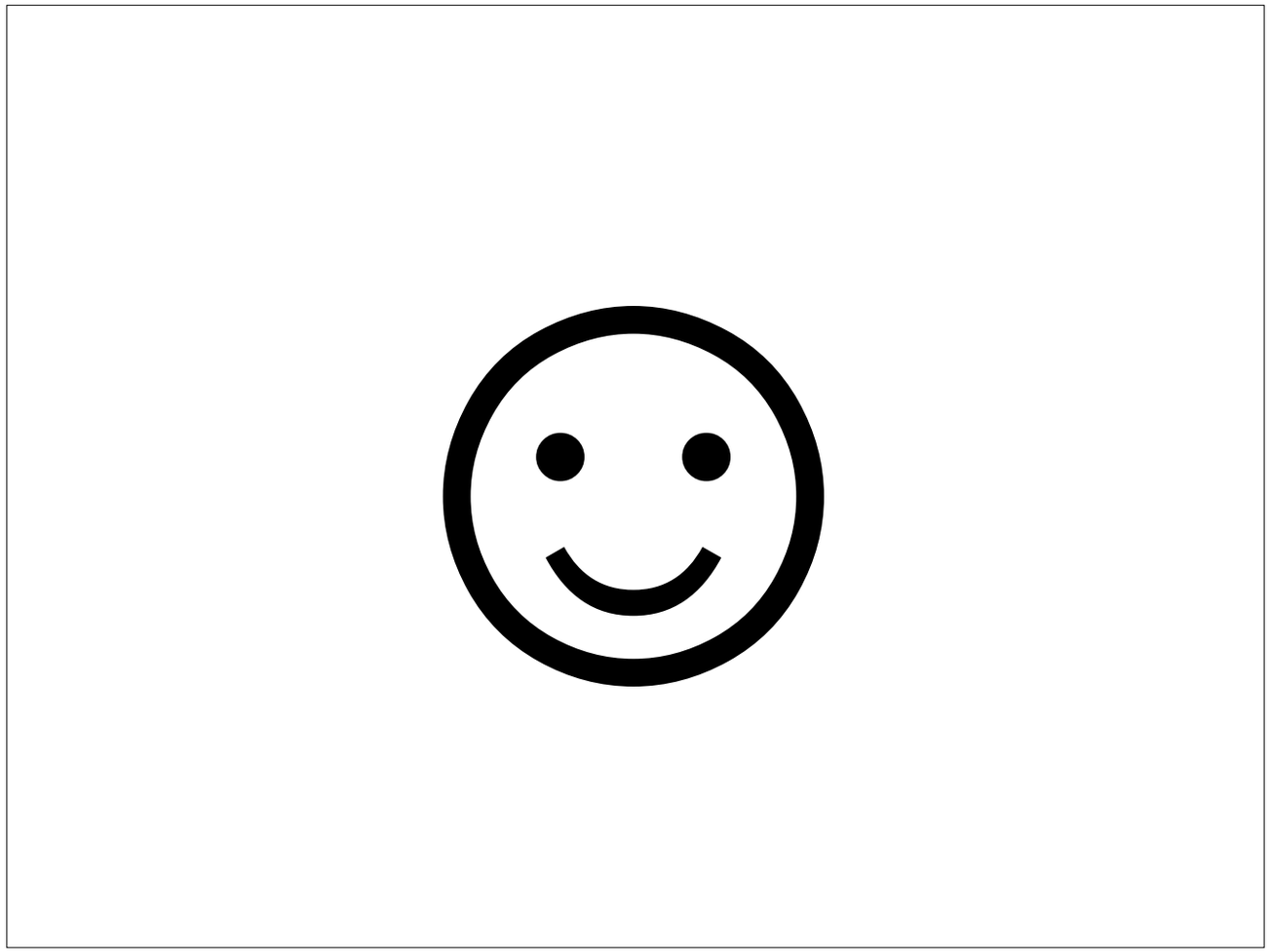
HIERARCHICAL SHAPE RETRIEVAL on a UNIT HYPERSPHERE

Yixin Lin
Glizela Taino
Mark Moyou
Adrian M. Peter

HAWAI'I PACIFIC UNIVERSITY

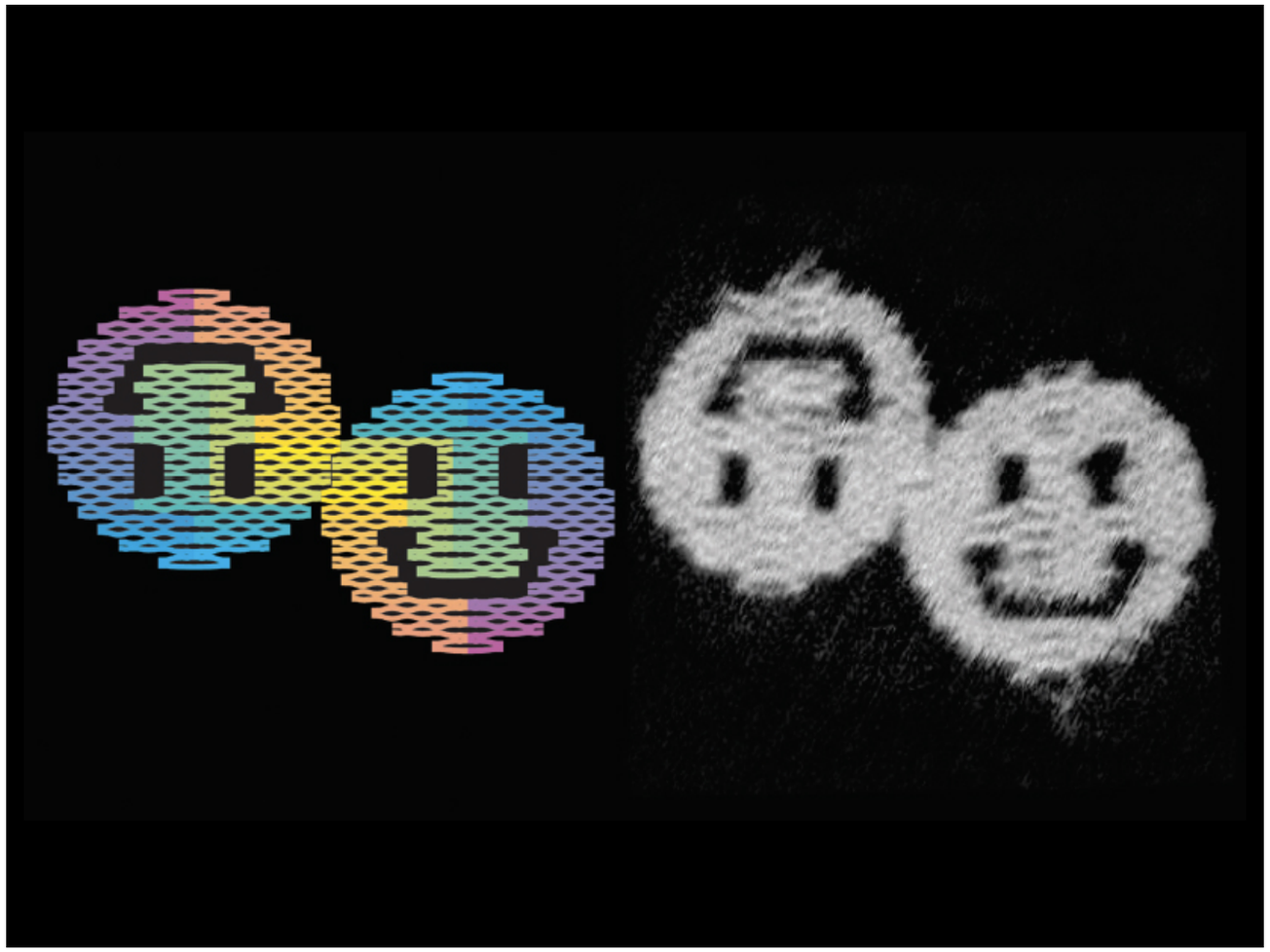
Duke UNIVERSITY



Yixin
Can you recognize what this is?
Zela
A smiley face, of course.

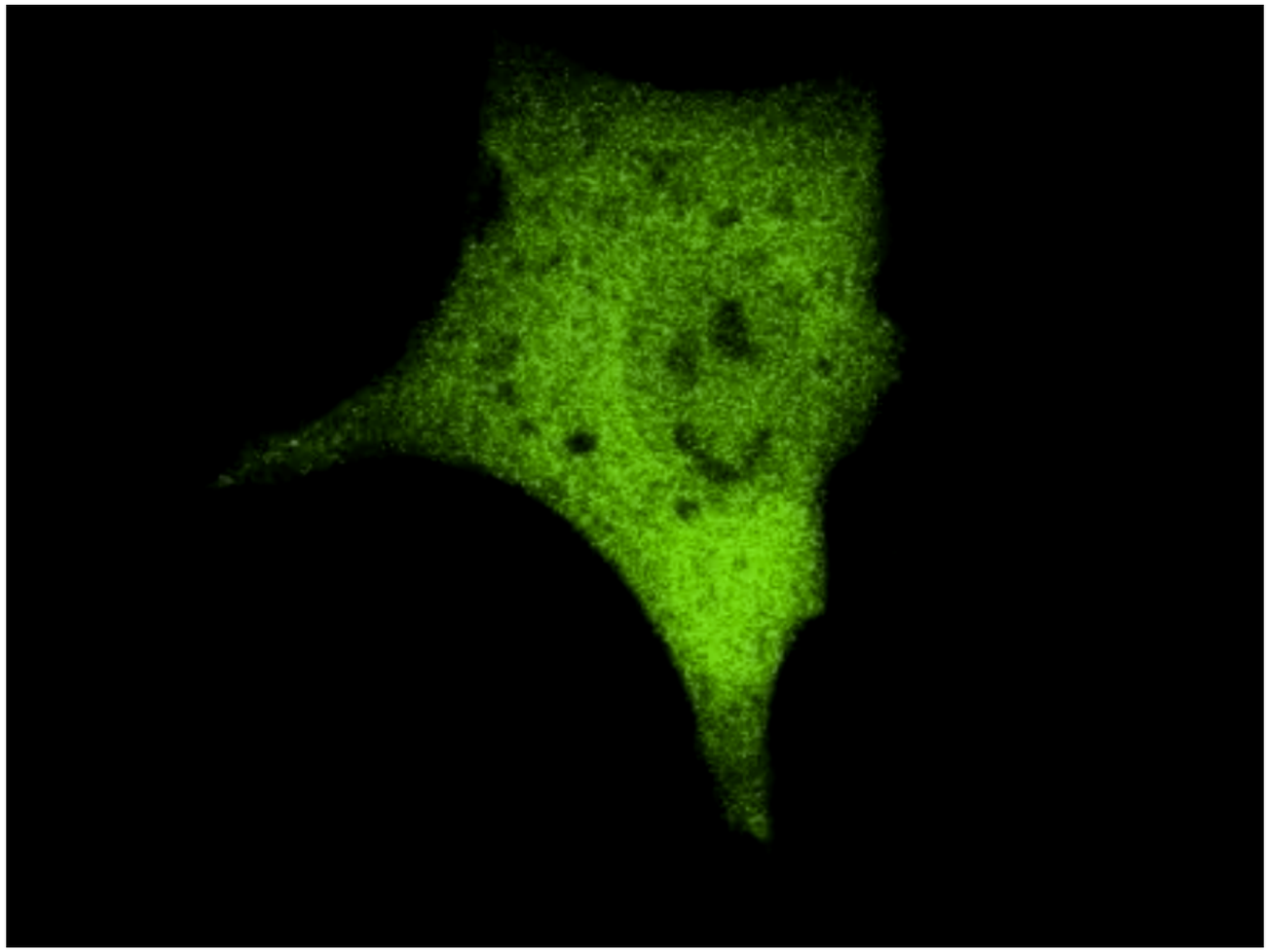
Yixin
What about this?

Zela
Of course. This is a famous symbol from the novel Watchmen, a smiley face pin defaced by a drop of blood. You can still recognize it despite the color change, rotation, scaling, and of course the blood.



Zela
What about this?

Yixin
This is an artificial nanostructure called scaffolded DNA origami. It’s at a scale millions of times smaller than the pin in the last slide. But despite the distortion from the electron microscope… and from being made out of nucleic acid, you can still tell what it is!

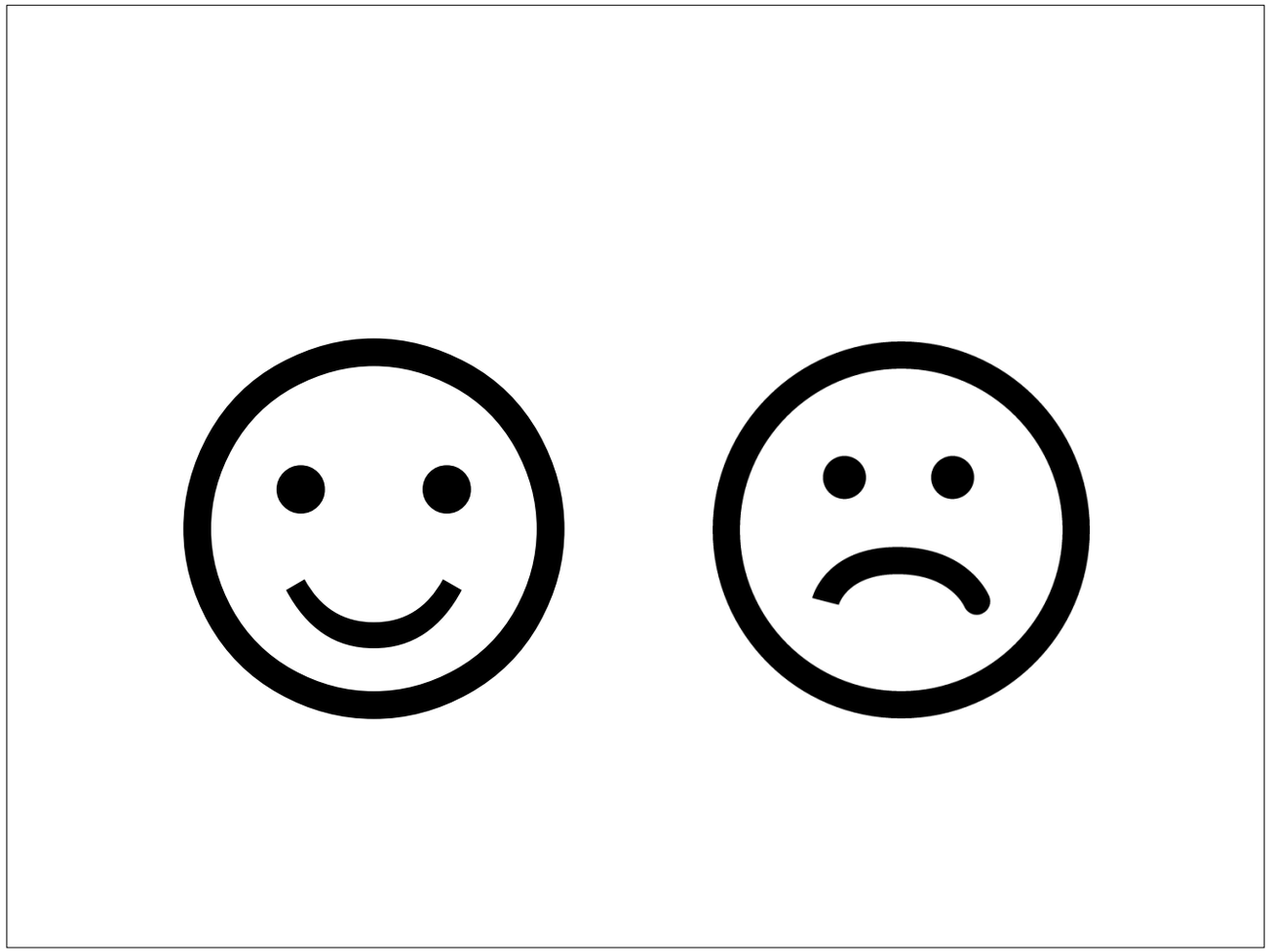


Yixin
This is a cancer cell literally laughing at human beings. Again a completely different scale, extreme color change, texture distortion, and warping of the image. Again, we can tell exactly how smug it’s being.

Zela
Here’s a smile in the sky.

Zela

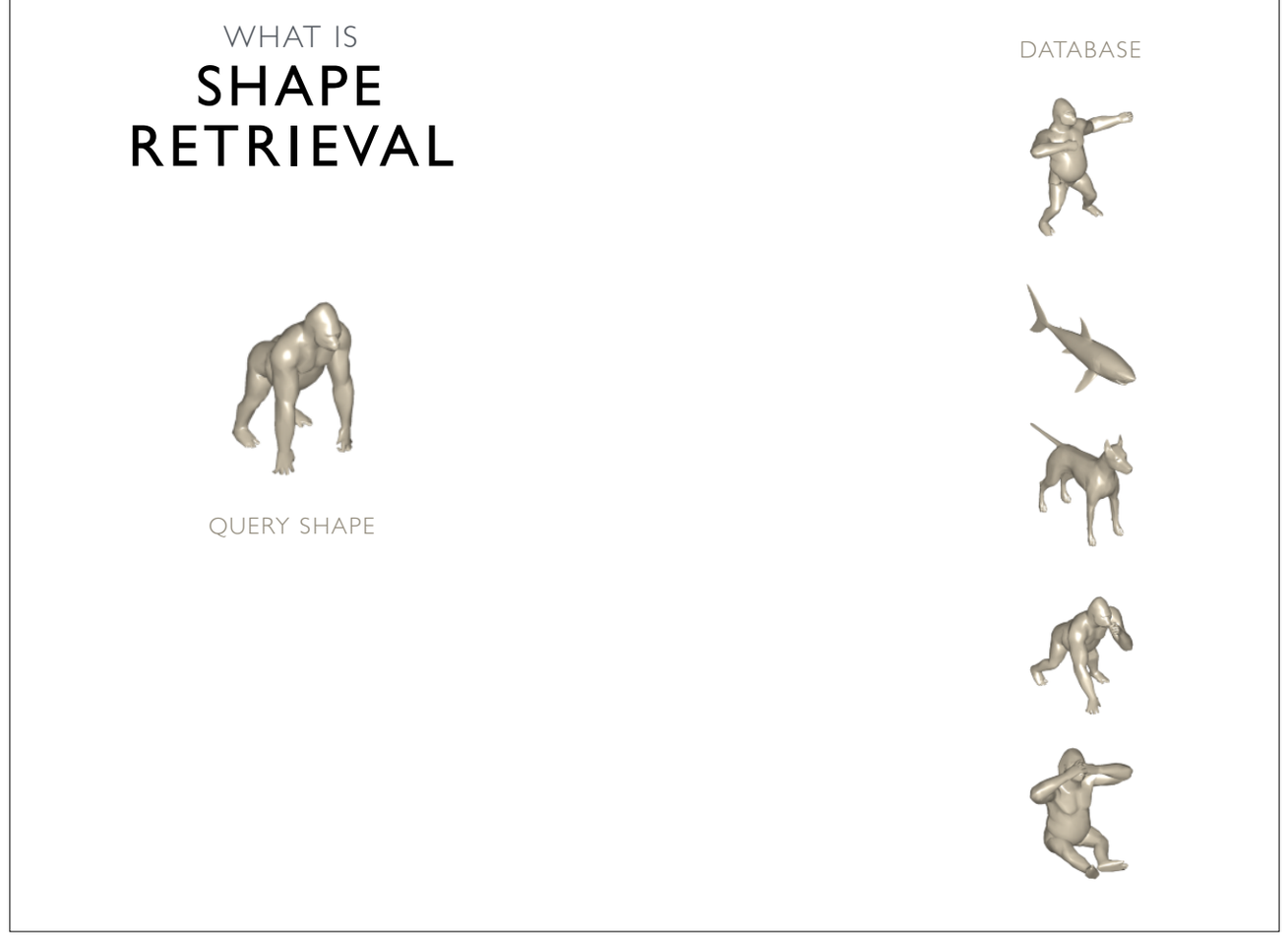
Here’s a smile from outer space, viewed by the Hubble space telescope.



Zela
And yet when we see an unhappy face that looks almost exactly identical to a happy one, we can immediately tell it apart.

