used MadGraph5 aMC@NLO 3.3.1 to produce 50,000 parton-level events of two processes: $ee \rightarrow jjj$ and $ee \rightarrow ZZ \rightarrow jjjj$, both with a collision energy of $\sqrt{s} = 1$ TeV.

We showered these parton-level events with three models: Pythia 8.306, Vincia, and Herwig 7.2.2, and we clustered the jets with FastJet using Delphes 3.5.1.

In $ee \rightarrow jjj$ (blue), the color rest frame is coincident with the lab frame and particle multiplicity is a function of momentum. However, in $ee \rightarrow ZZ \rightarrow jjjj$ (orange), in which jets obtain large momentum from boosted color rest frames, the dependence is significantly weaker. The mean, $\langle n_{90} \rangle$, is shown as a function of lab-frame jet momentum for different shower models (Bottom). Herwig, Pythia, and Vincia show similar behaviors.

Conclusion

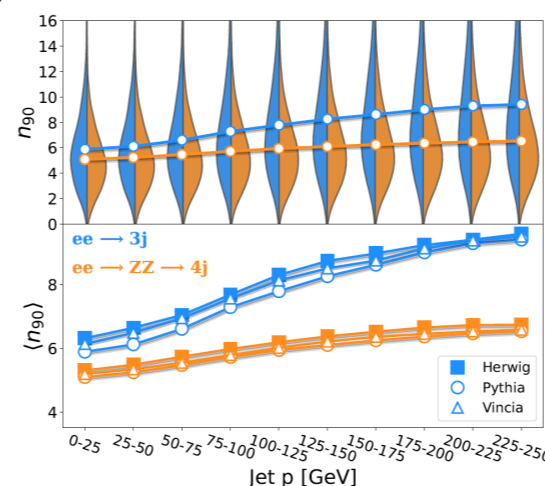
This effect should have significant implications on how jet tagging is done. Jet tagging algorithms try to gain insight into the origin of a jet using the observable properties of the jet shower. In the design, training, calibration, and validation of these taggers, the jet individualism assumption is heavily used.

The training of jet-by-jet taggers should consider the effect of boosted color rest frames, and the language around jet physics should be made more precise.

This effect also represents an under-explored opportunity for discriminating jets from boosted color singlet decays, especially in BSM searches.

green
anti-green

blue
anti-blue



(Top) Split violin plots showing the normalized n_{90} distributions for two ee collider processes as a function of lab-frame momentum.

(Background)
The collision of two leptons (purple) produces two color singlets, each decaying to a quark pair. The color rest frames are the frames where each of the color-connected qq systems are at rest

Motivation

At particle colliders, collimated sprays of hadrons known as jets are commonly produced.

QCD confinement forbids free particles from carrying color charge, i.e., jets consist of color-singlet hadrons.

Experiments measure the momenta and properties of these hadrons and cluster the reconstructed objects into jets using various algorithms.

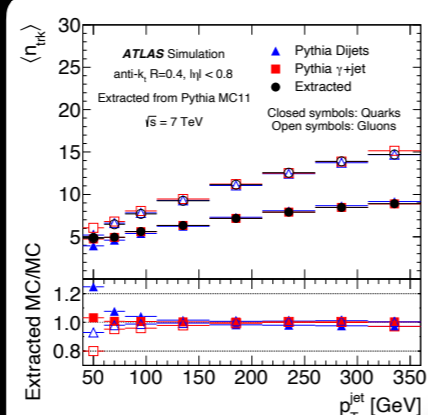
In the jet physics industry, it is common to consider a jet to be defined by its progenitor particle species and its momentum; e.g., a light quark jet of a particular momentum should have a set of properties drawn from the same distributions as every other light quark jet of that momentum scale.

The Problem

Assumption – Kinematically similar jets of similar origin are all produced from the same underlying physics distributions.

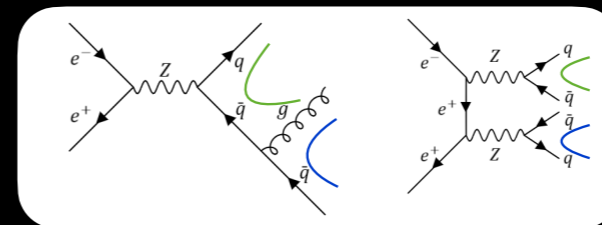
This assumption fails when requiring that observers in all frames must have a Lorentz-consistent view of these jets, e.g., all observers should agree on a jet's particle multiplicity.

There must be a special frame in which a parton's fragmentation occurs. The simplest self-consistent frame would be the rest frame of color-connected particles.



We consider particle multiplicity to be a function of jet momentum.
(Eur. Phys. J. C (2014) 74: 3023)

All edges nicely motivated and purposeful



Feynman diagrams of our processes.
Left: $ee \rightarrow jjj$ Right: $ee \rightarrow ZZ \rightarrow jjjj$

Eye catching from a distance!

Analysis

We used MadGraph5 aMC@NLO 3.3.1 to produce 50,000 parton-level events of two processes: $ee \rightarrow jjj$ and $ee \rightarrow ZZ \rightarrow jjjj$, both with a collision energy of $\sqrt{s} = 1$ TeV.

We showered these parton-level events with three models: Pythia 8.306, Vincia, and Herwig 7.2.2, and we clustered the jets with FastJet using Delphes 3.5.1.

In $ee \rightarrow jjj$ (blue), the color rest frame is coincident with the lab frame and particle multiplicity is a function of momentum. However, in $ee \rightarrow ZZ \rightarrow jjjj$ (orange), in which jets obtain large momentum from boosted color rest frames, the dependence is significantly weaker. The mean, $\langle n_{90} \rangle$, is shown as a function of lab-frame jet momentum for different shower models (Bottom). Herwig, Pythia, and Vincia show similar behaviors.

