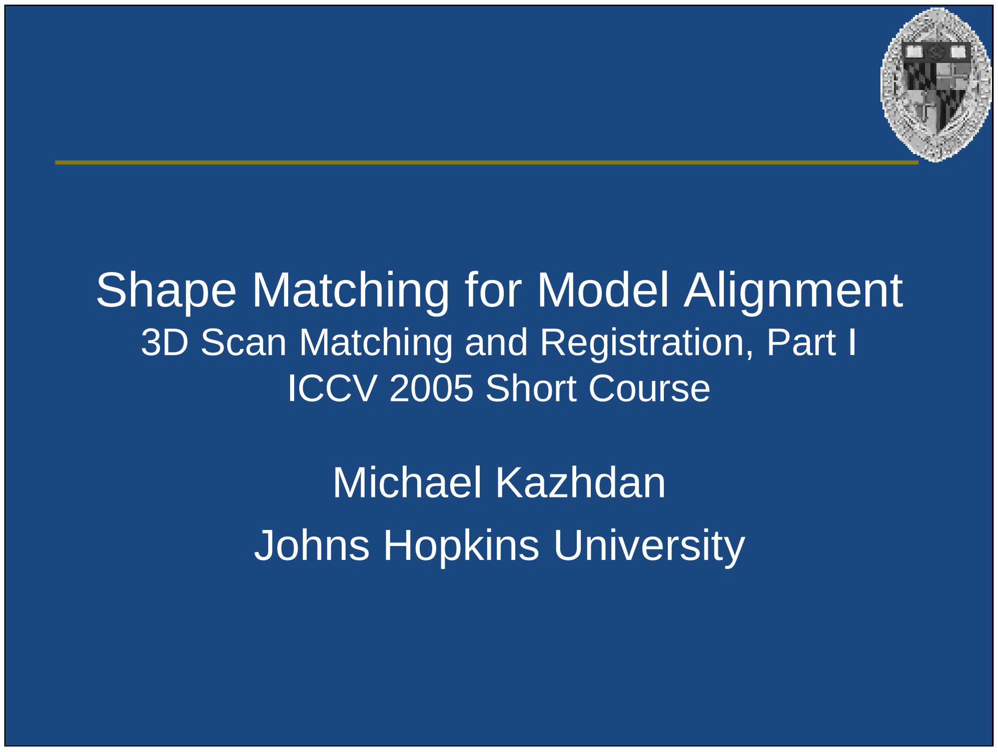


# Shape Matching for Model Alignment

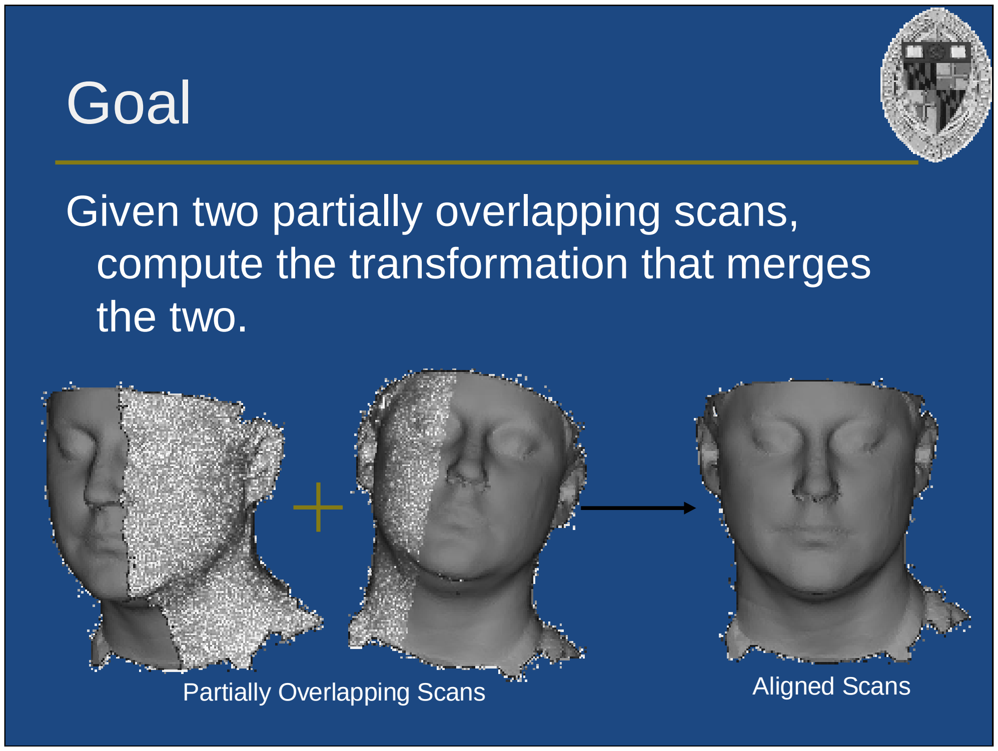
3D Scan Matching and Registration, Part I

ICCV 2005 Short Course

Michael Kazhdan

Johns Hopkins University

This section of the course will focus on a discussion of how the problem of shape matching relates to the task of model alignment



## Goal

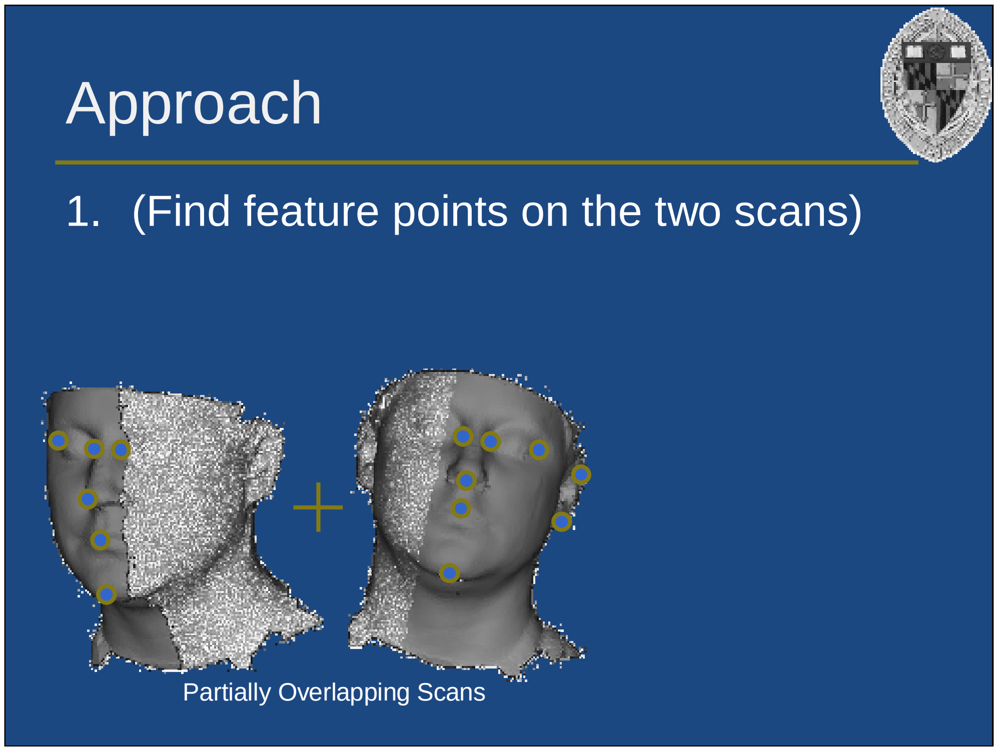
Given two partially overlapping scans, compute the transformation that merges the two.

Partially Overlapping Scans

Aligned Scans

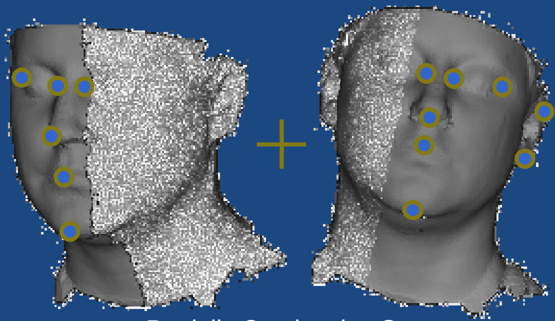
The goal of 3D scan registration is to be able to take in two overlapping subsets of a 3D surface and return the surface obtained by optimally aligning the overlapping regions.

As an example, given two scans of a human face, the first capturing a little more than the right half of the face and the second capturing a little more than the left hand side of the face, the goal is to identify the overlapping features (the chin, mouth, nose and interior corners of the eyes) and to use those shared to features to align the two separate scans into a shared coordinate frame.



## Approach

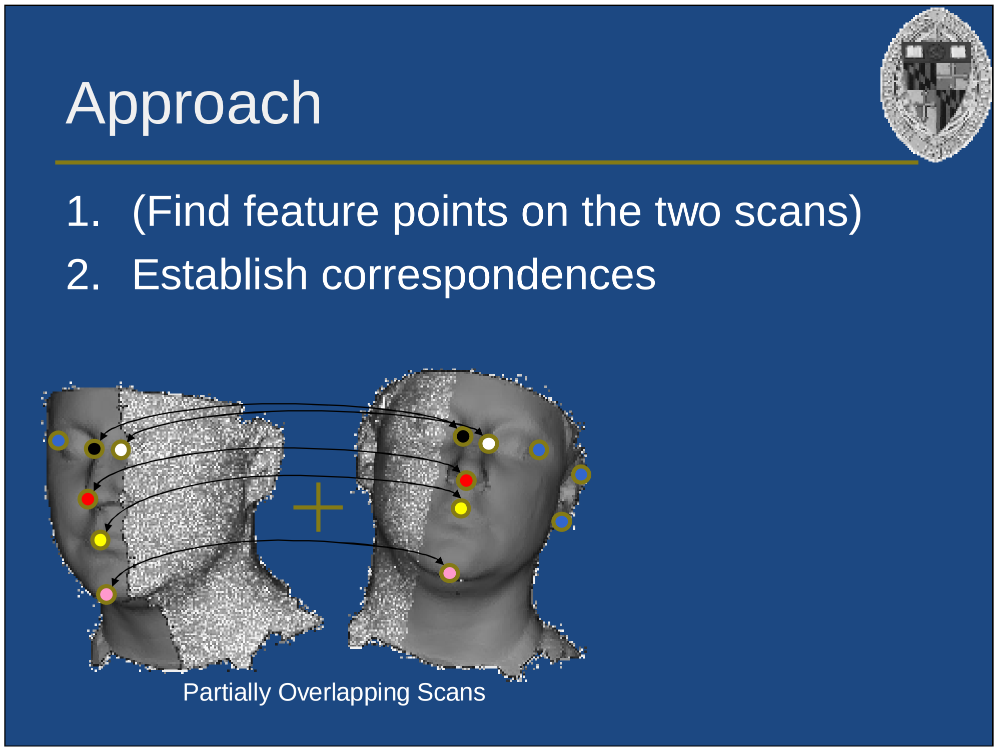
1. (Find feature points on the two scans)



Partially Overlapping Scans

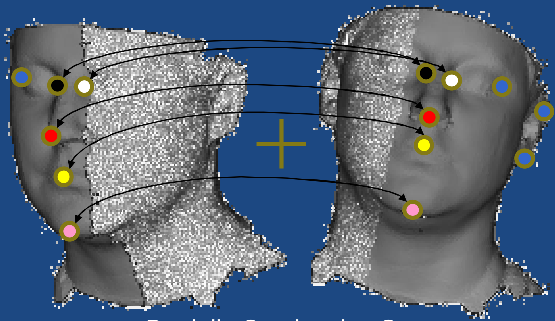
In general, the alignment problem is solved in three steps.

First, the features in the scans are independently identified.



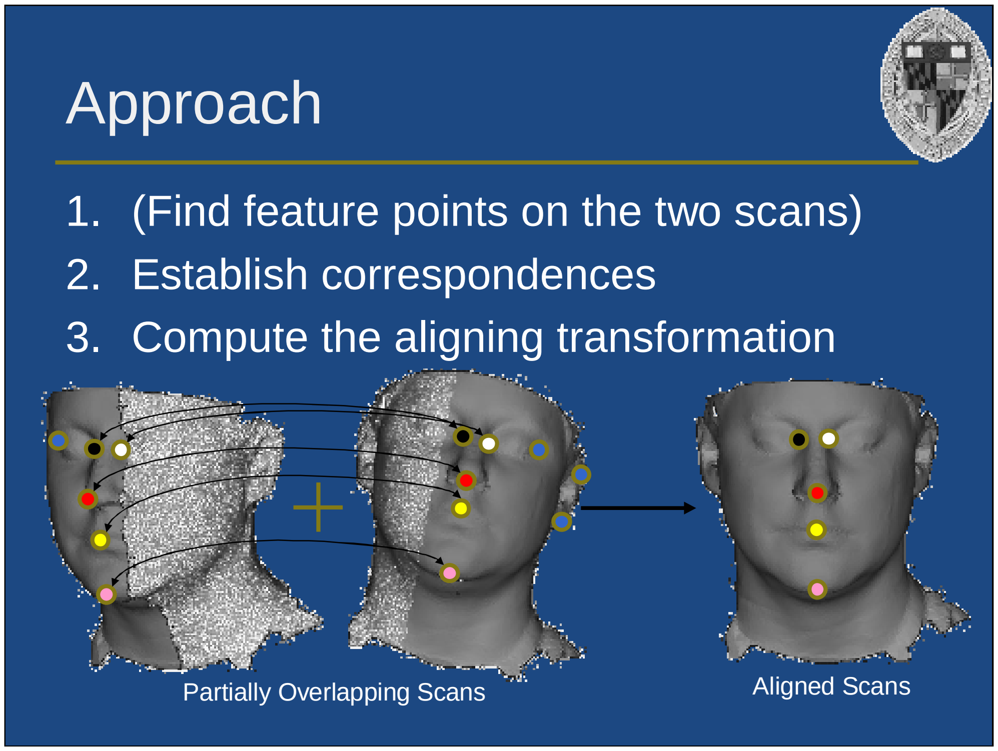
## Approach

1. (Find feature points on the two scans)
2. Establish correspondences



Partially Overlapping Scans

Next, point-correspondences are established between shared features in the two scans.



## Approach

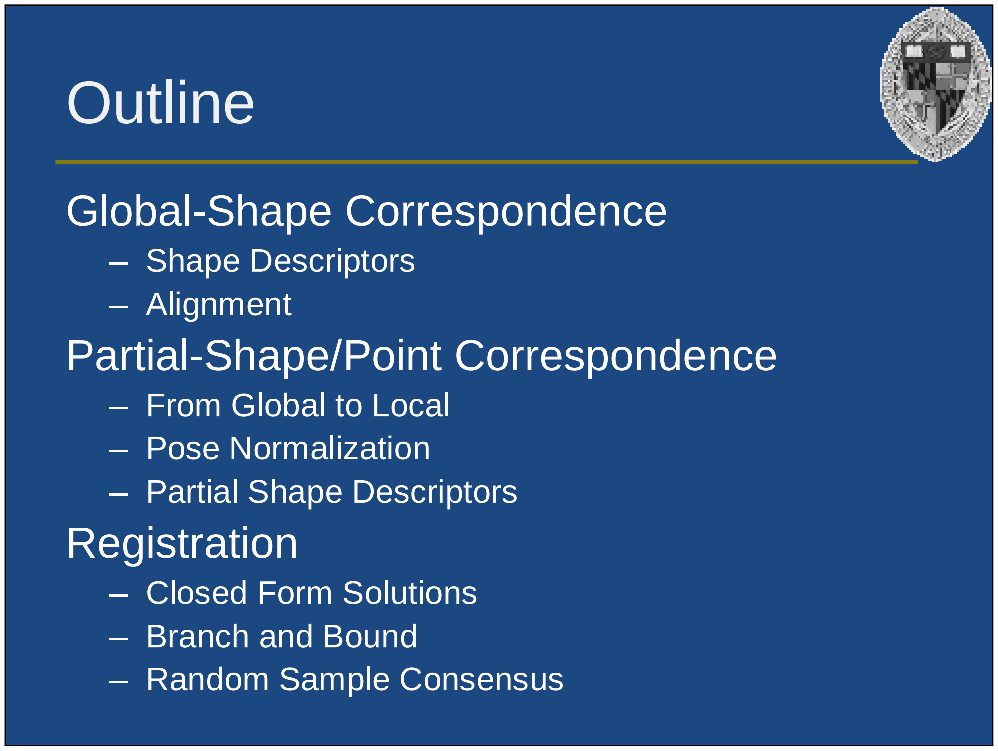
1. (Find feature points on the two scans)
2. Establish correspondences
3. Compute the aligning transformation

Partially Overlapping Scans
Aligned Scans

Finally, the computed set of correspondences are used to solve for the optimal aligning transformation and this transformation is used to place the two scans into a shared coordinate frame.

While finding features on both scans is an important problem and can help facilitate the task of registration by reducing noise and increasing efficiency, we will not focus on this aspect of shape registration in this course.

Instead, we will focus on the second and third steps of the registration pipeline, (2) establishing correspondences, and (3) using the established correspondences to define a transformation that places both scans into a shared coordinate system.

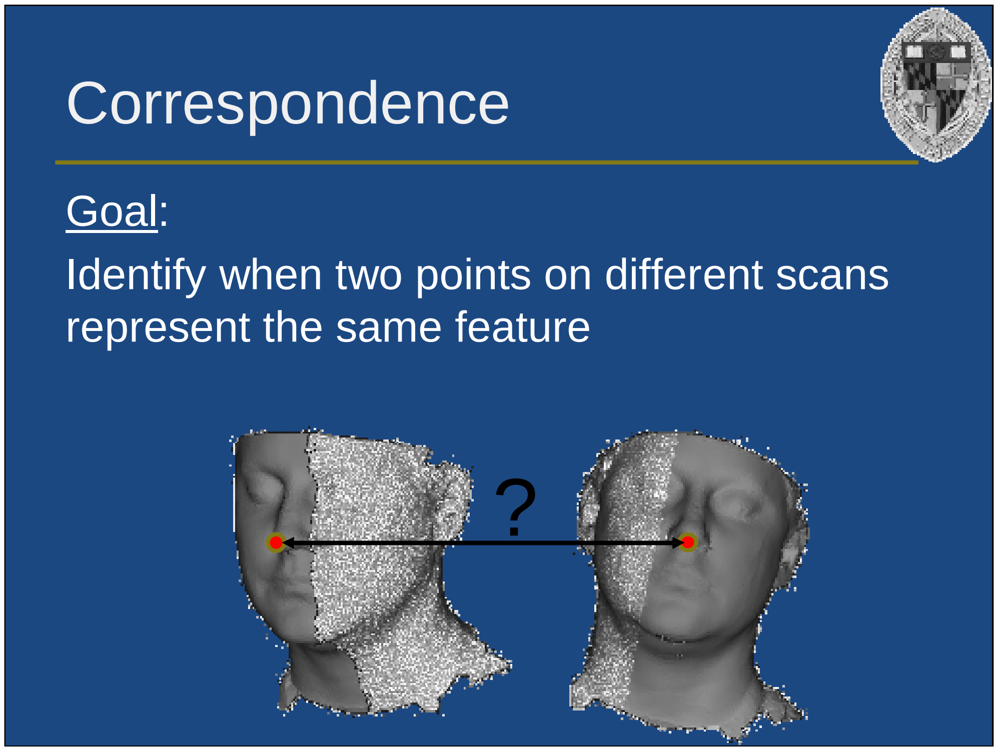

## Outline

Global-Shape Correspondence
- Shape Descriptors
- Alignment

Partial-Shape/Point Correspondence
- From Global to Local
- Pose Normalization
- Partial Shape Descriptors

Registration
- Closed Form Solutions
- Branch and Bound
- Random Sample Consensus

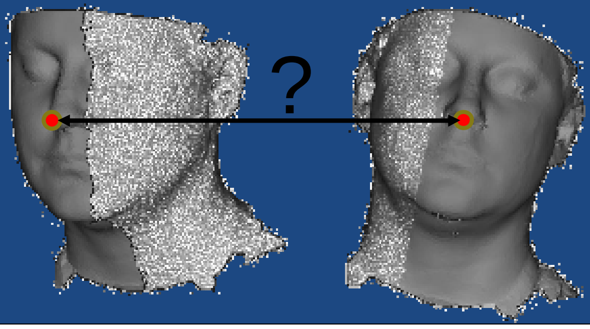
So, we will begin the discussion of correspondence establishment by considering a slightly more general problem: The problem of whole-shape matching. We will then discuss how work in the area of whole-shape matching can be used to drive algorithms for the establishment of local correspondences, and we will conclude by describing methods for using the correspondences to register the scans.



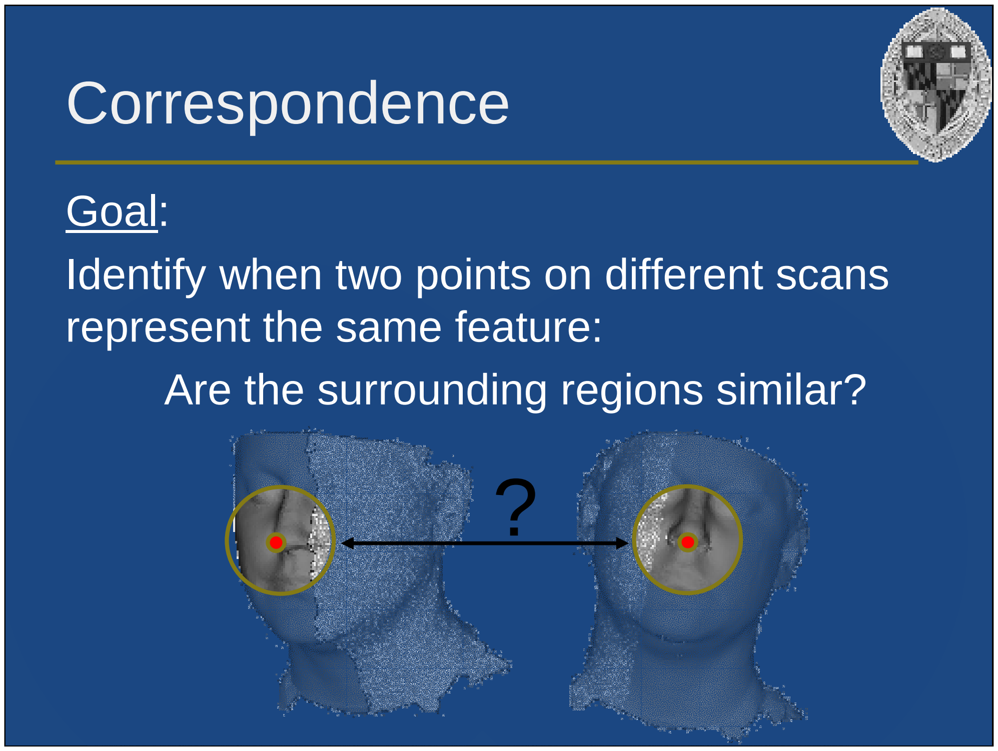
## Correspondence

Goal:

Identify when two points on different scans represent the same feature



In order to establish correspondences between points on the two scans, we need to identify when two points on different scans represent the same feature.

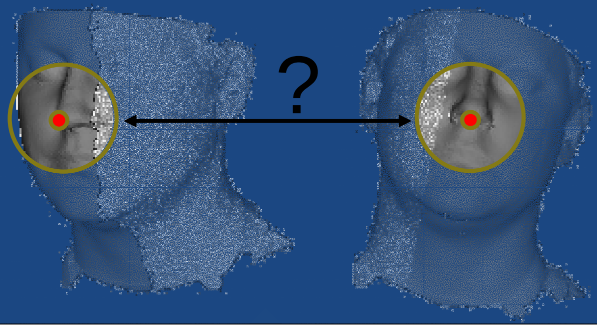


## Correspondence

Goal:

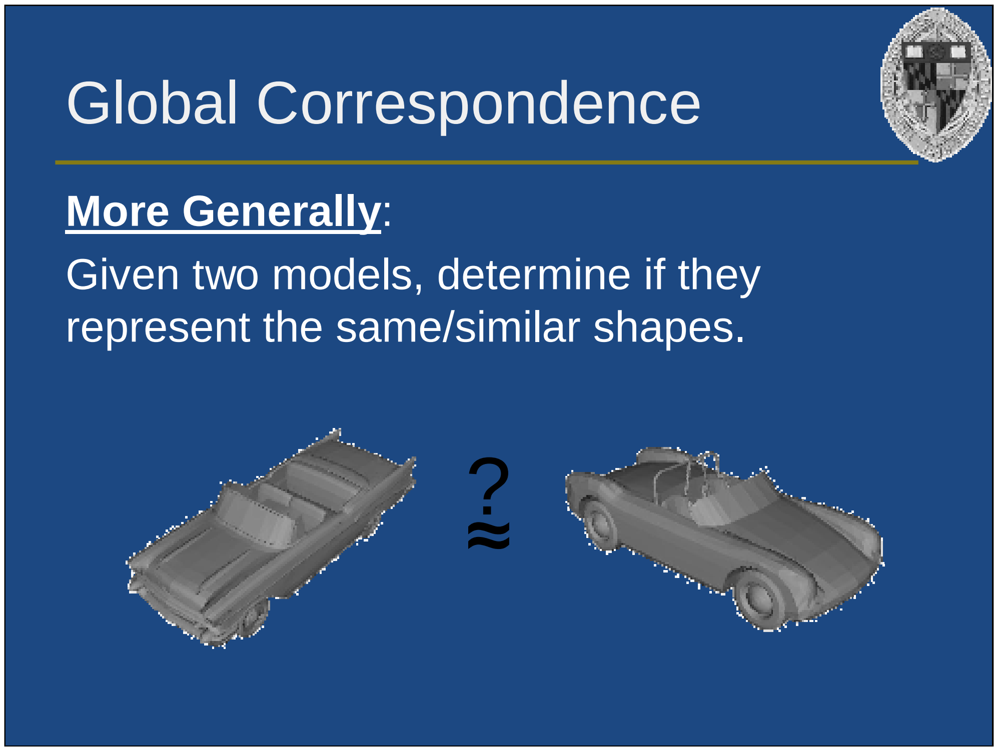
Identify when two points on different scans represent the same feature:

Are the surrounding regions similar?