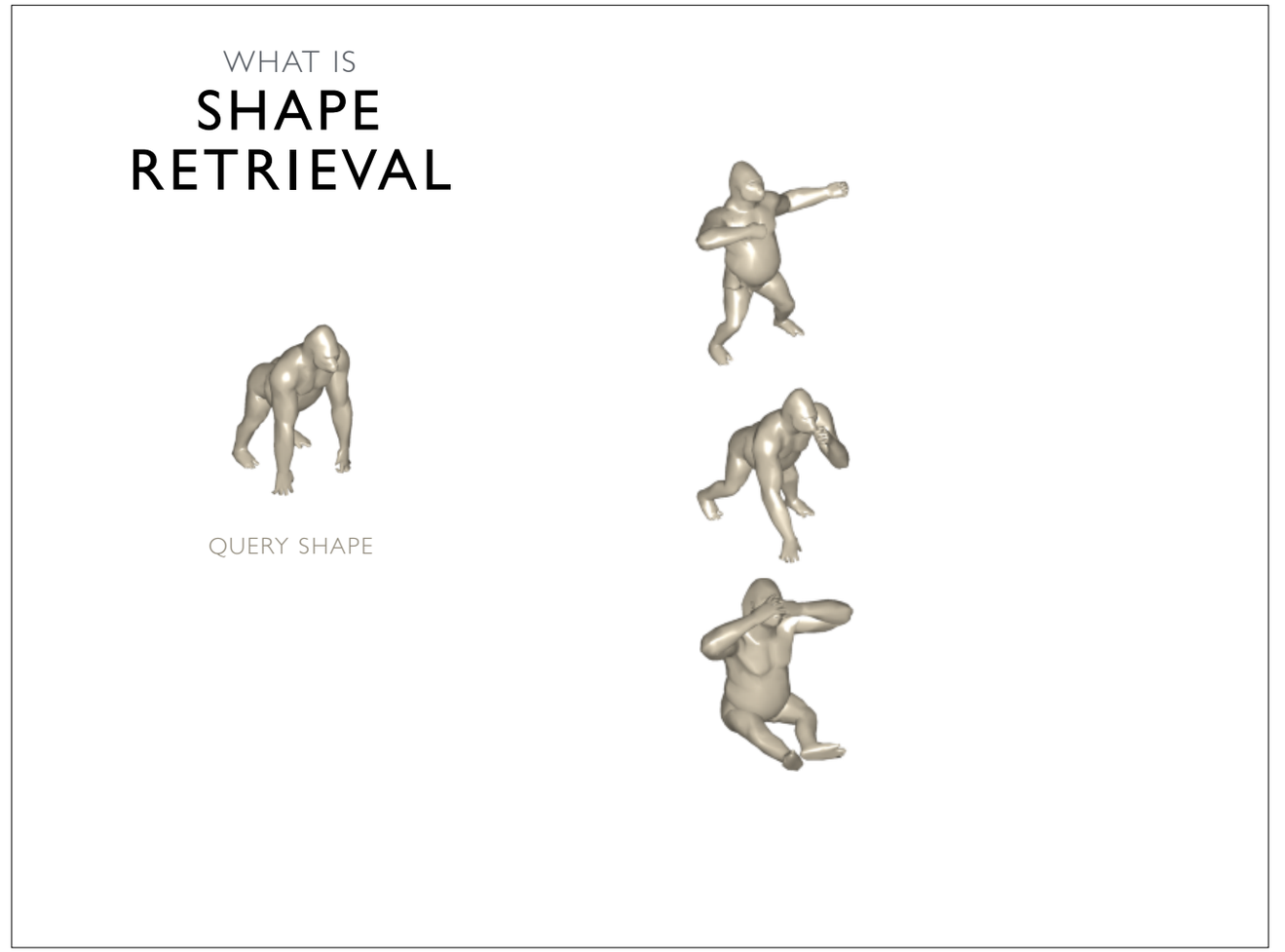
What is it about these images that let us understand their meaning? It’s certainly not color, nor scale, nor texture. What allows us see through any amount of distortions, imperfections, or warping?

It’s shape. If you don’t understand shape, you’re almost blind.



Yixin

Our goal is to allow computers to understand shape like humans naturally do, to see through all the distortions to capture the essence of an object, an important piece of the quest to make machines more intelligent. The question of shape retrieval is: given a shape, how can we find the most similar shapes from our database? How can we understand these shapes using geometry alone, without any textual metadata or context?



Yixin

Our goal is to allow computers to understand shape like humans naturally do, to see through all the distortions to capture the essence of an object, an important piece of the quest to make machines more intelligent. The question of shape retrieval is: given a shape, how can we find the most similar shapes from our database? How can we understand these shapes using geometry alone, without any textual metadata or context?

3D ANIMATION

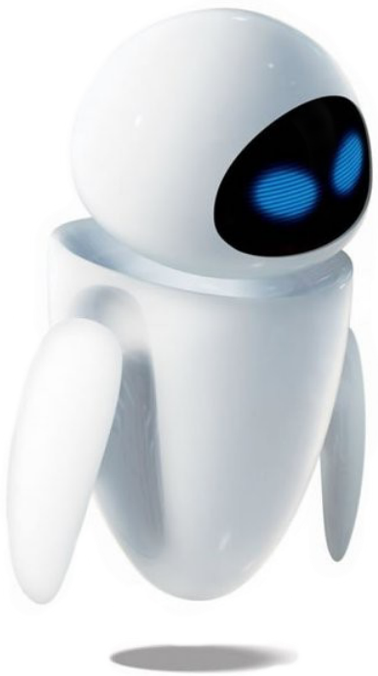
Yixin

One potential application is for use in 3D design and animation. Every billion dollar animated film has millions of 3D shapes that let the story come to life. If you can instantly find the similar ones among those millions of shapes, you can accelerate that creative process tremendously.

3D ANIMATION

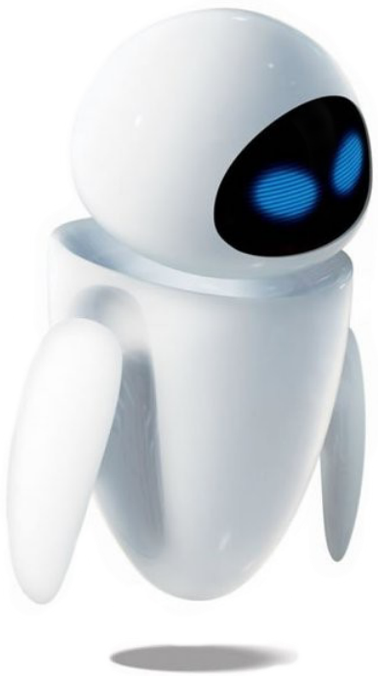
Yixin

One potential application is for use in 3D design and animation. Every billion dollar animated film has millions of 3D shapes that let the story come to life. If you can instantly find the similar ones among those millions of shapes, you can accelerate that creative process tremendously.



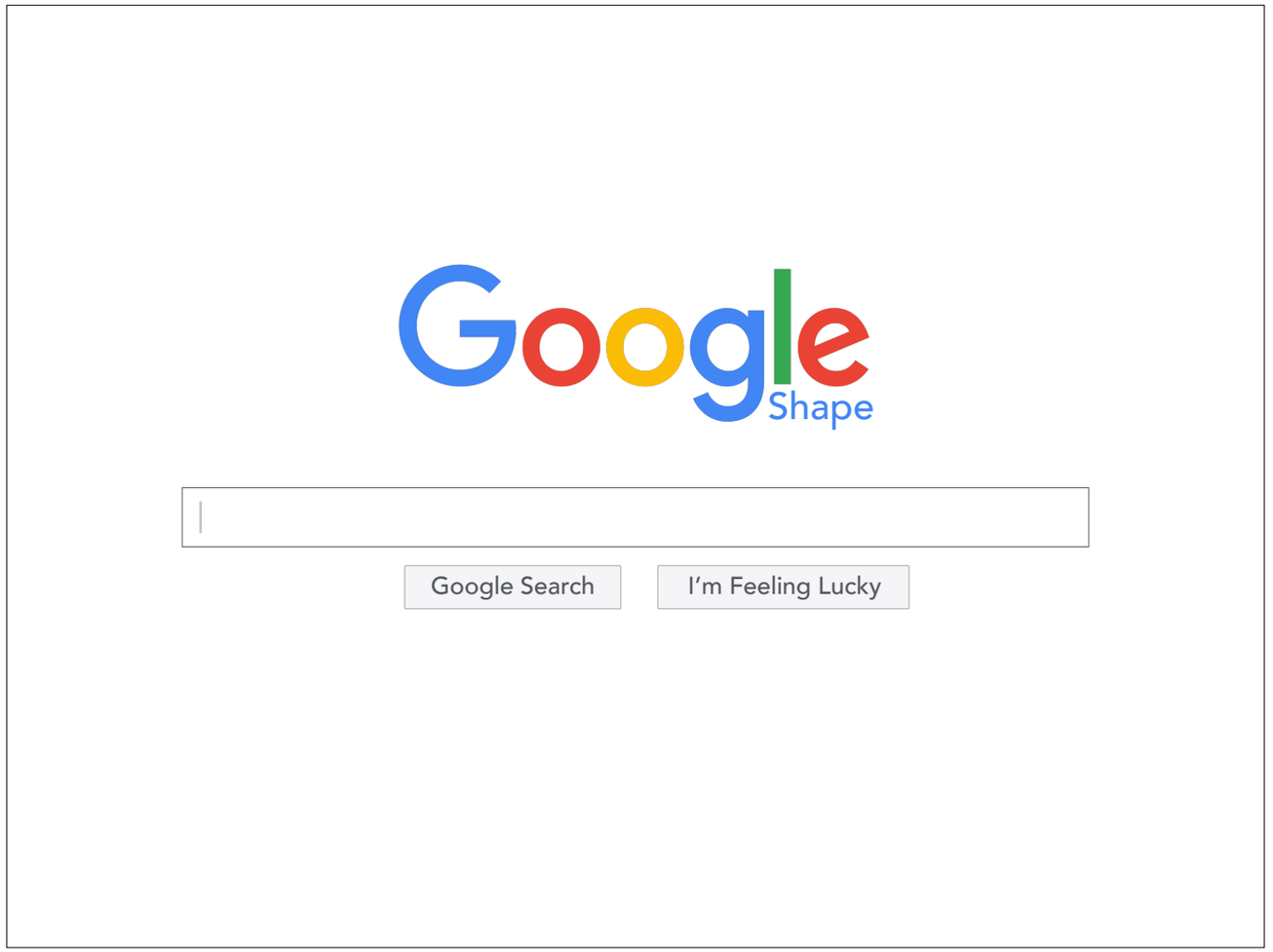
ROBOT
VISION

Zela
The potentials for robot and computer vision are even more exciting. Imagine a robot with with a pair of stereoscopic cameras. It can then create point clouds as it explores its environment. If it can use depth and 3D shape **in addition to** 2D images, it can understand the world it in a more human way.



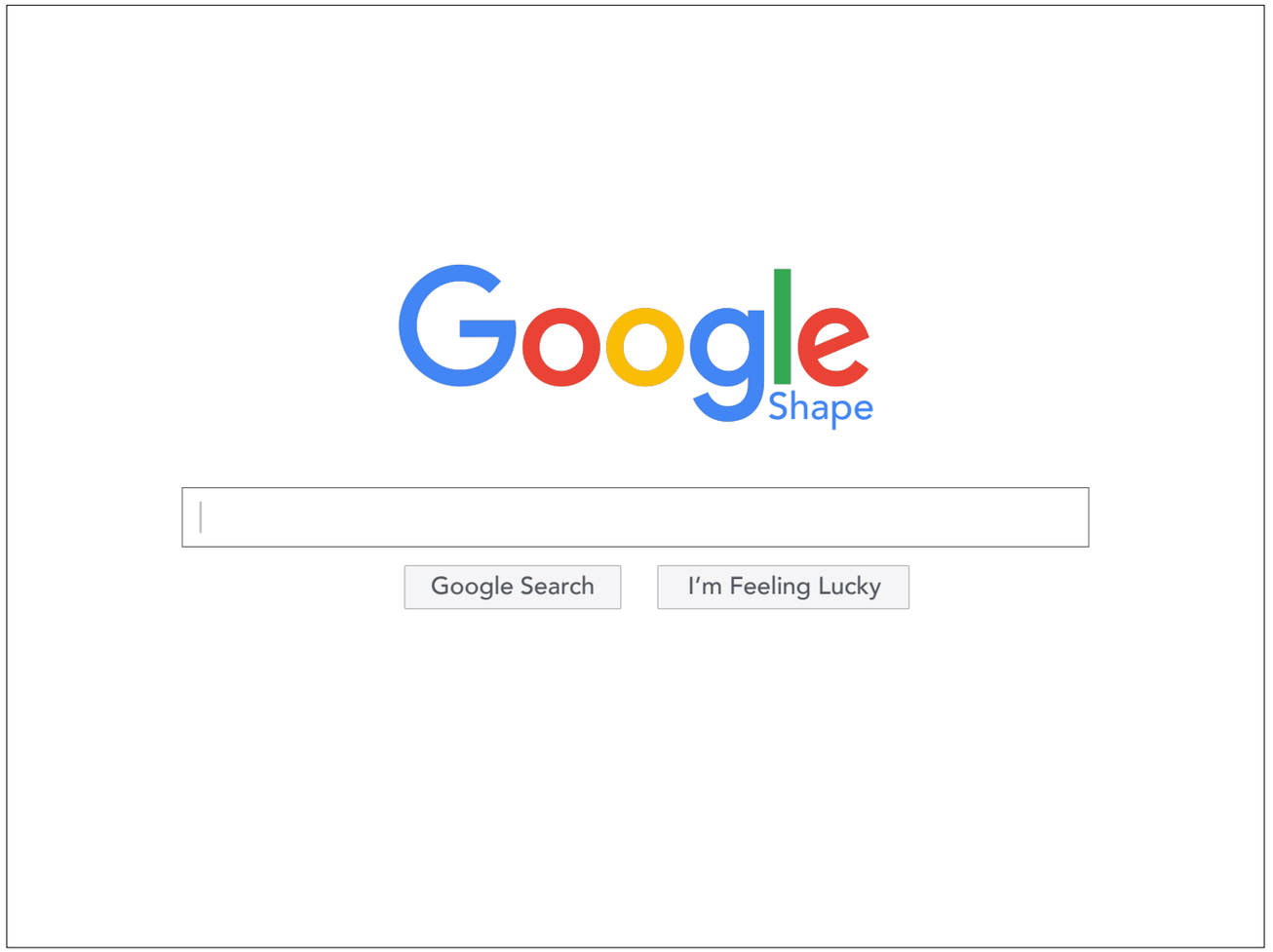
ROBOT
VISION

Zela
The potentials for robot and computer vision are even more exciting. Imagine a robot with with a pair of stereoscopic cameras. It can then create point clouds as it explores its environment. If it can use depth and 3D shape **in addition to** 2D images, it can understand the world it in a more human way.



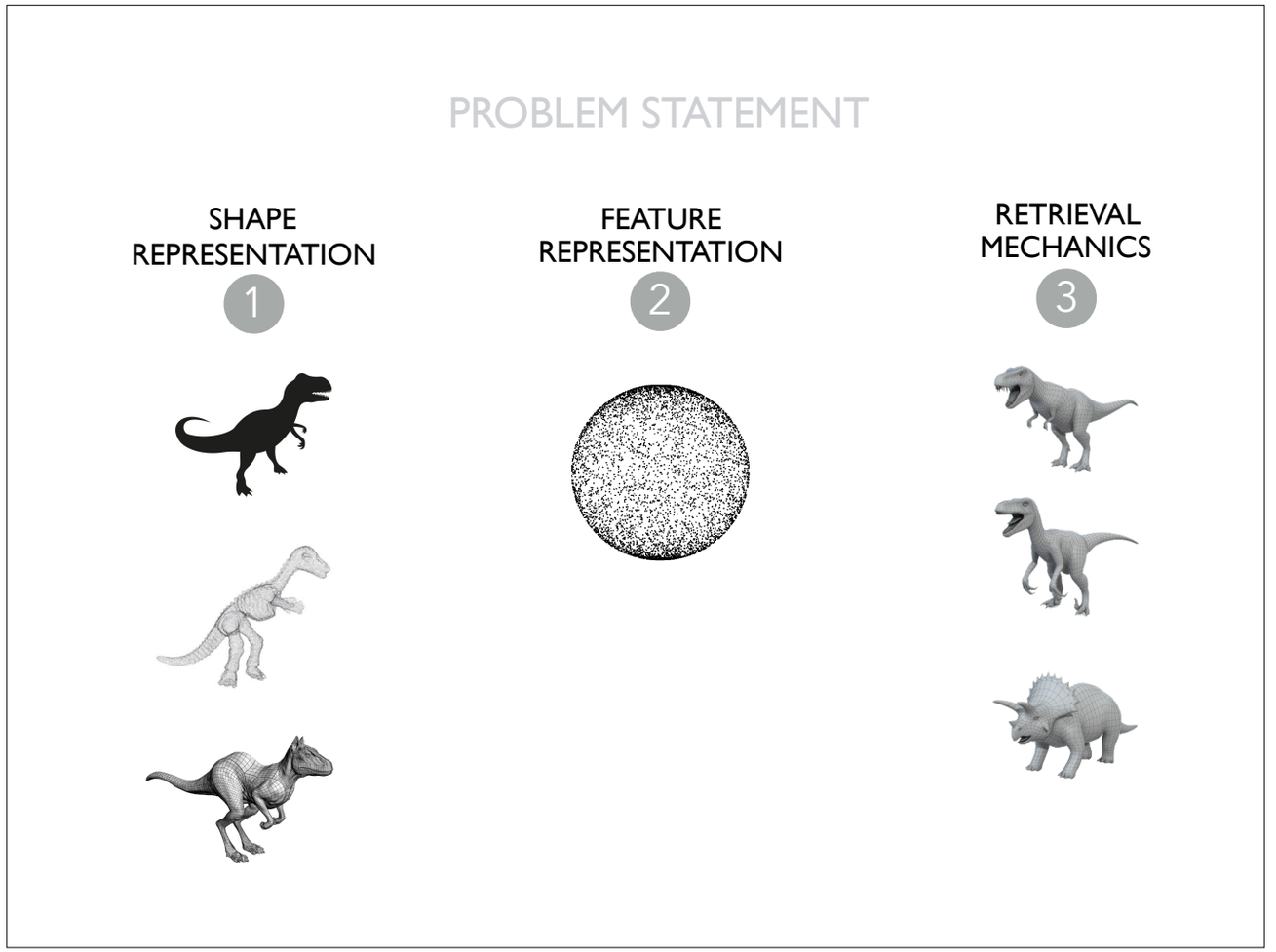
6MIN
Zela
Ultimately, understanding shape is a fundamental problem in computer vision.

Yixin
So if we solve this problem, we get closer to solving other problems we can't even imagine, like how search engines did for the Internet. The goal is to let another dimension of the world be accessible and retrievable by anyone.



6MIN
Zela
Ultimately, understanding shape is a fundamental problem in computer vision.

Yixin
So if we solve this problem, we get closer to solving other problems we can't even imagine, like how search engines did for the Internet. The goal is to let another dimension of the world be accessible and retrievable by anyone.

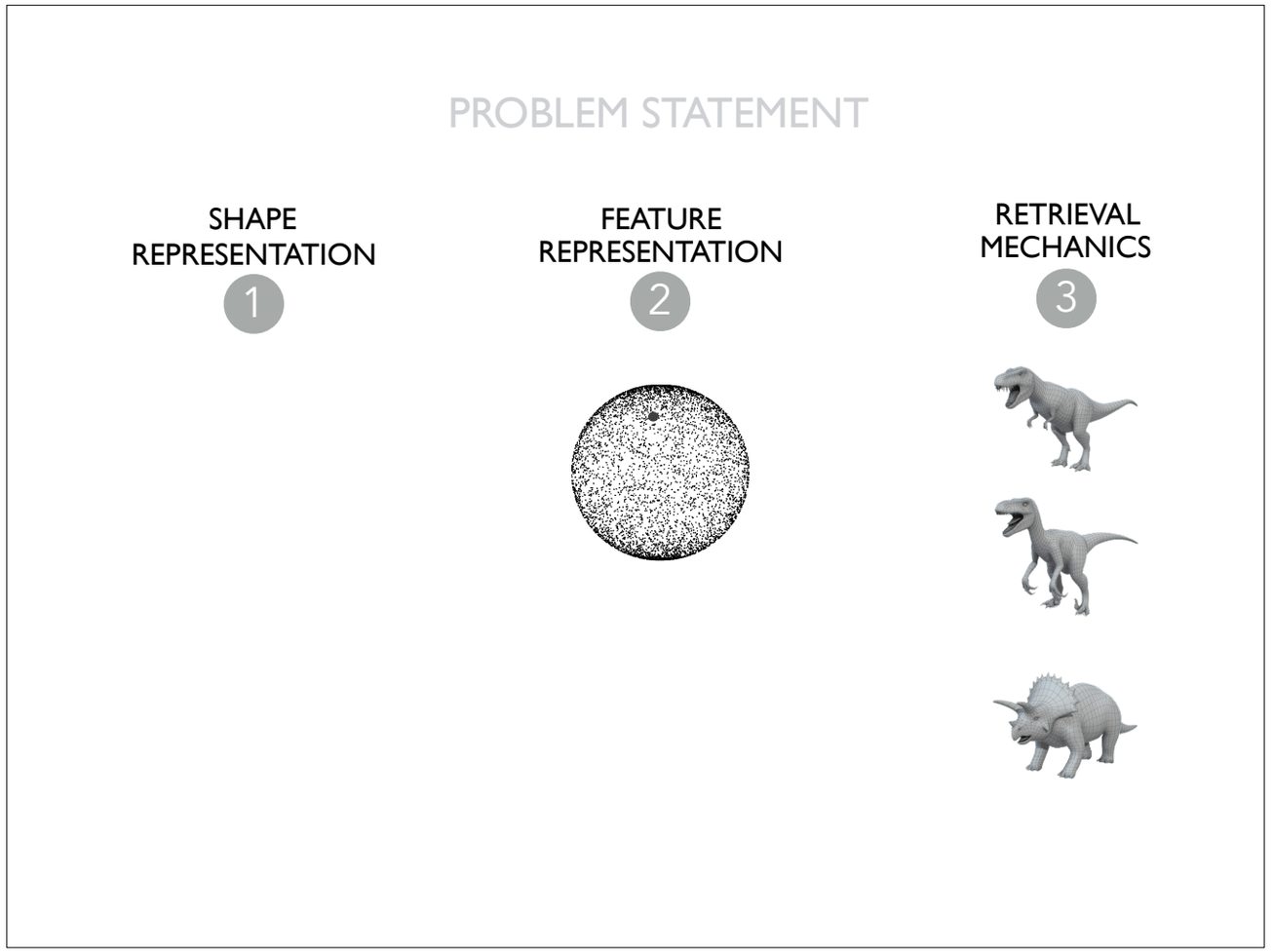


Yixin

- Shape representation
    - 2d points
    - 3d point cloud
    - 3d mesh
- We map this onto our feature representation, which is constrained to the surface of a unit hypersphere
    - Goal is to capture the essence of the data
- With this nice feature representation, we can bring all of our machine learning techniques
- Our main work is retrieval: given a completely new query shape, what are the ones closest to it?
- Let’s first look at the feature representation.