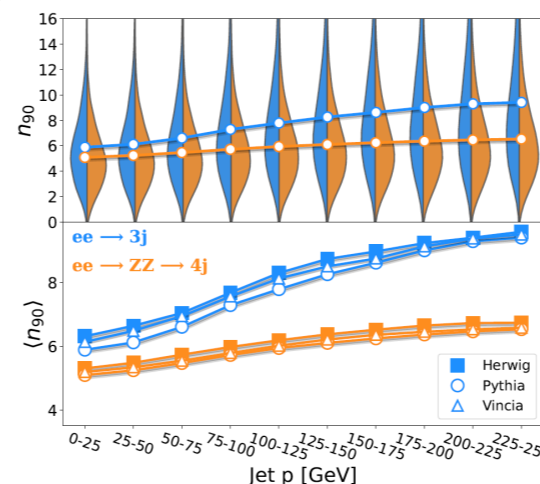
Conclusion

This effect should have significant implications on how jet tagging is done. Jet tagging algorithms try to gain insight into the origin of a jet using the observable properties of the jet shower. In the design, training, calibration, and validation of these taggers, the jet individualism assumption is heavily used.

The training of jet-by-jet taggers should consider the effect of boosted color rest frames, and the language around jet physics should be made more precise.

This effect also represents an under-explored opportunity for discriminating jets from boosted color singlet decays, especially in BSM searches.

green
anti-green



(Top) Split violin plots showing the normalized n_{90} distributions for two ee collider processes as a function of lab-frame momentum.

(Background)
The collision of two leptons (purple) produces two color singlets, each decaying to a quark pair. The color rest frames are the frames where each of the color-connected qq systems are at rest

Motivation

At particle colliders, collimated sprays of hadrons known as jets are commonly produced.

QCD confinement forbids free particles from carrying color charge, i.e., jets consist of color-singlet hadrons.

Experiments measure the momenta and properties of these hadrons and cluster the reconstructed objects into jets using various algorithms.

In the jet physics industry, it is common to consider a jet to be defined by its progenitor particle species and its momentum; e.g., a light quark jet of a particular momentum should have a set of properties drawn from the same distributions as every other light quark jet of that momentum scale.

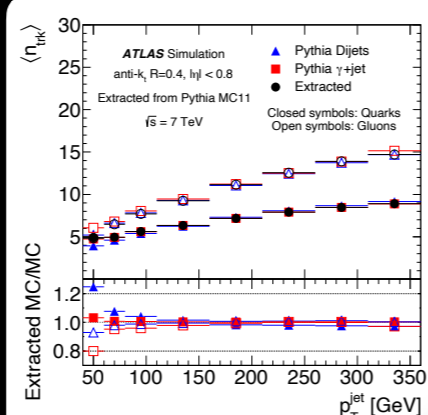
The Problem

Assumption – Kinematically similar jets of similar origin are all produced from the same underlying physics distributions.

This assumption fails when requiring that observers in all frames must have a Lorentz-consistent view of these jets, e.g., all observers should agree on a jet's particle multiplicity.

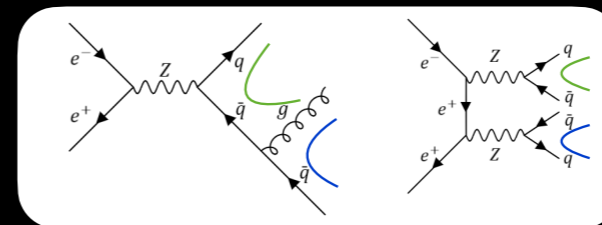
There must be a special frame in which a parton's fragmentation occurs. The simplest self-consistent frame would be the rest frame of color-connected particles.

This “hero image” useful in presentation spiel



We consider particle multiplicity to be a function of jet momentum.
(Eur. Phys. J. C (2014) 74: 3023)

All edges nicely motivated and purposeful



Feynman diagrams of our processes.
Left: $ee \rightarrow jjj$ Right: $ee \rightarrow ZZ \rightarrow jjjj$

Eye catching from a distance!

Analysis

We used MadGraph5 aMC@NLO 3.3.1 to produce 50,000 parton-level events of two processes: $ee \rightarrow jjj$ and $ee \rightarrow ZZ \rightarrow jjjj$, both with a collision energy of $\sqrt{s} = 1$ TeV.

We showered these parton-level events with three models: Pythia 8.306, Vincia, and Herwig 7.2.2, and we clustered the jets with FastJet using Delphes 3.5.1.

In $ee \rightarrow jjj$ (blue), the color rest frame is coincident with the lab frame and particle multiplicity is a function of momentum. However, in $ee \rightarrow ZZ \rightarrow jjjj$ (orange), in which jets obtain large momentum from boosted color rest frames, the dependence is significantly weaker. The mean, $\langle n_{90} \rangle$, is shown as a function of lab-frame jet momentum for different shower models (Bottom). Herwig, Pythia, and Vincia show similar behaviors.

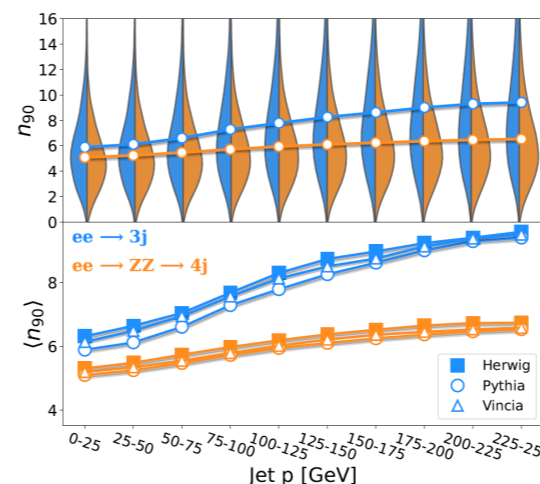
Conclusion

This effect should have significant implications on how jet tagging is done. Jet tagging algorithms try to gain insight into the origin of a jet using the observable properties of the jet shower. In the design, training, calibration, and validation of these taggers, the jet individualism assumption is heavily used.