In order to establish correspondences between points on the two scans, we need to identify when two points on different scans represent the same feature.

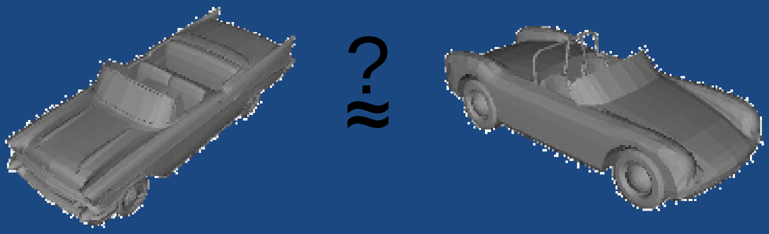
To do this we need to determine if the surface geometry about the two points is similar.



## Global Correspondence

**More Generally**:

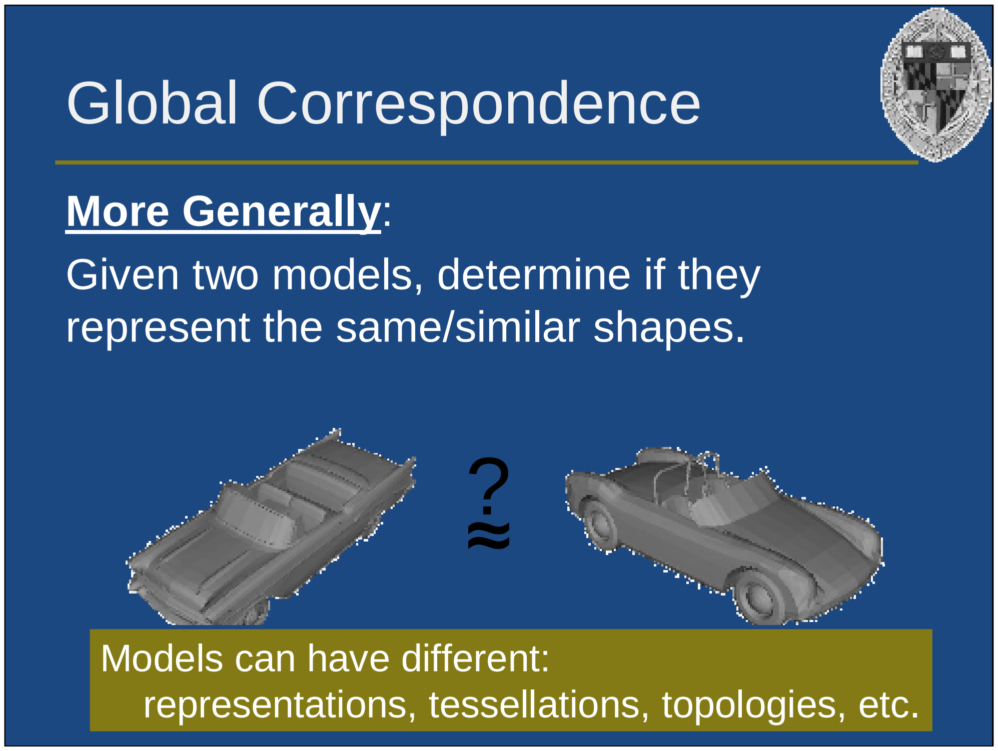
Given two models, determine if they represent the same/similar shapes.



The problem is closely related to the more general problem of whole shape matching that has been well studied.

The more general problem can be stated as follows:

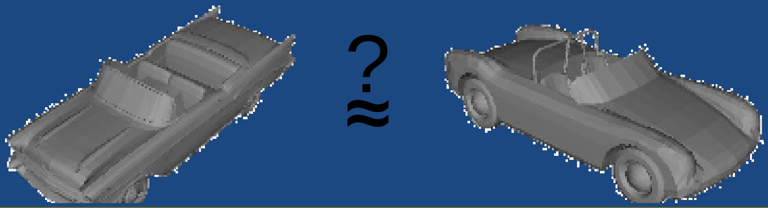
Given two models, determine if the two models are similar.



## Global Correspondence

**More Generally**:

Given two models, determine if they represent the same/similar shapes.



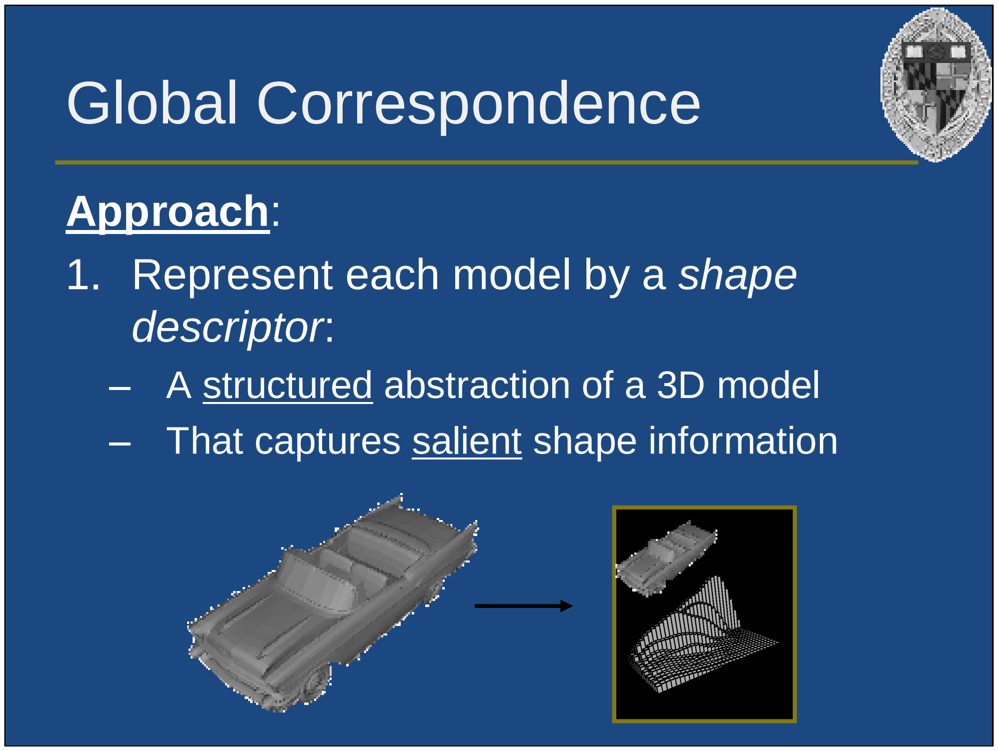
Models can have different:
representations, tessellations, topologies, etc.

The problem is closely related to the more general problem of whole shape matching that has been well studied.

The more general problem can be stated as follows:

Given two models, determine if the two models are similar.

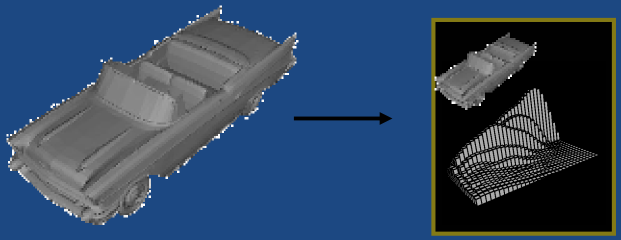
Trying to solve this problem directly in shape space can often be quite challenging: The models may be represented in different ways, they may have different topologies, they may be tessellated at different levels of detail, and so on.



## Global Correspondence

**Approach**:

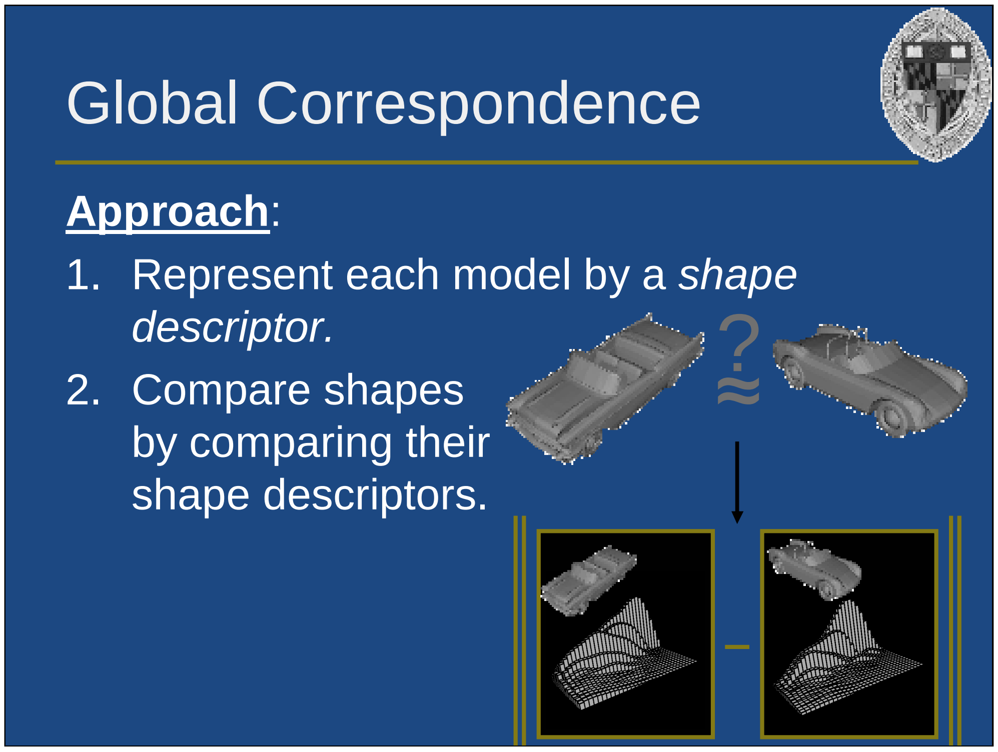
1. Represent each model by a *shape descriptor*:
   - A structured abstraction of a 3D model
   - That captures salient shape information



Instead, this problem is addressed with the use of a shape descriptor.

This is a structured representation of a 3D model capturing information about the salient features of the shape.

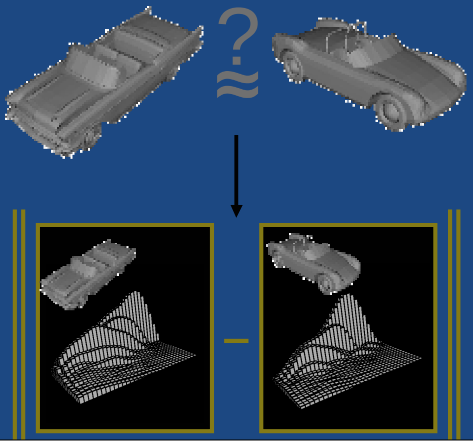
Most often, the shape descriptor is some fixed dimensional vector (either an array, or a function defined either on a sphere or in Euclidean Space).



## Global Correspondence

**Approach**:

1. Represent each model by a *shape descriptor.*
2. Compare shapes by comparing their shape descriptors.

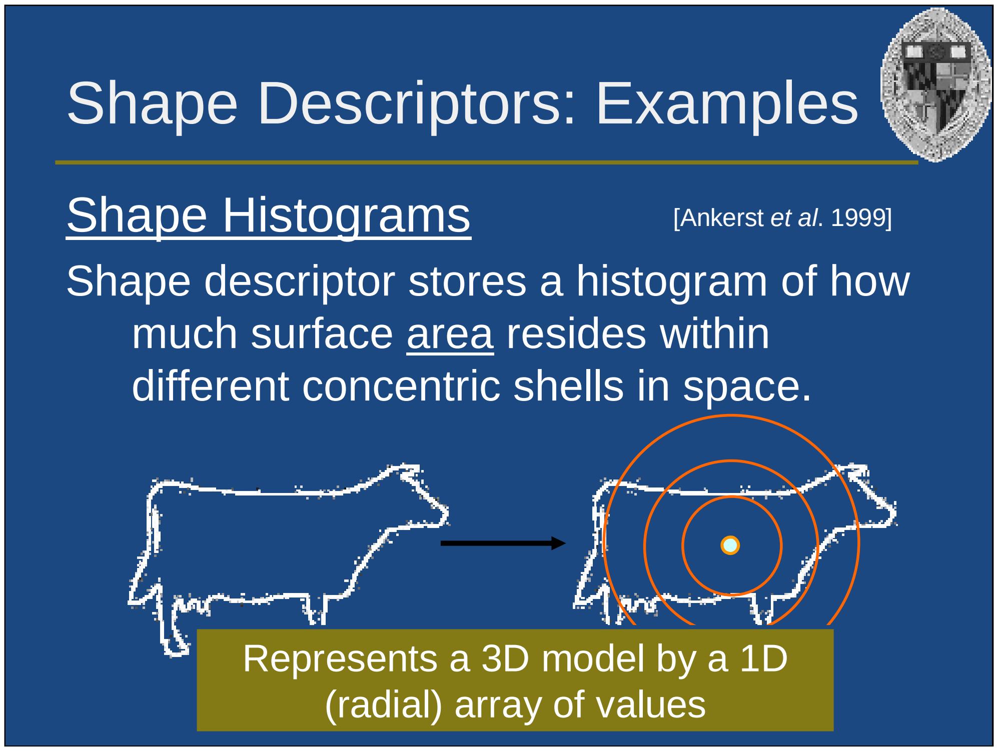


By structuring the shape information, it becomes much easier to compare the 3D models.

For example, if the shape descriptor represents models by a 2D array, we can simply define the distance between two shapes as the L2-distance between their corresponding descriptors.

In this case, the problem of shape matching is reduced to a simple problem of finding the Euclidean distance between two vectors and a variety of standard compression and indexing techniques can be used to assist in the matching.

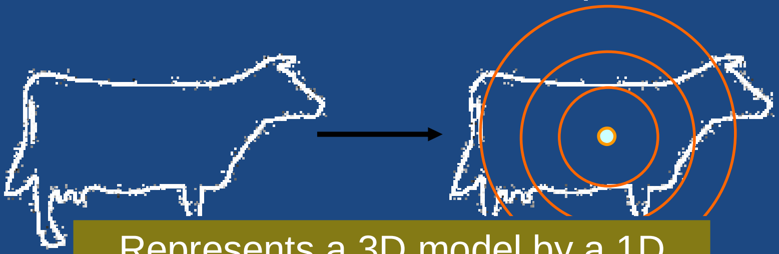
As examples, we consider three simple shape descriptors.



## Shape Descriptors: Examples

Shape Histograms [Ankerst *et al.* 1999]

Shape descriptor stores a histogram of how much surface area resides within different concentric shells in space.

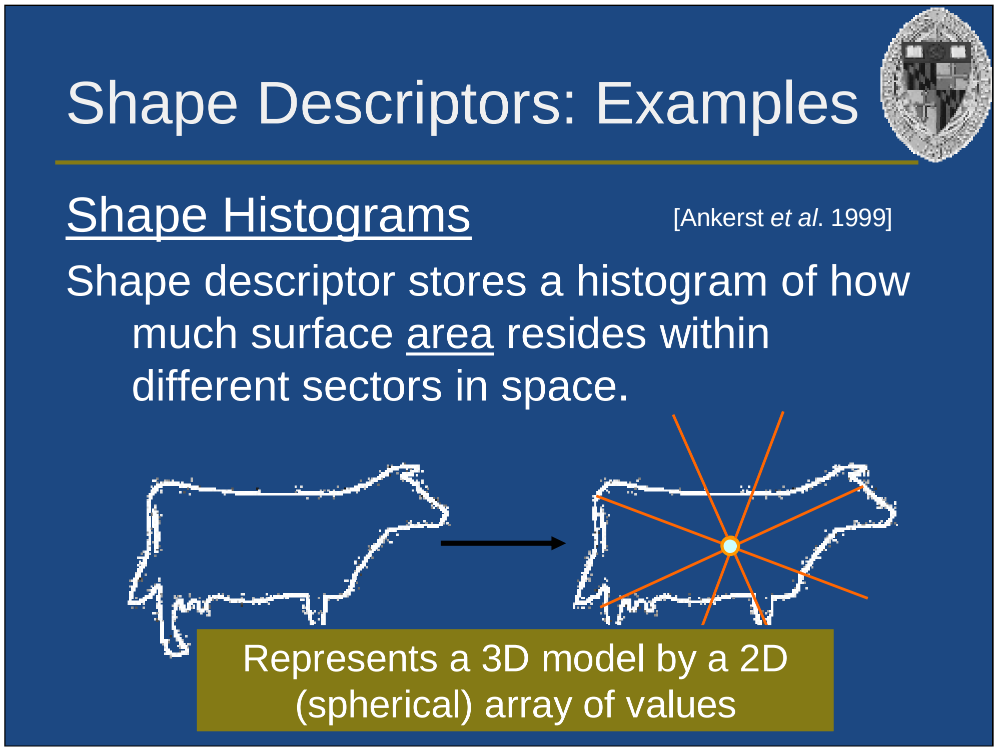


Represents a 3D model by a 1D (radial) array of values

In general, most shape descriptors represent a 3D model by the distribution of surface area across space.

As an example, Ankerst's Shape Histograms characterizes a 3D model by decomposing space into a collection of concentric shells.

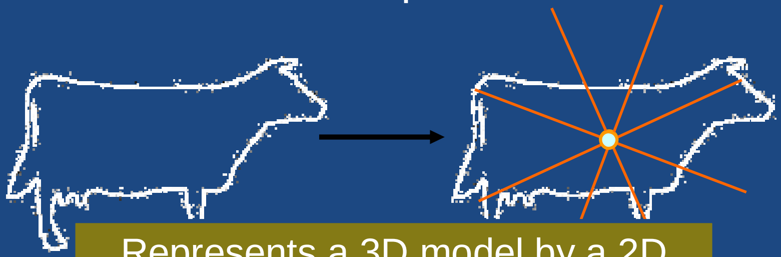
The area of the surface intersected by each shell is stored in a histogram indexed by radius, and this 1D array of values is used to represent the 3D shape.



## Shape Descriptors: Examples

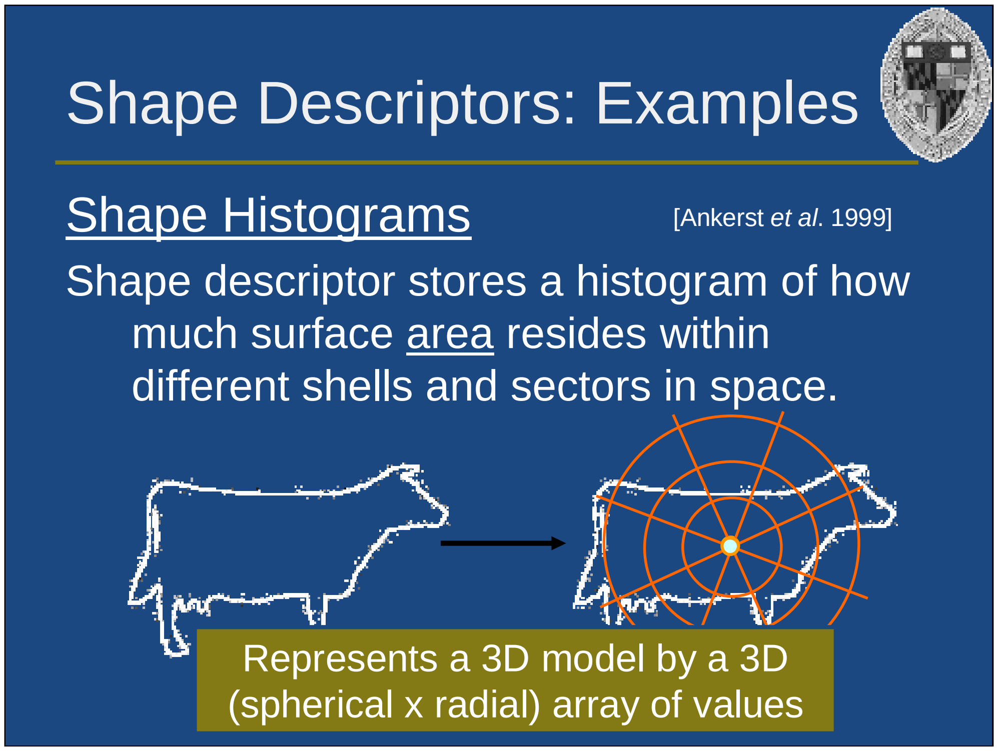
<u>Shape Histograms</u> [Ankerst *et al*. 1999]

Shape descriptor stores a histogram of how much surface <u>area</u> resides within different sectors in space.



Represents a 3D model by a 2D (spherical) array of values

In another example presented by Ankerst, space is decomposed into spherical sectors, and the shape is represented by a spherical array whose value in each bin corresponds to the area of the surface that falls into each sector.

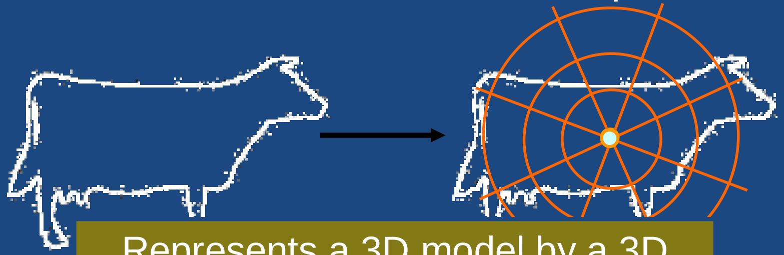


## Shape Descriptors: Examples

### Shape Histograms

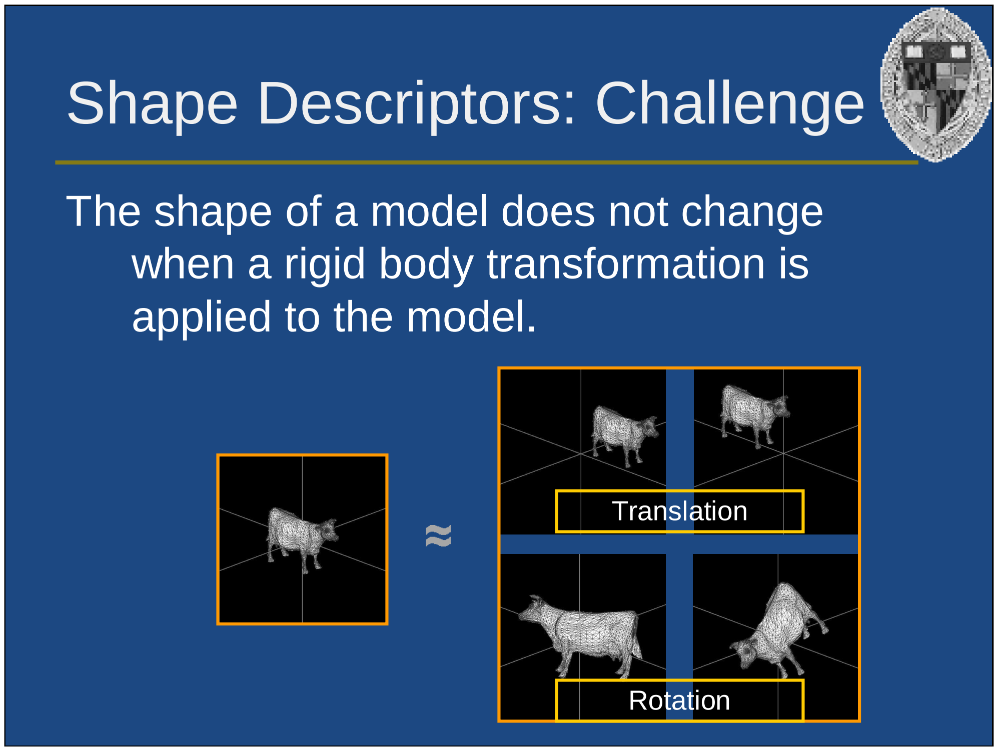
[Ankerst *et al*. 1999]

Shape descriptor stores a histogram of how much surface area resides within different shells and sectors in space.



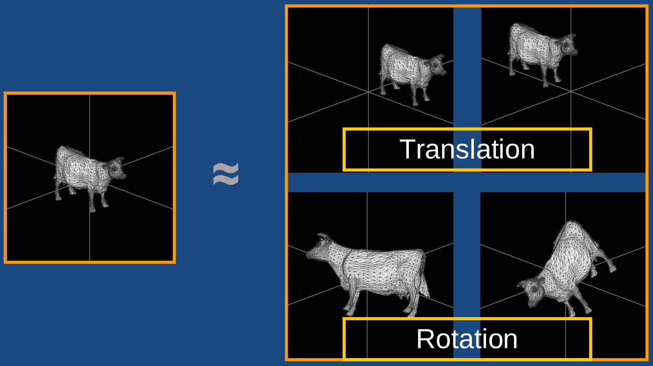
Represents a 3D model by a 3D (spherical x radial) array of values

Finally, combining the sectors and shells representation, we can obtain a representation of the 3D model by a 3D array of bins, with the value in each bin corresponding to the area of the surface intersected by a given sector and shell.