- Shape representation given as 2D point sets, 3D point sets, or 3D meshes
- Unsupervised data —> have to classify
- From SR we transform raw data into a FR that can be utilized in Shape Classification
- Not focused on feature representation. Focus on retrieval.

Transition: Important point, SR to FR onto Hypersphere.



Yixin

- Shape representation
    - 2d points
    - 3d point cloud
    - 3d mesh
- We map this onto our feature representation, which is constrained to the surface of a unit hypersphere
    - Goal is to capture the essence of the data
- With this nice feature representation, we can bring all of our machine learning techniques
- Our main work is retrieval: given a completely new query shape, what are the ones closest to it?
- Let’s first look at the feature representation.

- Shape representation given as 2D point sets, 3D point sets, or 3D meshes
- Unsupervised data —> have to classify
- From SR we transform raw data into a FR that can be utilized in Shape Classification
- Not focused on feature representation. Focus on retrieval.

Transition: Important point, SR to FR onto Hypersphere.

FEATURE REPRESENTATION

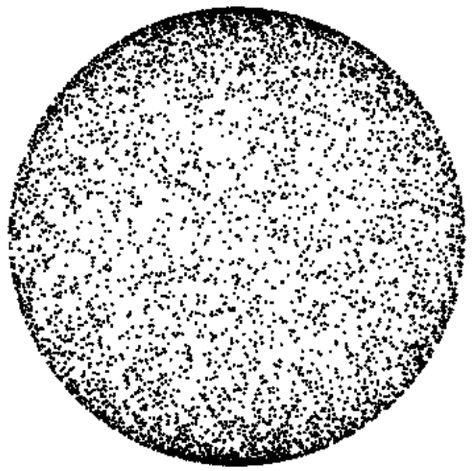
ISOMETRY INVARIANCE

Yixin

- Nice properties of our feature representation:
  - Brings us to unit hypersphere geometry, simple distance metrics
  - Isometry invariance: isometries map to the same point

FEATURE REPRESENTATION

ISOMETRY INVARIANCE

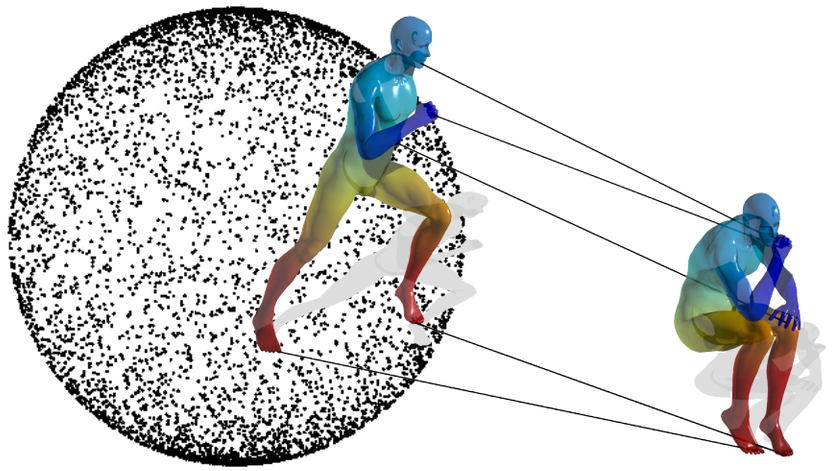


Yixin

- Nice properties of our feature representation:
  - Brings us to unit hypersphere geometry, simple distance metrics
  - Isometry invariance: isometries map to the same point

FEATURE REPRESENTATION

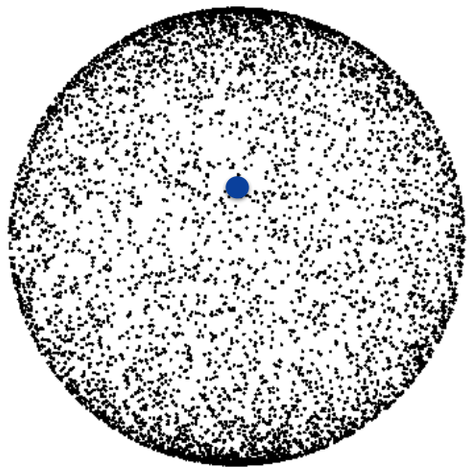
ISOMETRY INVARIANCE



Yixin

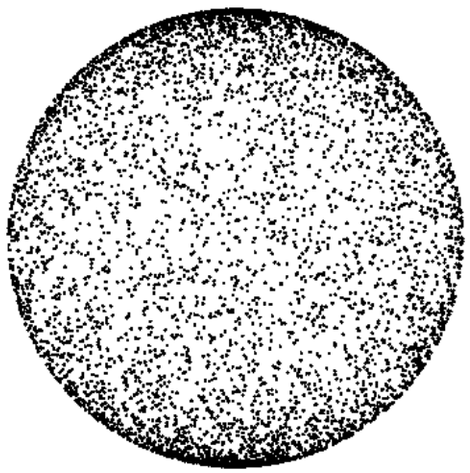
- Nice properties of our feature representation:
  - Brings us to unit hypersphere geometry, simple distance metrics
  - Isometry invariance: isometries map to the same point

FEATURE REPRESENTATION



Yixin

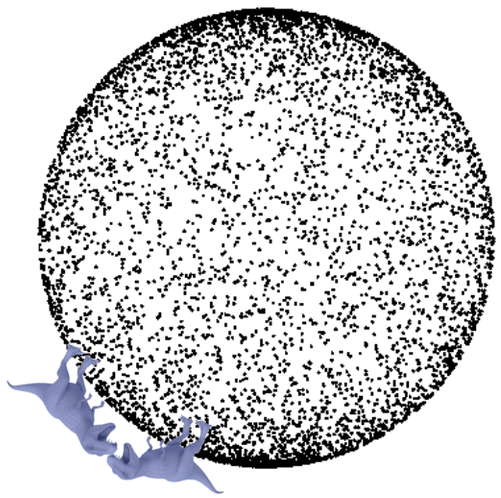
- Nice properties of our feature representation:
  - Brings us to unit hypersphere geometry, simple distance metrics
  - Isometry invariance: isometries map to the same point



COMPARING SHAPES ON THE UNIT HYPERSPHERE

Zela

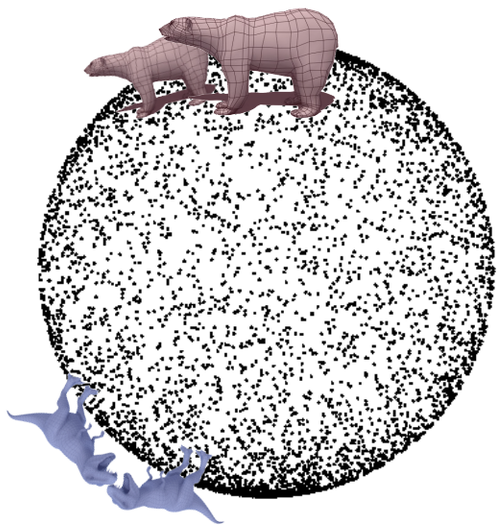
- Distances between points signify similarity
- Form cluster representations of shape classes, e.g. dinosaurs, bears
- Given a new query, we can classify it based on our previous clusters



COMPARING SHAPES ON THE UNIT HYPERSPHERE

Zela

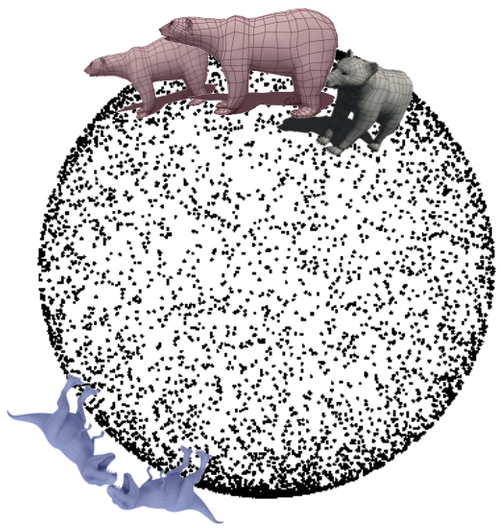
- Distances between points signify similarity
- Form cluster representations of shape classes, e.g. dinosaurs, bears
- Given a new query, we can classify it based on our previous clusters



COMPARING SHAPES ON THE UNIT HYPERSPHERE

Zela

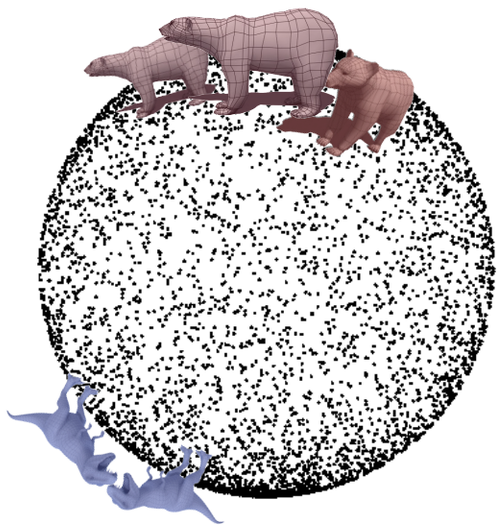
- Distances between points signify similarity
- Form cluster representations of shape classes, e.g. dinosaurs, bears
- Given a new query, we can classify it based on our previous clusters



COMPARING SHAPES ON THE UNIT HYPERSPHERE

Zela

- Distances between points signify similarity
- Form cluster representations of shape classes, e.g. dinosaurs, bears
- Given a new query, we can classify it based on our previous clusters

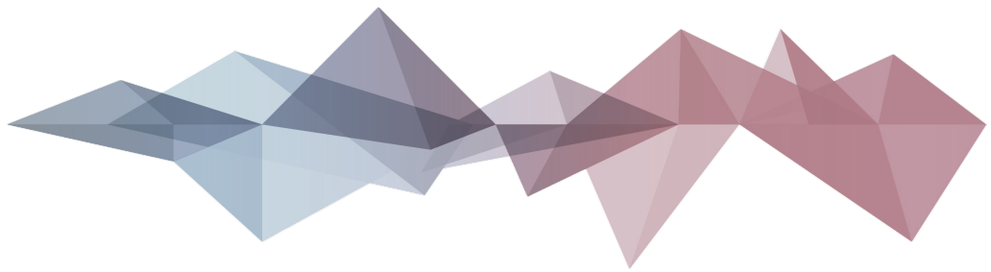


COMPARING SHAPES ON THE UNIT HYPERSPHERE

Zela

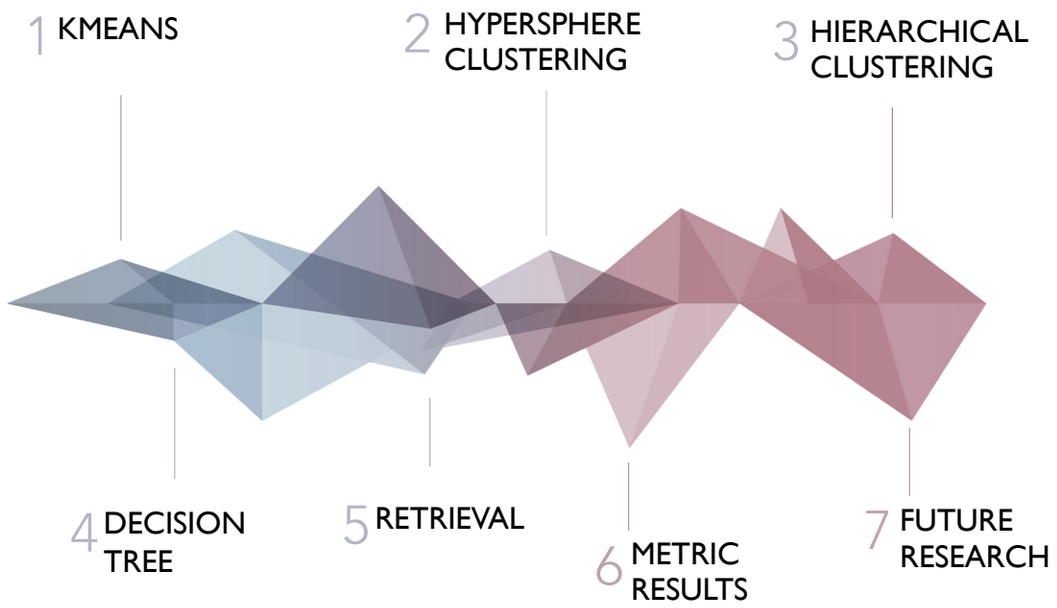
- Distances between points signify similarity
- Form cluster representations of shape classes, e.g. dinosaurs, bears
- Given a new query, we can classify it based on our previous clusters

ROADMAP FOR RETRIEVAL



- Zela

ROADMAP FOR RETRIEVAL



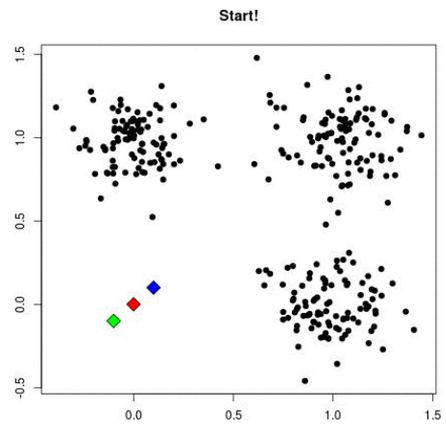
- Zela

UNDERSTANDING
KMEANS

KMEANS CLUSTERING

EUCLIDEAN SPACE

1: generate an initial set of *k* means
2: **while** not converged **do**
3: assign points to nearest centroid
4: calculate new centroids of the points in the cluster
5: **end while**



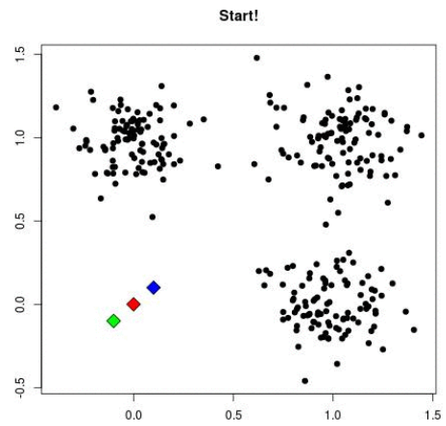
Zela
We’re all familiar with the traditional k-means clustering algorithm on a flat space. It picks some initial means, reclusters each datapoint to the closest mean, and recomputes new means, repeating until convergence.

UNDERSTANDING
KMEANS

KMEANS CLUSTERING

EUCLIDEAN SPACE

1: generate an initial set of *k* means
2: **while** not converged **do**
3: assign points to nearest centroid
4: calculate new centroids of the points in the cluster
5: **end while**



Zela
We’re all familiar with the traditional k-means clustering algorithm on a flat space. It picks some initial means, reclusters each datapoint to the closest mean, and recomputes new means, repeating until convergence.