The training of jet-by-jet taggers should consider the effect of boosted color rest frames, and the language around jet physics should be made more precise.

This effect also represents an under-explored opportunity for discriminating jets from boosted color singlet decays, especially in BSM searches.

green
anti-green



(Top) Split violin plots showing the normalized n_{90} distributions for two ee collider processes as a function of lab-frame momentum.

(Background)
The collision of two leptons (purple) produces two color singlets, each decaying to a quark pair. The color rest frames are the frames where each of the color-connected qq systems are at rest

Motivation

At particle colliders, collimated sprays of hadrons known as jets are commonly produced.

QCD confinement forbids free particles from carrying color charge, i.e., jets consist of color-singlet hadrons.

Experiments measure the momenta and properties of these hadrons and cluster the reconstructed objects into jets using various algorithms.

In the jet physics industry, it is common to consider a jet to be defined by its progenitor particle species and its momentum; e.g., a light quark jet of a particular momentum should have a set of properties drawn from the same distributions as every other light quark jet of that momentum scale.

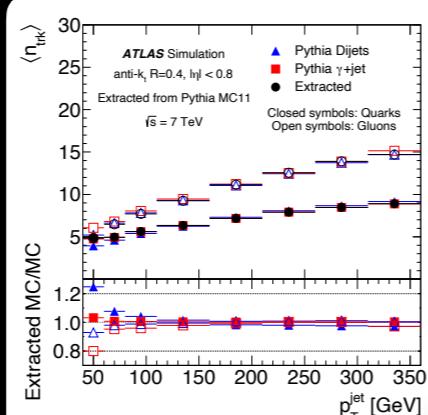
The Problem

Assumption – Kinematically similar jets of similar origin are all produced from the same underlying physics distributions.

This assumption fails when requiring that observers in all frames must have a Lorentz-consistent view of these jets, e.g., all observers should agree on a jet's particle multiplicity.

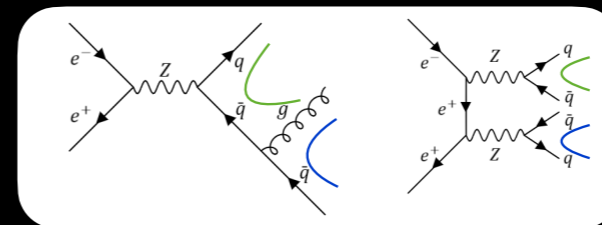
There must be a special frame in which a parton's fragmentation occurs. The simplest self-consistent frame would be the rest frame of color-connected particles.

This “hero image” useful in presentation spiel



We consider particle multiplicity to be a function of jet momentum.
(Eur. Phys. J. C (2014) 74: 3023)

All edges nicely motivated and purposeful



Feynman diagrams of our processes.
Left: $ee \rightarrow jjj$ Right: $ee \rightarrow ZZ \rightarrow jjjj$

Eye catching from a distance!

Analysis

We used MadGraph5 aMC@NLO 3.3.1 to produce 50,000 parton-level events of two processes: $ee \rightarrow jjj$ and $ee \rightarrow ZZ \rightarrow jjjj$, both with a collision energy of $\sqrt{s} = 1$ TeV.

We showered these parton-level events with three models: Pythia 8.306, Vincia, and Herwig 7.2.2, and we clustered the jets with FastJet using Delphes 3.5.1.

In $ee \rightarrow jjj$ (blue), the color rest frame is coincident with the lab frame and particle multiplicity is a function of momentum. However, in $ee \rightarrow ZZ \rightarrow jjjj$ (orange), in which jets obtain large momentum from boosted color rest frames, the dependence is significantly weaker. The mean, $\langle n_{90} \rangle$, is shown as a function of lab-frame jet momentum for different shower models (Bottom). Herwig, Pythia, and Vincia show similar behaviors.

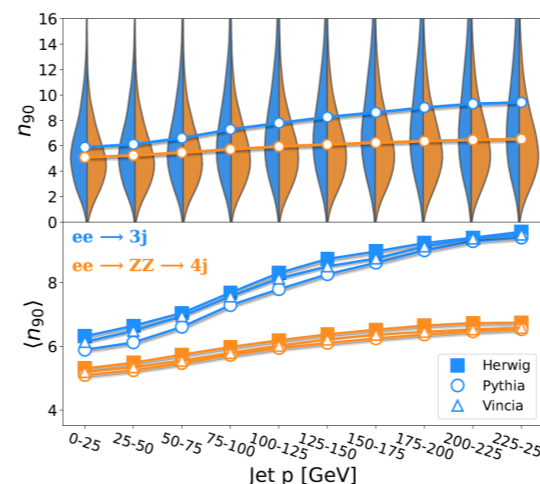
Conclusion

This effect should have significant implications on how jet tagging is done. Jet tagging algorithms try to gain insight into the origin of a jet using the observable properties of the jet shower. In the design, training, calibration, and validation of these taggers, the jet individualism assumption is heavily used.

The training of jet-by-jet taggers should consider the effect of boosted color rest frames, and the language of color flow is more precise.