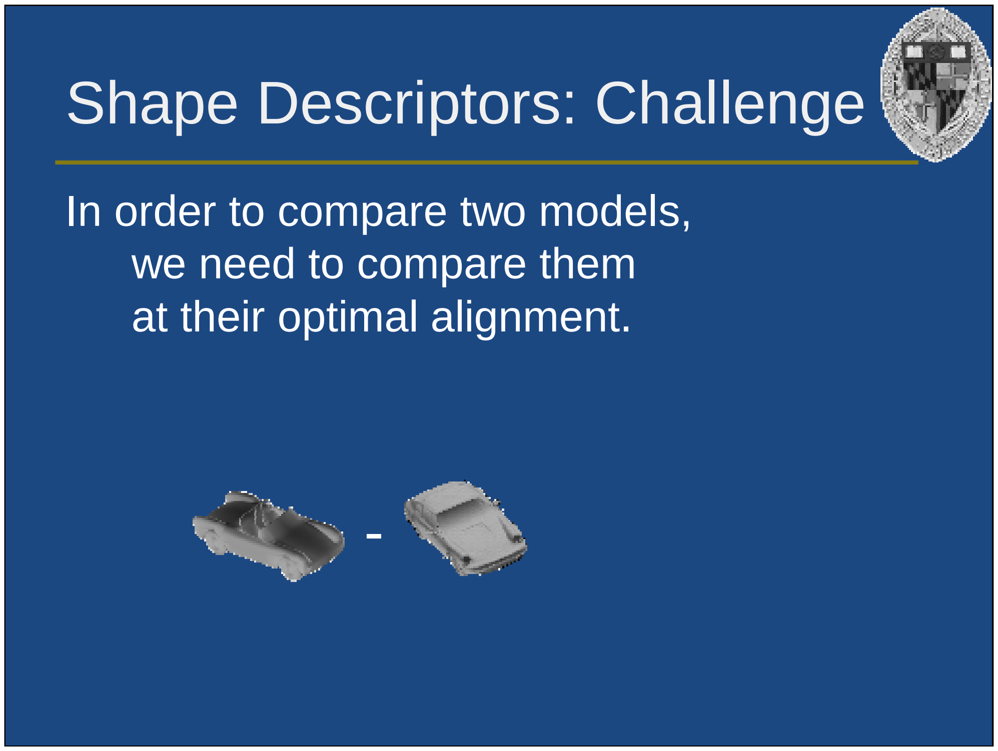
## Shape Descriptors: Challenge

The shape of a model does not change when a rigid body transformation is applied to the model.



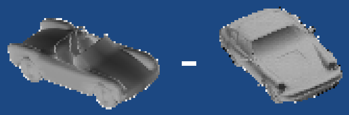
Independent of the choice of shape descriptor, what makes shape matching particularly challenging is the fact that the notion of the shape of a model is often independent of the model’s pose.

For example, regardless of how the geometry of the cow on the left is translated and rotated through space, it still represents the same shape.



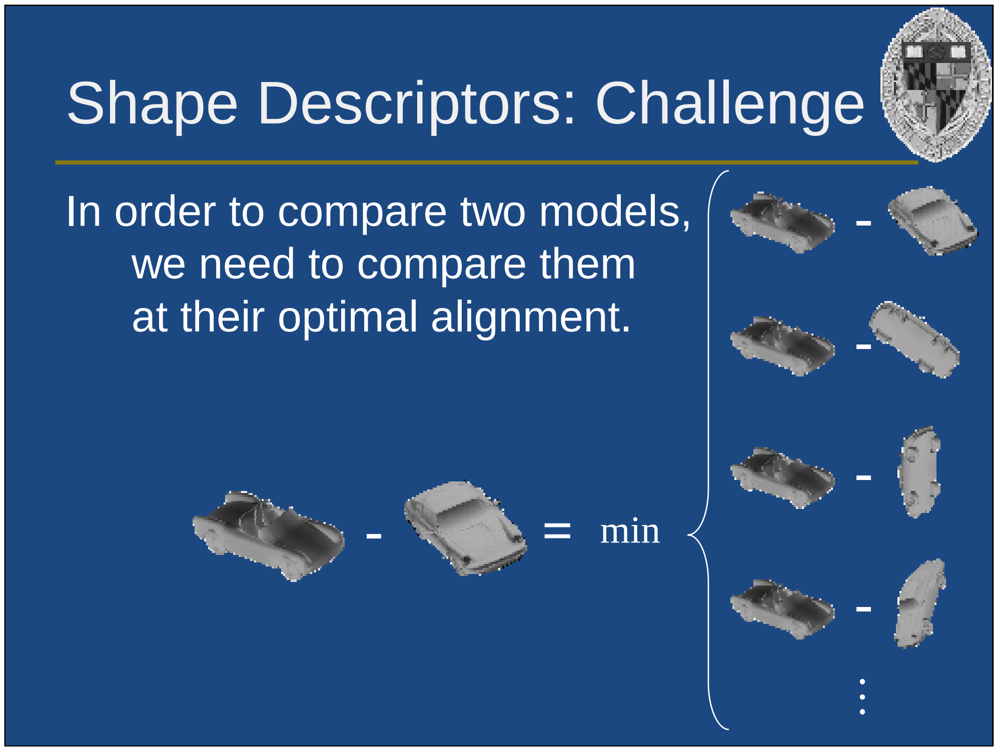
## Shape Descriptors: Challenge

In order to compare two models, we need to compare them at their optimal alignment.



Consequently, when we compare two models, we need to compute the optimal measure of similarity, over all possible poses.

For example, if we would like to compare these two car models, it would not be sufficient to compare them in their initial poses.



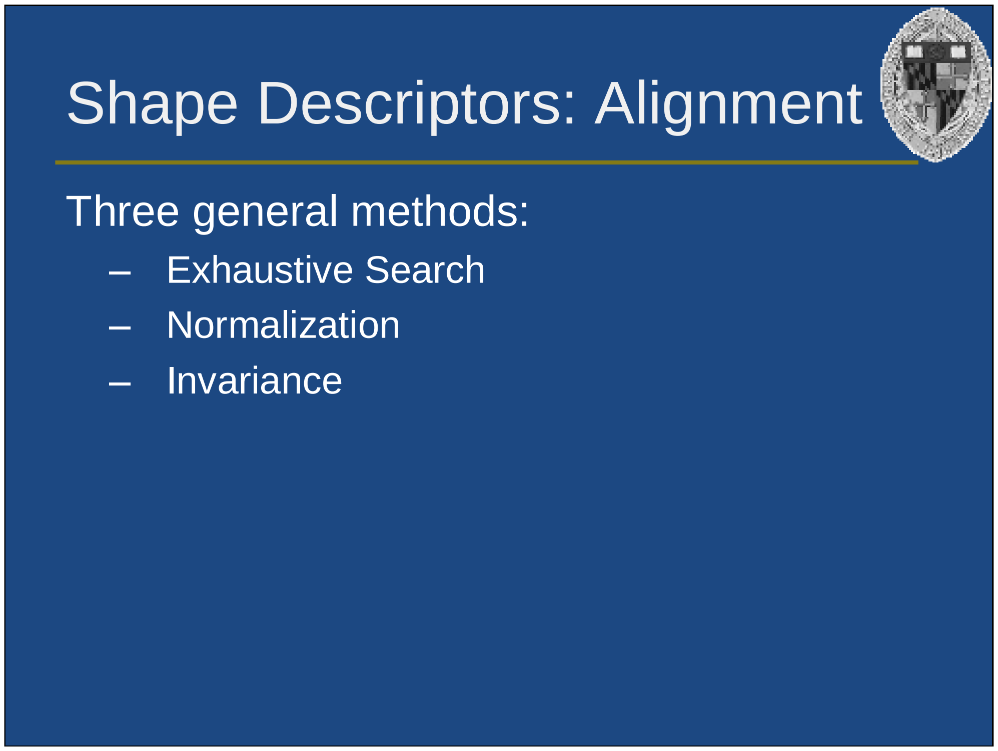
## Shape Descriptors: Challenge

In order to compare two models, we need to compare them at their optimal alignment.

Consequently, when we compare two models, we need to compute the optimal measure of similarity, over all possible poses.

For example, if we would like to compare these two car models, it would not be sufficient to compare them in their initial poses.

Instead, we would have to compare them at all possible alignments and use the smallest distance between the two models, over all possible poses, as the measure of their similarity

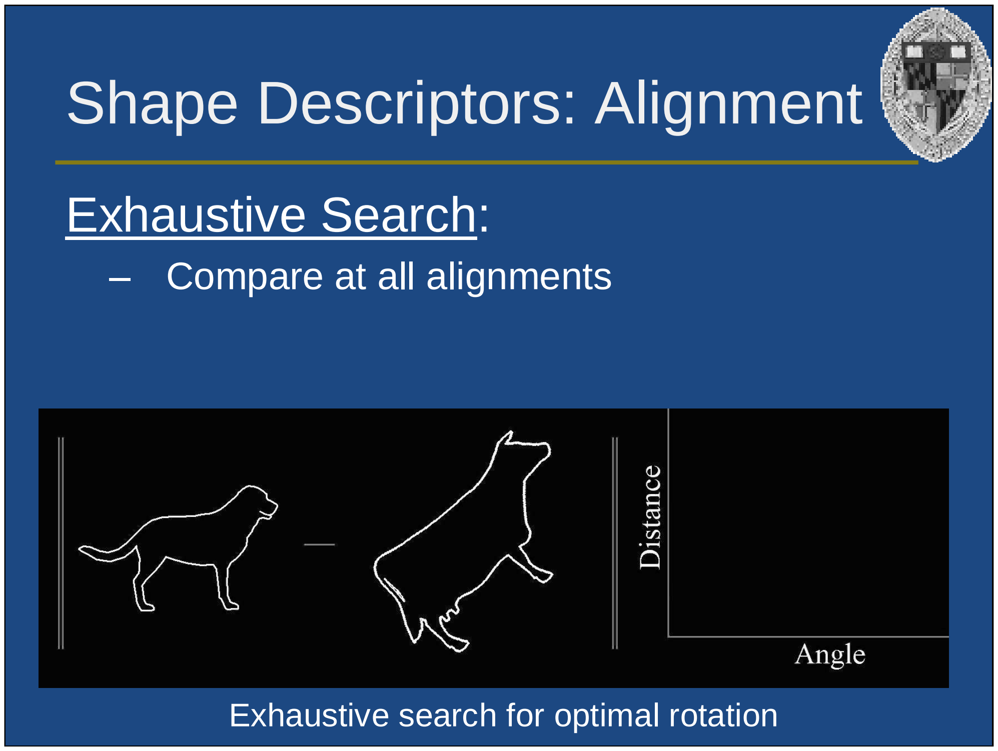


## Shape Descriptors: Alignment

Three general methods:

- Exhaustive Search
- Normalization
- Invariance

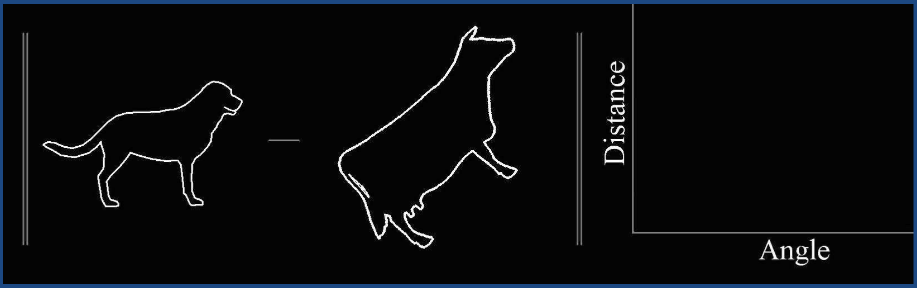
This problem has been addressed in three different ways:



## Shape Descriptors: Alignment

Exhaustive Search:

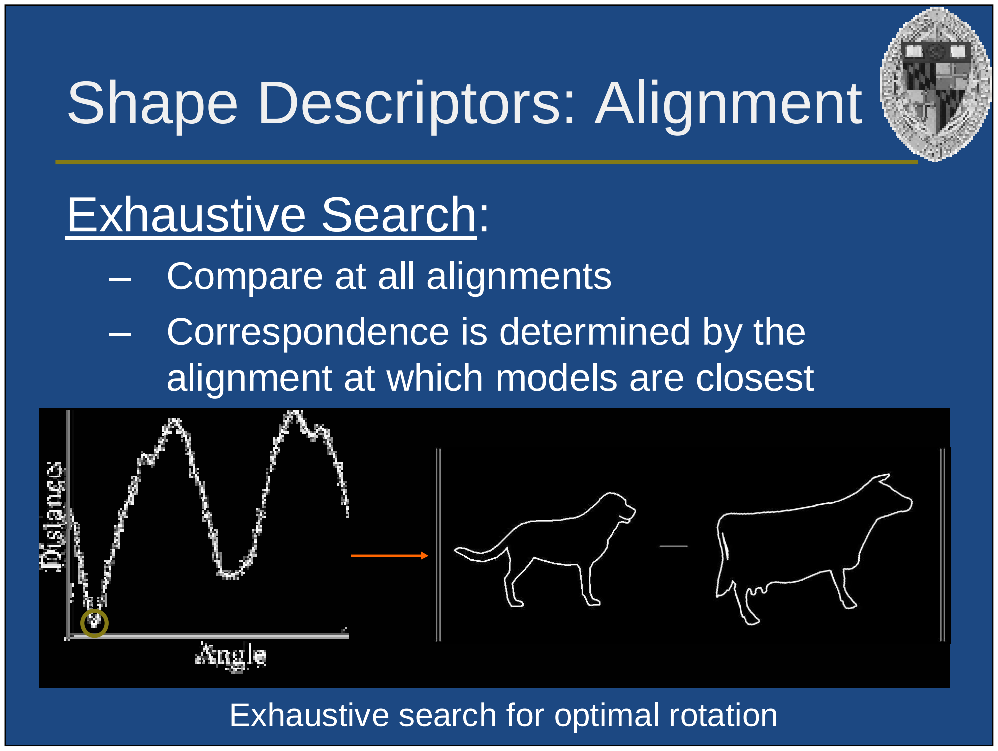
– Compare at all alignments



Exhaustive search for optimal rotation

The first, and most direct approach, is the brute force moethod. To find the measure of similarity,  at the optimal alignment, we explicitly compute the distance between the two models at every possible alignment and then define the measure of similarity as the minimum of these distances.

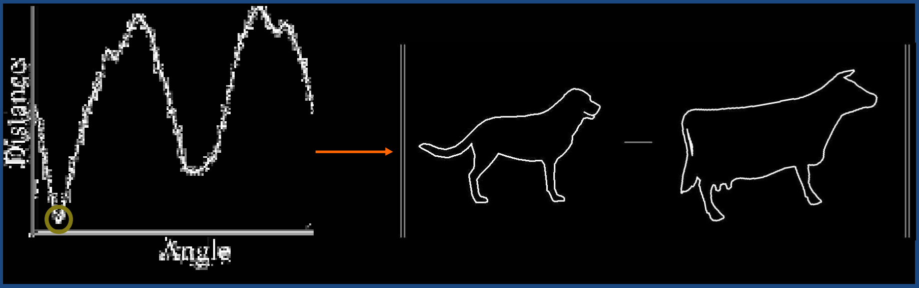
For example, if we wanted to compare the dog model with the cow model  over the space of rotations, we would compute the distance between the two models at every alignment…



## Shape Descriptors: Alignment

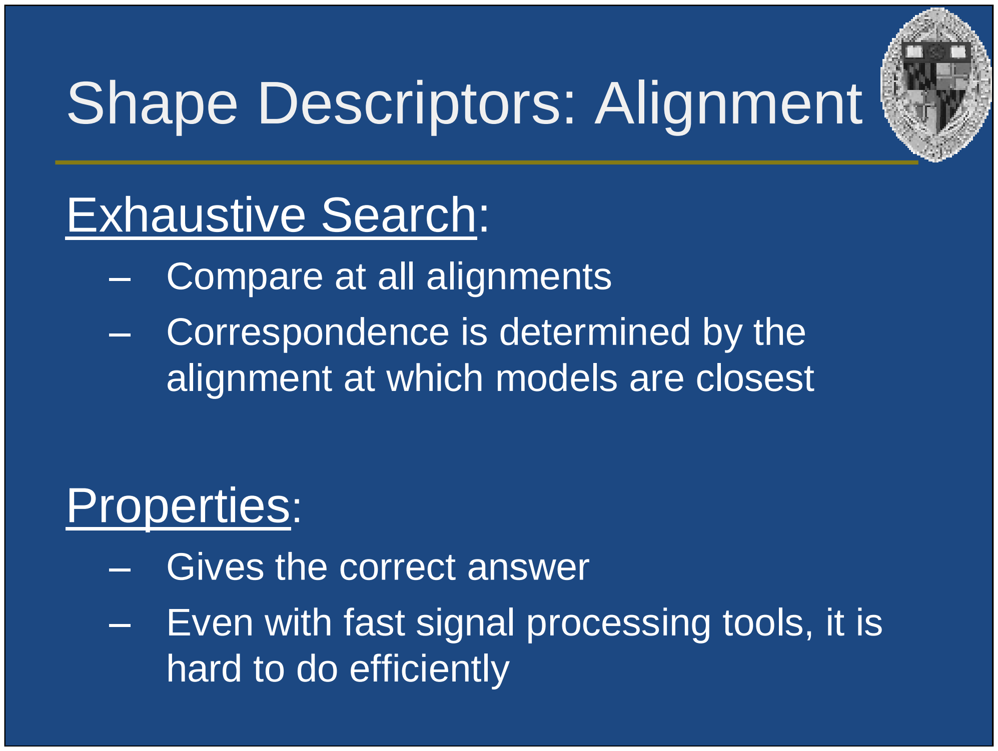
Exhaustive Search:

- Compare at all alignments
- Correspondence is determined by the alignment at which models are closest



Exhaustive search for optimal rotation

… And then find the alignment at which the distance between the two models is minimized. The value at the optimal alignment can then be used as the measure of model similarity.

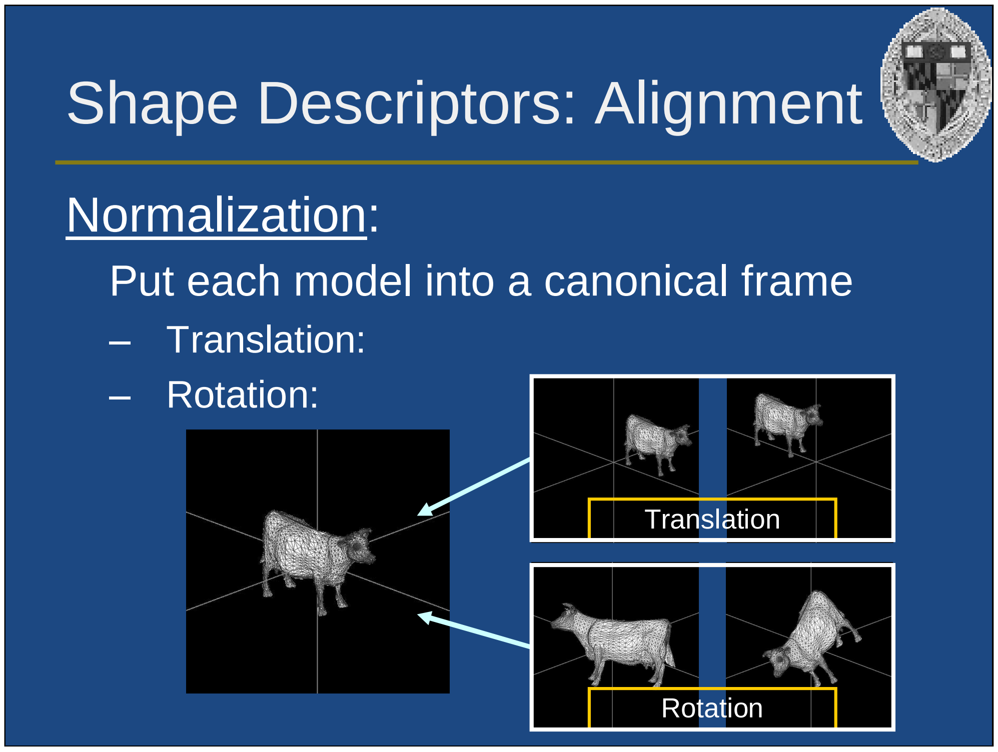


## Shape Descriptors: Alignment

Exhaustive Search:
- Compare at all alignments
- Correspondence is determined by the alignment at which models are closest

Properties:
- Gives the correct answer
- Even with fast signal processing tools, it is hard to do efficiently

While this method has the advantage of always returning the correct answer, it is usually too slow to be useful in practical settings.



## Shape Descriptors: Alignment

Normalization:

Put each model into a canonical frame

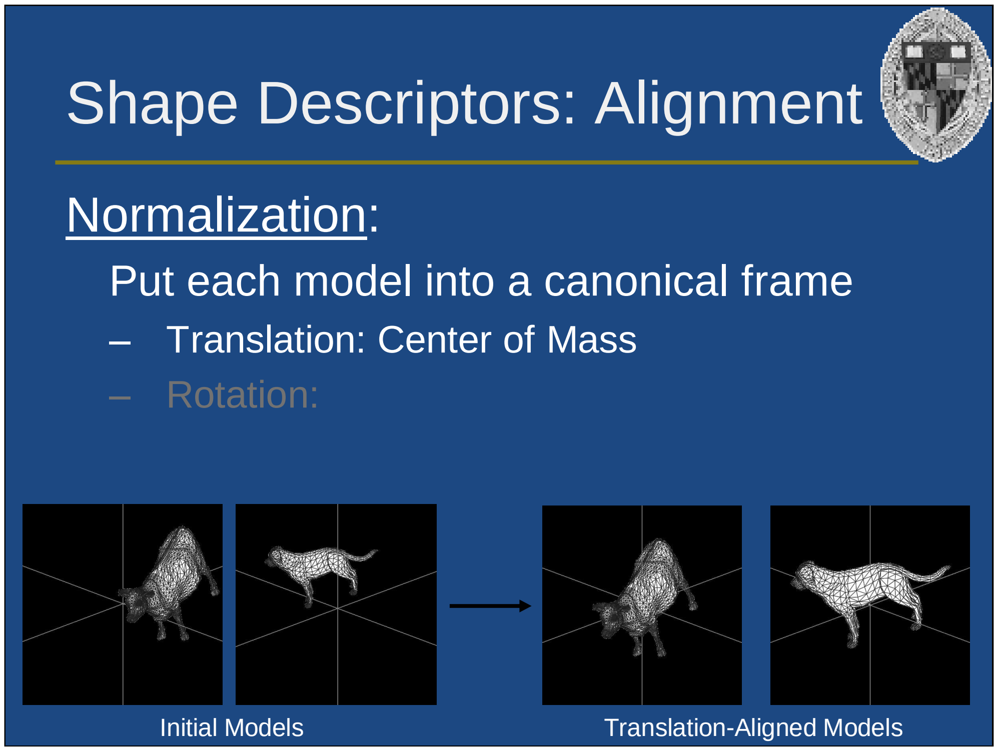
- Translation:
- Rotation:

Translation
Rotation

The second method for addressing the alignment problem is to try to find the optimal alignment between the two models without explicitly testing all possible transformations.

To do this, each model is placed into a canonical coordinate frame, normalizing for translation and rotation.

Then, when two normalized models are compared they will already be optimally aligned, and it will not be necessary to perform a search over the space of transformations.

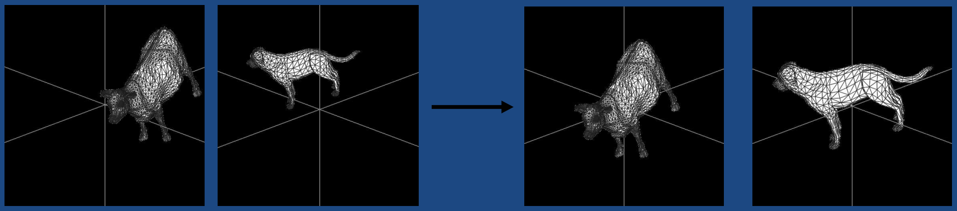


## Shape Descriptors: Alignment

Normalization:

Put each model into a canonical frame

- Translation: Center of Mass
- Rotation:



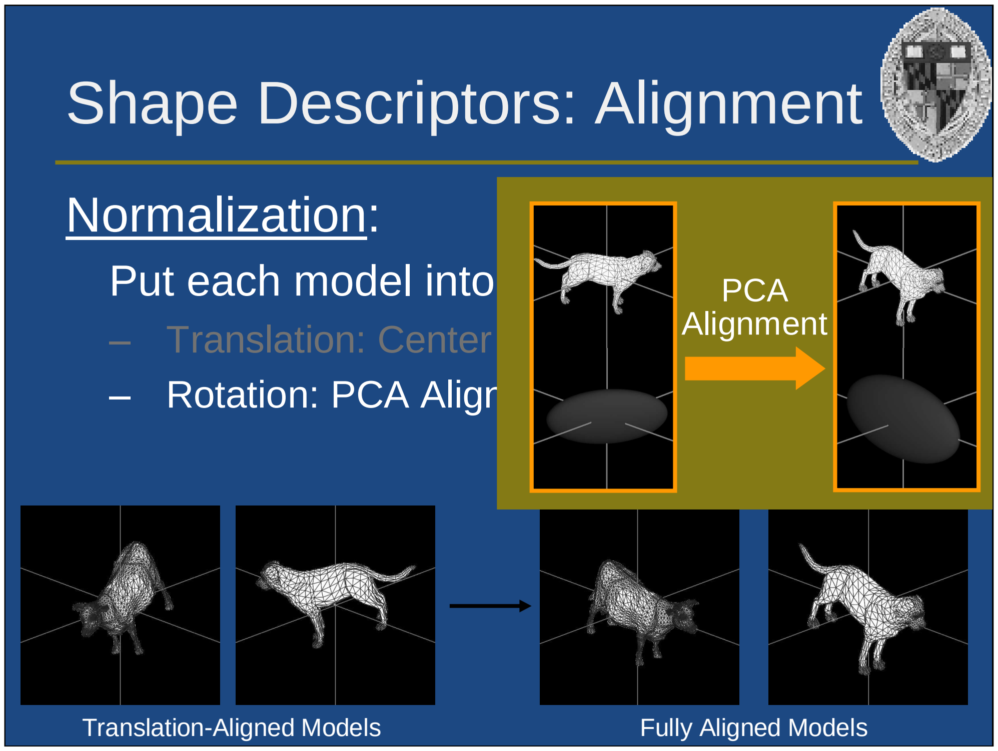
Initial Models

Translation-Aligned Models

In practice, normalization is performed for the different components of the transformation independently.

First, the models are normalized for translation by shifting them so that the center of mass is aligns with the origin.

In fact, most shape descriptor do this implicitly by centering the shape descriptor about the model's center of mass.