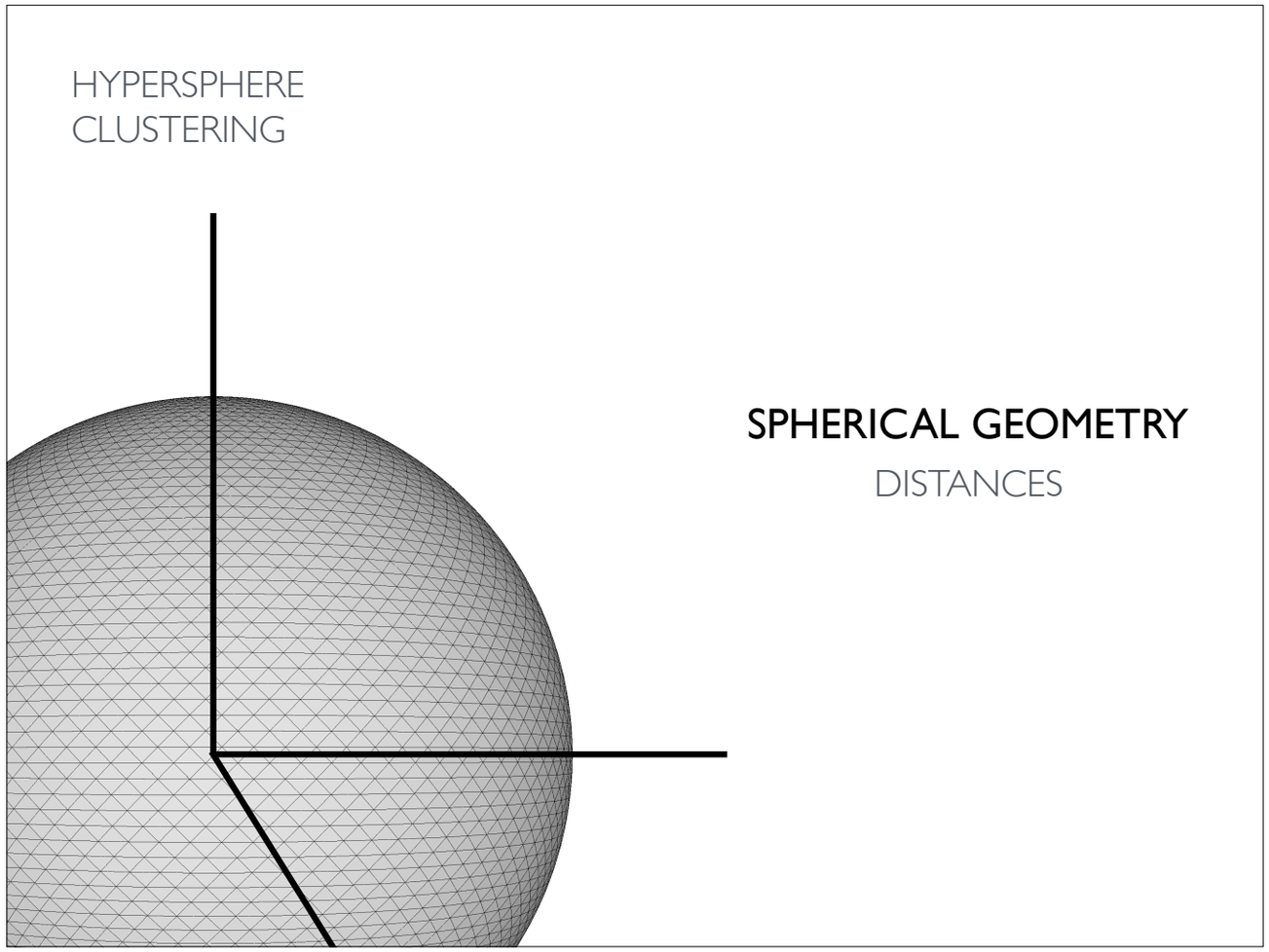
HYPERSPHERE
CLUSTERING

SPHERICAL GEOMETRY

DISTANCES

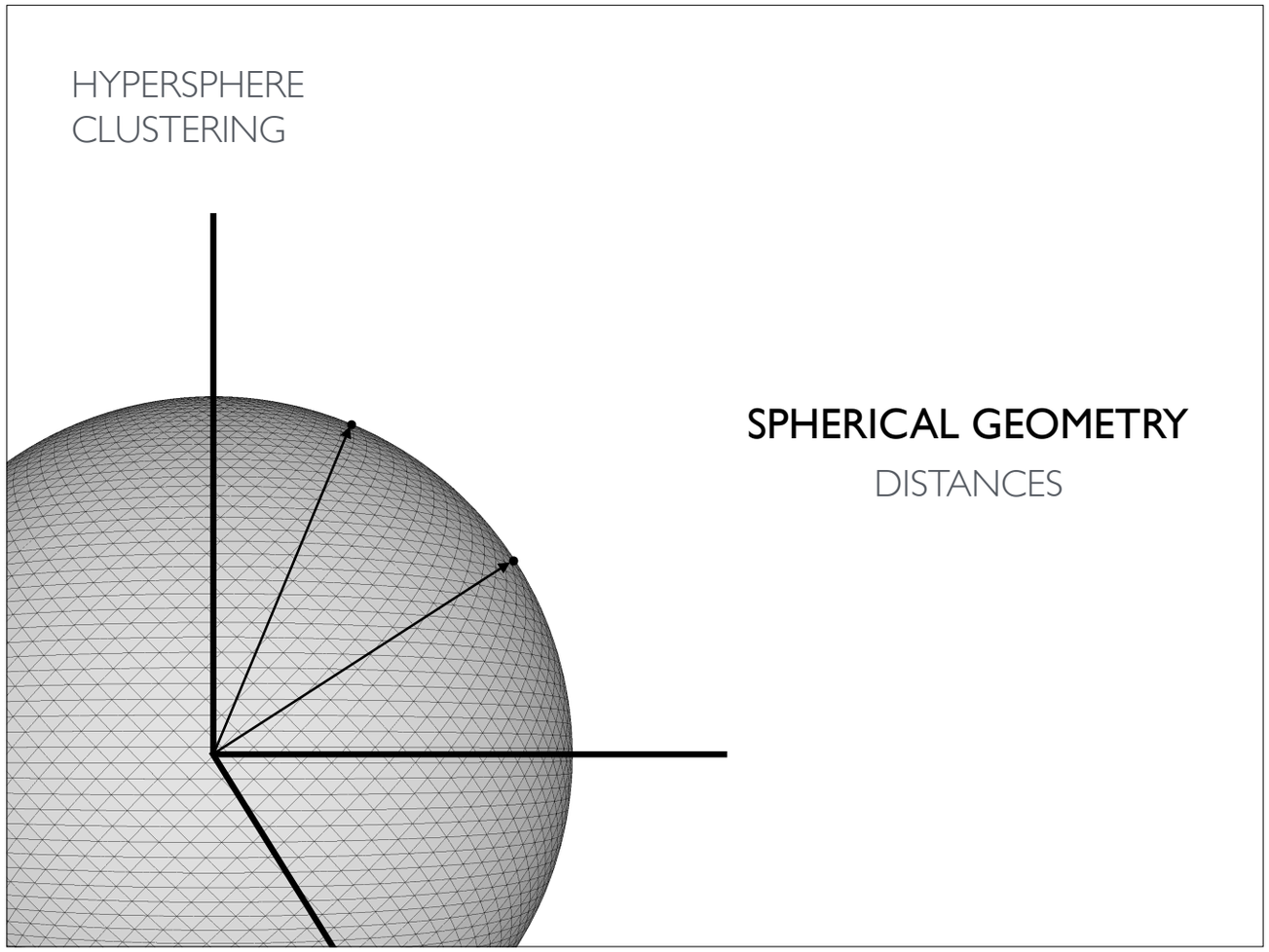
Zela

- However, our problem is on a hypersphere, which is a curved surface.
- First, we need to figure out how to find distances between two points
  - Use the angle between them



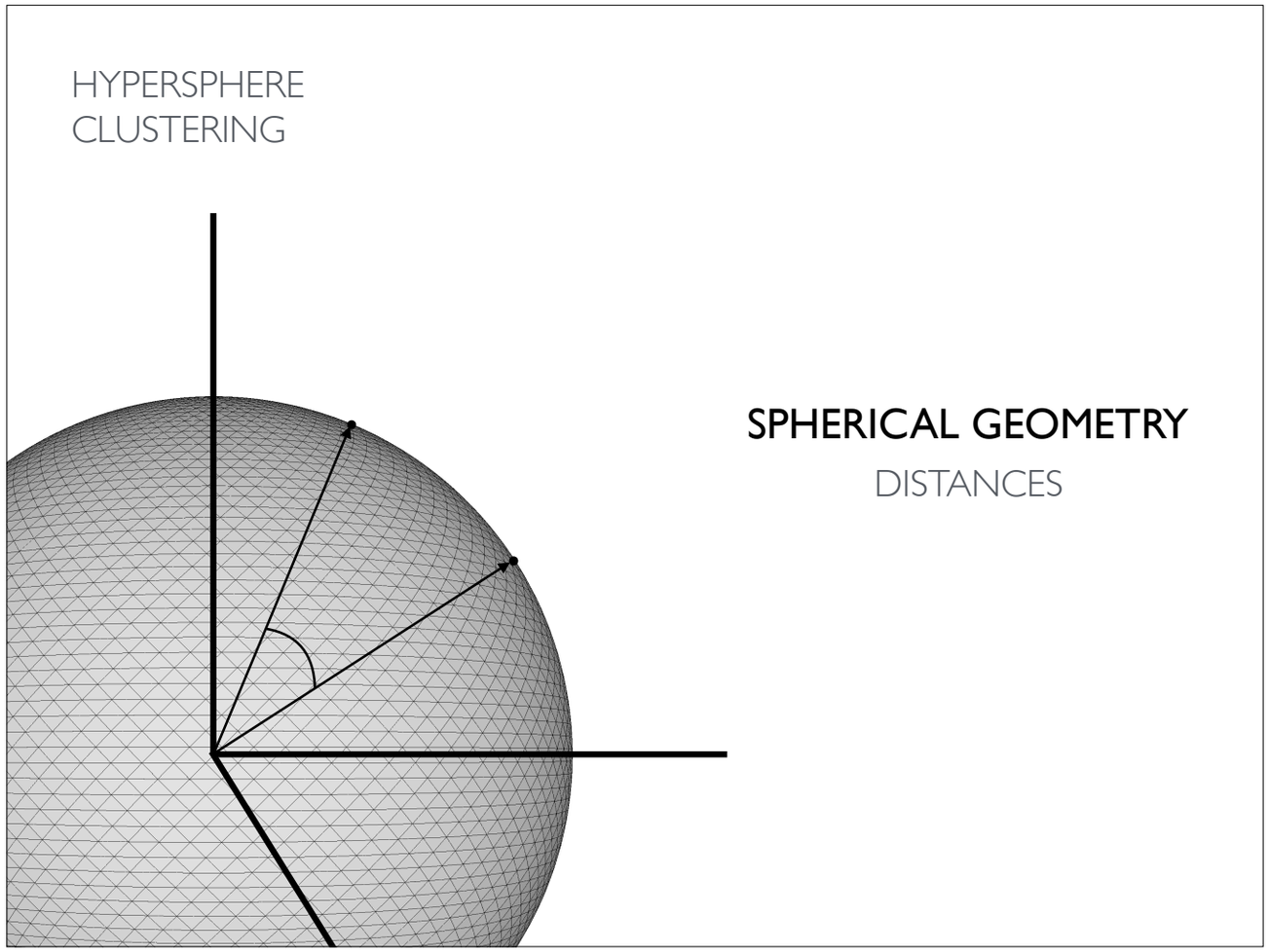
Zela

- However, our problem is on a hypersphere, which is a curved surface.
- First, we need to figure out how to find distances between two points
  - Use the angle between them



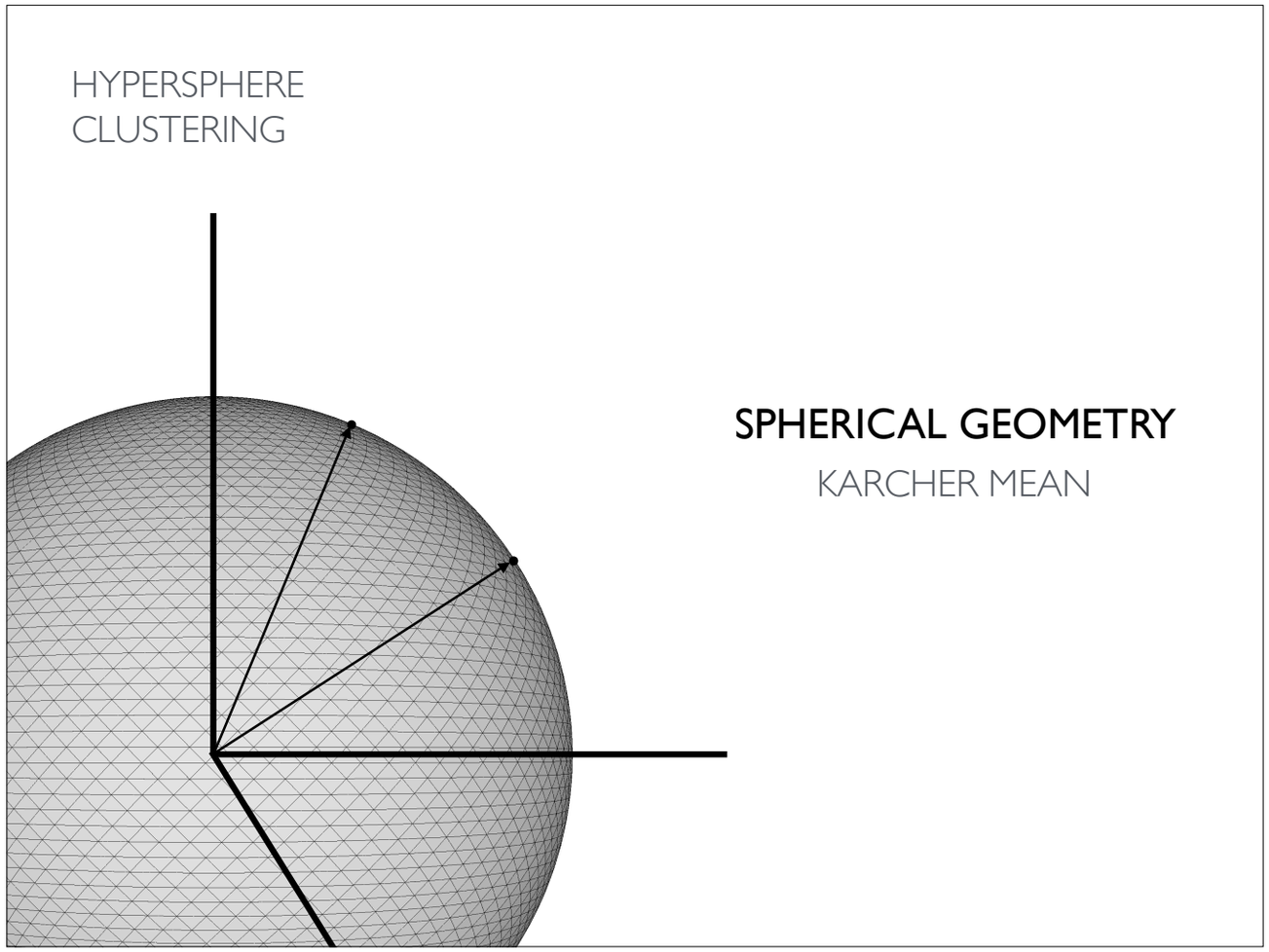
Zela

- However, our problem is on a hypersphere, which is a curved surface.
- First, we need to figure out how to find distances between two points
  - Use the angle between them



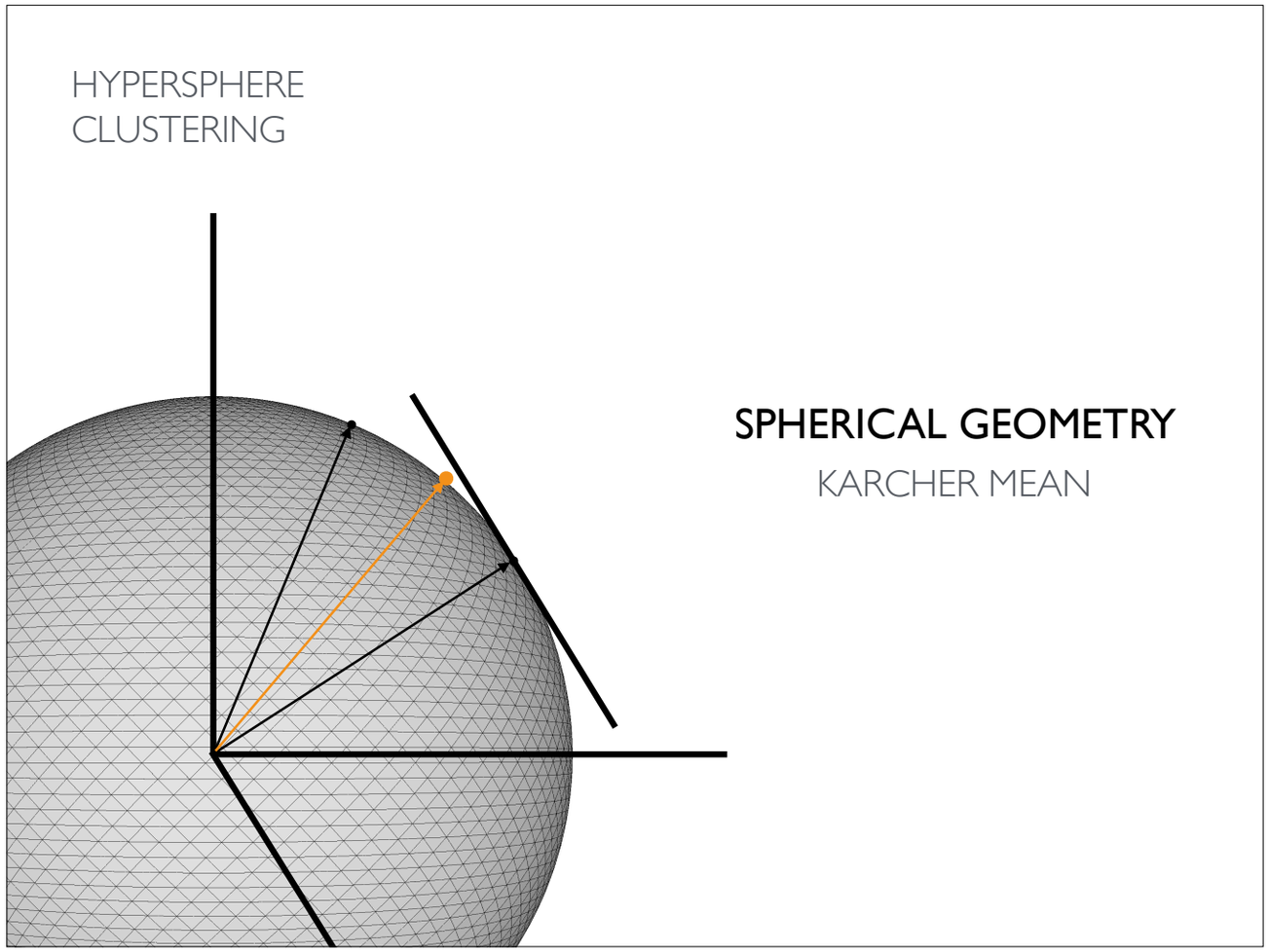
Zela

- However, our problem is on a hypersphere, which is a curved surface.
- First, we need to figure out how to find distances between two points
  - Use the angle between them



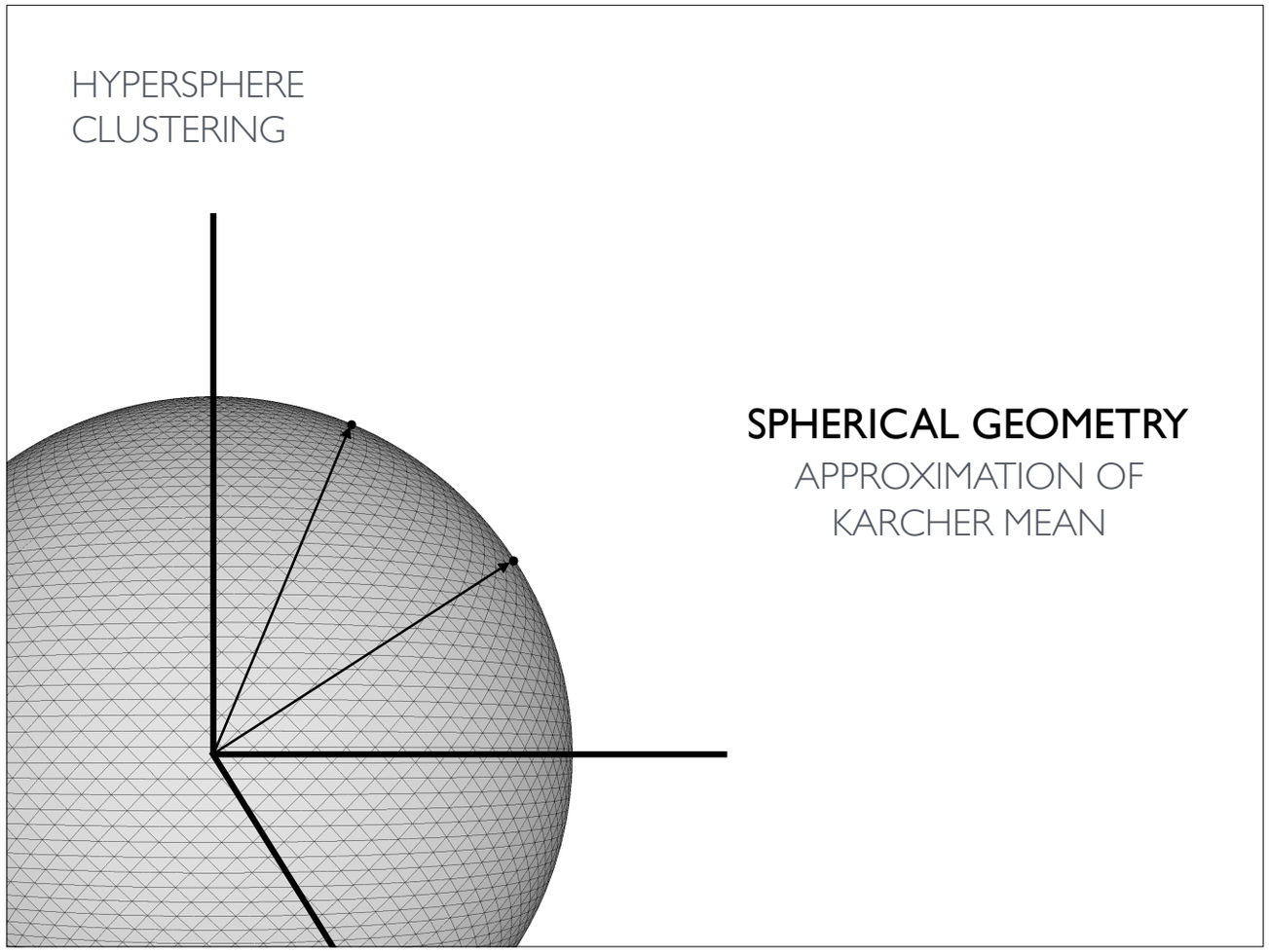
Yixin

- Second, we need to figure out how to find means
- Can’t simply take the Euclidean mean: use Karcher mean from differential geometry, which lets you take means on curved surfaces