Stretches beyond the canvas

green
anti-green



(Top) Split violin plots showing the normalized n_{90} distributions for two ee collider processes as a function of lab-frame momentum.

(Background)
The collision of two leptons (purple) produces two color singlets, each decaying to a quark pair. The color rest frames are the frames where each of the color-connected qq systems are at rest

OUR SIGNATURES

ELECTRONS 270 HZ

MUONS 290 HZ

TAUS 160 HZ

JETS & MET 630 HZ

THE ATLAS RUN 3 TRIGGER MENU

MARCO MONTELLA [1] ON BEHALF OF THE ATLAS TRIGGER GROUP

ABOUT US

ATLAS runs a two-level triggering strategy:

- L1** → Hardware-based, coarse reconstruction, total accepted rate is 90-100 KHz
- HLT** → Software-based, reconstruction precision approaching offline reconstruction

The Trigger Menu is limited by:

- L1 RATE** → Constrained by dead time. Impacts the range of physics accessible
- HLT CPU** → Limits the execution rate of high-precision reconstruction algorithms
- TO CPU** → Limits the data volume that can be reconstructed promptly for endpoint analysis

L1/HLT limitations scale with luminosity:

- END OF FILL** → Enhance signatures limited by L1 rate and/or HLT rate & CPU

OUR STREAMS

MAIN 1.7 KHZ

For prompt reconstruction
Rate limited by L1, CPU & TO Resources

DELAYED 900 HZ

Delayed Processing
when TO CPU available

TRIGGER-LEVEL ANALYSIS (TLA) 5+ KHZ

Reduced event content,
HLT Objects only.
Minimal burden on bandwidth

PARTIAL EVENT BUILDING (PEB) 0.5 KHZ

Regional data around near
physics objects identified by trigger

SPECIALTY TRIGGERS

EMERGING JETS 10 HZ

HIGHLY IONIZING TRACK 5 HZ

ISOLATED TRACK 1 HZ

DISAPPEARING TRACK 4 HZ

DISPLACED OBJECTS 40 HZ

PARTIAL EVENT BUILDING 200 HZ

HEAVY IONS: RUN 2 THRESHOLDS PRESERVED!

CHEF'S NOTES

AN UPGRADED SYSTEM

Meeting the challenges of higher luminosity & beam intensity:

L1

- NEW DIGITIZED LAR CALORIMETER READOUT → x10 Calo Granularity!
- NEW L1CALO FEATURE EXTRACTORS:
 - eFEX → High granularity EM & Tau core reconstruction
 - jFEX → L1 Jet & hadronic Tau shower isolation
 - gFEX → Coarser granularity for large-scale reconstruction (large-R jets, MET)
- OVERHAULED MUON TRIGGER: → NSW + New EndCap Processor
- RENEWED L1TOPO → New logic for input data from Phase-1 FEXes

HLT

- TRACKING IMPROVEMENTS:
 - LARGE RADIUS → High efficiency reconstruction for LLP signatures
- FULL SCAN TRACKING+VERTEXING FOR HLT JETS:
 - Particle Flow Reconstruction
 - High Performance Flavour Tagging
- 'REAL TIME ANALYSIS':
 - MULTI-OBJECT TLA → Combined signatures (e.g. Photon + Jets)
 - PARTIAL EVENT BUILDING → Regional data around near physics objects identified by trigger

ELECTRONS & MUONS THE STAPLES NEW & IMPROVED IN '22

ATLAS Preliminary, $\sqrt{s} = 13.6$ TeV, 295 pb⁻¹

ATLAS Data 2022, $\sqrt{s} = 13.6$ TeV Z- $\mu\mu$ tag-and-probe, $|\eta| < 1.05$

Run 3 Entrées LRT & B-TAGGING AND MUCH MORE!

ATLAS Data 2022, $\sqrt{s} = 13.6$ TeV

ATLAS Simulation Preliminary $\sqrt{s} = 13.6$ TeV, $\Phi_{flow} = 0.018$

CERTIFIED PERFORMANCE!

- Innovative
- Funny and Engaging
- Informative without being wordy!
- Notice: No sentences!
- Skimmable, visual
- Engaging presentation
- Judges immediately agreed this would win

[1], The Ohio State University
All Photo and Figures from
and the public ATLAS Trigger Plot Repository: <https://wiki.cern.ch/view/AtlasPublic/TriggerPublic/Results>