## Shape Descriptors: Alignment

Normalization:

Put each model into

– Translation: Center

– Rotation: PCA Align

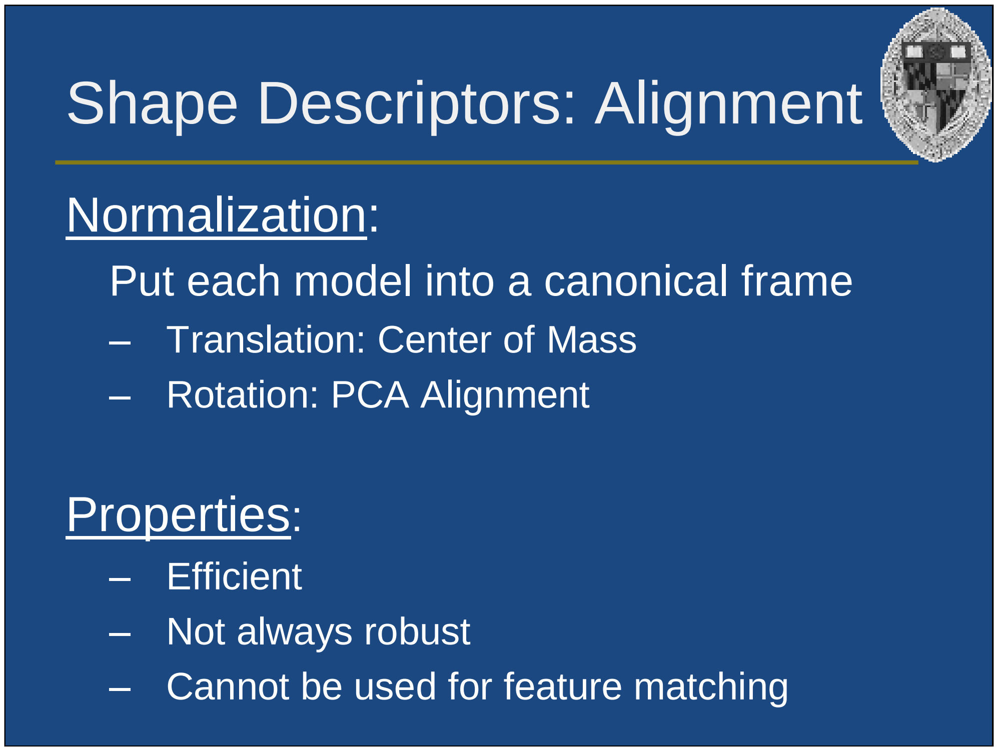
Next, the models are normalized for rotation by aligning the principal axes of the model with the x-, y- and z-axes.

An example of this type of normalization is shown for the dog.

On the left we see the dog in its initial pose and a visualization of its covariance ellipsoid – the ellipsoid that best fits the surface of the model.

By rotating the dog so that the major axis of the ellipsoid aligns with the x-axis, and the 2nd major axis aligns with the y-axes, we obtain a model of the dog in a normalized coordinate frame.

As the bottom row indicates, after normalizing for translation and rotation, the two models are (near) optimally aligned and can be directly compared in their normalized poses.



## Shape Descriptors: Alignment

Normalization:

Put each model into a canonical frame

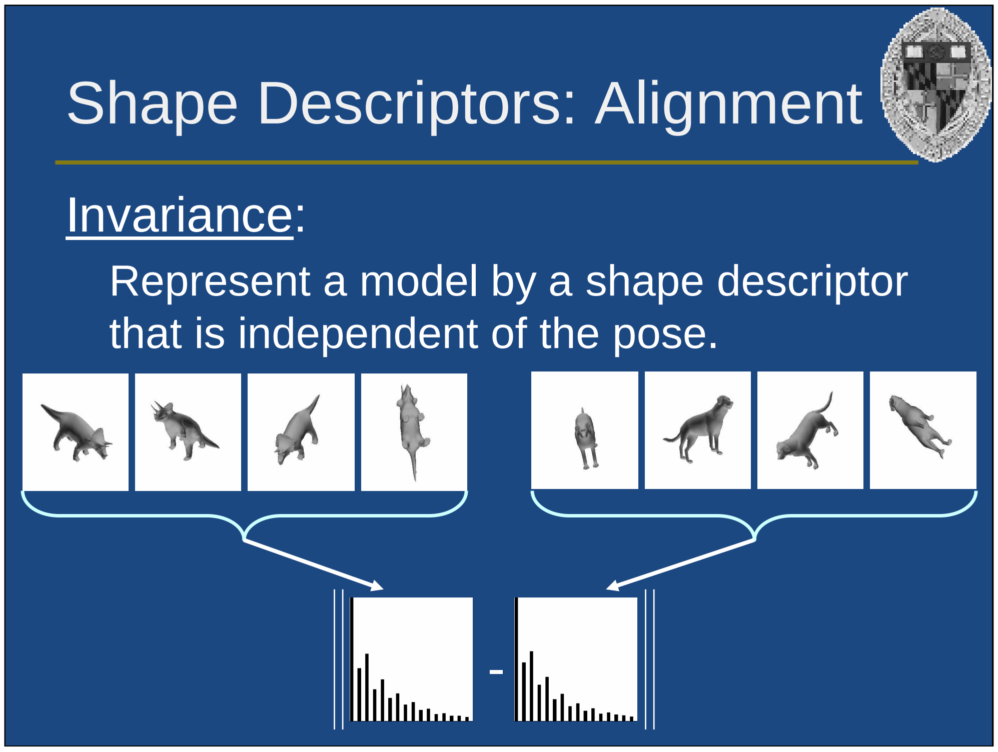
- Translation: Center of Mass
- Rotation: PCA Alignment

Properties:

- Efficient
- Not always robust
- Cannot be used for feature matching

While this method bypasses any searching over the space of transformations and provides an efficient method for addressing the alignment problem, care must be taken in using the method as it is not always true that when each of the two models are in its own canonical fame then they are also optimally aligned to each other.

Furthermore, some of these normalization techniques are difficult to translate to the context of feature point matching



## Shape Descriptors: Alignment

Invariance:

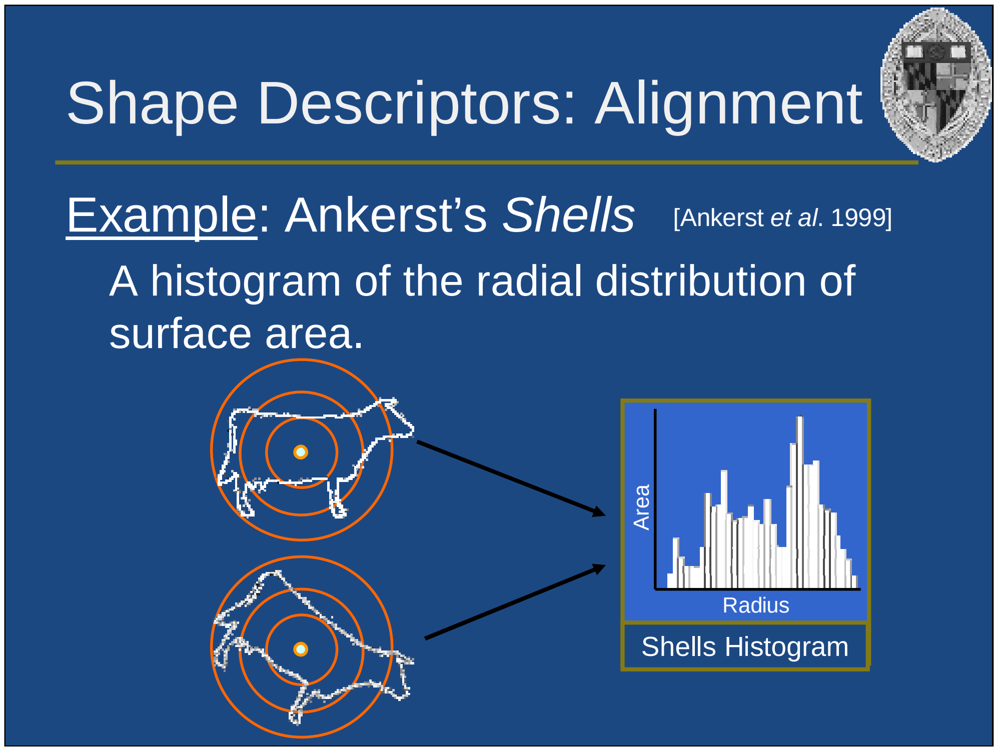
Represent a model by a shape descriptor that is independent of the pose.

The last approach designs a shape descriptor so as to avoid the alignment problem altogether.

That is, the shape descriptor is designed to be invariant to rigid body transformations.

Thus, regardless of the pose of the model, the measure of similarity remains unchanged, and any pose can be used for matching.

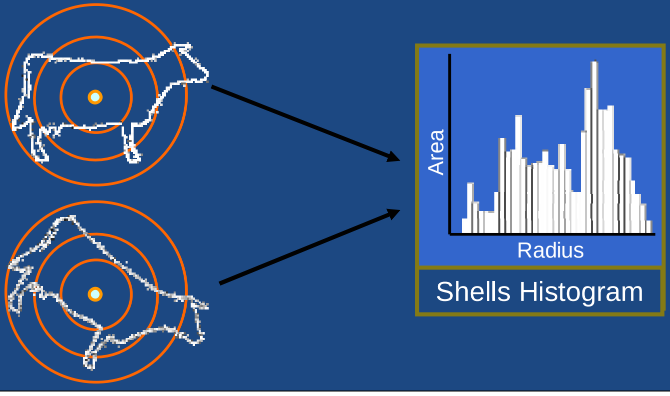
For example, if we wanted to compare the triceratops model to the dog model, then if the shape descriptor is transformation invariant, the shape descriptors of the triceratops and dog will not change with pose and the same measure of similarity is computed at all alignments.



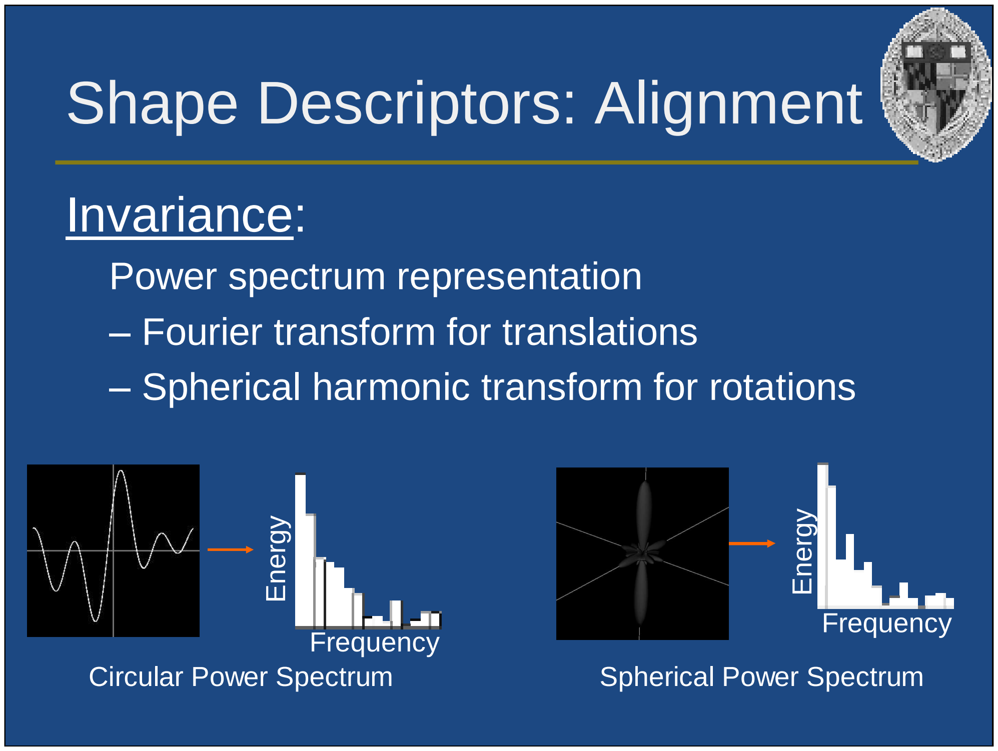
## Shape Descriptors: Alignment

Example: Ankerst's *Shells* [Ankerst *et al*. 1999]

A histogram of the radial distribution of surface area.



An example of such a shape descriptor is Ankerst's Shells descriptor mentioned earlier, obtained by decomposing space into concentric shells about the model's center of mass, and then computing the amount of surface area that falls within each shell. Since rotations preserve distances, any rotation about the center of mass will give rise to the same distribution, and hence the shape descriptor is invariant to rotations.



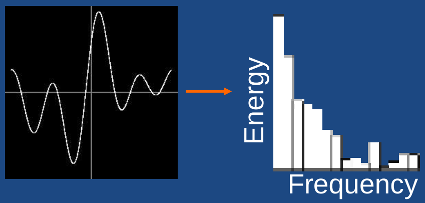
## Shape Descriptors: Alignment

Invariance:

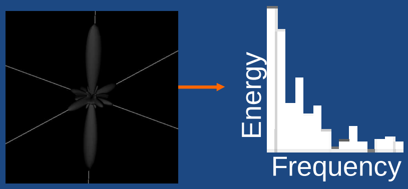
Power spectrum representation

– Fourier transform for translations

– Spherical harmonic transform for rotations



Circular Power Spectrum



Spherical Power Spectrum

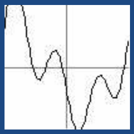
More generally, a transformation independent representation can be generated from a transformation dependent one by using Signal Processing techniques to extract the power spectrum. The new descriptor discards all the data that is pose-dependent and results in a transformation invariant shape representation.

For translations, this is done by computing the Fourier transform and storing only the amplitudes of the different frequency components, discarding phase.

For rotations, an analogous approach is taken, using the Spherical Harmonic transform to discard the spherical phase.

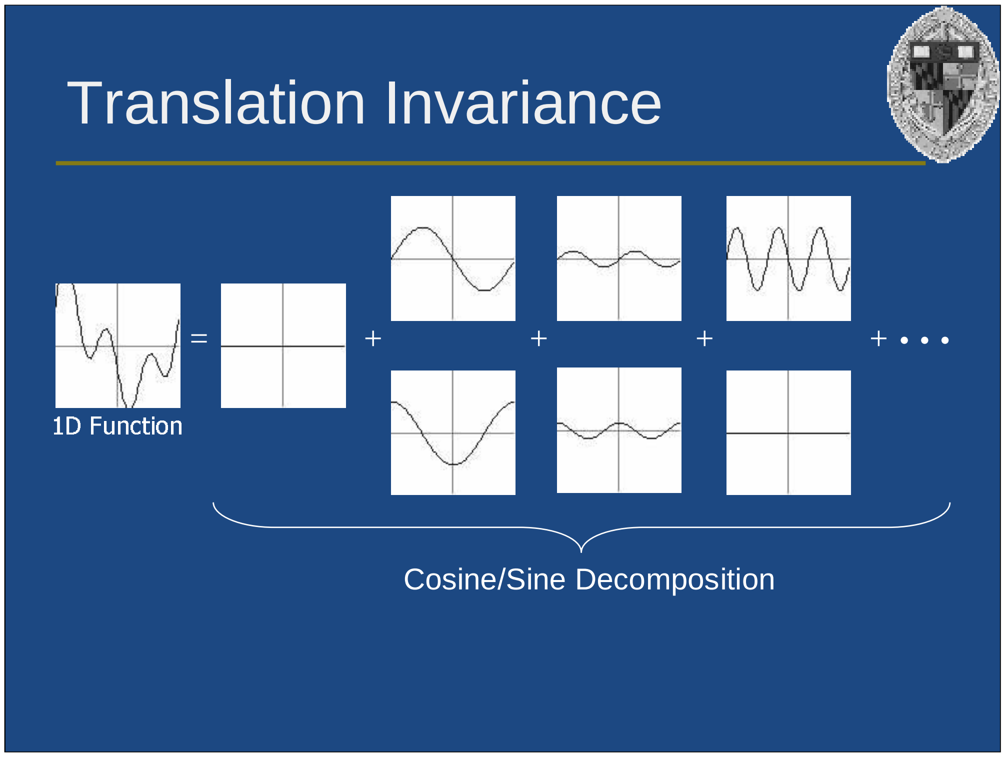
We briefly review these two approaches:

## Translation Invariance



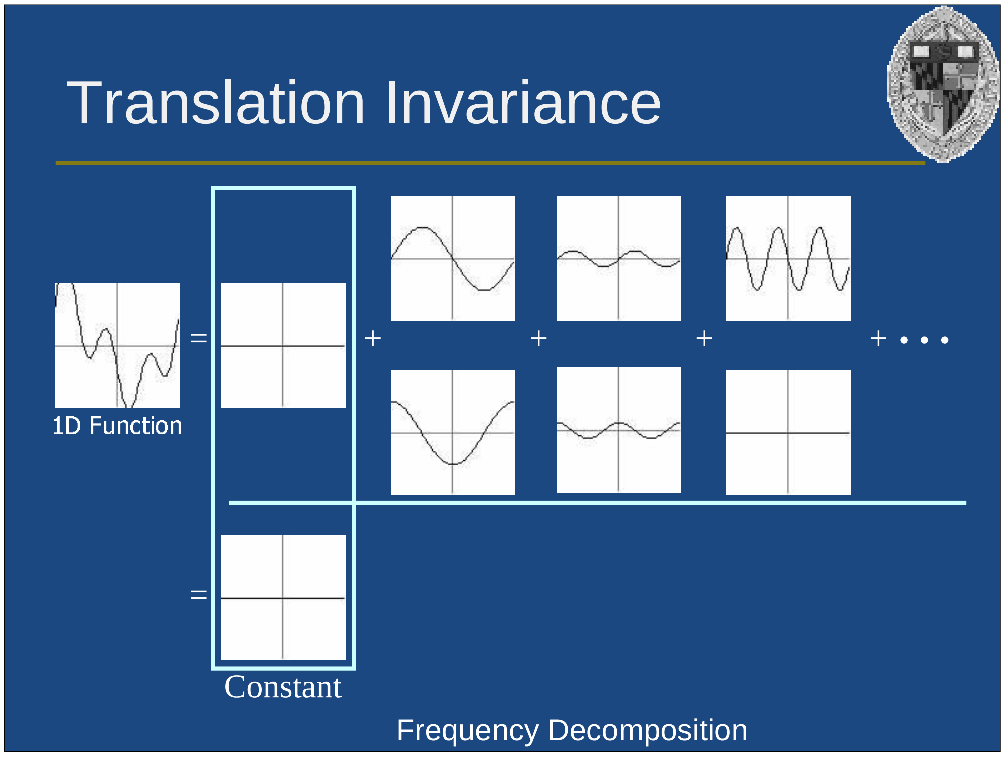
1D Function

In the one-dimensional case, our input is a function defined on an interval. By using the Fourier decomposition, we can express the initial function in terms of its cosine and sine expansion:



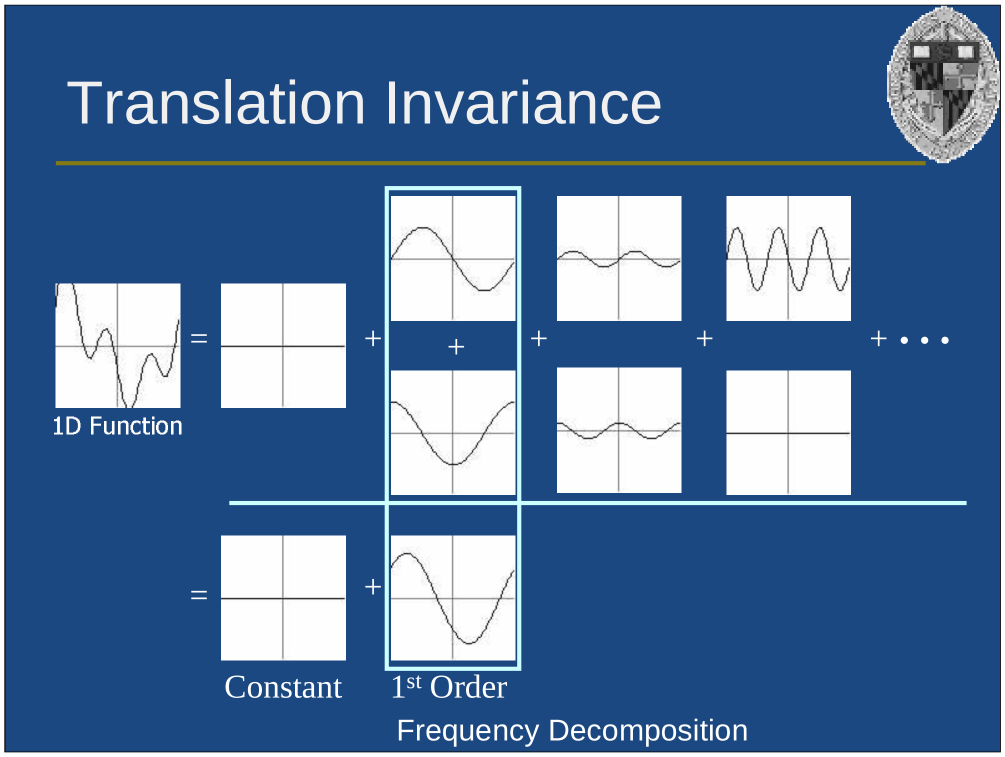
## Translation Invariance

The important fact about this expansion is that we can combine the cosine and sine terms to obtain a frequency decomposition, expressing the initial function as a sum of:



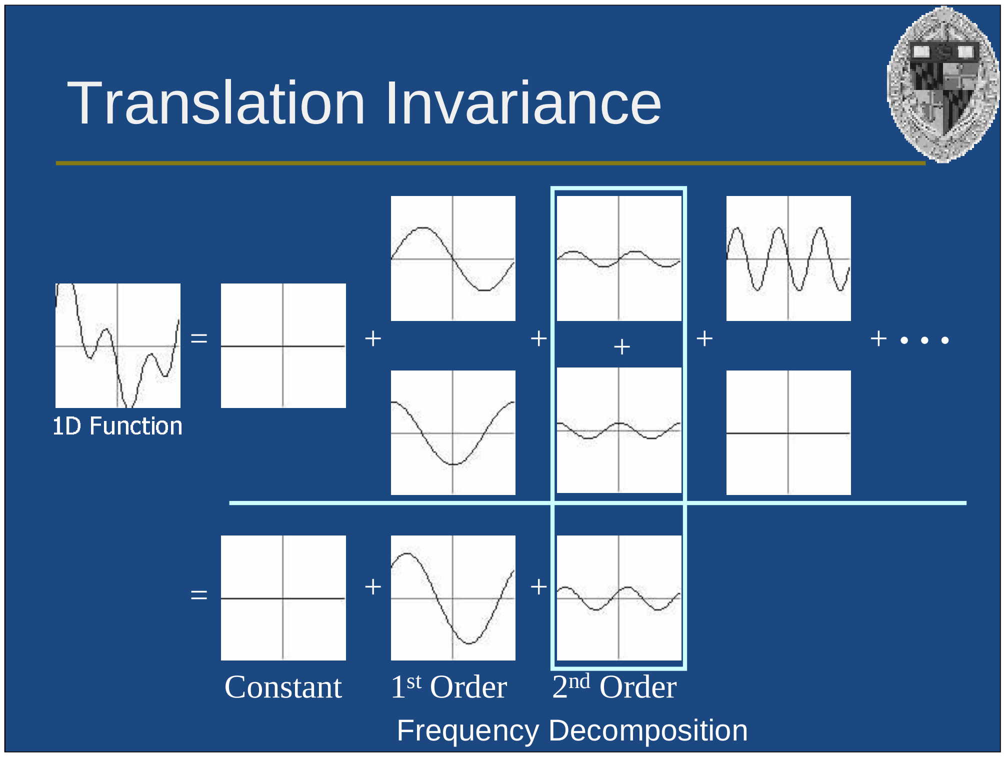
## Translation Invariance

A constant term,



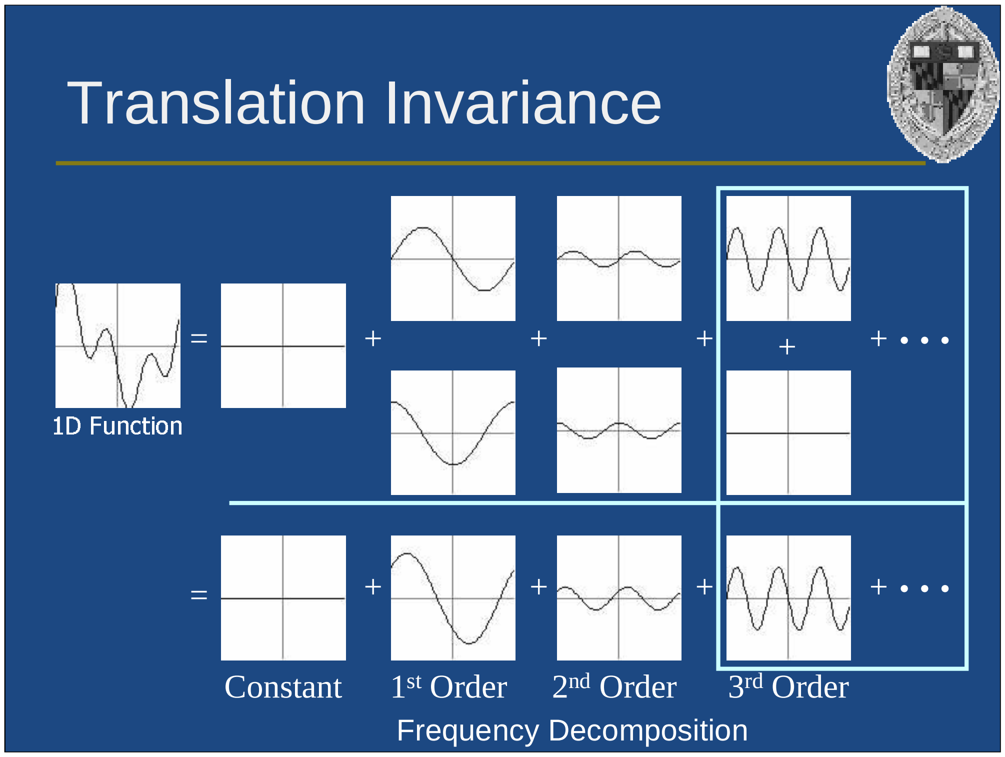
## Translation Invariance

A first order term,

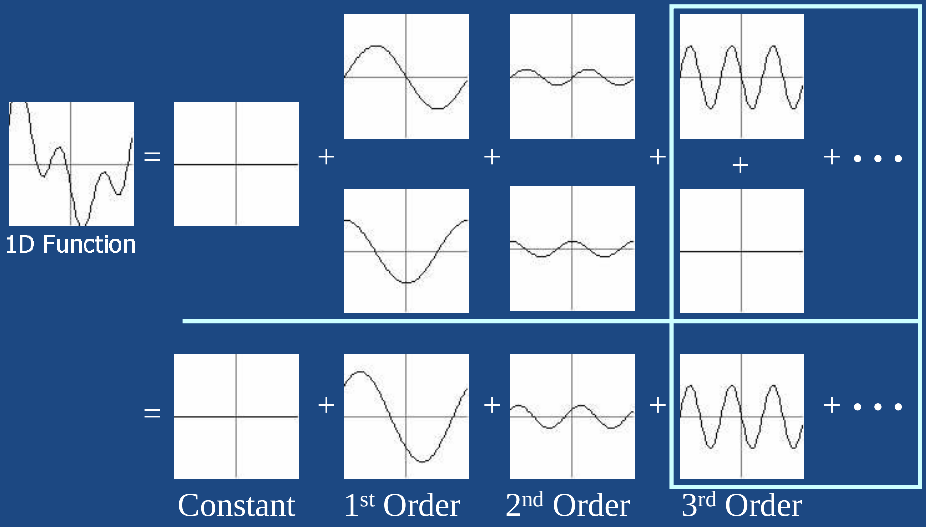


## Translation Invariance

A second order term,

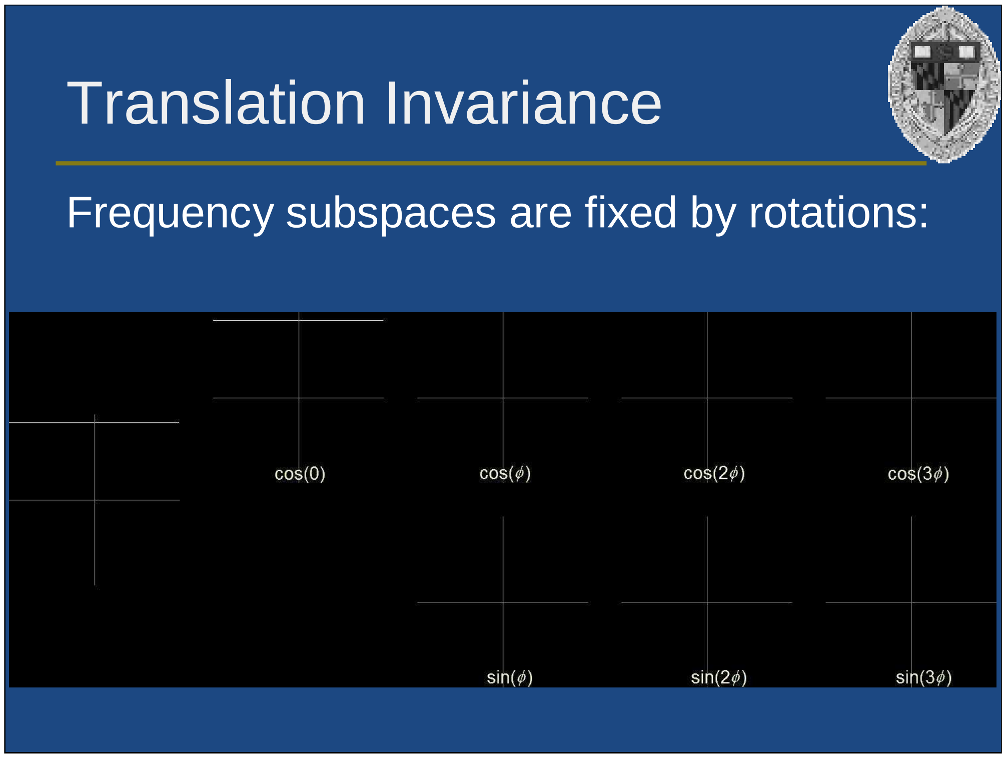


## Translation Invariance



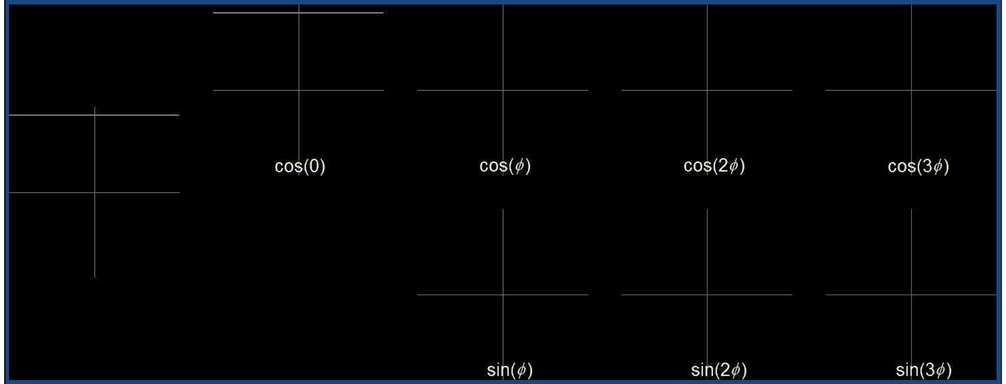
Frequency Decomposition

And so on.

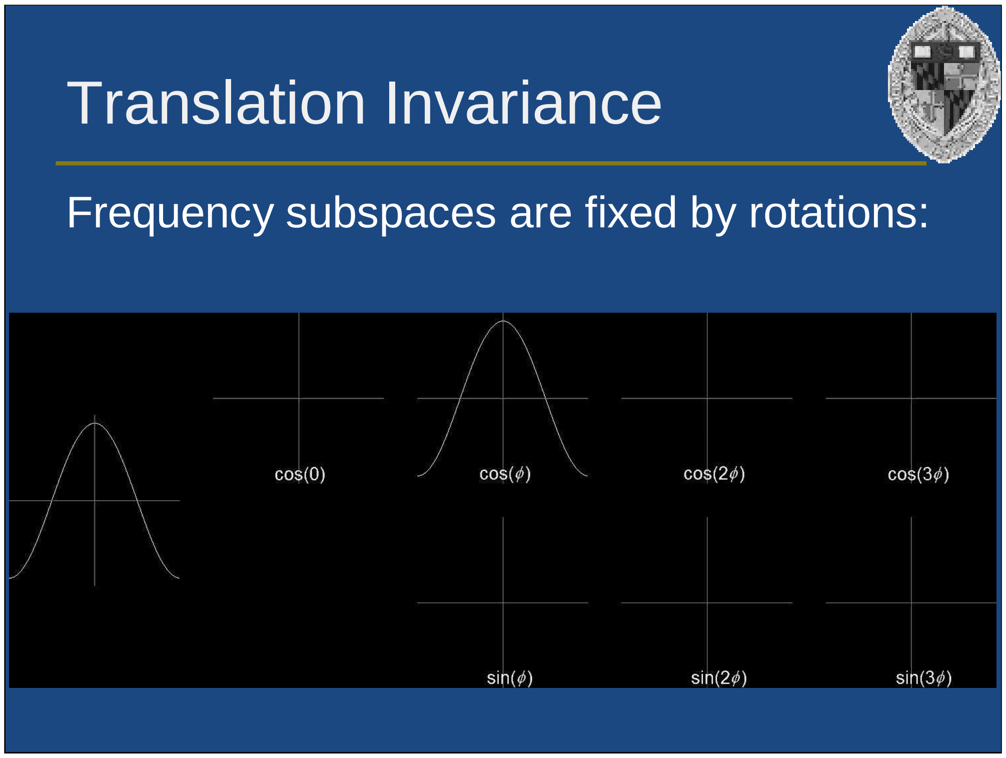


## Translation Invariance

Frequency subspaces are fixed by rotations:

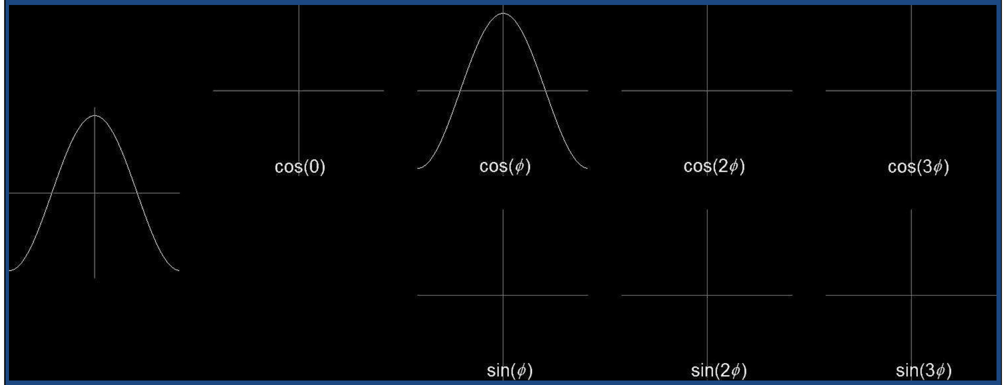


For example, if we start with the constant-order function shown on the left, and we start applying translations to it, the resultant function can always be expressed as a sum of the constant-order sines and cosines.

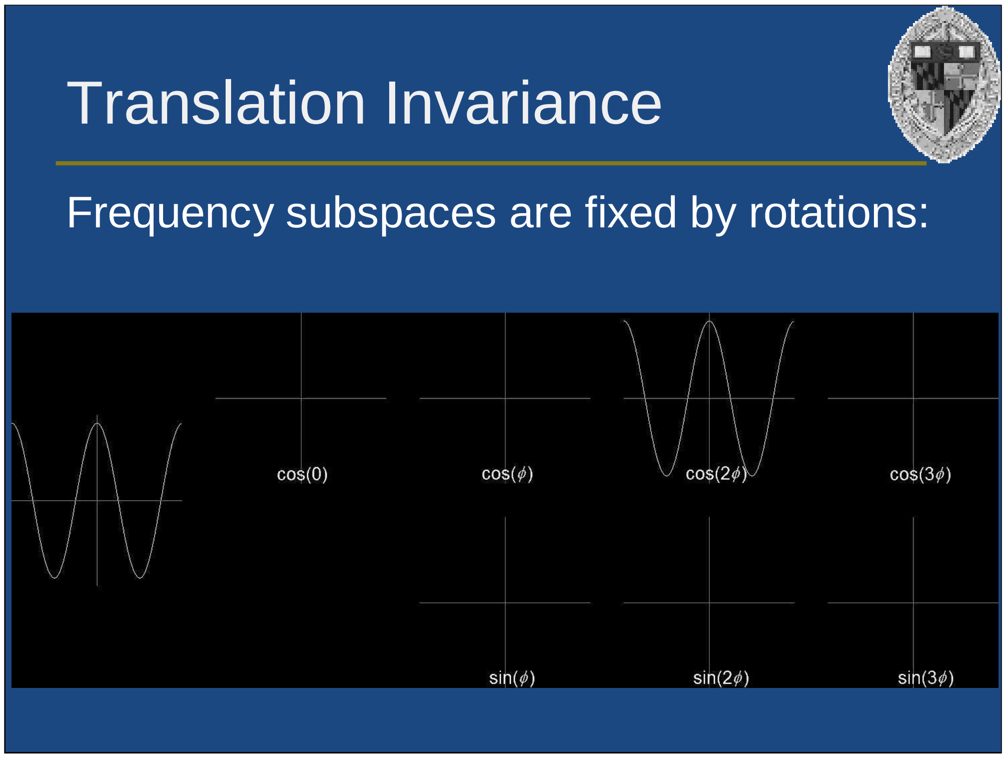


## Translation Invariance

Frequency subspaces are fixed by rotations:

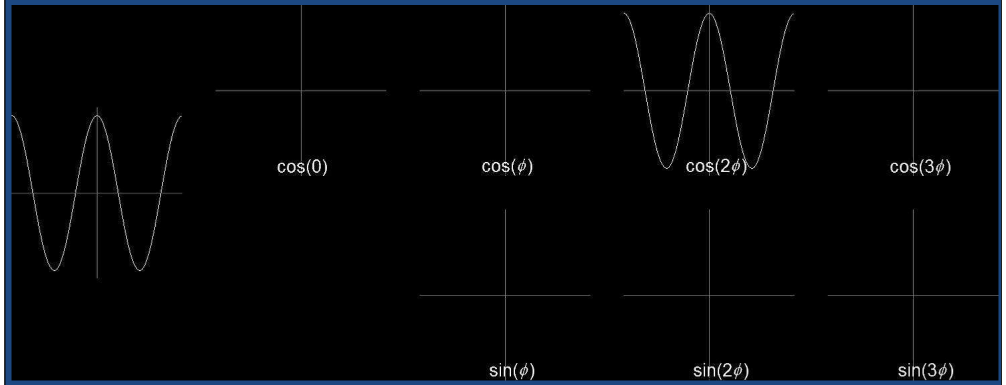


Likewise, a first-order function, when translated remains a first-order function, and can be expressed as a linear sum of the first-order sines and cosines.

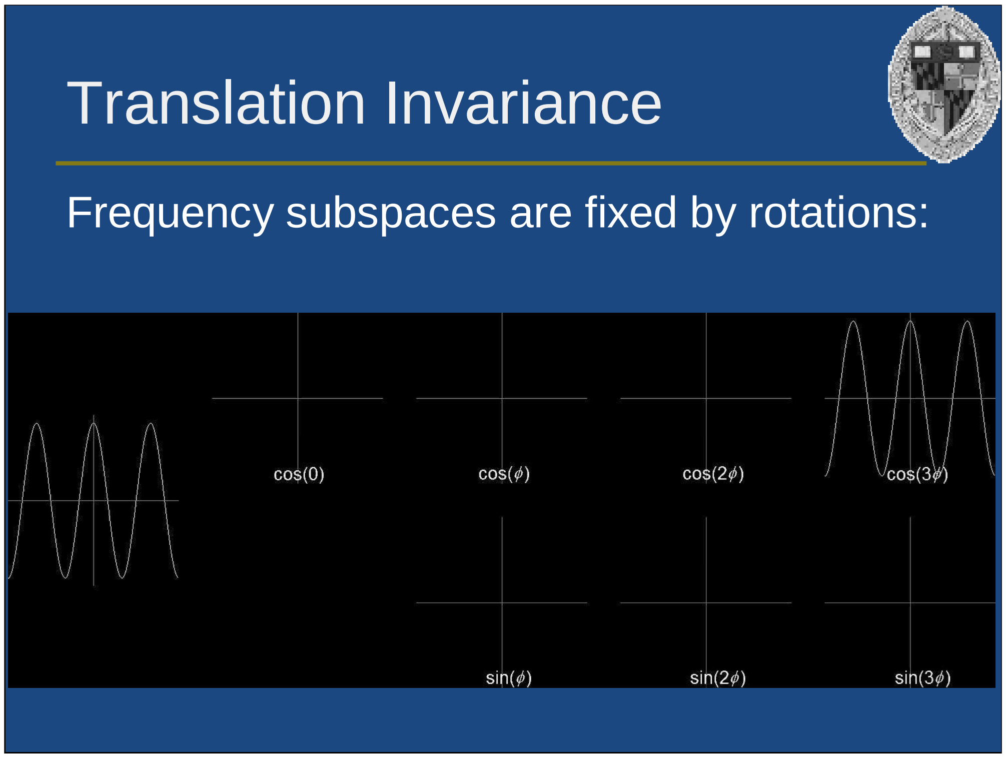


## Translation Invariance

Frequency subspaces are fixed by rotations:

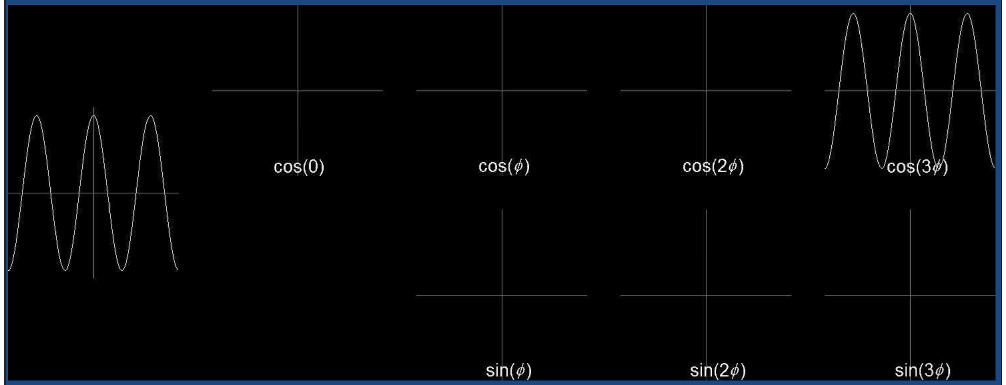


A second-order function, when translated remains a second-order function, and can be expressed as a linear sum of the second-order snes and cosines.



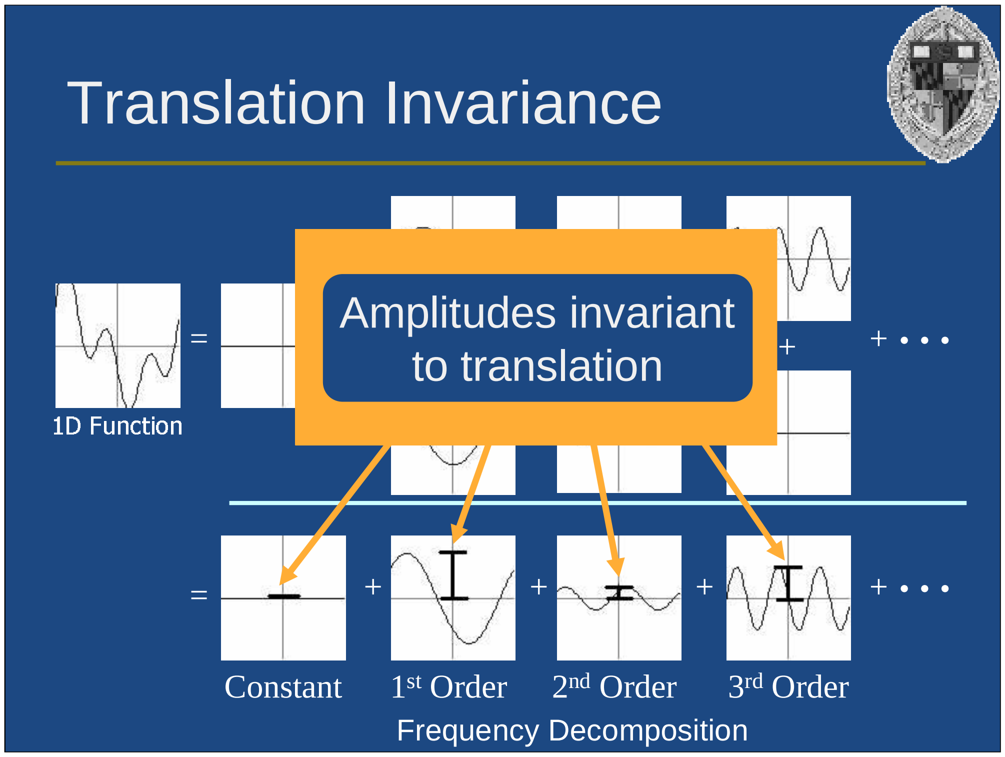
## Translation Invariance

Frequency subspaces are fixed by rotations:



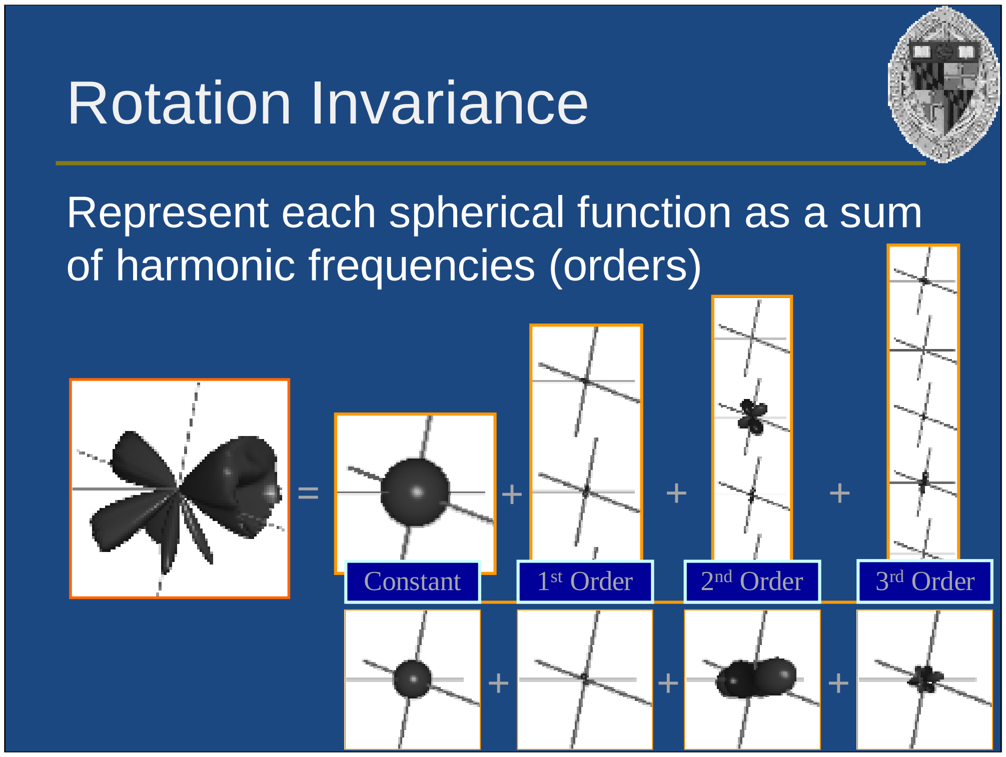
And so on

## Translation Invariance



In particular, this implies that the amplitude (or L2-norm) of each frequency component does not change when the initial function is translated.

Thus, we can obtain a translation invariant representation of any 1D function by storing only the power spectrum.



## Rotation Invariance

Represent each spherical function as a sum of harmonic frequencies (orders)

In a similar fashion, given a spherical function we can obtain a rotation invariant representation using the amplitudes of its frequency components.

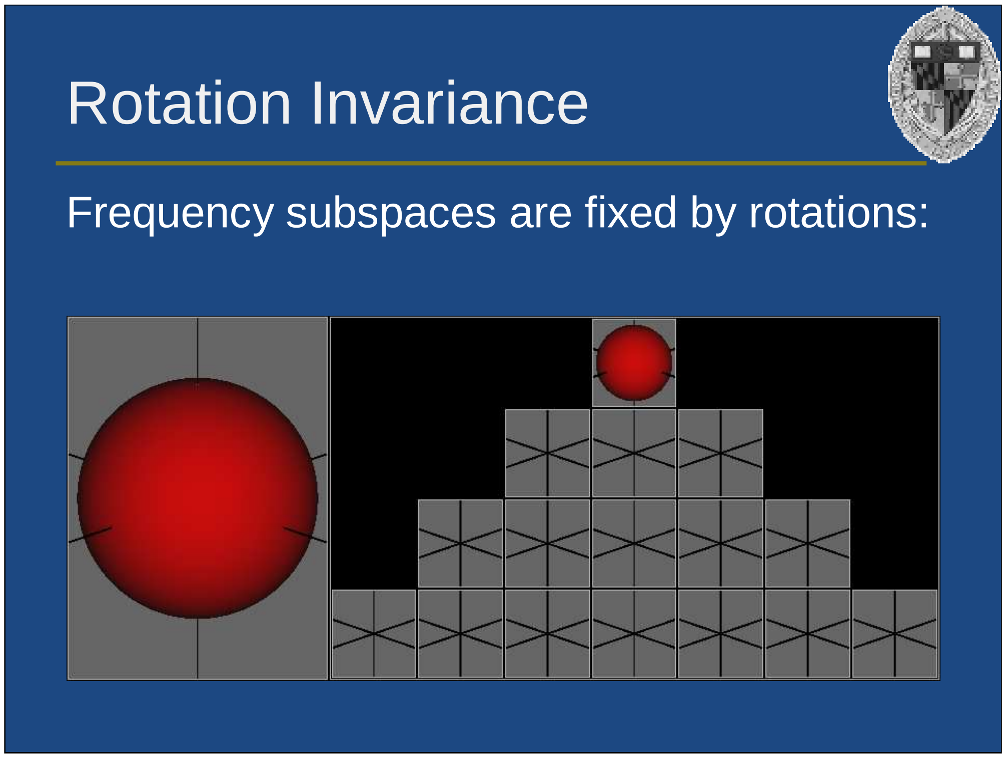
In this image, I am representing a spherical function by scaling points on the surface of the sphere in proportion to their value. So that points on the sphere that have large value are pushed away from the center, and points with small value are pulled in. Points with positive value are drawn in red, and points with negative value are drawn in blue.

To obtain the decomposition of the spherical function into frequency components, we first express the function as a sum of its spherical harmonics.

These are functions on the sphere that play the same role as the cosine and sine function do for a function on a circle. The only real difference is that as apposed to the cosine and sine functions, of which there are only two for each frequency, the spherical harmonics have the property that as the frequency increases, you need more and more of them.

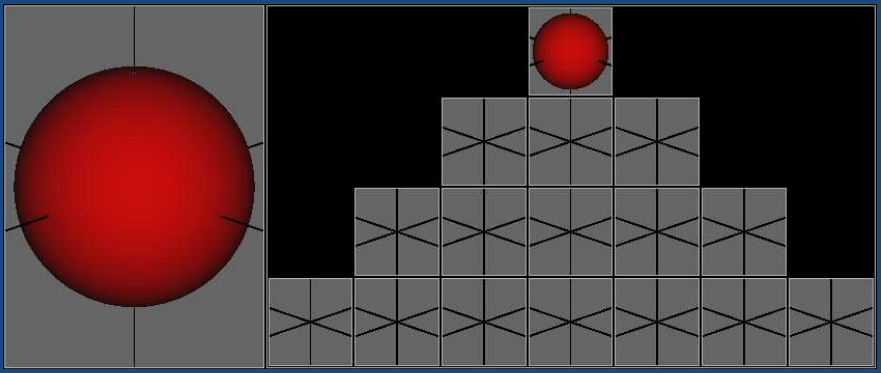
So that, for example, the 1st order frequency functions can be expressed as the sum of 3 harmonics, while the 2nd order frequency functions can be expressed as the sum of 5 harmonics, and so on.