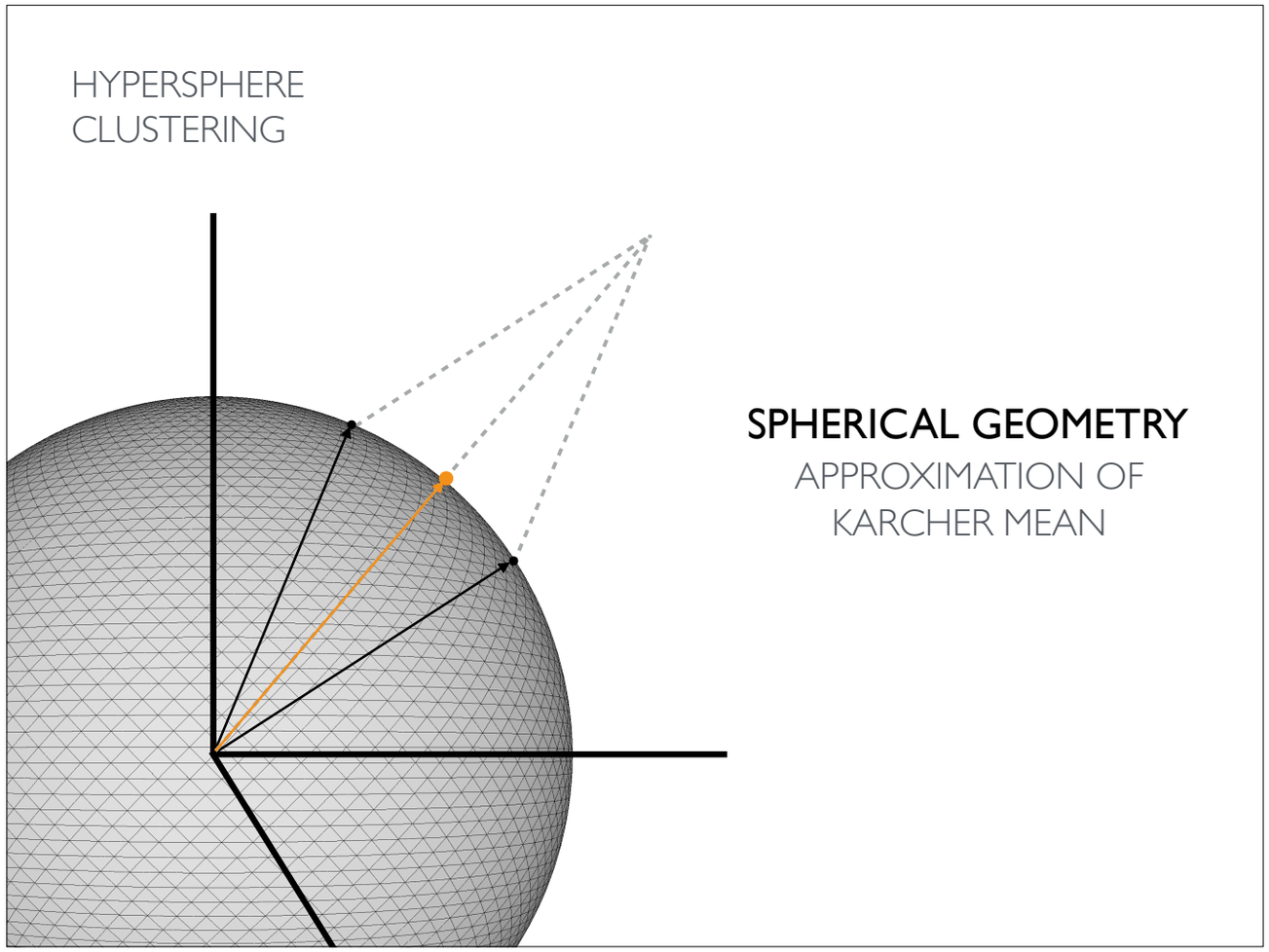
Yixin

- Second, we need to figure out how to find means
- Can’t simply take the Euclidean mean: use Karcher mean from differential geometry, which lets you take means on curved surfaces



Zela

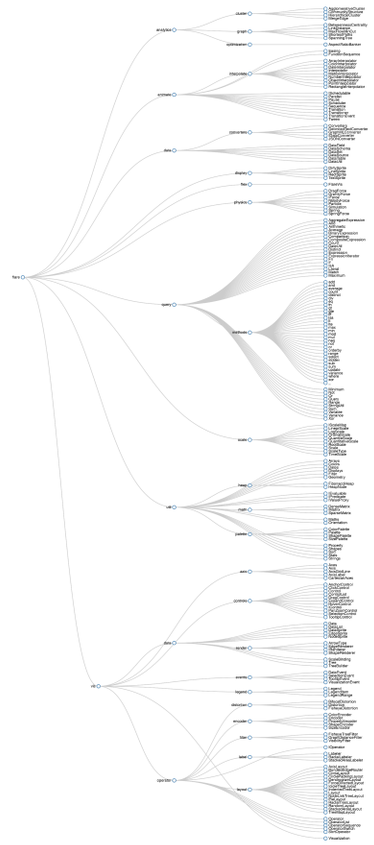
- We used a good approximation that avoids the differential geometry:
- This second method finds a centroid using the normalized gravity center. It adds the two vectors and then normalizes it to map back onto the unit hypersphere



Zela

- We used a good approximation that avoids the differential geometry:
- This second method finds a centroid using the normalized gravity center. It adds the two vectors and then normalizes it to map back onto the unit hypersphere

HIERARCHICAL CLUSTERING



Yixin 10MIN

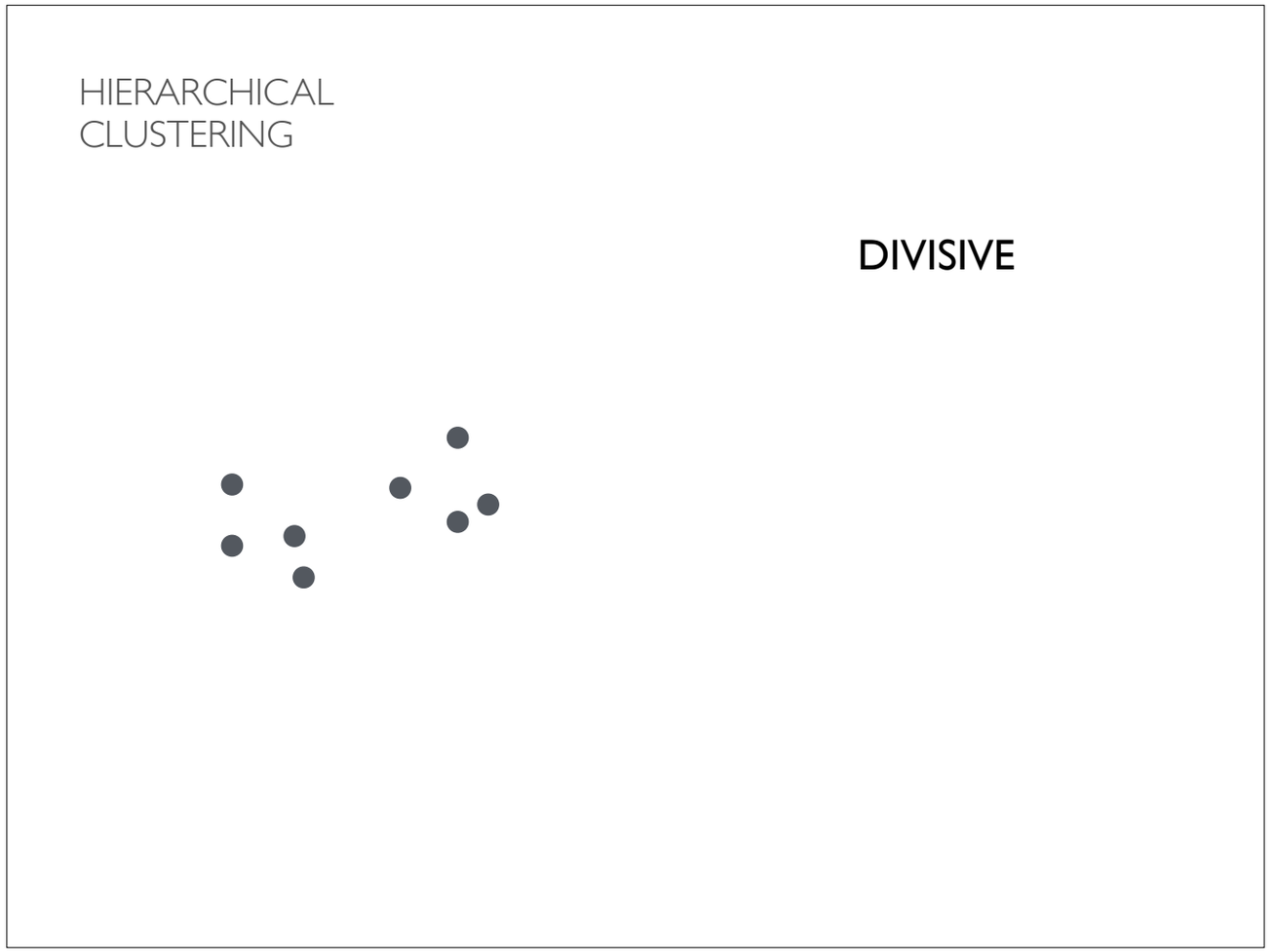
- Now we know how to cluster on the surface
- Shows organization and relationship among clusters
- We explored hierarchical clustering as one efficient way to do this
    - Cluster data points
    - Then recursively cluster those clusters
- There are two algorithms to construct hierarchical clusters

Add graphic that looks like a dendrogram. Epic intro

Yixin

- Multilevel hierarchy representing a group of data
- Shows organization and relationship among clusters with a dendrogram
- Two approaches: Divisive and Agglomerative
- Linkage criterion: Determines how to link or split clusters. Using centroid linkage involves taking the average of all points in a cluster and assigning that value as the mean centroid. Then the centroids with the shortest distances are merged together.

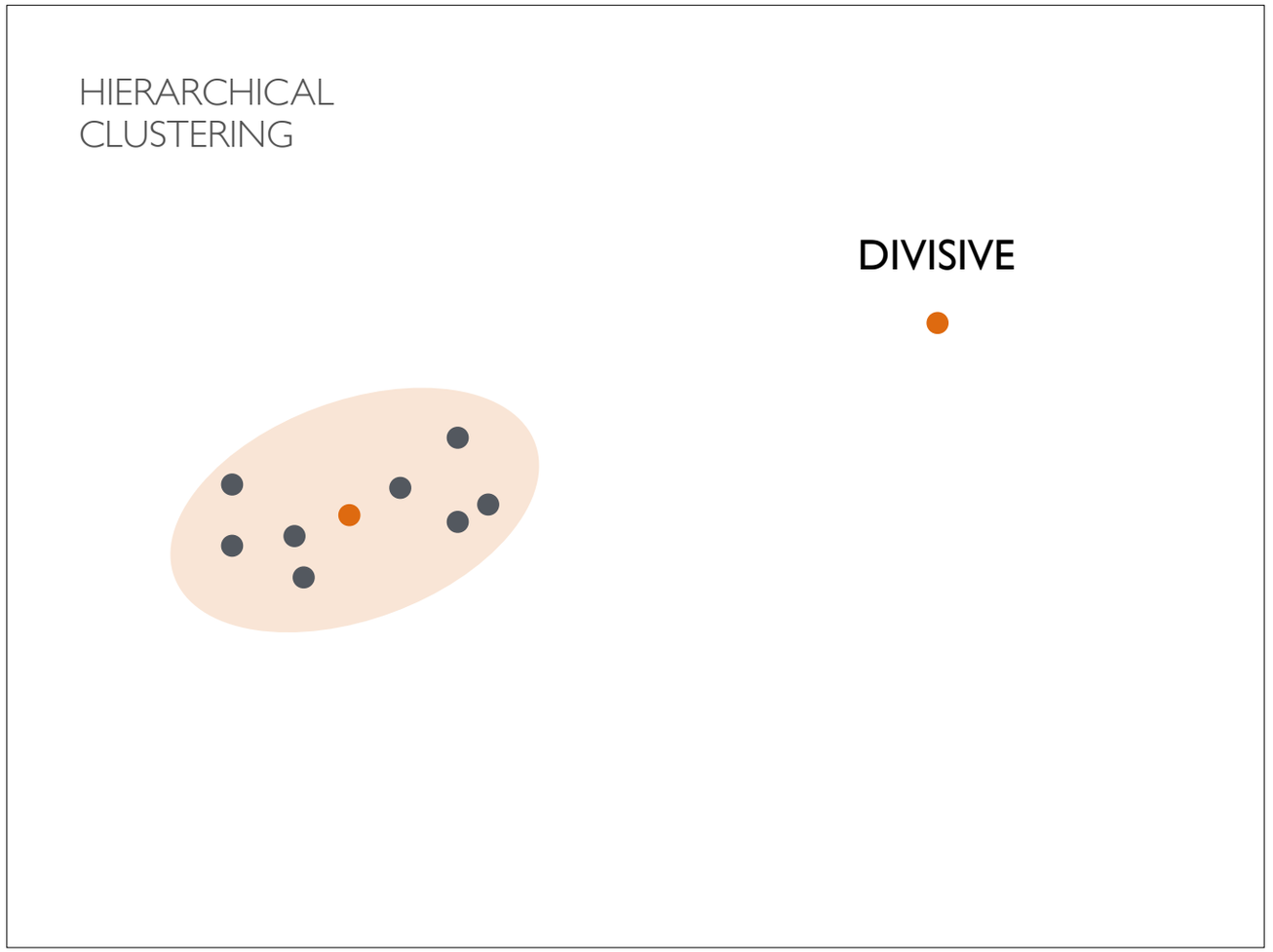
Add graphic that looks like a dendrogram. Epic intro



Zela

Divisive hierarchical clustering is thought of as a top down approach. This takes in a data set as one large cluster with a single centroid (animation). These nodes are important because they are the average representation of the points beneath it. We'll keep track of nodes on the tree to the right. This, then branches out and splits the cluster into two more clusters with its own average representation until it reaches the individual observations.

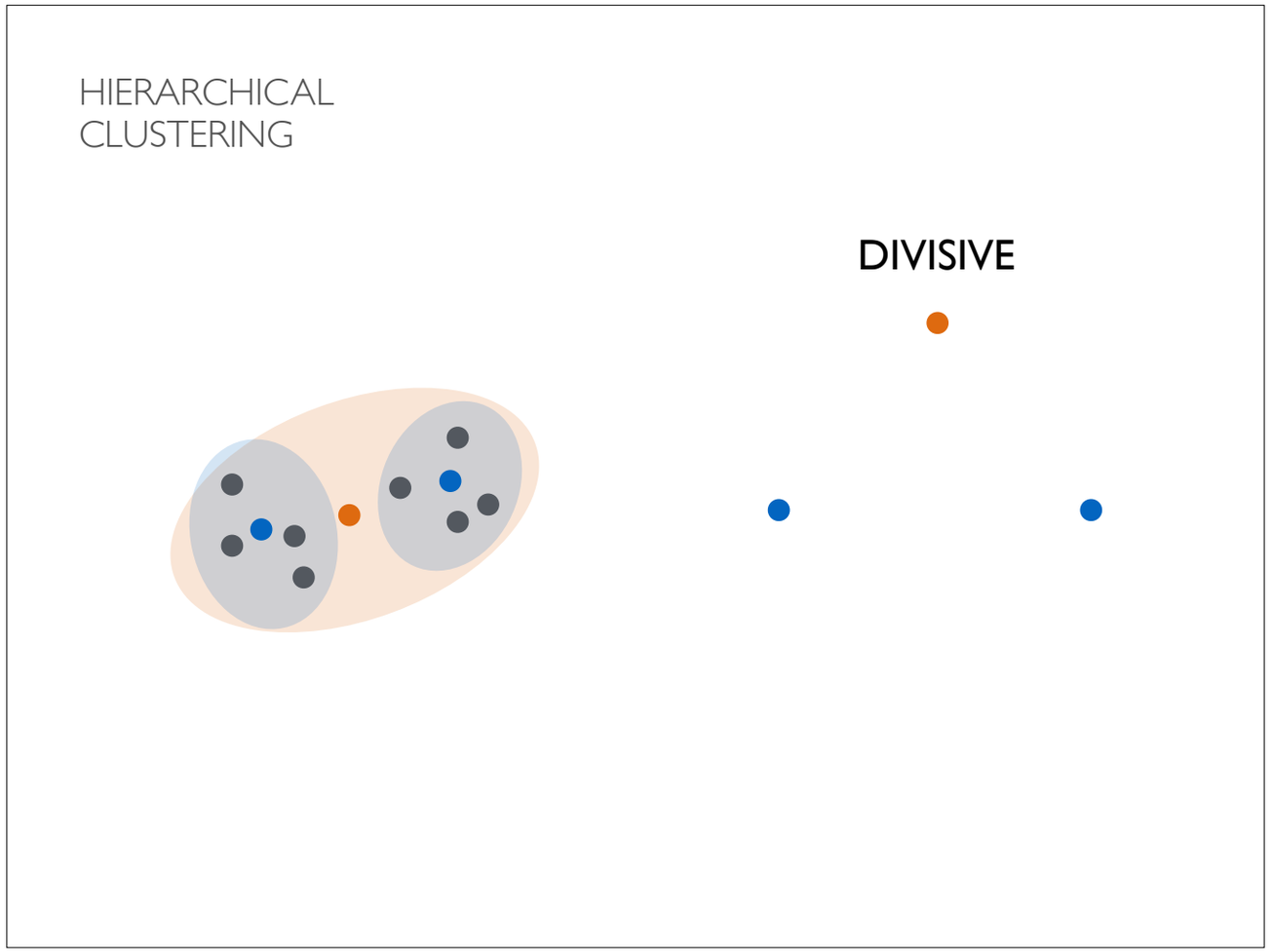
- Top Down Approach
- This takes in a data set as one large cluster with a single centroid. Then it recursively splits into two until it reaches the single points.
- Each centroid acts as a node



Zela

Divisive hierarchical clustering is thought of as a top down approach. This takes in a data set as one large cluster with a single centroid (animation). These nodes are important because they are the average representation of the points beneath it. We'll keep track of nodes on the tree to the right. This, then branches out and splits the cluster into two more clusters with its own average representation until it reaches the individual observations.

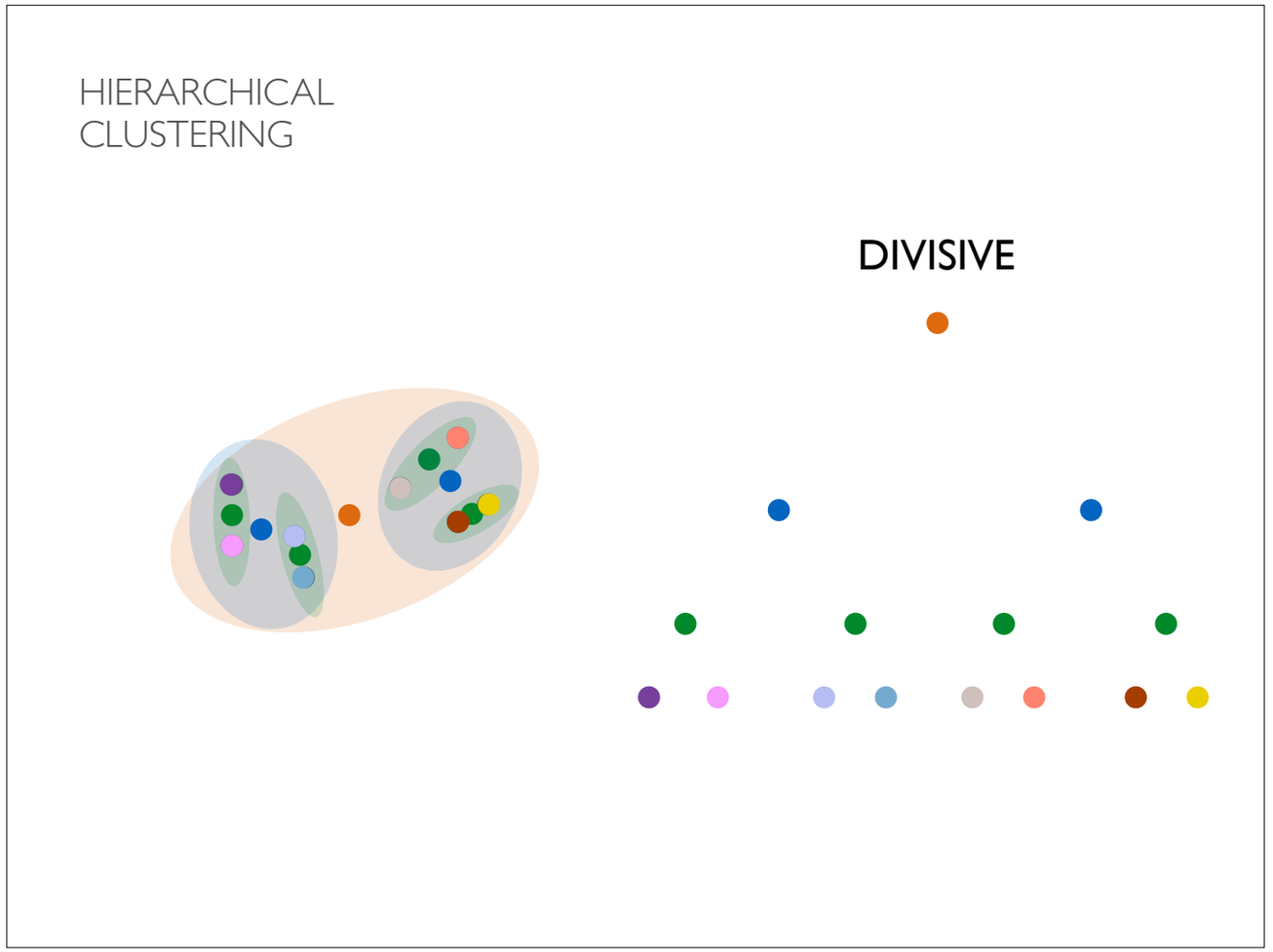
- Top Down Approach
- This takes in a data set as one large cluster with a single centroid. Then it recursively splits into two until it reaches the single points.
- Each centroid acts as a node



Zela

Divisive hierarchical clustering is thought of as a top down approach. This takes in a data set as one large cluster with a single centroid (animation). These nodes are important because they are the average representation of the points beneath it. We'll keep track of nodes on the tree to the right. This, then branches out and splits the cluster into two more clusters with its own average representation until it reaches the individual observations.

- Top Down Approach
- This takes in a data set as one large cluster with a single centroid. Then it recursively splits into two until it reaches the single points.
- Each centroid acts as a node



Zela

Divisive hierarchical clustering is thought of as a top down approach. This takes in a data set as one large cluster with a single centroid (animation). These nodes are important because they are the average representation of the points beneath it. We'll keep track of nodes on the tree to the right. This, then branches out and splits the cluster into two more clusters with its own average representation until it reaches the individual observations.