Template Design by Katharine Leney & Marco Montella

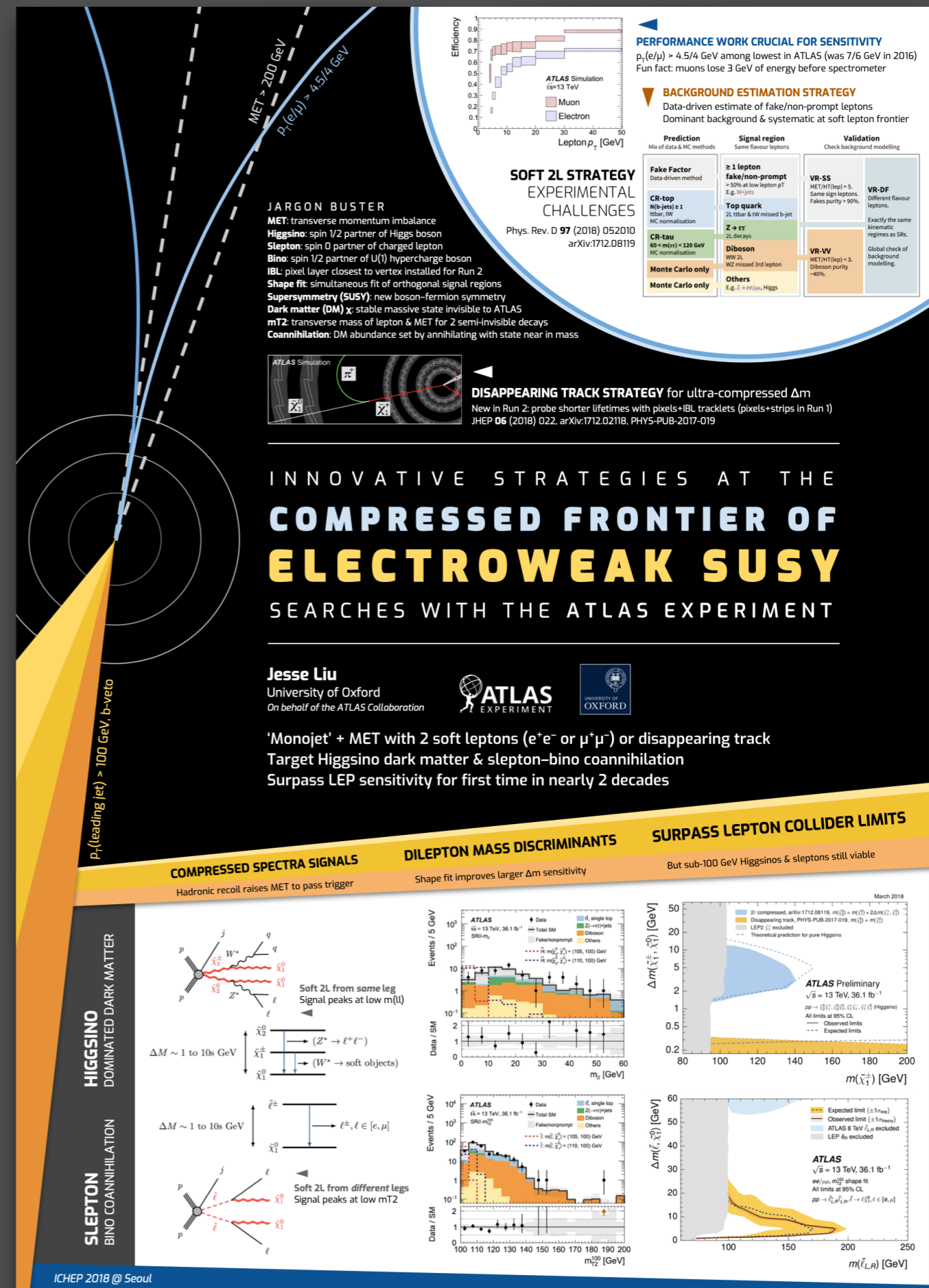
ATLAS EXPERIMENT LHCp2024

23

- **Beautiful** and attention grabbing from across the room
- Had a **huge crowd** the whole session!

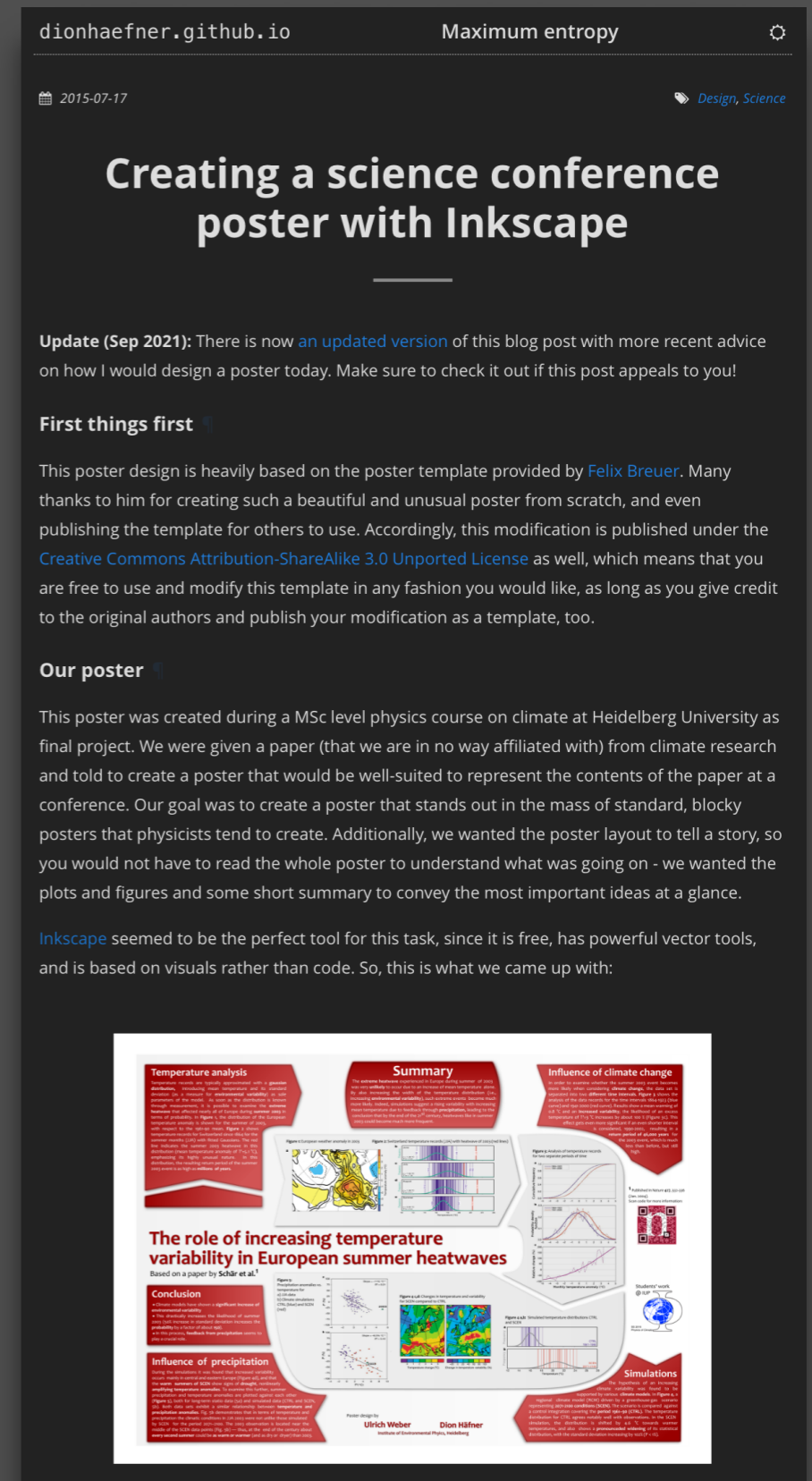
- All graphics serve to **communicate**
- Surprising, unique, characterful