Now just as with the cosine and sine function, we can take the sum of the harmonics within each frequency to obtain an expression of the initial spherical function as the sum of constant, first order, second order, and so on,

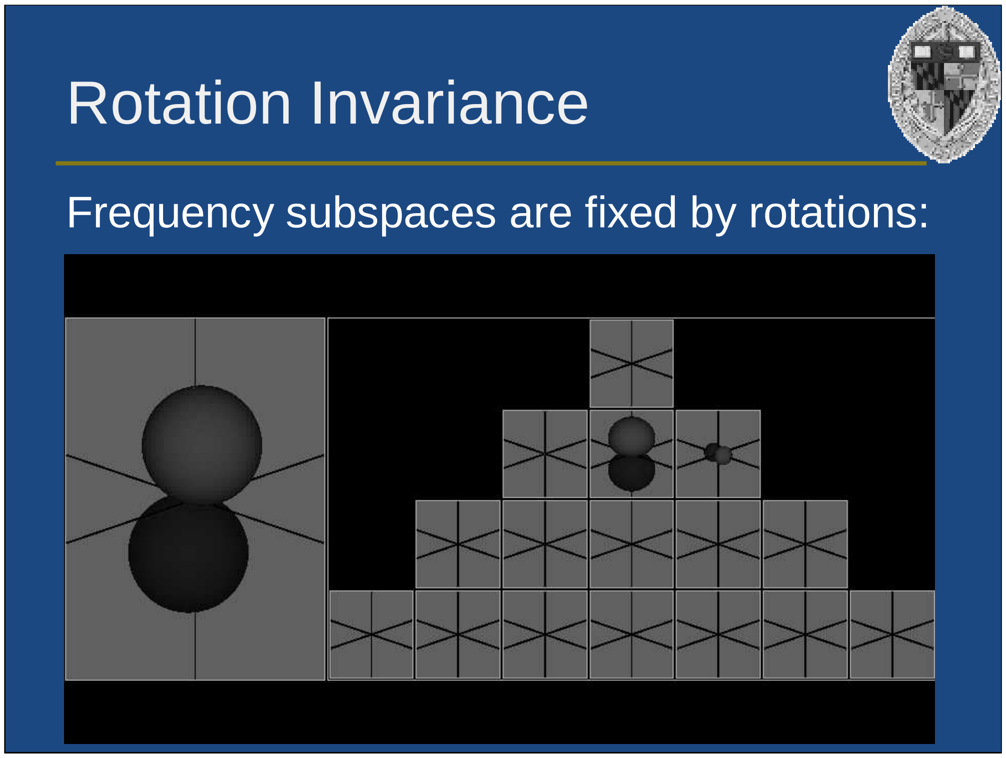


## Rotation Invariance

Frequency subspaces are fixed by rotations:

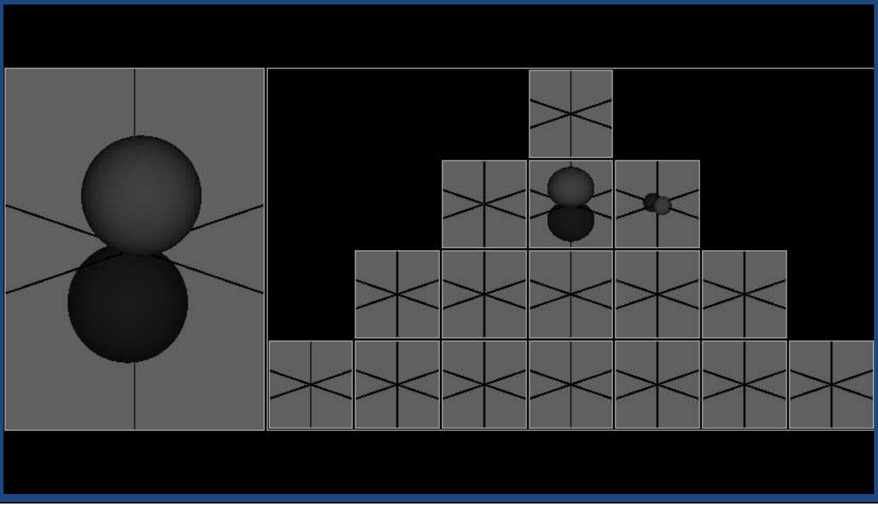


As with the Fourier decomposition, the spherical frequency decomposition has the property that the individual frequency components are fixed by rotation.

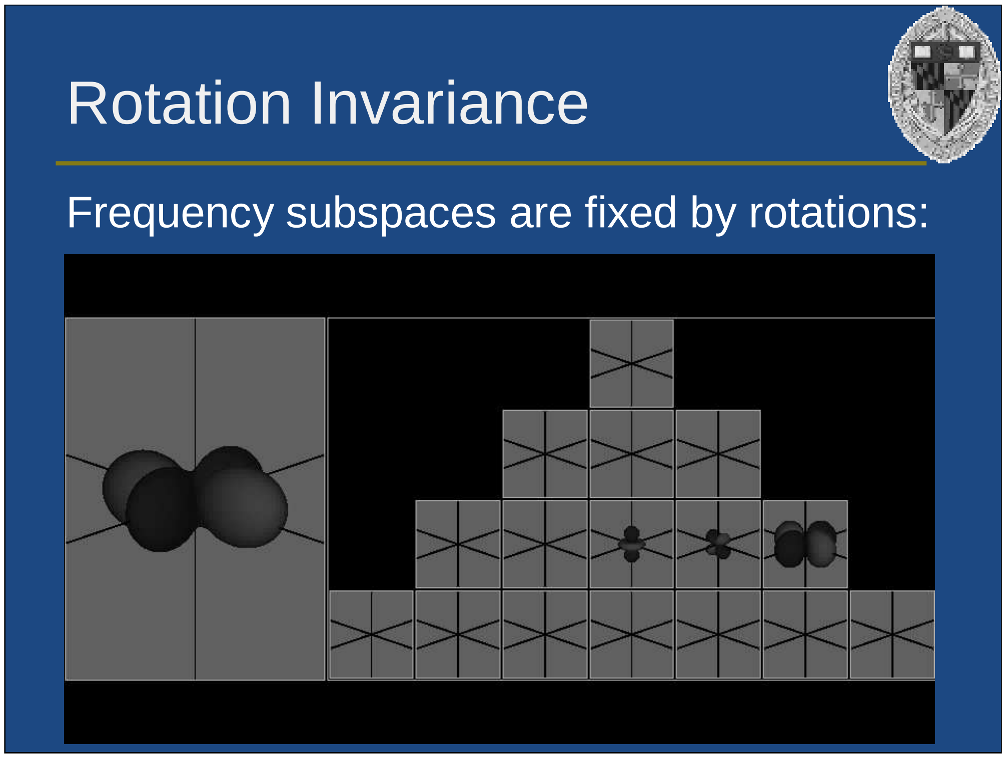


## Rotation Invariance

Frequency subspaces are fixed by rotations:

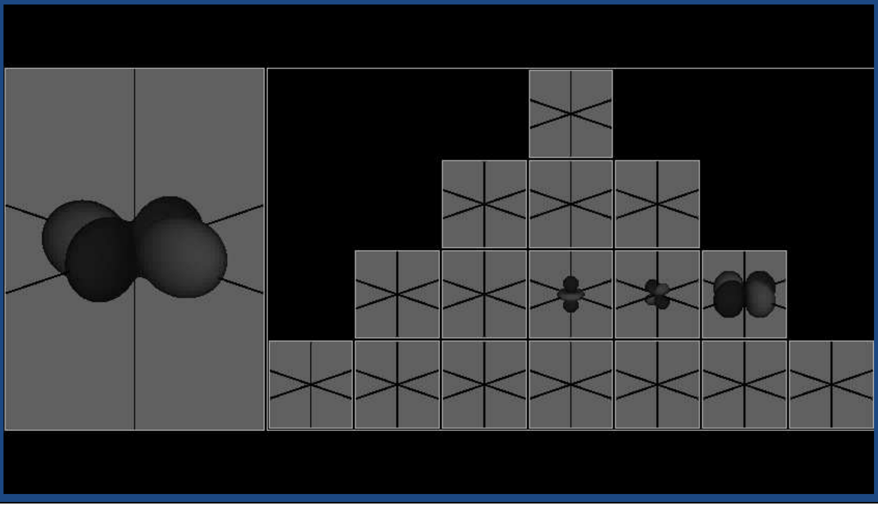


Likewise, a first-order function, when rotated remains a first-order function, and can be expressed as a linear sum of the first-order harmonics.

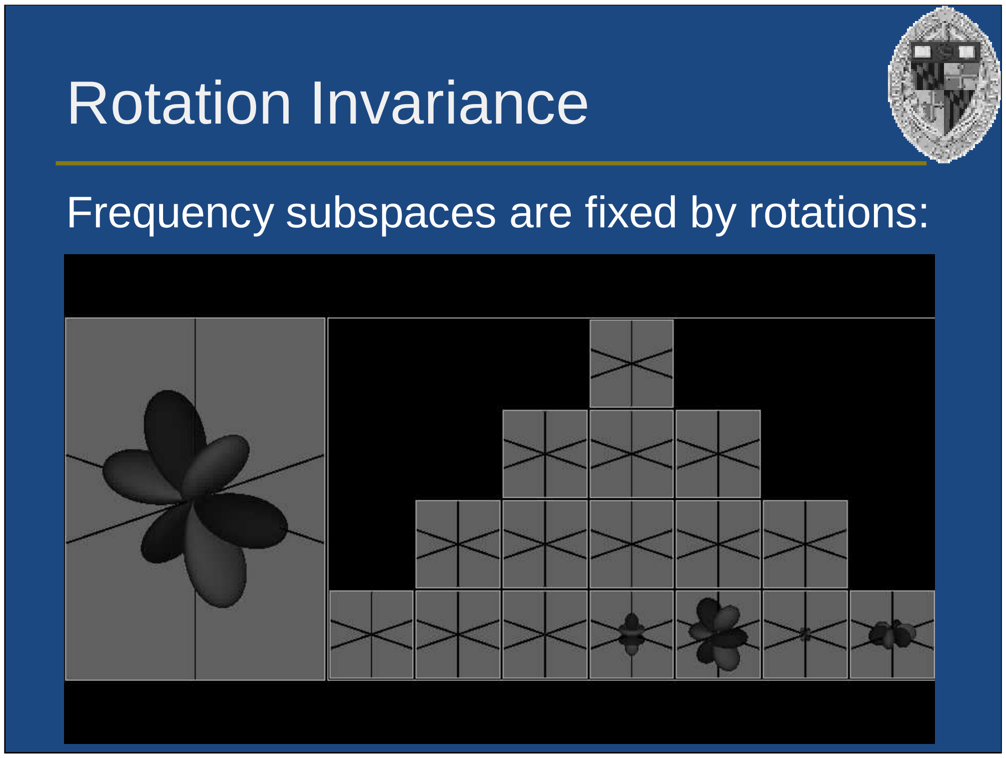


## Rotation Invariance

Frequency subspaces are fixed by rotations:

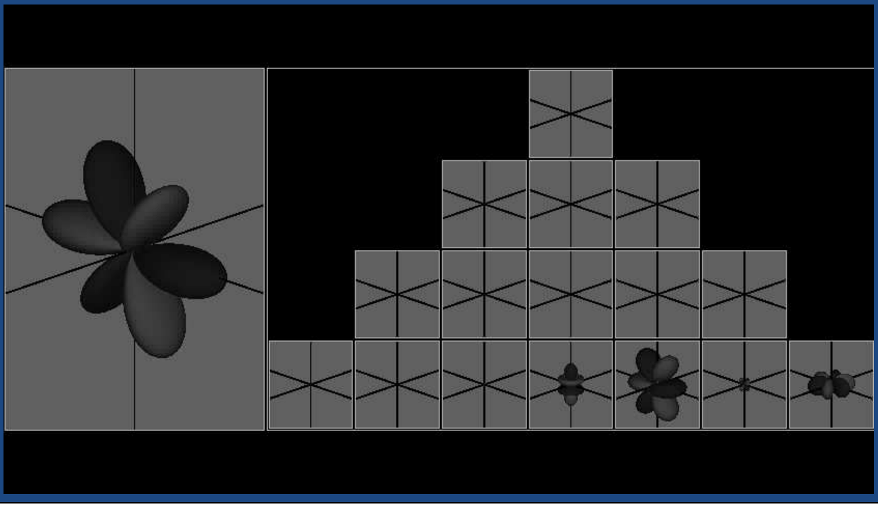


A second-order function, when rotated remains a second-order function, and can be expressed as a linear sum of the second-order harmonics.

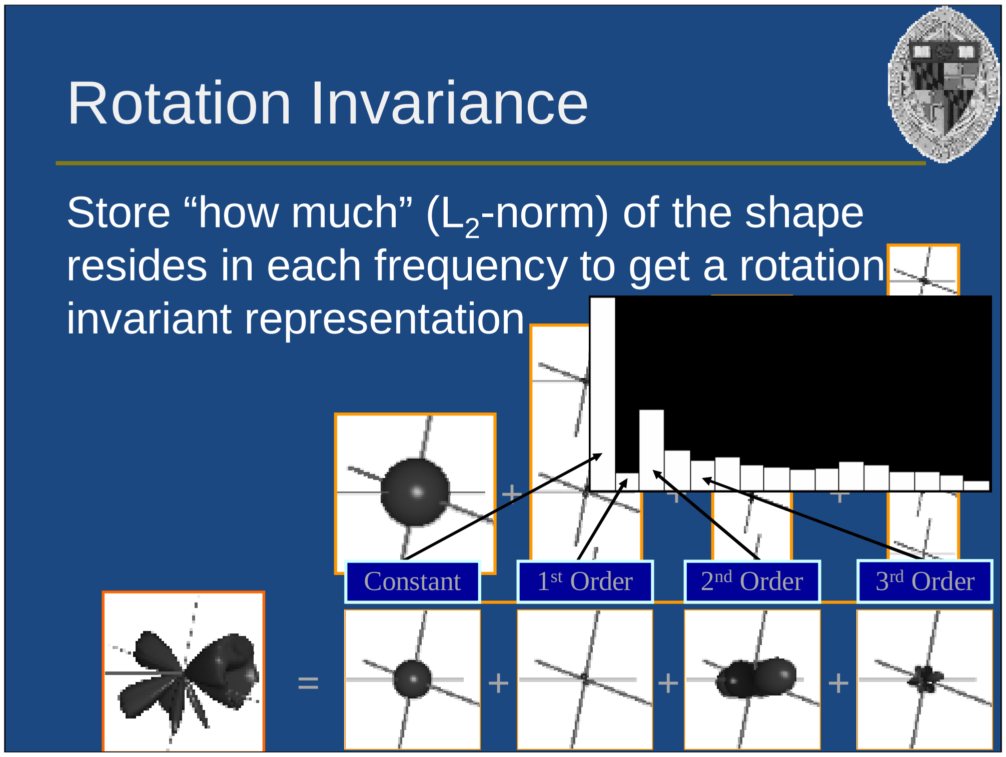


## Rotation Invariance

Frequency subspaces are fixed by rotations:



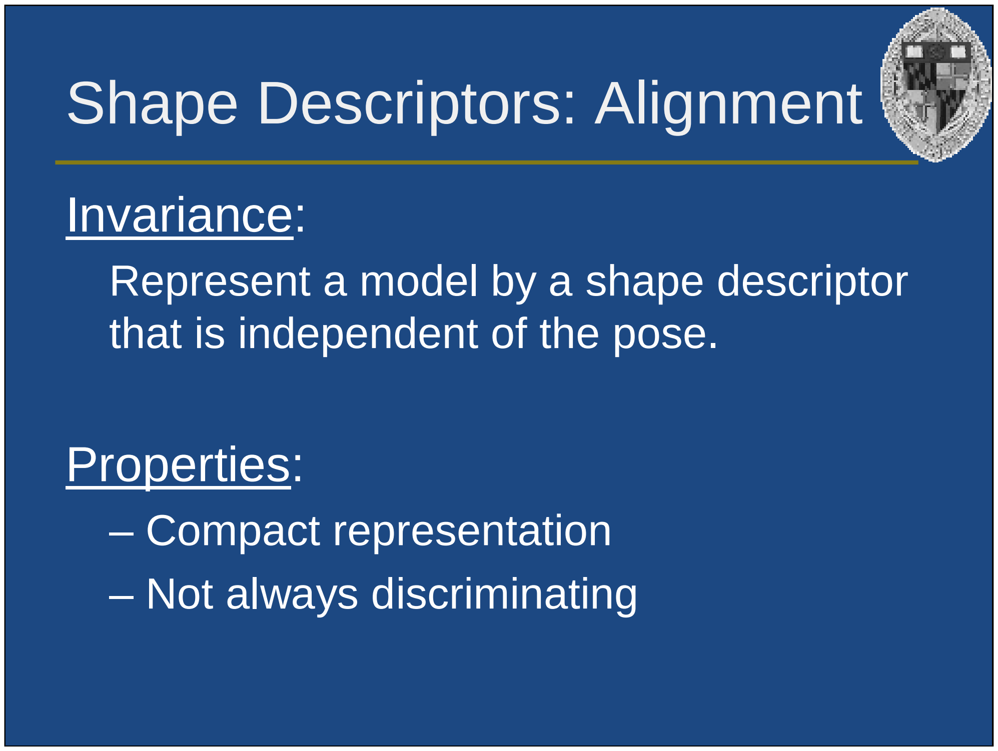
And so on



## Rotation Invariance

Store "how much" ($L_2$-norm) of the shape resides in each frequency to get a rotation invariant representation

Consequently, we can obtain a rotation invariant representation of the spherical function by storing only the energy of each frequency component.



## Shape Descriptors: Alignment

Invariance:

Represent a model by a shape descriptor that is independent of the pose.

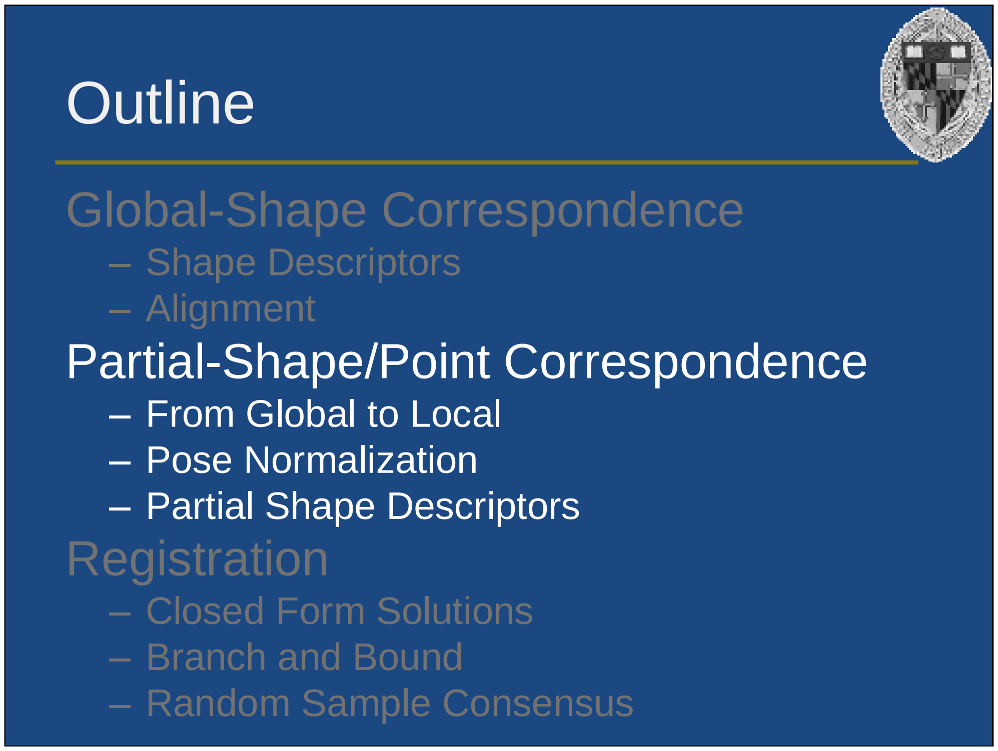
Properties:

– Compact representation

– Not always discriminating

While this method for obtaining a transformation invariant representation is based on a well understood mathematical theory it has a property that is at once an advantage and a disadvantage.

In turning a pose-dependent descriptor into a pose-independent one, all of the pose-dependent information is discarded. As a result, it is often the case that the invariant descriptor is more compact than the original, resulting in smaller storage and faster comparison times. However, it is also often the case that in discarding the pose-dependent information, a certain amount of pose-independent information is also lost and the resulting descriptor can be less discriminating than the original.



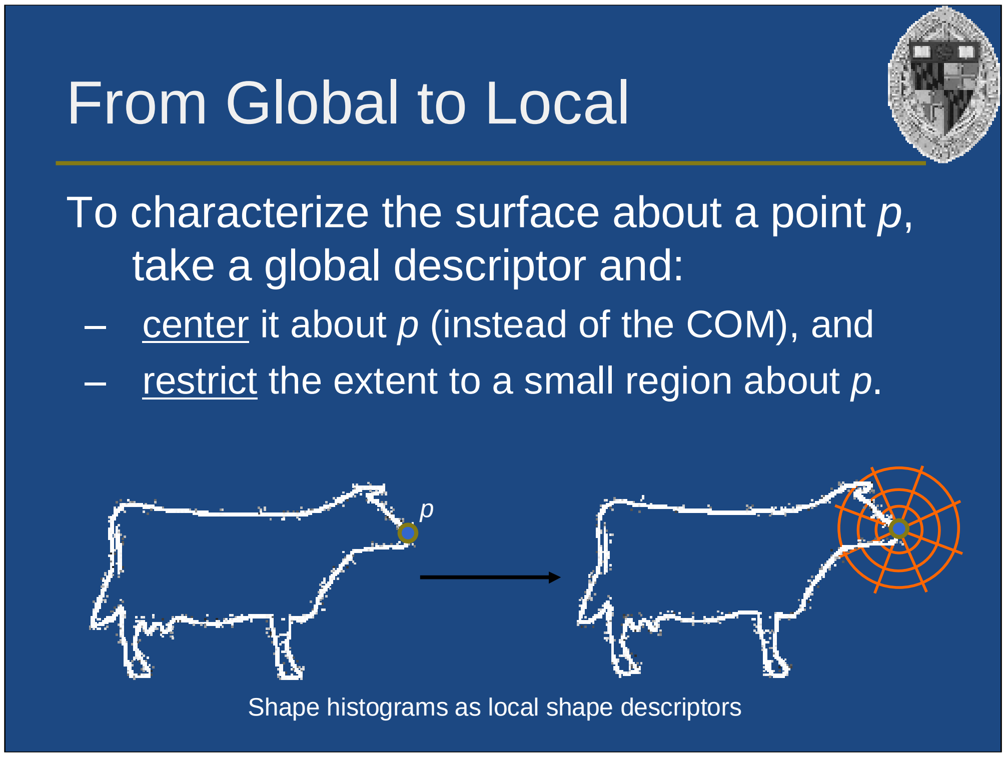
## Outline

Global-Shape Correspondence
- Shape Descriptors
- Alignment

Partial-Shape/Point Correspondence
- From Global to Local
- Pose Normalization
- Partial Shape Descriptors

Registration
- Closed Form Solutions
- Branch and Bound
- Random Sample Consensus

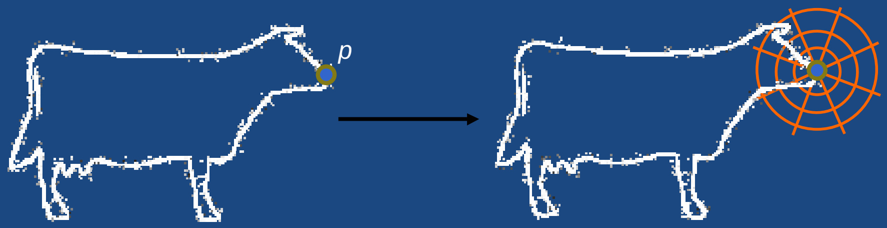
Having described how shape descriptors are used for addressing the whole-object matching problem, we will now discuss applications of these methods to the task of establishing point-correspondences between two overlapping surfaces.



## From Global to Local

To characterize the surface about a point *p*, take a global descriptor and:

- center it about *p* (instead of the COM), and
- restrict the extent to a small region about *p*.