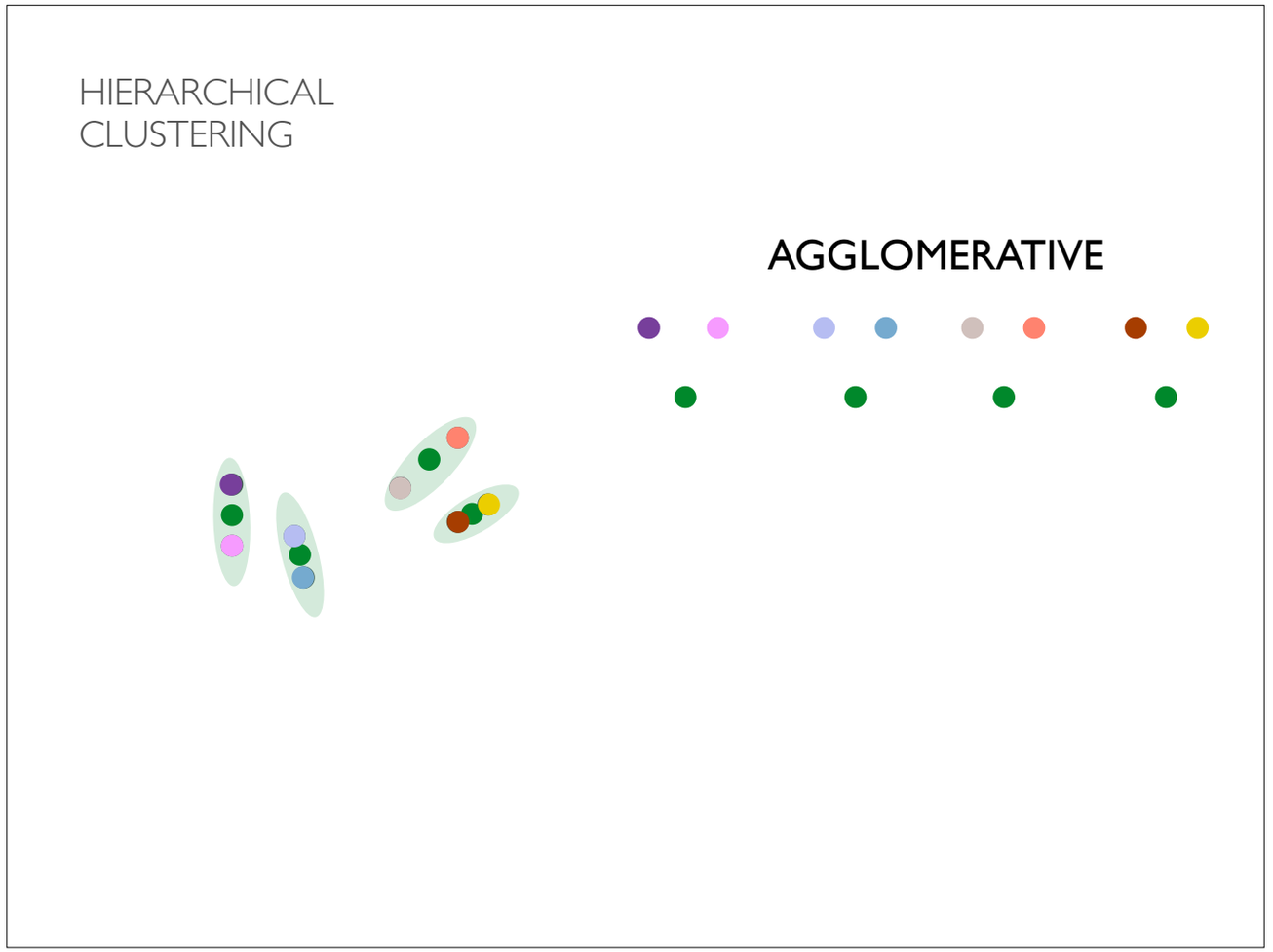
- Top Down Approach
- This takes in a data set as one large cluster with a single centroid. Then it recursively splits into two until it reaches the single points.
- Each centroid acts as a node

Zela

Agglomerative hierarchical clustering is similar but approach the problem bottom up. It initially looks at each data point as its own cluster. Then it clusters all the data into groups of two, which form the next layer. The means of those clusters are the data points of the next layer. This continues until the entire dataset is represented by a single average centroid. Now we're going to compare the two methods.

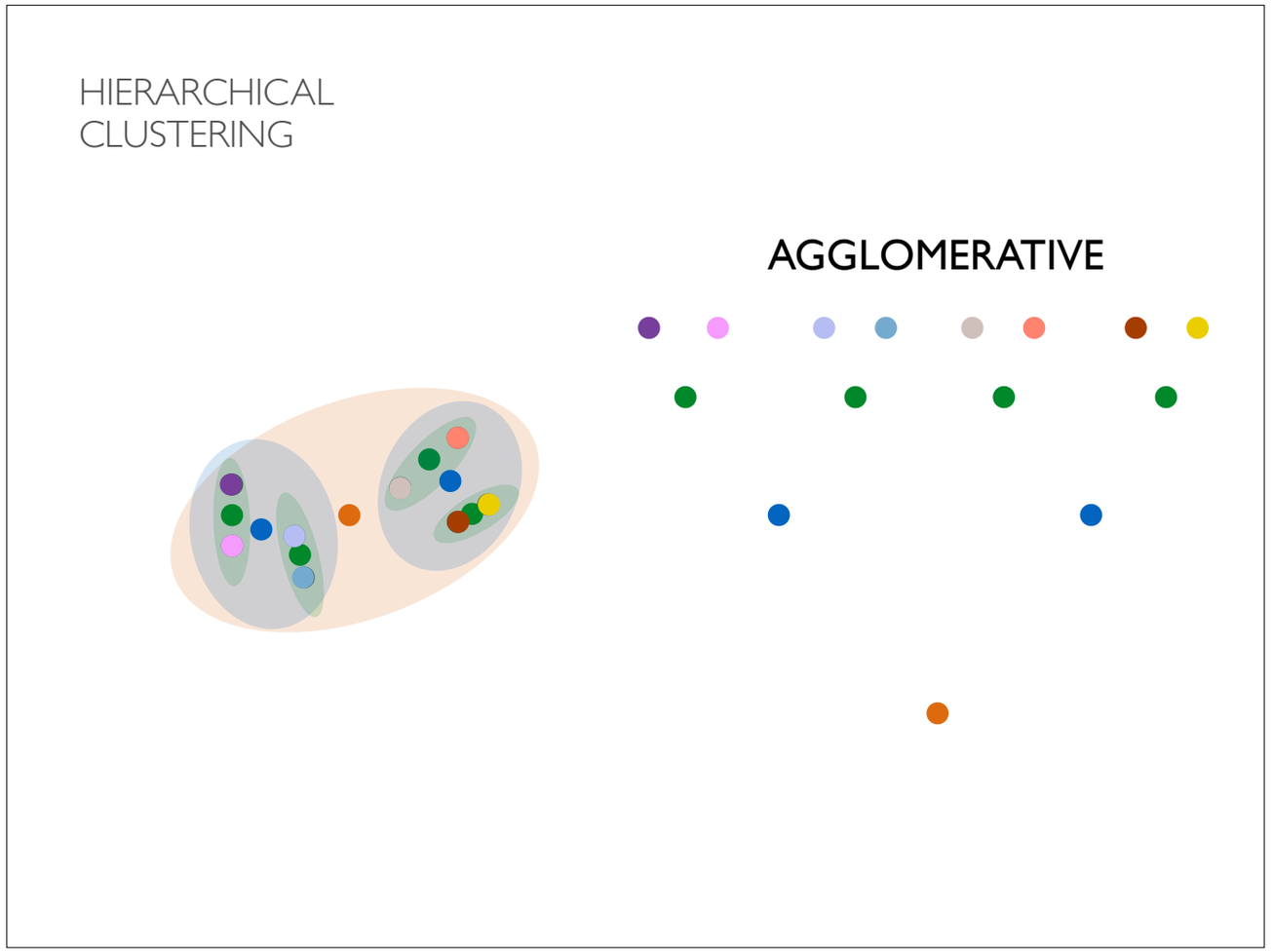
Zela

Agglomerative hierarchical clustering is similar but approach the problem bottom up. It initially looks at each data point as its own cluster. Then it clusters all the data into groups of two, which form the next layer. The means of those clusters are the data points of the next layer. This continues until the entire dataset is represented by a single average centroid. Now we're going to compare the two methods.



Zela

Agglomerative hierarchical clustering is similar but approach the problem bottom up. It initially looks at each data point as its own cluster. Then it clusters all the data into groups of two, which form the next layer. The means of those clusters are the data points of the next layer. This continues until the entire dataset is represented by a single average centroid. Now we're going to compare the two methods.



Zela

Agglomerative hierarchical clustering is similar but approach the problem bottom up. It initially looks at each data point as its own cluster. Then it clusters all the data into groups of two, which form the next layer. The means of those clusters are the data points of the next layer. This continues until the entire dataset is represented by a single average centroid. Now we're going to compare the two methods.

ALGORITHM ANALYSIS

n = number of points
k = number of clusters
d = dimensionality of the data
i = number of iterations until convergence

k-means complexity: $O(nkdi)$

Yixin
k - means complexity
agglomerative
divisive

ALGORITHM ANALYSIS

n = number of points
k = number of clusters
d = dimensionality of the data
i = number of iterations until convergence

Master theorem: Solves $T(n) = aT(n/b) + f(n)$

Yixin
k - means complexity
agglomerative
divisive

ALGORITHM ANALYSIS

n = number of points
k = number of clusters
d = dimensionality of the data
i = number of iterations until convergence

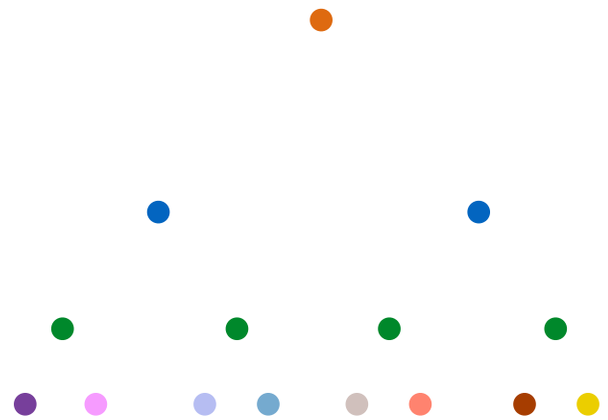
$$T_{div}(n) = 2T(\frac{n}{2}) + nkdi$$

Yixin
k - means complexity
agglomerative
divisive

ALGORITHM ANALYSIS

n = number of points
k = number of clusters
d = dimensionality of the data
i = number of iterations until convergence

$$T_{div}(n) = 2T(\frac{n}{2}) + nkdi$$



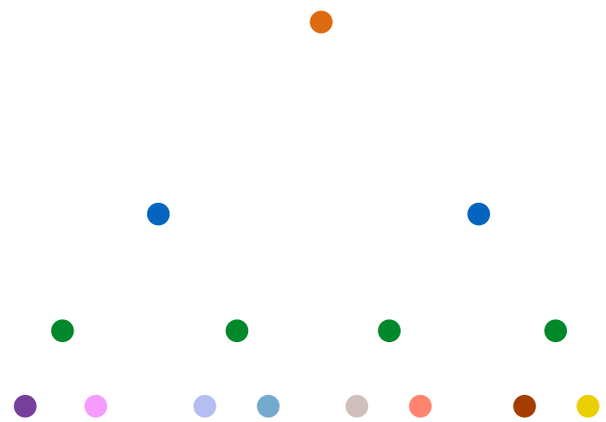
Yixin
k - means complexity
agglomerative
divisive

ALGORITHM ANALYSIS

n = number of points
k = number of clusters
d = dimensionality of the data
i = number of iterations until convergence

$$T_{div}(n) = 2T(\frac{n}{2}) + nkdi$$

$$T_{div}(n) = 2T(\frac{n}{2}) + 2ndi$$



Yixin
k - means complexity
agglomerative
divisive

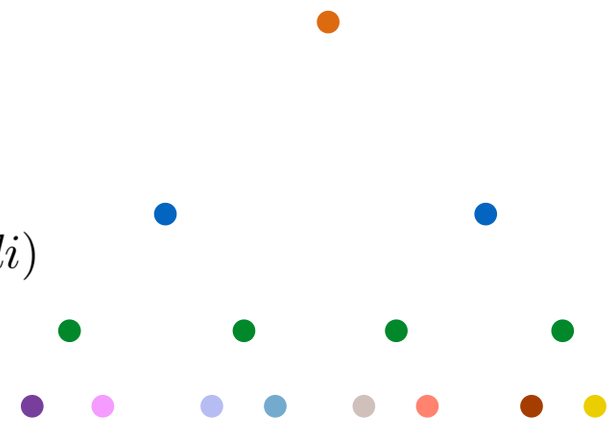
ALGORITHM ANALYSIS

n = number of points
k = number of clusters
d = dimensionality of the data
i = number of iterations until convergence

$$T_{div}(n) = 2T(\frac{n}{2}) + nkdi$$

$$T_{div}(n) = 2T(\frac{n}{2}) + 2ndi$$

$$T_{div}(n) = \Theta(n\log(n)di)$$



Yixin
k - means complexity
agglomerative
divisive

ALGORITHM ANALYSIS

n = number of points
k = number of clusters
d = dimensionality of the data
i = number of iterations until convergence

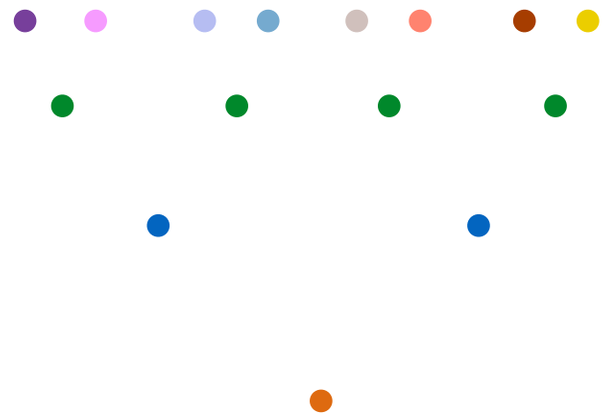
$$T_{aggl}(n) = T(\frac{n}{2}) + nkdi$$

Yixin
k - means complexity
agglomerative
divisive

ALGORITHM ANALYSIS

n = number of points
k = number of clusters
d = dimensionality of the data
i = number of iterations until convergence

$$T_{aggl}(n) = T(\frac{n}{2}) + nkdi$$



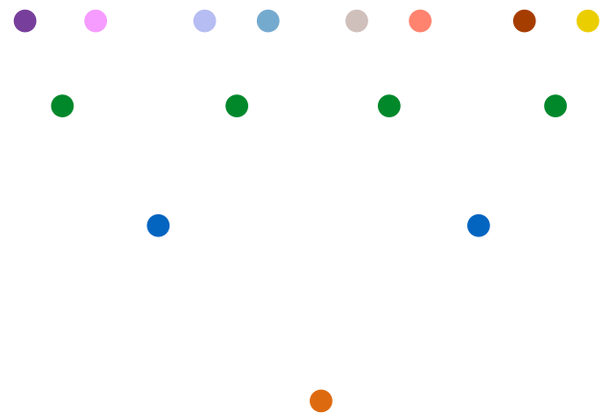
Yixin
k - means complexity
agglomerative
divisive

ALGORITHM ANALYSIS

n = number of points
k = number of clusters
d = dimensionality of the data
i = number of iterations until convergence

$$T_{aggl}(n) = T(\frac{n}{2}) + nkdi$$

$$T_{aggl}(n) = T(\frac{n}{2}) + \frac{n^2di}{2}$$



Yixin
k - means complexity
agglomerative
divisive

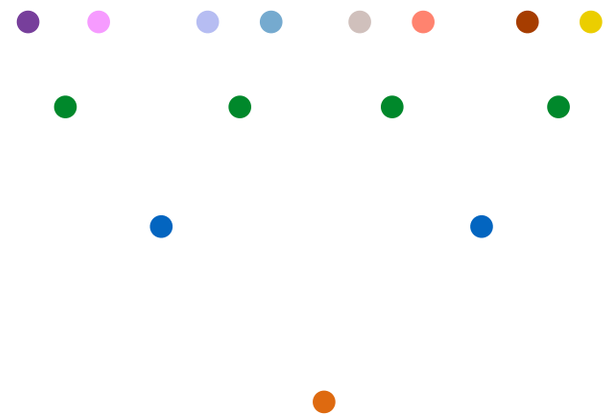
ALGORITHM ANALYSIS

n = number of points
k = number of clusters
d = dimensionality of the data
i = number of iterations until convergence

$$T_{aggl}(n) = T(\frac{n}{2}) + nkdi$$

$$T_{aggl}(n) = T(\frac{n}{2}) + \frac{n^2di}{2}$$

$$T_{aggl}(n) = \Theta(n^2di)$$



Yixin
k - means complexity
agglomerative
divisive

ALGORITHM ANALYSIS

n = number of points
k = number of clusters
d = dimensionality of the data
i = number of iterations until convergence

$$T_{div}(n) = \Theta(n \log(n) di)$$

$$T_{aggl}(n) = \Theta(n^2 di)$$

Yixin
k - means complexity
agglomerative
divisive

ALGORITHM ANALYSIS

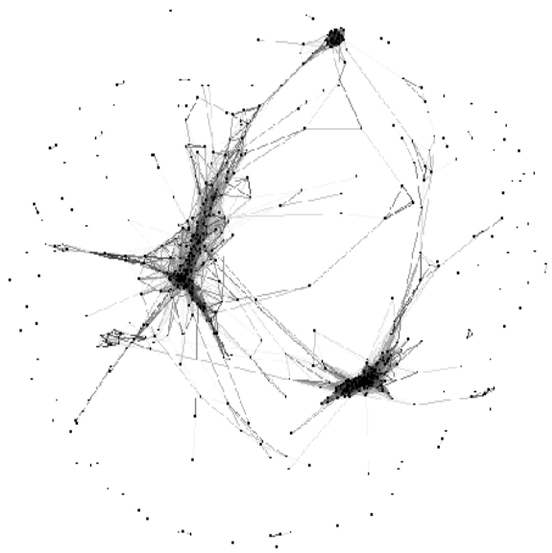
n = number of points
k = number of clusters
d = dimensionality of the data
i = number of iterations until convergence

$$T_{div}(n) = \Theta(n \log(n) di)$$

$$T_{aggl}(n) = \Theta(n^2 di)$$

Yixin
k - means complexity
agglomerative
divisive

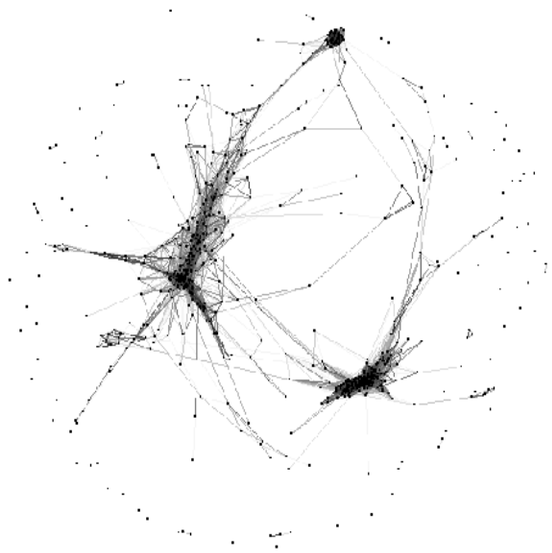
HIERARCHICAL CLUSTERING



14MIN Yixin

- We wrote both hierarchical clustering algorithms
- We tested them, as well as a built-in MATLAB one, on standard datasets
- Thousands of data points which live on 13,000 dimensional
- Illustration of tree structure with means
- Example Voronoi diagram: which points are clustered with which means