- Arrows and shapes direct the viewer's attention
- Shadows used to convey depth



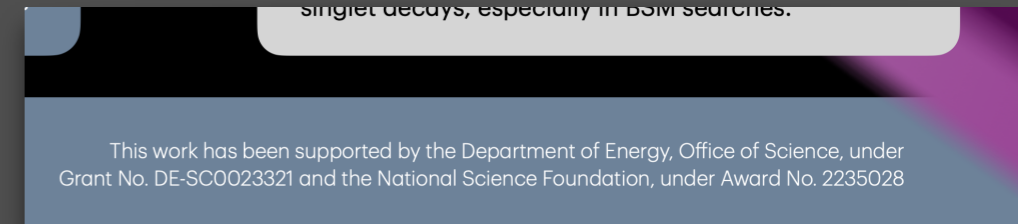
Technical Advice

- Most people use **Powerpoint** to create their poster
 - If you're on a Mac, **Keynote** is much better for this
- If you have access to advanced software, I recommend it (w/ learning curve):
 - Adobe Products (\$\$\$)
 - Affinity Products (\$\$) [LL]
 - GIMP/Inkscape (Free/OS)
- Some people do it in LaTeX — as much as I advocate that you all learn LaTeX, **it's not a good tool for this**
- **Pixelation looks unprofessional**
 - Make sure all images are vector (SVG, EPS, PDF, ...)
 - Or if they're raster (JPG, PNG, ...) ensure high resolution
 - If you can't find a high quality version, consider making your own vector version!



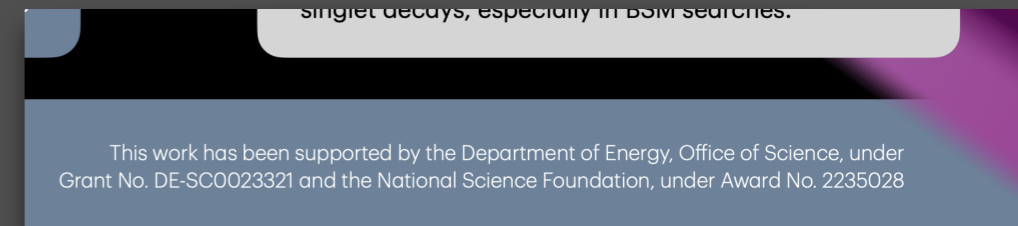
Practical Tips

- Check with your advisor about any needed funding **acknowledgements**!
- Usually: Have a **UTK logo** and any other relevant logos
- Some sessions have visual requirements and some have required templates. Make sure you're compliant.
- They should communicate allowable sizes. In US, common standards:
 - 3' x 4' (sometimes a pain to carry)
 - 2' x 3'
 - Either portrait or landscape
- For our dept business, we have a **free poster printer**. Front office or faculty can show you.
 - External printing is weirdly expensive!



Practical Tips

- Check with your advisor about any needed funding **acknowledgements**!
- Usually: Have a **UTK logo** and any other relevant logos
- Some sessions have visual requirements and some have required templates. Make sure you're compliant.
- They should communicate allowable sizes. In US, common standards:
 - 3' x 4' (sometimes a pain to carry)
 - 2' x 3'
 - Either portrait or landscape
- For our dept business, we have a **free poster printer**. Front office or faculty can show you.
 - External printing is weirdly expensive!