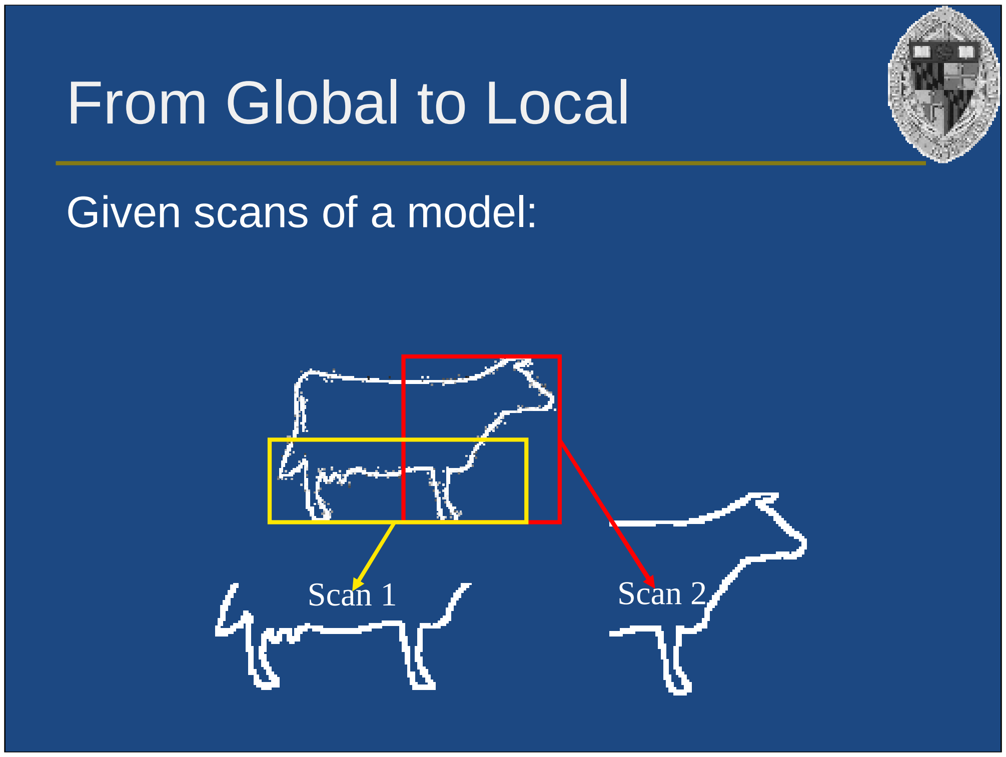


Shape histograms as local shape descriptors

In general, a shape descriptor designed to capture the global properties of a shape can often be directly transformed into a shape descriptor target at characterizing the surface in a region about some point.

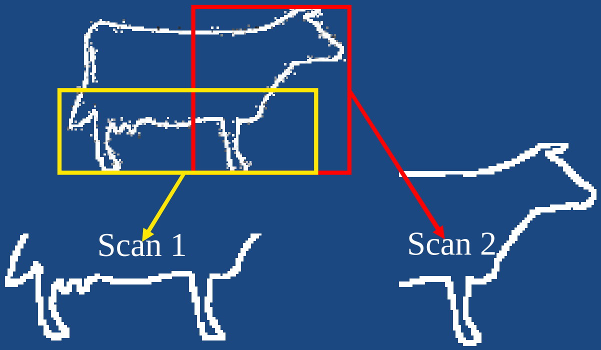
This can be done by changing the point at which the descriptor is centered and changing the radius of its extent so that it only characterizes model properties within a fixed distance about the point of interest.

As an example, the image at the bottom shows the use of Ankerst's Shape Histogram to characterize the region of the model about the point *p*. The shape descriptor is shifted so that it is centered about *p* and its radius of extent is changed to be roughly the size of the head. Thus, the shape descriptor will capture information about the snout of the cow and will not incorporate any information about the legs or back.



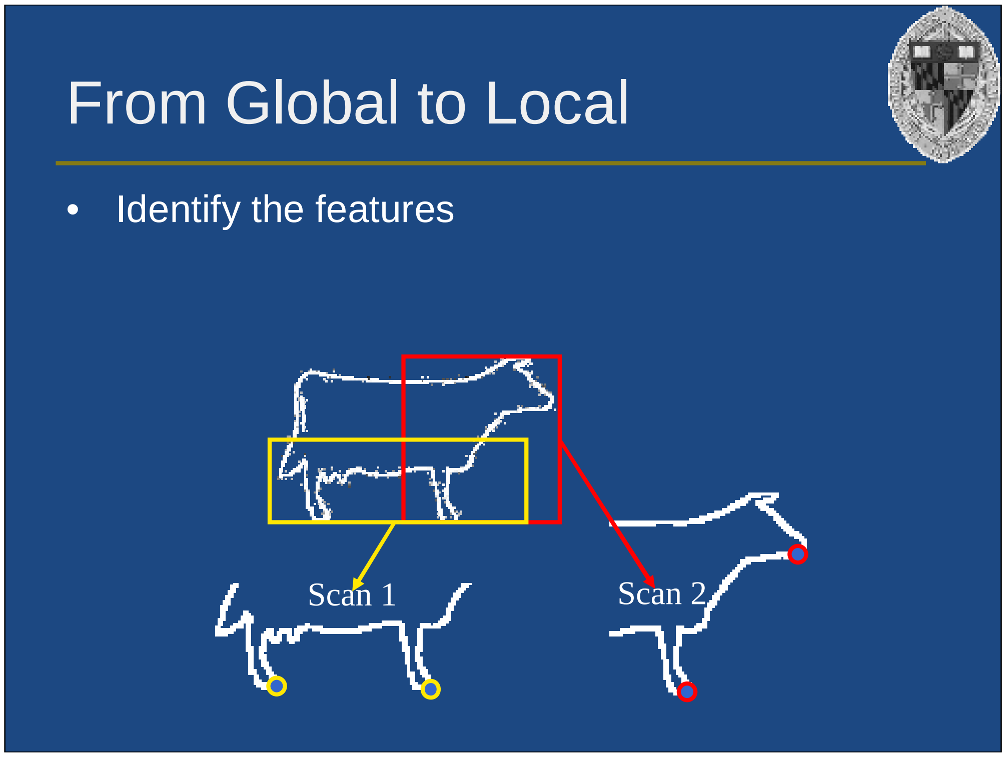
## From Global to Local

Given scans of a model:



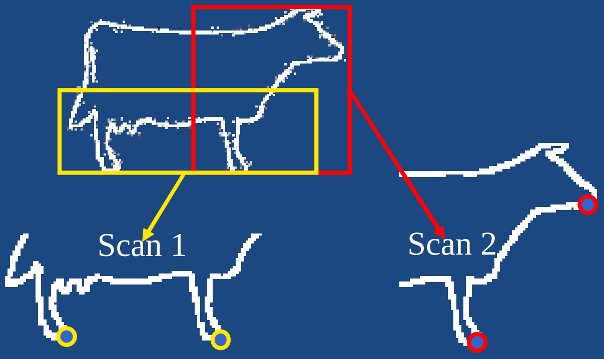
Now, we can use these local shape descriptors to establish corresponding point between scans.

Consider the example of this cow model and the two partial scans show in the bottom.



## From Global to Local

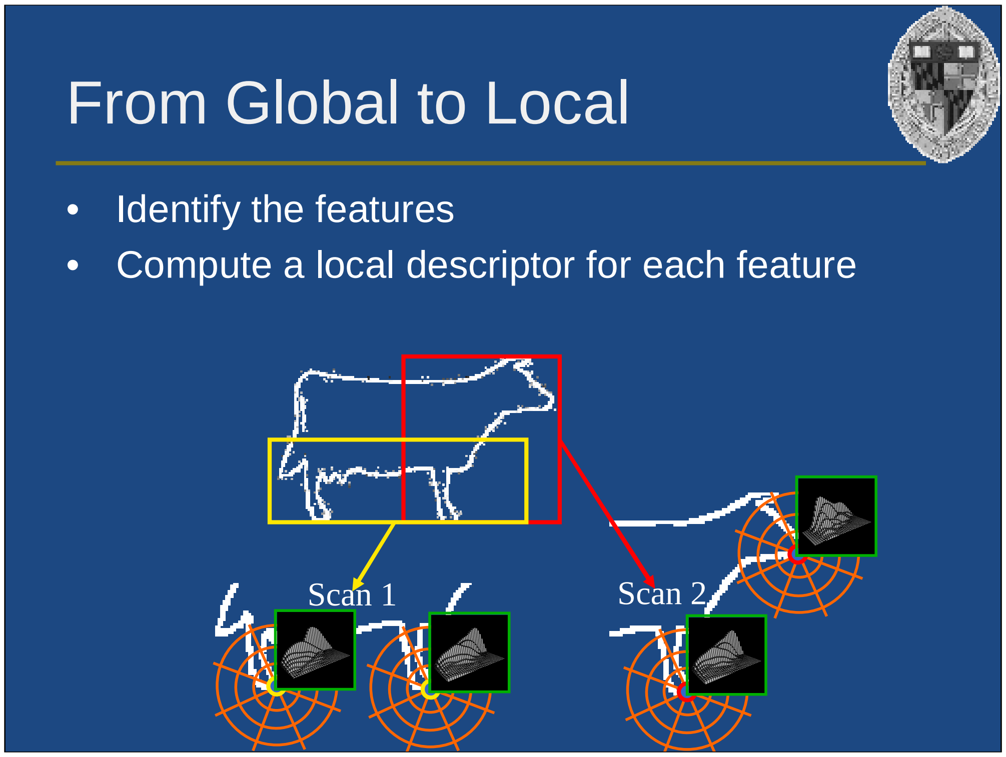
- Identify the features



Now, we can use these local shape descriptors to establish corresponding point between scans.

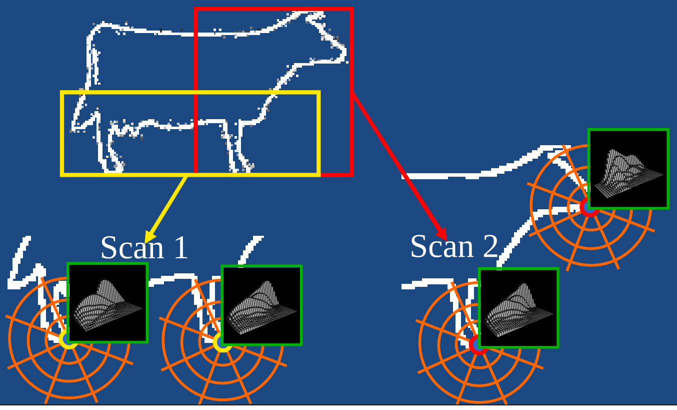
Consider the example of this cow model and the two partial scans show in the bottom.

We independently identify the feature points on the two models.



## From Global to Local

- Identify the features
- Compute a local descriptor for each feature

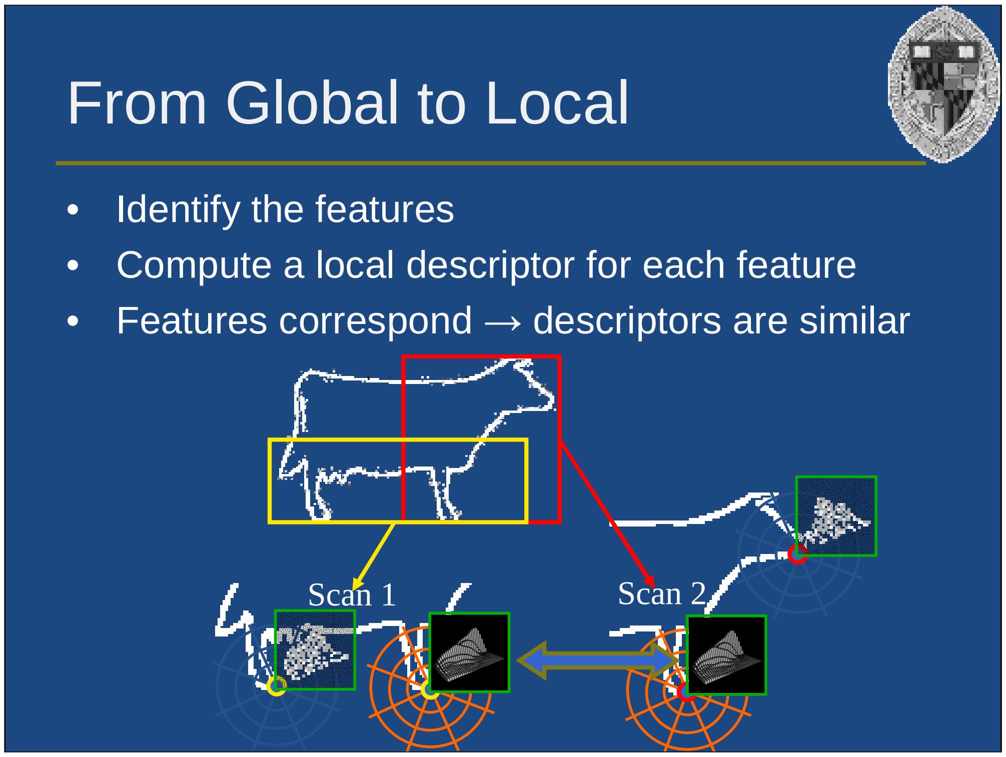


Now, we can use these local shape descriptors to establish corresponding point between scans.

Consider the example of this cow model and the two partial scans show in the bottom.

We independently identify the feature points on the two models.

Then, we characterize the geometry of scans about each feature point with a local shape descriptor



## From Global to Local

- Identify the features
- Compute a local descriptor for each feature
- Features correspond → descriptors are similar

Scan 1
Scan 2

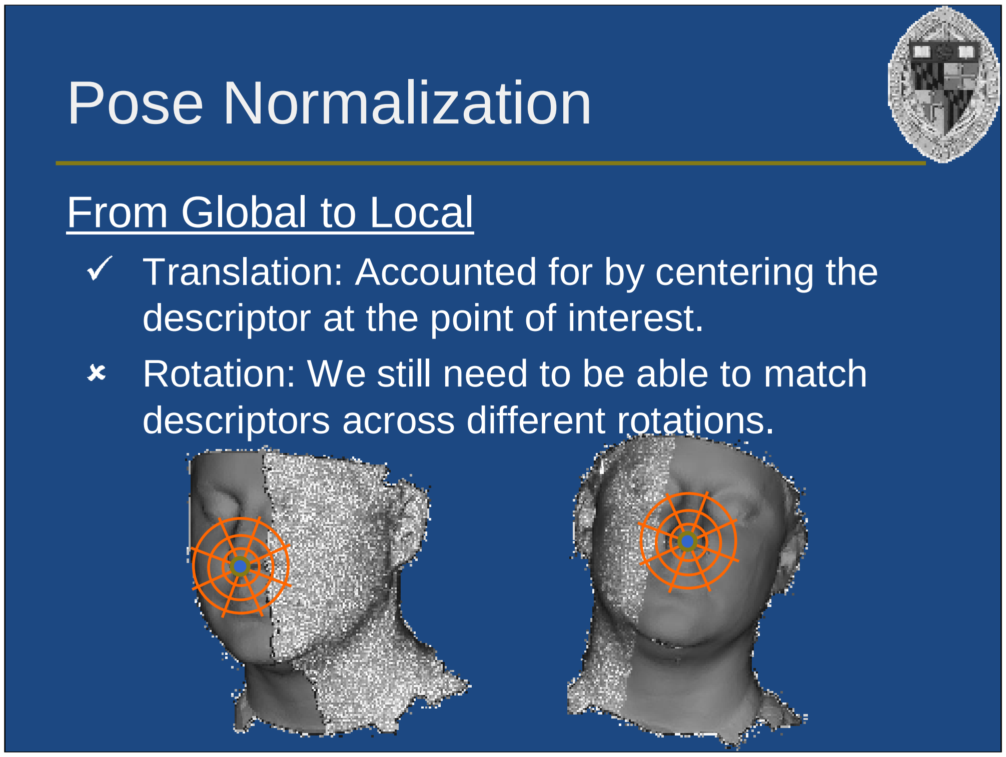
Now, we can use these local shape descriptors to establish corresponding point between scans.

Consider the example of this cow model and the two partial scans show in the bottom.

We independently identify the feature points on the two models.

Then, we characterize the geometry of scans about each feature point with a local shape descriptor

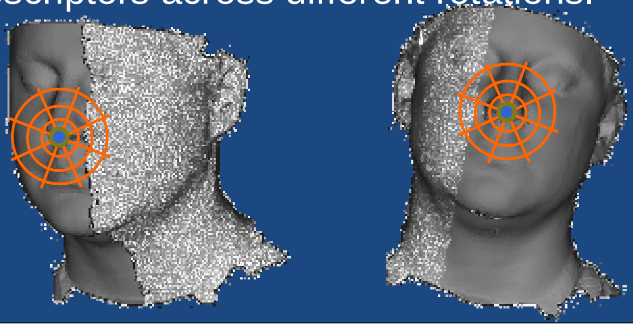
And finally, we identify corresponding features by looking for similar local shape descriptors across the two scans.



## Pose Normalization

From Global to Local

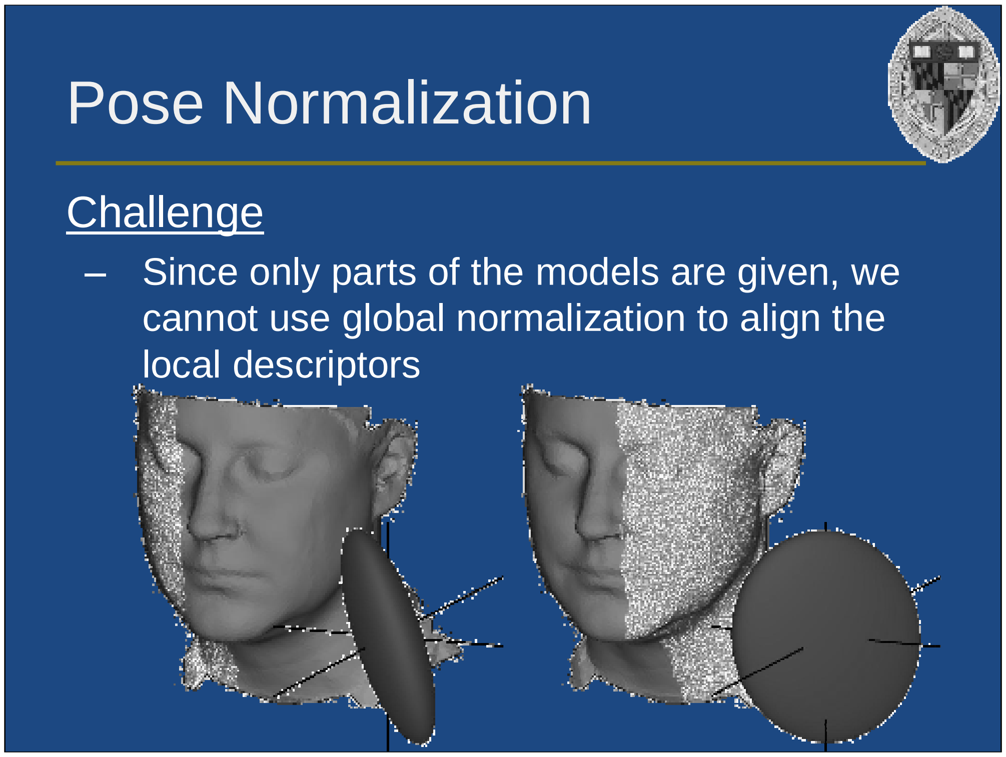
- ✓ Translation: Accounted for by centering the descriptor at the point of interest.
- ✗ Rotation: We still need to be able to match descriptors across different rotations.



While the shape descriptor itself carries over directly from the global to the local context, the alignment normalization methods are not so straightforward.

In particular, methods for aligning models for whole-object matching have relied on normalization techniques that use global shape information to compute a robust coordinate frame.

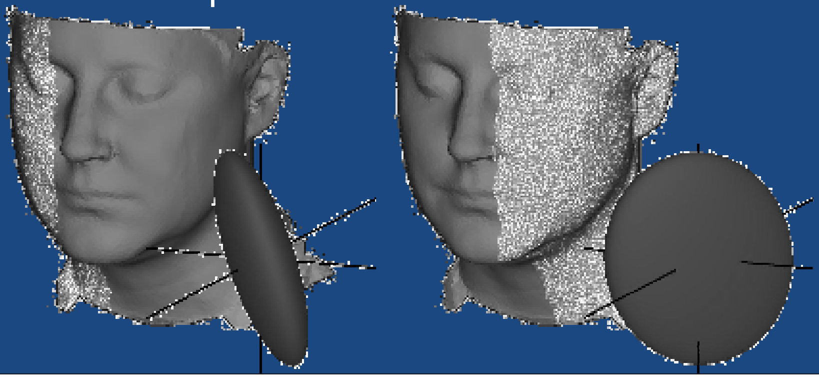
While this is not an issue for translation, since we explicitly translate the descriptor so that it is centered about the feature point, this does become a problem for rotation normalization.



## Pose Normalization

Challenge

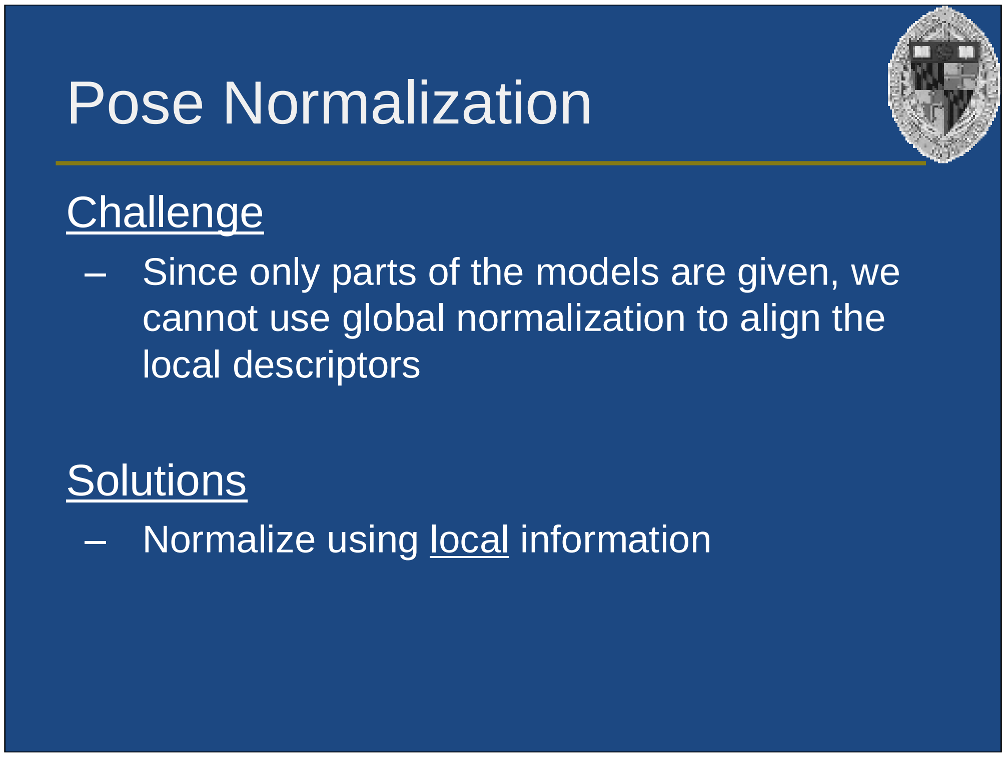
– Since only parts of the models are given, we cannot use global normalization to align the local descriptors



In particular, when matching local features, we can no longer use global methods such as PCA alignment since the covariance matrix of part of a model will generally vary depending on which part of the model is represented.

As an example, if we consider the left and right scans of the face shown on the bottom, (which are rendered in a common coordinate frame) we see that their associated covariance ellipsoids are quite different.

Thus, normalizing using these ellipsoids would result in misaligned scans.



## Pose Normalization

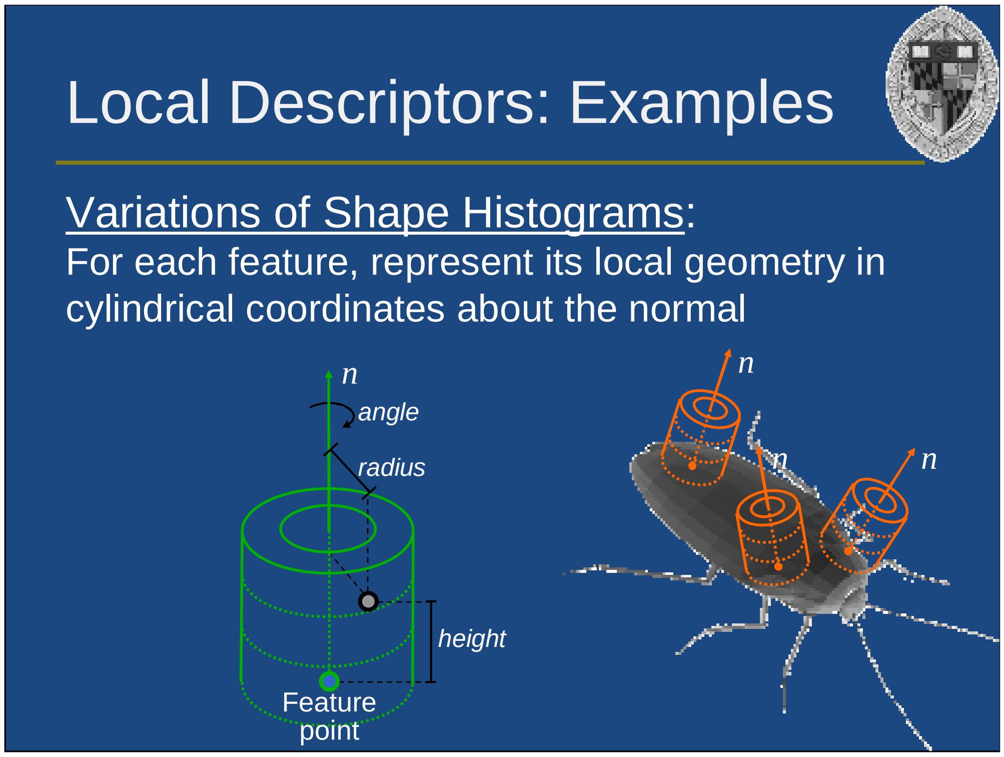
<u>Challenge</u>

– Since only parts of the models are given, we cannot use global normalization to align the local descriptors

<u>Solutions</u>

– Normalize using <u>local</u> information

To address this problem, methods for local shape descriptor normalization have relied on the use of local shape information to define canonical frames.



## Local Descriptors: Examples

Variations of Shape Histograms:
For each feature, represent its local geometry in cylindrical coordinates about the normal

n
angle
radius
height
Feature point
n
n
n

As an example, we will consider three different local shape descriptors.