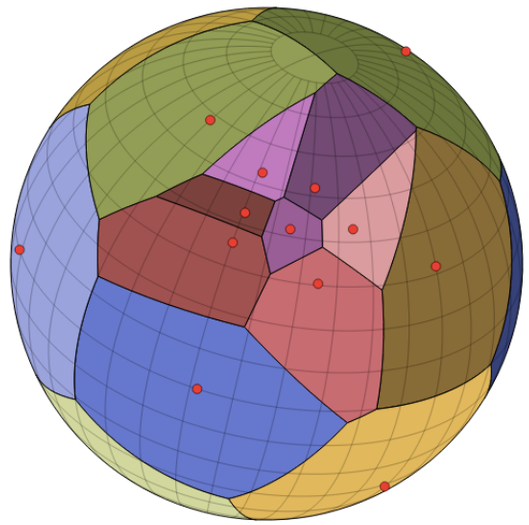
HIERARCHICAL CLUSTERING



14MIN Yixin

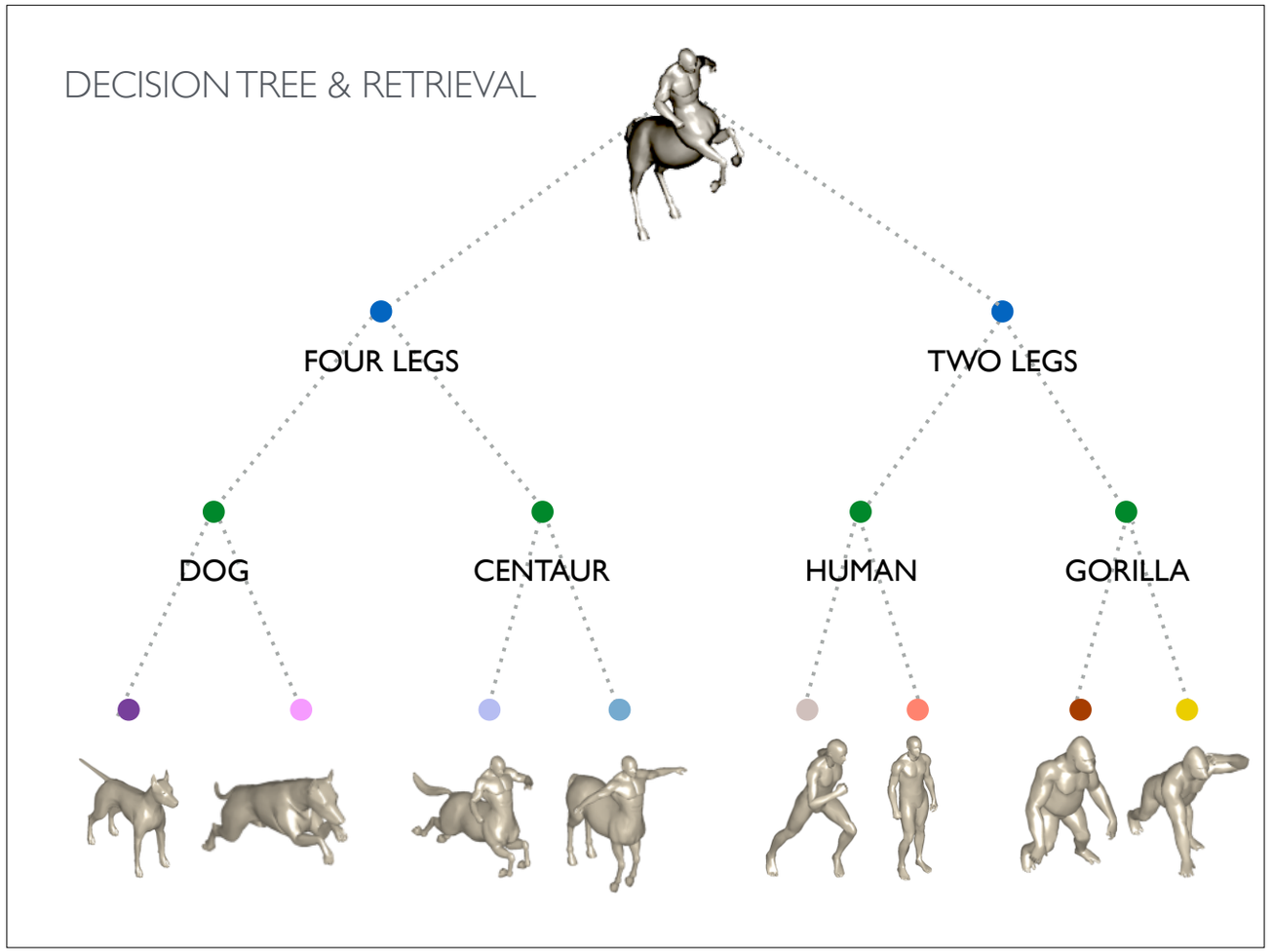
- We wrote both hierarchical clustering algorithms
- We tested them, as well as a built-in MATLAB one, on standard datasets
- Thousands of data points which live on 13,000 dimensional
- Illustration of tree structure with means
- Example Voronoi diagram: which points are clustered with which means

HIERARCHICAL CLUSTERING



14MIN Yixin

- We wrote both hierarchical clustering algorithms
- We tested them, as well as a built-in MATLAB one, on standard datasets
- Thousands of data points which live on 13,000 dimensional
- Illustration of tree structure with means
- Example Voronoi diagram: which points are clustered with which means

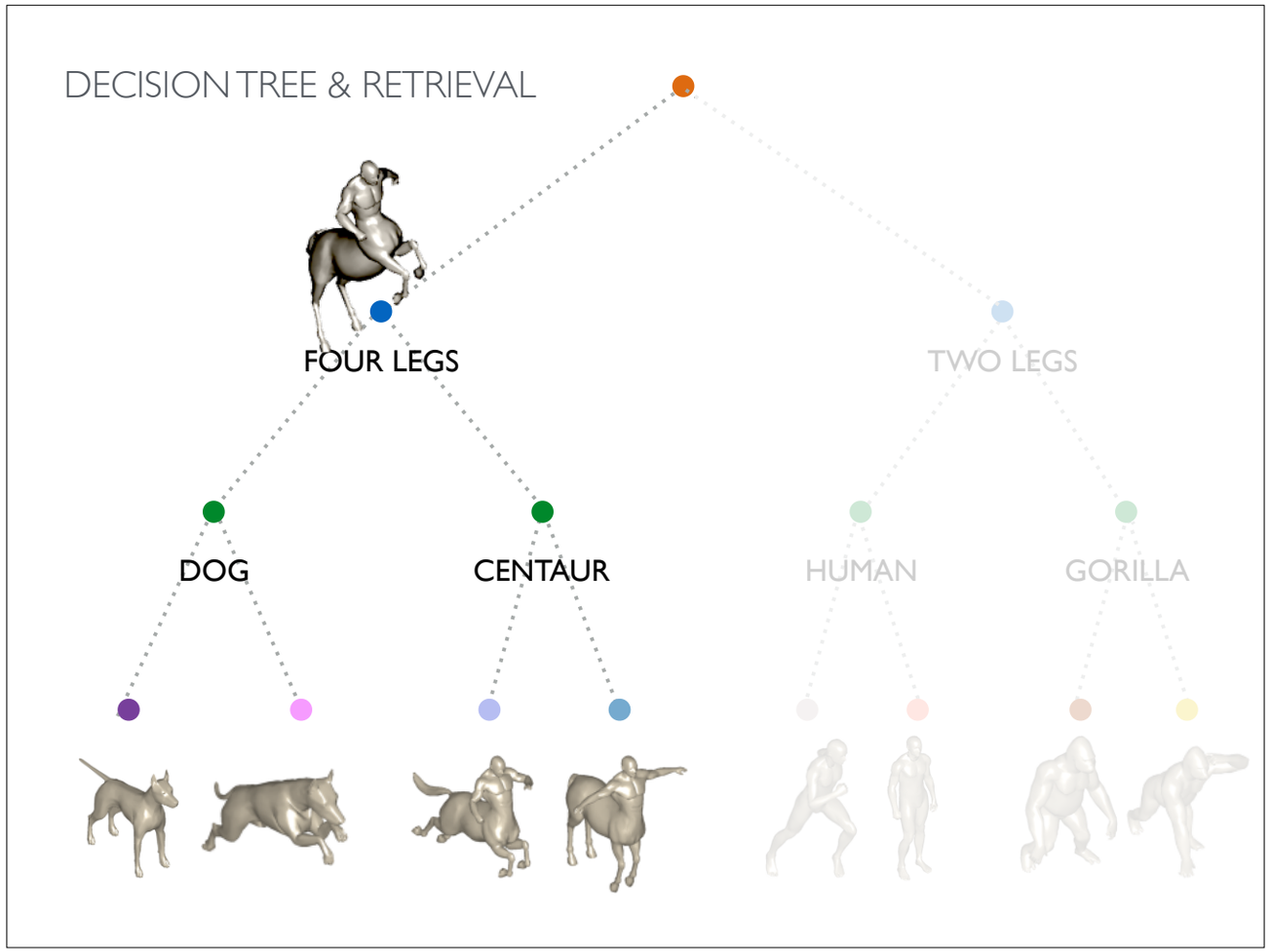


Zela

So how does all of this help us in our overall goal? Let’s do a quick over view of whats happening. We we’re given a data base. We clustered each shape in the data base. When you give us a query, we run it through the decision tree to give you back the most similar object as efficiently as possible. (Animation) a query shape comes in and begins to traverse the tree. Is the object closer to this point or this point? It will traverse again and ask is the object closer to this point or this point? And once it hits the leaves it will compare to the objects in the subtree and retrieve the closest one. And because this is a balanced tree, it takes log(n) time.

- The goal is to figure out what the closest points are, efficiently
- For each internal node in the tree, find the closest subtree mean (using spherical distance, aka Karcher mean)
- Once you get to a leaf, you have the “closest” object, but in log(n) time

log(amount of data points) 2^n = h
h = log2(n)

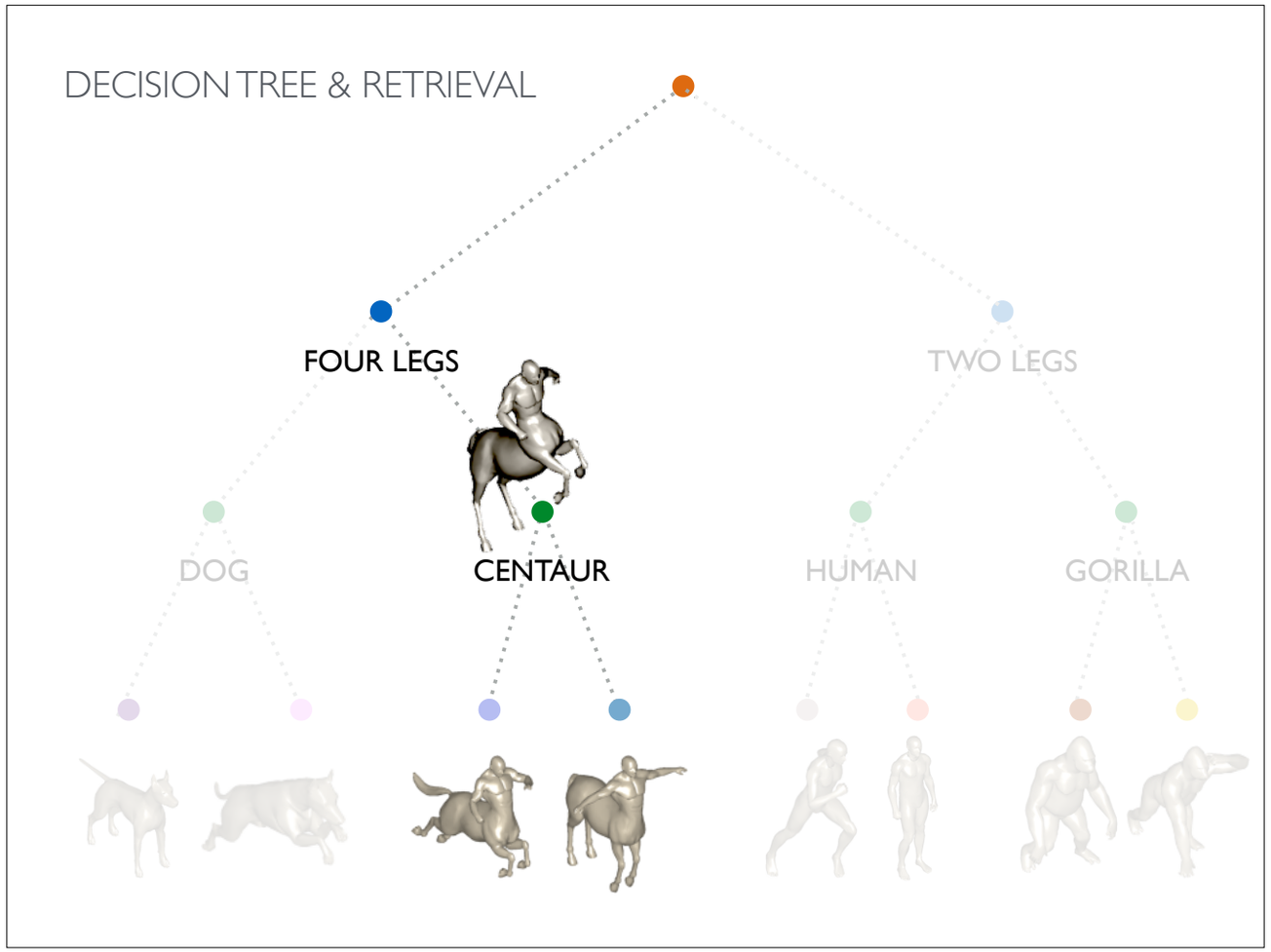


Zela

So how does all of this help us in our overall goal? Let’s do a quick over view of whats happening. We we’re given a data base. We clustered each shape in the data base. When you give us a query, we run it through the decision tree to give you back the most similar object as efficiently as possible. (Animation) a query shape comes in and begins to traverse the tree. Is the object closer to this point or this point? It will traverse again and ask is the object closer to this point or this point? And once it hits the leaves it will compare to the objects in the subtree and retrieve the closest one. And because this is a balanced tree, it takes log(n) time.

- The goal is to figure out what the closest points are, efficiently
- For each internal node in the tree, find the closest subtree mean (using spherical distance, aka Karcher mean)
- Once you get to a leaf, you have the “closest” object, but in log(n) time

log(amount of data points) 2^n = h
h = log2(n)

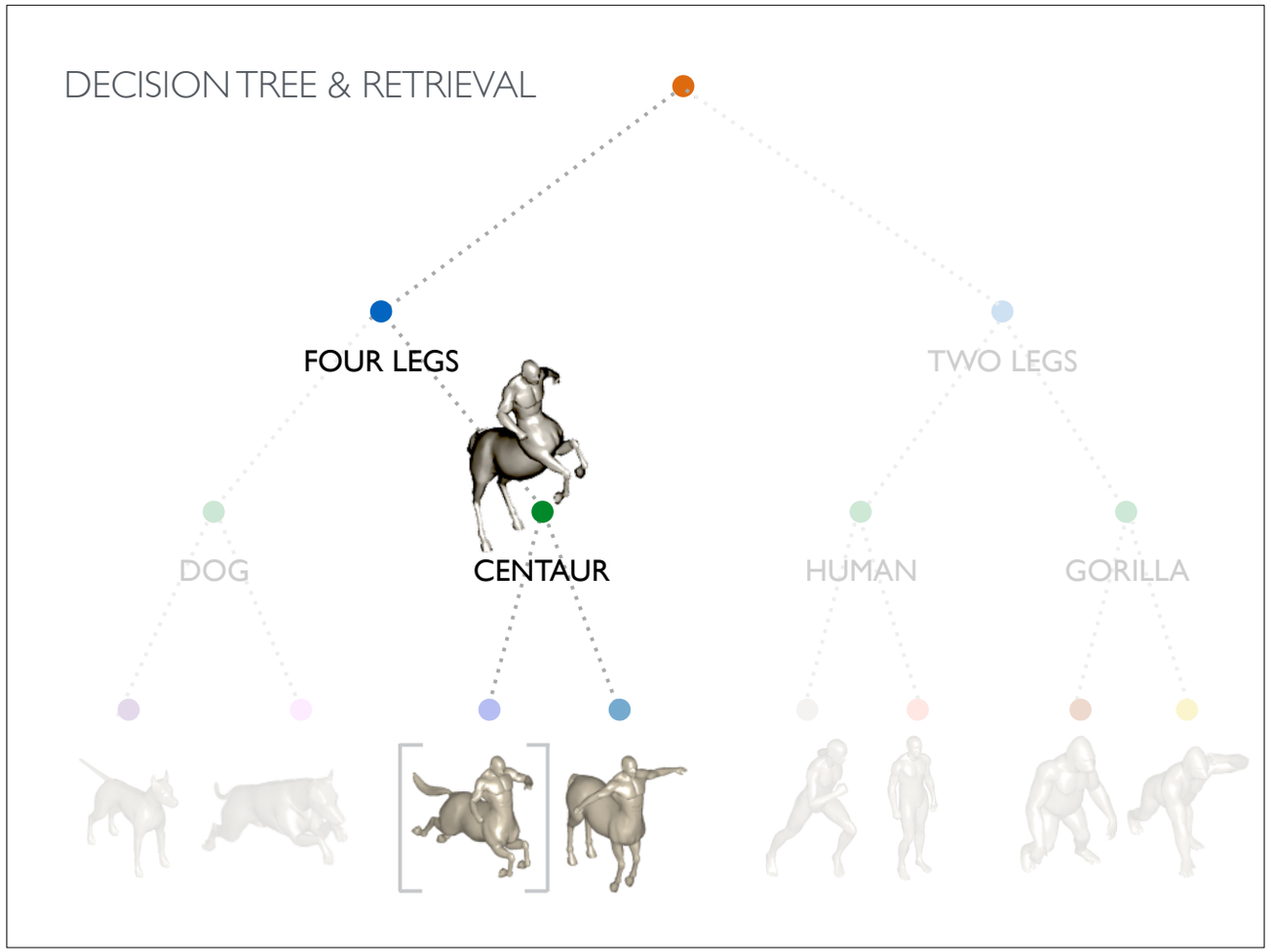


Zela

So how does all of this help us in our overall goal? Let's do a quick over view of whats happening. We we're given a data base. We clustered each shape in the data base. When you give us a query, we run it through the decision tree to give you back the most similar object as efficiently as possible. (Animation) a query shape comes in and begins to traverse the tree. Is the object closer to this point or this point? It will traverse again and ask is the object closer to this point or this point? And once it hits the leaves it will compare to the objects in the subtree and retrieve the closest one. And because this is a balanced tree, it takes log(n) time.

- The goal is to figure out what the closest points are, efficiently
- For each internal node in the tree, find the closest subtree mean (using spherical distance, aka Karcher mean)
- Once you get to a leaf, you have the "closest" object, but in log(n) time

log(amount of data points) 2^n = h
h = log2(n)

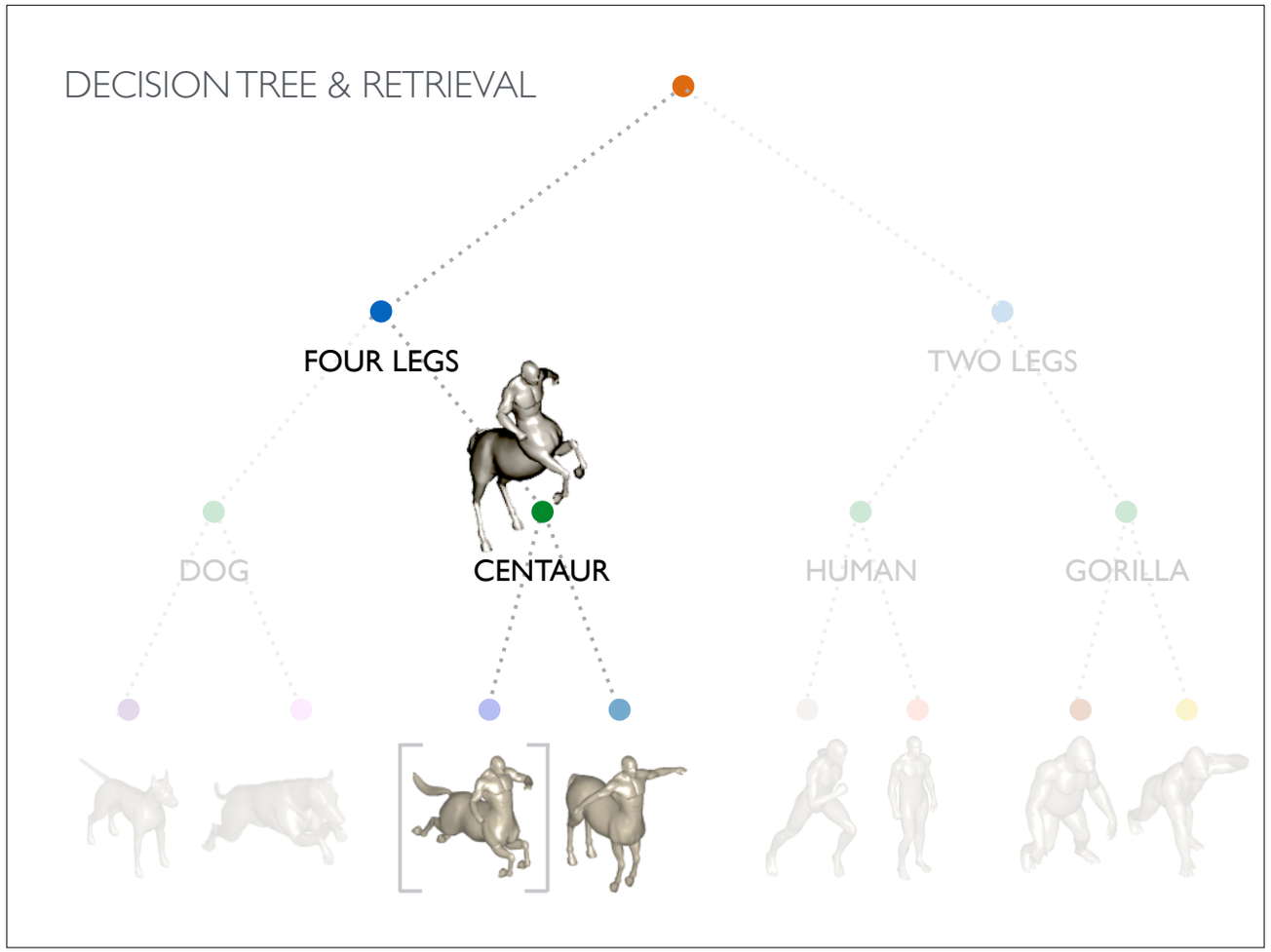


Zela

So how does all of this help us in our overall goal? Let's do a quick over view of whats happening. We we're given a data base. We clustered each shape in the data base. When you give us a query, we run it through the decision tree to give you back the most similar object as efficiently as possible. (Animation) a query shape comes in and begins to traverse the tree. Is the object closer to this point or this point? It will traverse again and ask is the object closer to this point or this point? And once it hits the leaves it will compare to the objects in the subtree and retrieve the closest one. And because this is a balanced tree, it takes log(n) time.