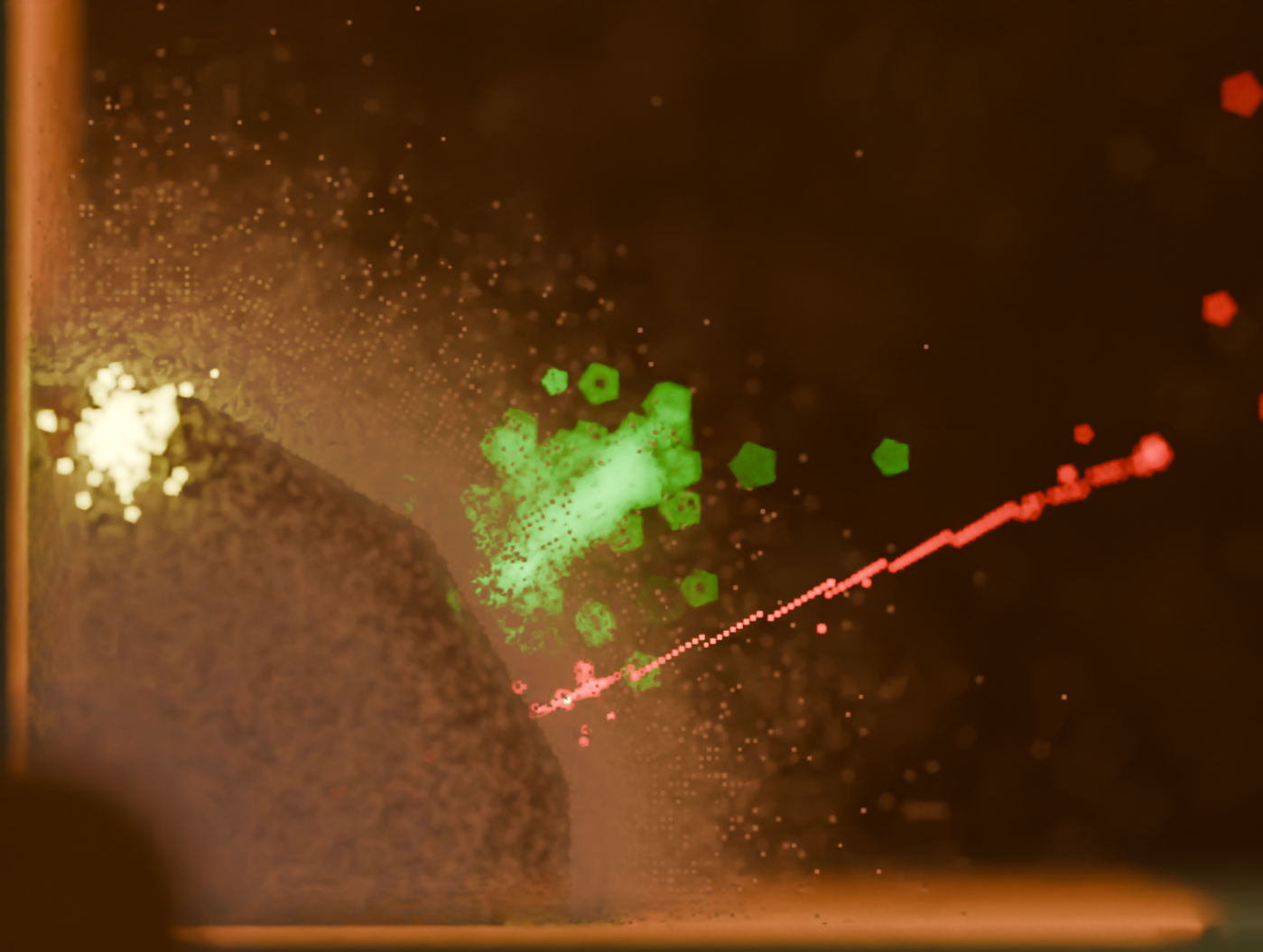


Dept has some loaner poster tubes for transport to conference (Can usually be brought on a plane, maybe as your personal item)

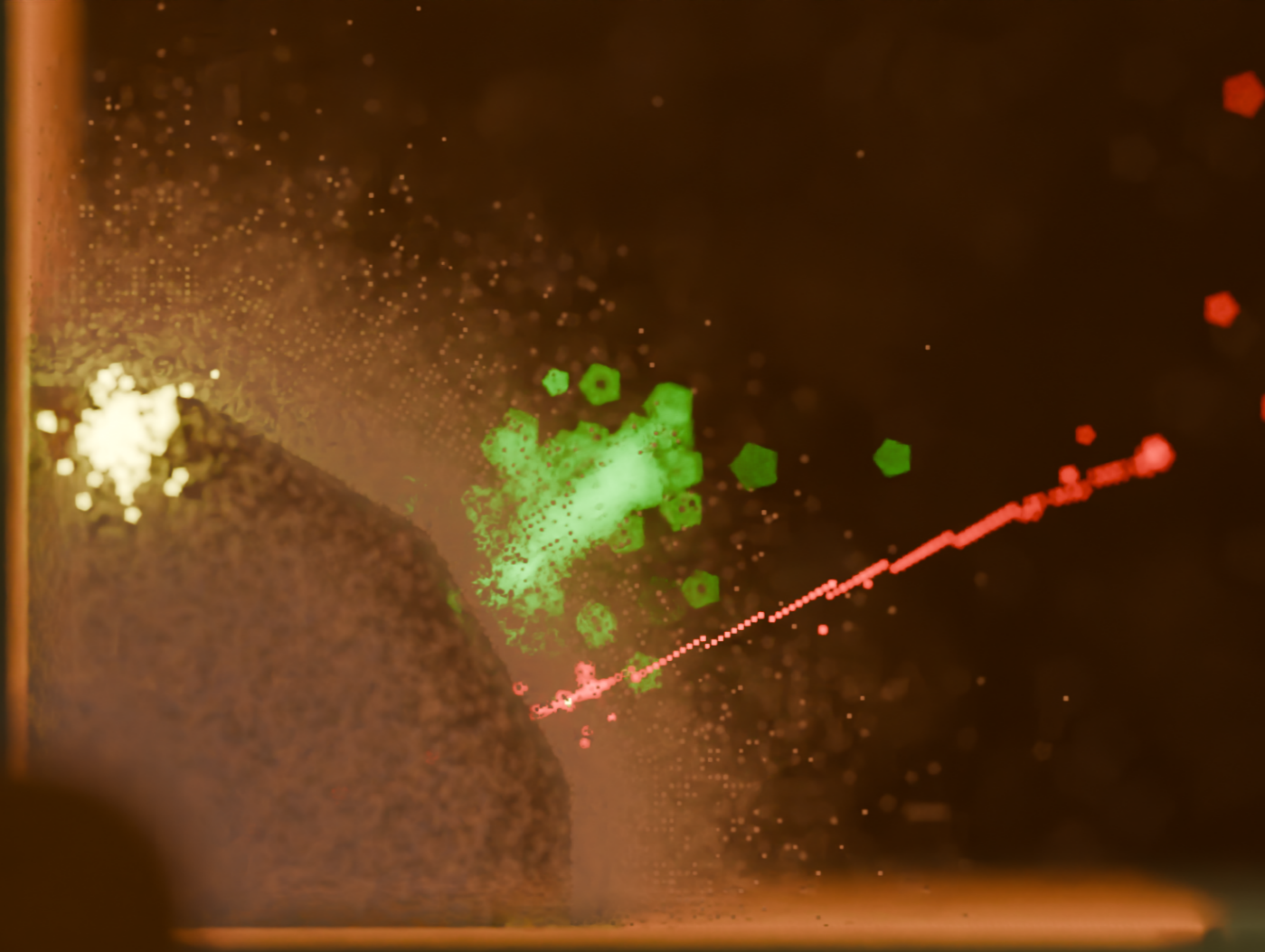


Final Words

- During SPS meetings, I'm happy to skim over poster drafts and give quick feedback
- Focus on the **communication** and practice, practice, practice
- Remember that scientific communication need not be dry
- Convey information, but also be emotionally and empathetically mindful of the viewer's experience
- And good luck with your poster prep!



Thanks for your
attention!



Backup

Big Ears Festival Stars the Avant-Garde

Share full article



3/11

Laurie Anderson performed with the Kronos Quartet. Jake Giles Netter for The New York Times

By Ben Ratliff

March 30, 2015

KNOXVILLE, Tenn. — The avant-garde is a notion shaped by discomfort, as well as defiance. No music gets that designation

BIG EARS

March 27-30 2025
KNOXVILLE / USA

ANOHN! and the Johnsons - Anoushka Shankar - Arooj Aftab
Béla Fleck, Edmar Castañeda, Antonio Sánchez Trio - Bill Frisell - DakhaBrakha
Esperanza Spalding - Explosions In The Sky - Jessica Pratt - Joe Lovano Paramount Quartet
King Britt - Lankum - Les Claypool's Bastard Jazz - Meshell Ndegeocello - múm
Nels Cline - Rufus Wainwright - Steve Roach - Sun Ra Arkestra & Yo La Tengo - Taj Mahal
Tessa Lark, Joshua Roman & Edgar Meyer - Tindersticks - Tortoise - Tyshawn Sorey
Vijay Iyer - Waxahatchee - Zakir Hussain & Masters of Percussion

Wadada Leo Smith:
CREATE

RedKoral Quartet - Orange Wave Electric
Revolutionary Love & More

Philip Glass:
Music in 12 Parts
(50th Anniversary)

Performed by the
Philip Glass Ensemble

Jonny Greenwood's
133 Years of Reverb
(N. American Premiere)

Performed by James McVinnie
& Eliza McCarthy

Kate Soper's
Ipsa Dixit

Performed by
Wet Ink Ensemble

Tyshawn Sorey

Monochromatic Light
(Afterlife)

Michael Rother

The Music of NEU!
& Harmonia

Across the Horizon

Ambient Americana Soundscapes
Curated by Bob Holmes and SUSS

Blacktronika

Afrofuturism in Electronic Music
Curated by King Britt

Adam Rudolph - [Ahmed] - Alabaster DePlume - Alan Sparhawk - Allison de Groot & Tatiana Hargreaves - Amaro Freitas Trio
Ambrose Akinmusire - Antipop Consortium - Asha Puthli - Astrid Sonne - Axiom 5 - Barry Altschul's 3 Dom Factor - Beak> - Bia Ferreira
Brigade Chaimbeul - Canzoniere Greco Salentino - Carlos Niño & Friends - Cassandra Jenkins - Chanel Beads - Chuck Johnson - Claire Chase
Clarice Jensen - clipping. - Cowboy Sadness - Dan Weiss Even Odds Trio - David Grubbs - Dawn Richard & Spencer Zahn - Dedicated Men of Zion
EMEL - Eucademix (Yuka Honda) - Fay Victor - Flore Laurentienne - Free Form Funky Freqs - Helado Negro - Ibelisse Guardia Ferragutti & Frank Rosaly
Immanuel Wilkins - Jeff Parker ETA IVtet - Jenny Scheinman - Joan as Police Woman - Joel Harrison - Joseph Keckler - Josh Johnson
Joy Guidry - Jules Reidy - Julia Holter - June McDoom - Kahil El'Zabar Ethnic Heritage Ensemble - Kalia Vandever - Kelly Moran
Knoxville Opera Gospel Choir - Kokayi - Kris Davis Trio - Lara Somogyi - Luke Stewart Silt Trio - Mabe Fratti - Macie Stewart - Magic Tuber Stringband
Maria Chávez / Victoria Shen / Mariam Rezaei - Marisa Anderson - Marissa Nadler - Mark Guiliana - Maruja - Mary Lattimore - Michael Hurley
Mike Reed's Separatist Party - ML Buch - Modney - Nanocluster (Immersion | SUSS) - Peni Candra Rini - Phantom Orchard - Phil Cook
Rachika Nayar - R.B. Morris & William Wright - Rich Ruth - Sam Bush Band - Shelley Hirsch - SML - Squanderers - Steve Coleman and Five Elements
Steve Lehman Trio + Mark Turner - Steven Schick - Still House Plants - Sunny War - Susan Alcorn - Sylvie Courvoisier
Tara Clerkin Trio - Tarta Relena - Tigran Hamasyan - Tilt - Water Damage - William Basinski - Yaya Bey - Zeena Parkins

With more to come!

Nearly 200 performances, 12+ venues PLUS films, conversations, & more
Passes and more info at BigEarsFestival.org

Listen.