- The goal is to figure out what the closest points are, efficiently
- For each internal node in the tree, find the closest subtree mean (using spherical distance, aka Karcher mean)
- Once you get to a leaf, you have the "closest" object, but in log(n) time

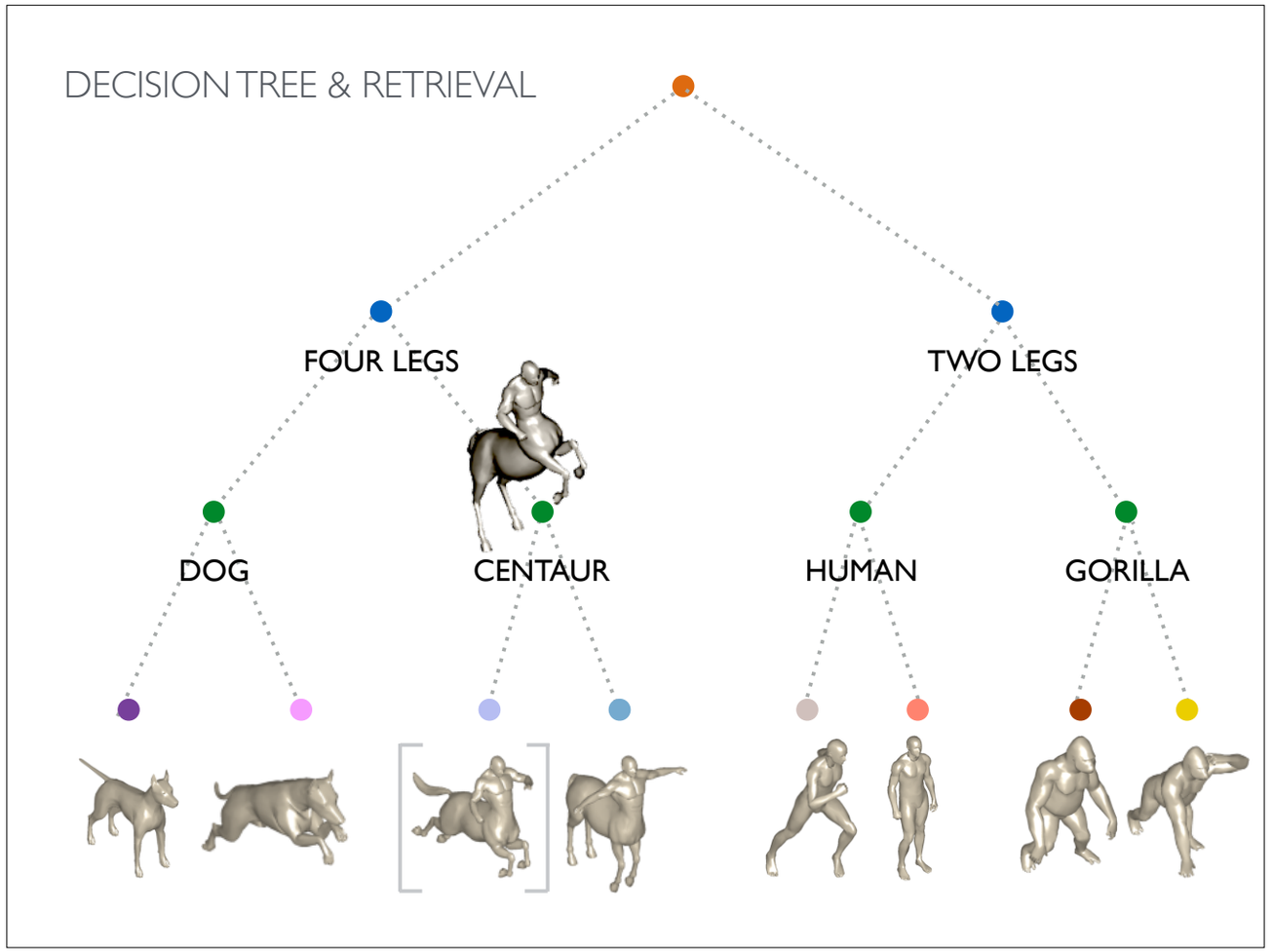
log(amount of data points) 2^n = h
h = log2(n)



Yixin

Retrieval implementation

- How do we grab multiple results? Retrieval, not classification
- Use a recursive mergesort-like algorithm (except not really)
    - At each level, return the concatenation of the closer and the farther subproblems



Yixin

Retrieval implementation

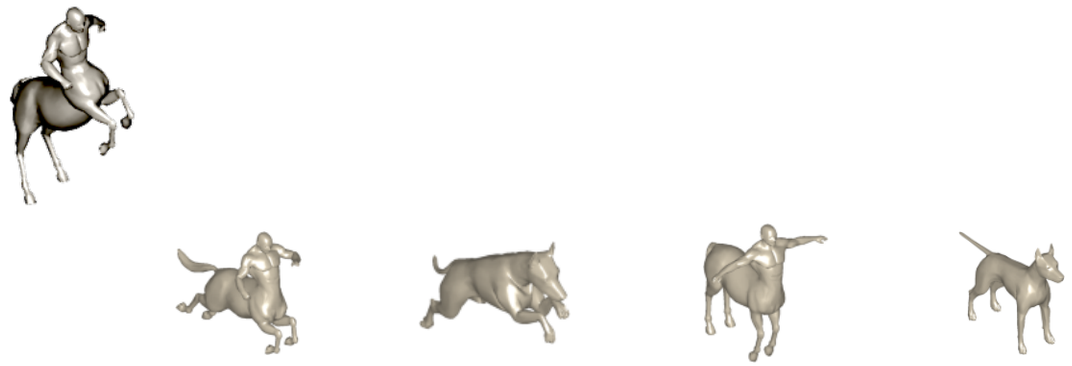
- How do we grab multiple results? Retrieval, not classification
- Use a recursive mergesort-like algorithm (except not really)
    - At each level, return the concatenation of the closer and the farther subproblems

METRICS AND RESULTS

Zela

- Suppose we got this result from retrieving with this centaur query
- We would then evaluate retrieval mechanisms on standardized metrics
- Precision recall

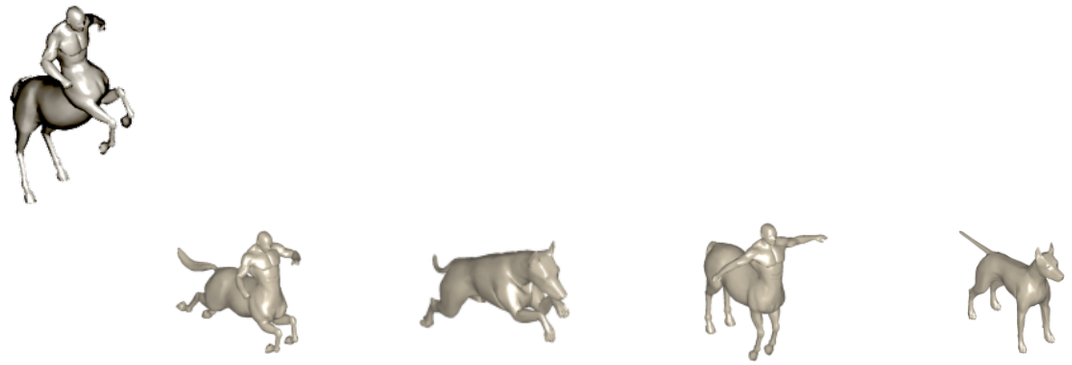
METRICS AND RESULTS



Zela

- Suppose we got this result from retrieving with this centaur query
- We would then evaluate retrieval mechanisms on standardized metrics
- Precision recall

METRICS AND RESULTS



$$\text{PRECISION} = \frac{\text{number of retrieved \& relevant}}{\text{number of retrieved}}$$

Zela

- Suppose we got this result from retrieving with this centaur query
- We would then evaluate retrieval mechanisms on standardized metrics
- Precision recall

METRICS AND RESULTS

PRECISION	1/1	1/2	2/3	2/4

Zela

- Suppose we got this result from retrieving with this centaur query
- We would then evaluate retrieval mechanisms on standardized metrics
- Precision recall

METRICS AND RESULTS

PRECISION 1/1 1/2 2/3 2/4

RECALL = $\frac{\text{number of retrieved \& relevant}}{\text{number of relevant}}$

Zela

- Suppose we got this result from retrieving with this centaur query
- We would then evaluate retrieval mechanisms on standardized metrics
- Precision recall

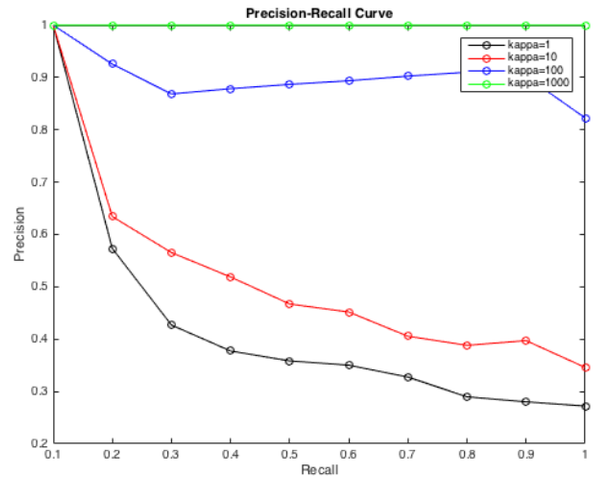
METRICS AND RESULTS

PRECISION	1/1	1/2	2/3	2/4
RECALL	1/2	1/2	2/2	2/2

Zela

- Suppose we got this result from retrieving with this centaur query
- We would then evaluate retrieval mechanisms on standardized metrics
- Precision recall

METRICS AND RESULTS



Yixin

- Precision recall curve
- First we randomly constructed spherical clusters using the von Mises distribution (generalization of the normal distribution on a circle)
- Depending on the kappa value (concentration parameter), it performed quite well
- Higher kappa means more concentrated, better retrieval (less overlap)
- When clusters don’t overlap, it gets nearly perfect scores

METRICS AND RESULTS

Yixin

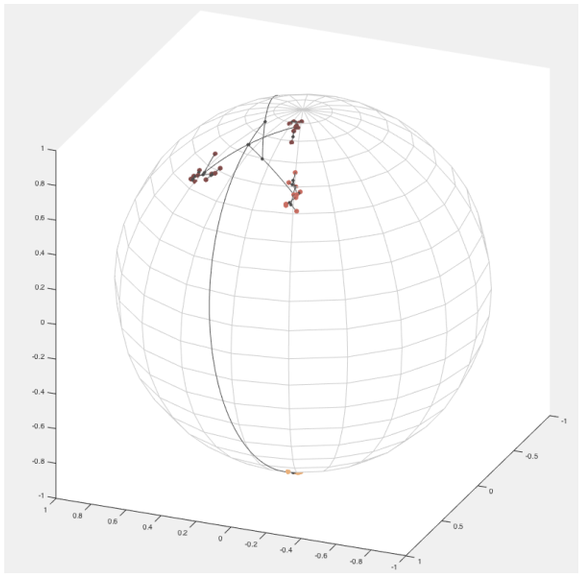
- Then we ran it on common datasets
- Very good at nearest neighbors (close to 1)
- Not so good on some of the other metrics (outperformed by state of the art)
- Clusters overlap in higher dimensions, explore different ways of fitting the structure of the data

Yixin

Clustering Graphics

Explain how well our approach worked

METRICS AND RESULTS



Yixin

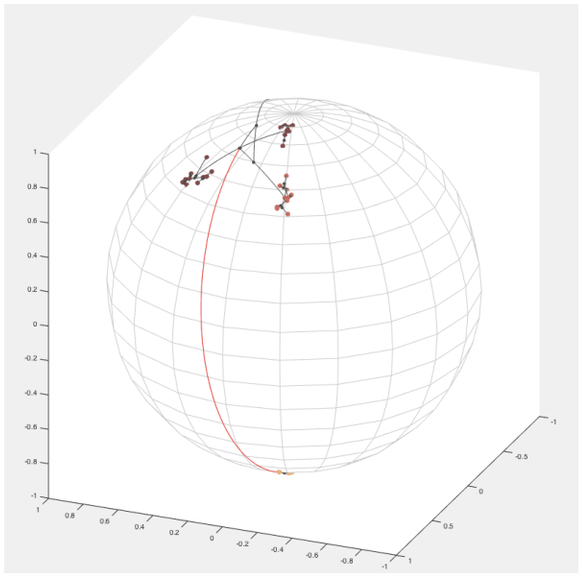
- Then we ran it on common datasets
- Very good at nearest neighbors (close to 1)
- Not so good on some of the other metrics (outperformed by state of the art)
- Clusters overlap in higher dimensions, explore different ways of fitting the structure of the data

Yixin

Clustering Graphics

Explain how well our approach worked

METRICS AND RESULTS



Yixin

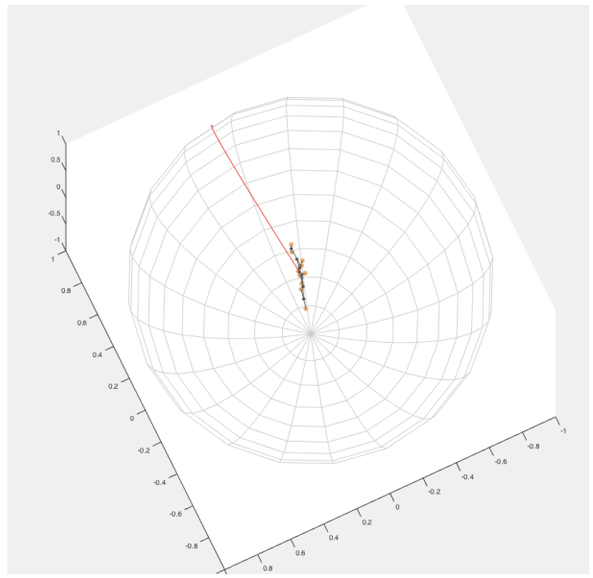
- Then we ran it on common datasets
- Very good at nearest neighbors (close to 1)
- Not so good on some of the other metrics (outperformed by state of the art)
- Clusters overlap in higher dimensions, explore different ways of fitting the structure of the data

Yixin

Clustering Graphics

Explain how well our approach worked

METRICS AND RESULTS



Yixin

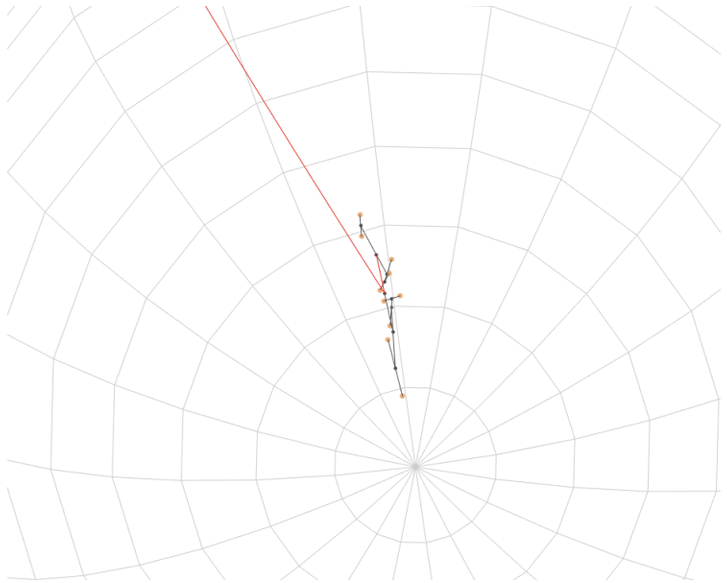
- Then we ran it on common datasets
- Very good at nearest neighbors (close to 1)
- Not so good on some of the other metrics (outperformed by state of the art)
- Clusters overlap in higher dimensions, explore different ways of fitting the structure of the data

Yixin

Clustering Graphics

Explain how well our approach worked

METRICS AND RESULTS



Yixin

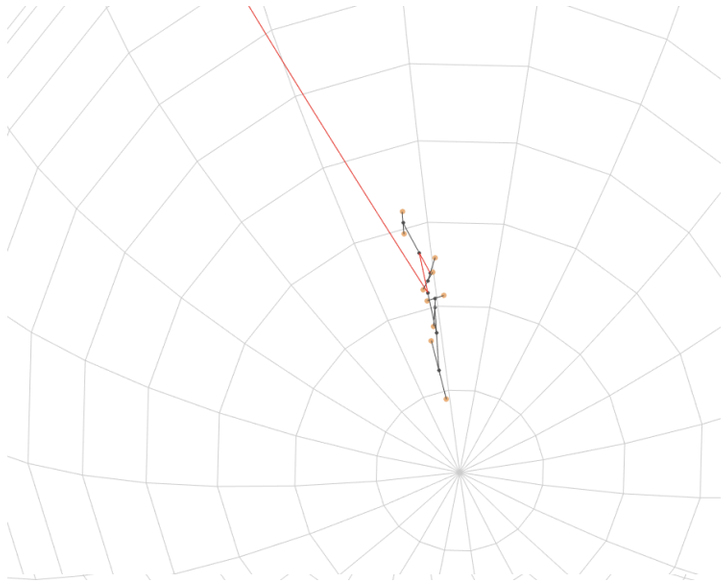
- Then we ran it on common datasets
- Very good at nearest neighbors (close to 1)
- Not so good on some of the other metrics (outperformed by state of the art)
- Clusters overlap in higher dimensions, explore different ways of fitting the structure of the data

Yixin

Clustering Graphics

Explain how well our approach worked

METRICS AND RESULTS



Yixin

- Then we ran it on common datasets
- Very good at nearest neighbors (close to 1)
- Not so good on some of the other metrics (outperformed by state of the art)
- Clusters overlap in higher dimensions, explore different ways of fitting the structure of the data

Yixin

Clustering Graphics

Explain how well our approach worked

CONCLUSION

Zixin

- We’ve explored the significant problem of shape analysis and retrieval, important for many applications including 3d animation and design and robot and computer vision
- We’ve implemented a faster retrieval data structure and retrieval mechanism that runs in O(log(n)) instead of O(n)
- We proved the running time of hierarchical clustering using spherical K-means

CONCLUSION

Zixin

- We’ve explored the significant problem of shape analysis and retrieval, important for many applications including 3d animation and design and robot and computer vision
- We’ve implemented a faster retrieval data structure and retrieval mechanism that runs in O(log(n)) instead of O(n)
- We proved the running time of hierarchical clustering using spherical K-means

FUTURE RESEARCH

- Run on standard retrieval datasets
- Investigate alternative retrieval structures

RETRIEVAL MECHANICS

- Delve deeper into the wavelet density estimation
- Optimize wavelet density estimation code
- Linear Assignment

FEATURE REPRESENTATION

Zixin

Next Phase

Improve shape retrievals

- Run on datasets
- Investigate alternative retrieval structures

Begin optimizing calculating feature representations

- Delve deeper into the wavelet density estimation
- Optimize wavelet density estimation code
- Linear assignment of wavelet coefficients: how can we “distort” or manipulate the feature representation so that close shapes are closer together?

FUTURE RESEARCH

- Run on standard retrieval datasets
- Investigate alternative retrieval structures

RETRIEVAL MECHANICS

- Delve deeper into the wavelet density estimation
- Optimize wavelet density estimation code
- Linear Assignment

FEATURE REPRESENTATION

Zixin

Next Phase

Improve shape retrievals

- Run on datasets
- Investigate alternative retrieval structures

Begin optimizing calculating feature representations

- Delve deeper into the wavelet density estimation
- Optimize wavelet density estimation code
- Linear assignment of wavelet coefficients: how can we “distort” or manipulate the feature representation so that close shapes are closer together?

THANK YOU