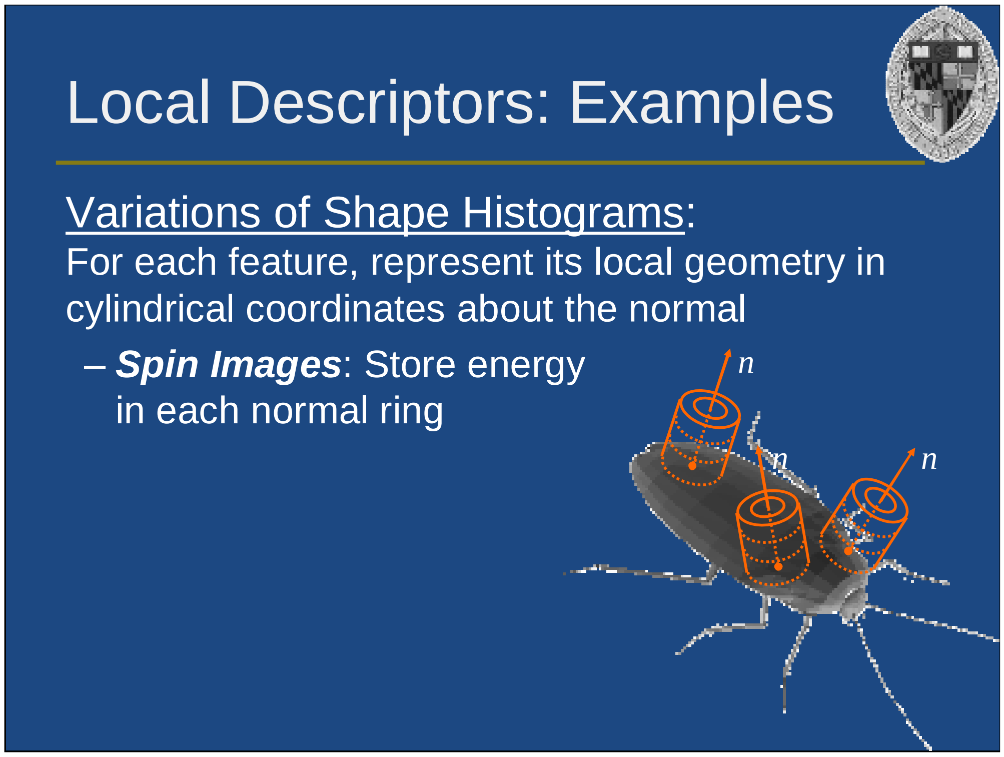
All three descriptors characterize the surface about a feature point by considering the local distribution of surface geometry about the point, and all three use the surface normal to normalize for two of the three degrees of rotational freedom.

In particular, all three methods define a cylindrical coordinate frame indexed by:

1. Height along the normal,
2. Radius from the normal, and
3. Angle about the normal

Since the surface normal is consistent across corresponding feature points, the height and radius are in normalized coordinates.

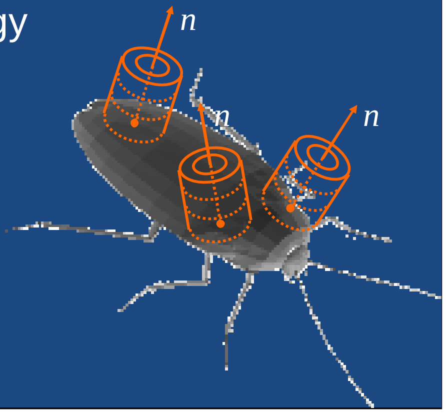
However, there is no normalization for the angle about the normal, and the three methods differ in terms of how they address the angular alignment problem.



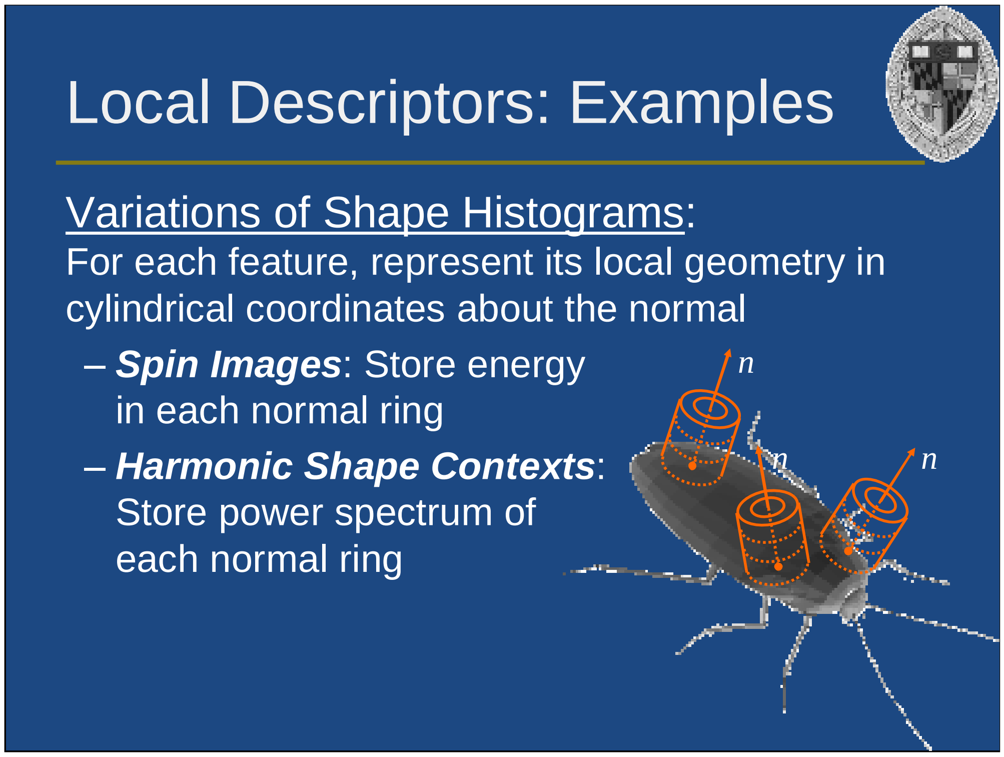
## Local Descriptors: Examples

Variations of Shape Histograms:
For each feature, represent its local geometry in cylindrical coordinates about the normal

– ***Spin Images***: Store energy in each normal ring



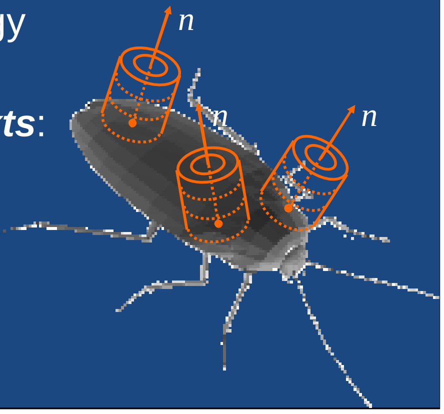
Introduced by Johnson in 1997, spin images address the angular alignment problem by storing the average of the surface area obtained by intersecting the local geometry with rings of fixed height and radius.



## Local Descriptors: Examples

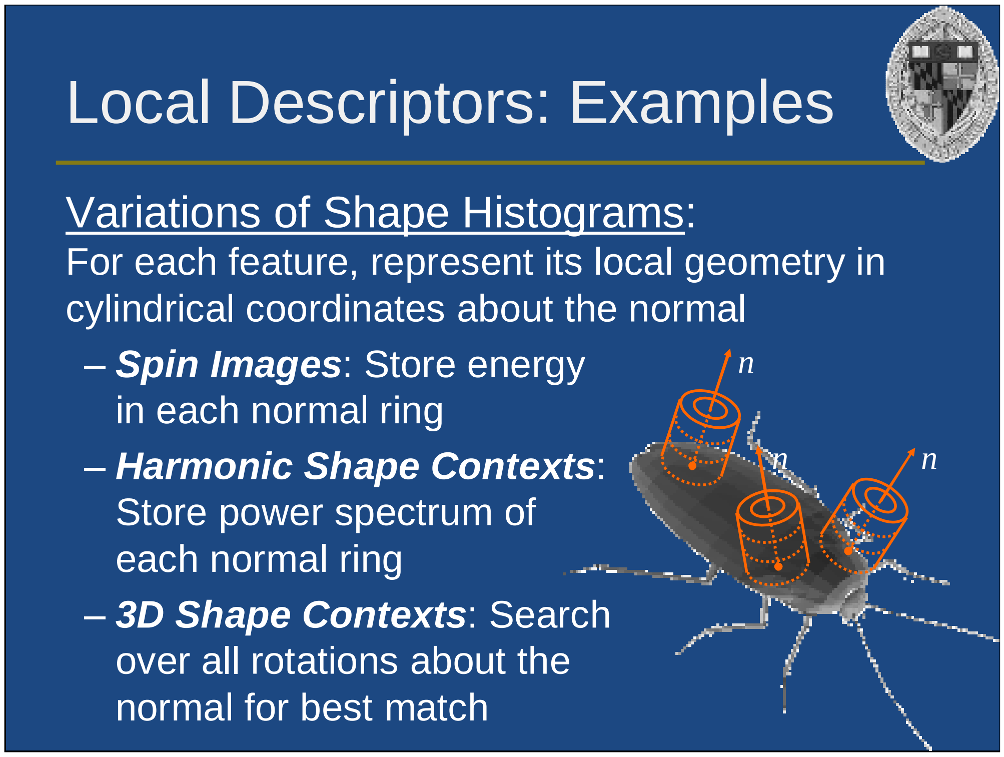
Variations of Shape Histograms:
For each feature, represent its local geometry in cylindrical coordinates about the normal

- ***Spin Images***: Store energy in each normal ring
- ***Harmonic Shape Contexts***: Store power spectrum of each normal ring



Recognizing that the average over normal rotations corresponds the constant order term of a more general power spectrum, the Harmonic Shape Contexts provide a more complete angular rotation invariant shape descriptor by storing the amplitude of all of the frequency components.

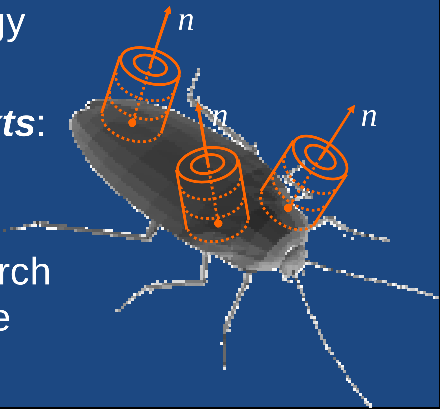
The resultant descriptor is still rotation invariant but now characterizes the region about a feature point by a 3D descriptor.



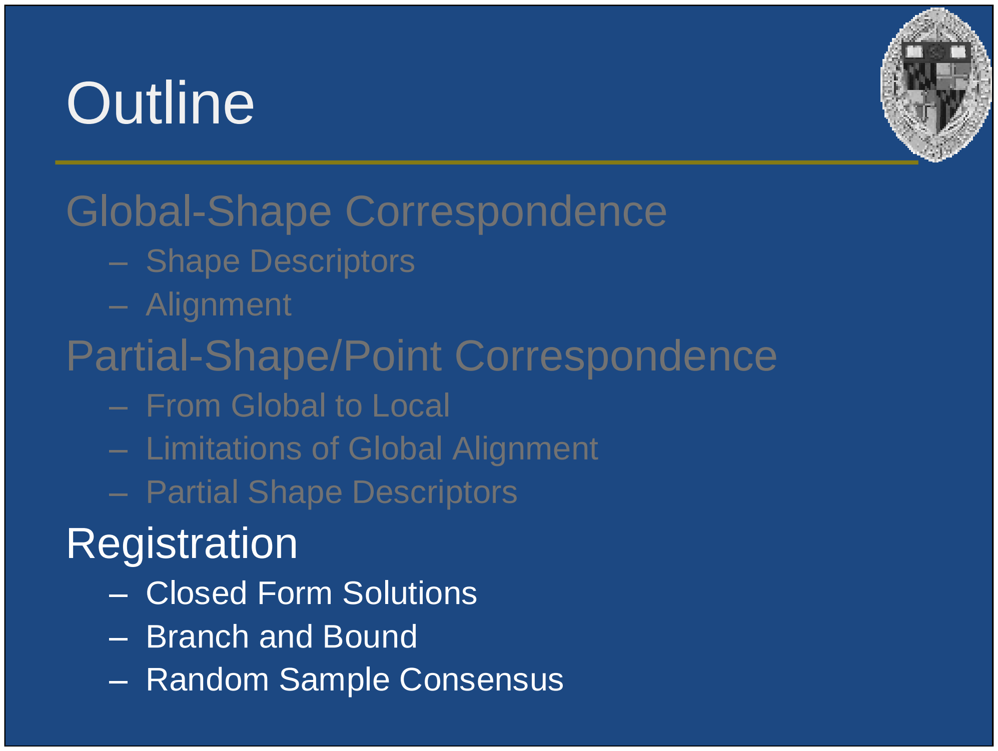
## Local Descriptors: Examples

Variations of Shape Histograms:
For each feature, represent its local geometry in cylindrical coordinates about the normal

- ***Spin Images***: Store energy in each normal ring
- ***Harmonic Shape Contexts***: Store power spectrum of each normal ring
- ***3D Shape Contexts***: Search over all rotations about the normal for best match



Finally, in the 3D analog of Belongie et al.'s Shape Contexts, an exhaustive search over normal angles of rotation is used to find the rotation that best aligns two local shape descriptors.



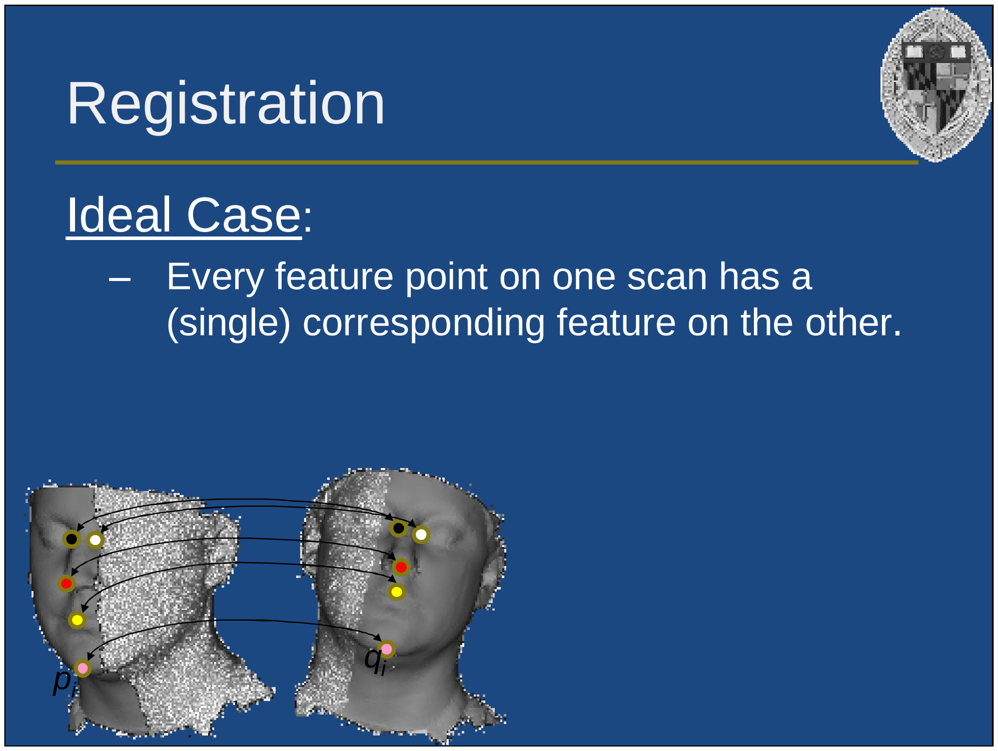
## Outline

Global-Shape Correspondence
- Shape Descriptors
- Alignment

Partial-Shape/Point Correspondence
- From Global to Local
- Limitations of Global Alignment
- Partial Shape Descriptors

Registration
- Closed Form Solutions
- Branch and Bound
- Random Sample Consensus

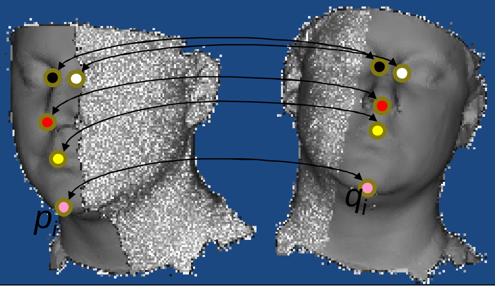
Having described methods for establishing potential correspondences between feature points on two different scans, we will now describe how these correspondences can be used to register the scans.



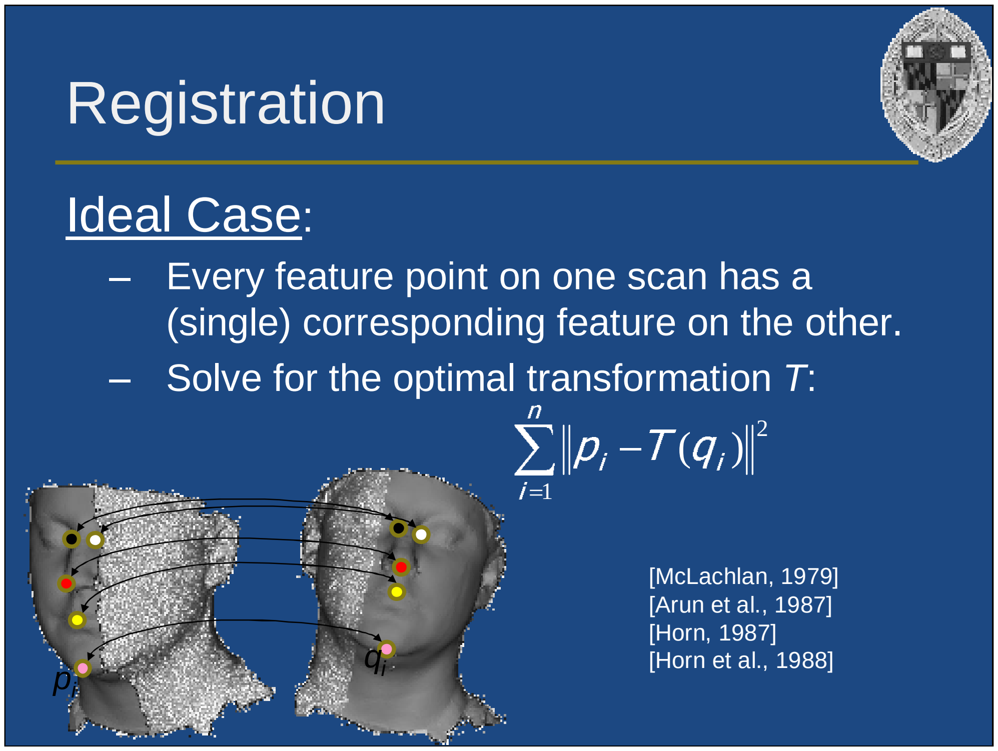
## Registration

Ideal Case:

– Every feature point on one scan has a (single) corresponding feature on the other.



In the ideal case, every feature point on one scan would have one, and only one, corresponding feature point on the second scan.

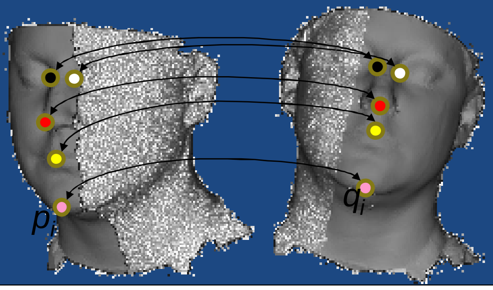


## Registration

Ideal Case:

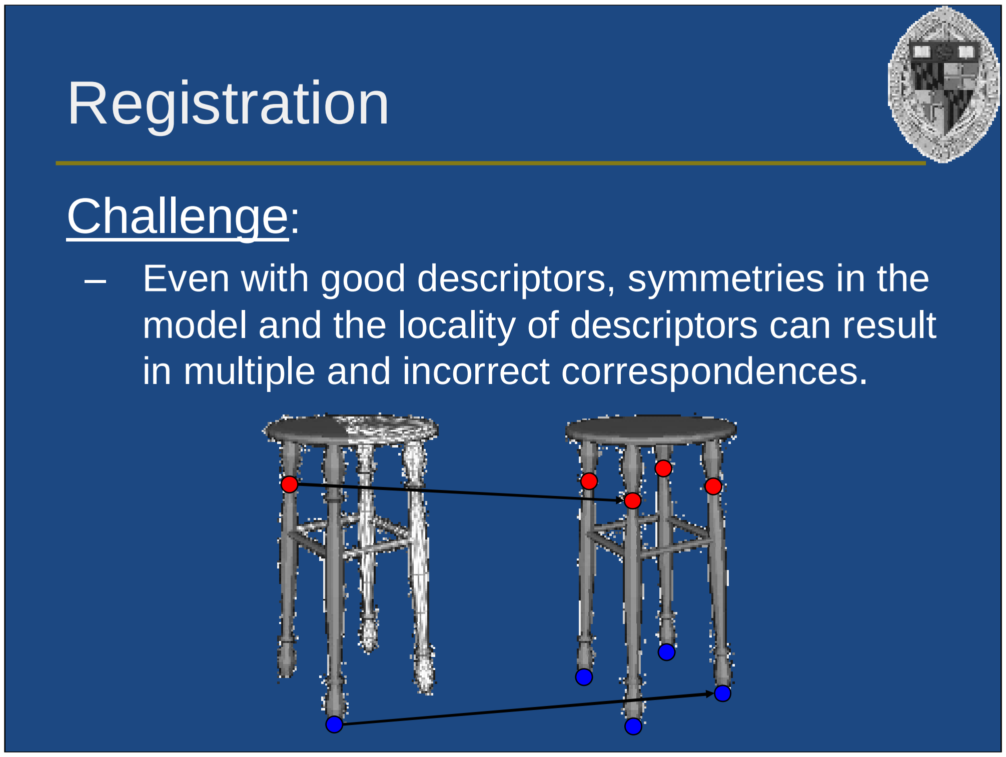
- Every feature point on one scan has a (single) corresponding feature on the other.
- Solve for the optimal transformation $T$:

$$\sum_{i=1}^{n} \left\| p_i - T(q_i) \right\|^2$$



[McLachlan, 1979]
[Arun et al., 1987]
[Horn, 1987]
[Horn et al., 1988]

In this case, the scans could be aligned by solving for the transformation that minimizes the sum of squared distances between corresponding points, which is a problem that has been well studied and well understood.



## Registration

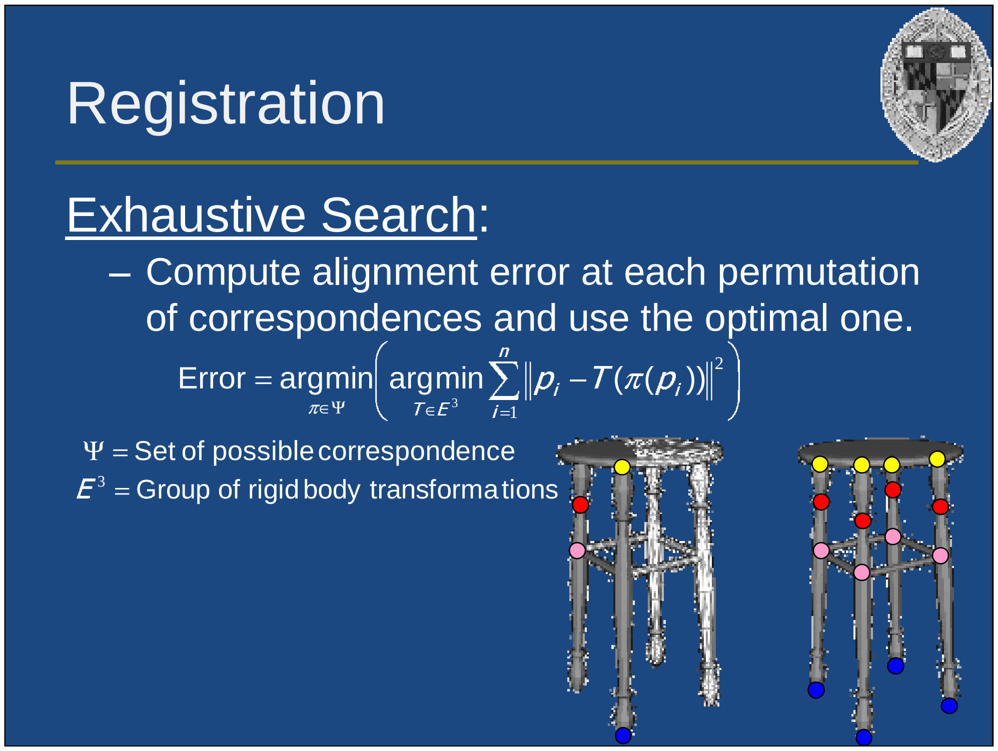
Challenge:

– Even with good descriptors, symmetries in the model and the locality of descriptors can result in multiple and incorrect correspondences.

The problem, however, is that it is often impossible to establish one-to-one correspondences between two scans.

As the figure indicates, even in the case that the local shape descriptors are perfectly discriminating, symmetries in the models can give rise to multiple correspondences. For example, the feature point in blue on the bottom of the partial scan on the left has four different corresponding features in the whole model on the right. Similarly, the feature point in red near the top of the partial scan has four different corresponding features in the whole model on the right.

The problem in this case is that we can no longer treat the correspondences independently. Thus, for example, if we arbitrarily chose to map the blue feature point on the partial scan to one of its corresponding features on the whole model, this would dictate quite strictly which of the four points the red feature point could map to.



## Registration

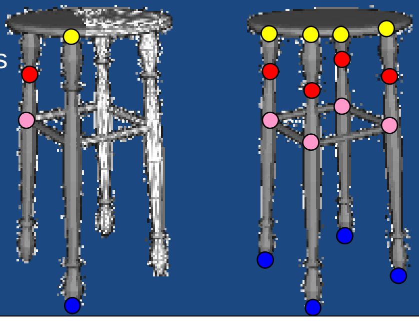
Exhaustive Search:

– Compute alignment error at each permutation of correspondences and use the optimal one.

$$\text{Error} = \underset{\pi \in \Psi}{\operatorname{argmin}} \left( \underset{T \in E^3}{\operatorname{argmin}} \sum_{i=1}^{n} \left\| p_i - T(\pi(p_i)) \right\|^2 \right)$$

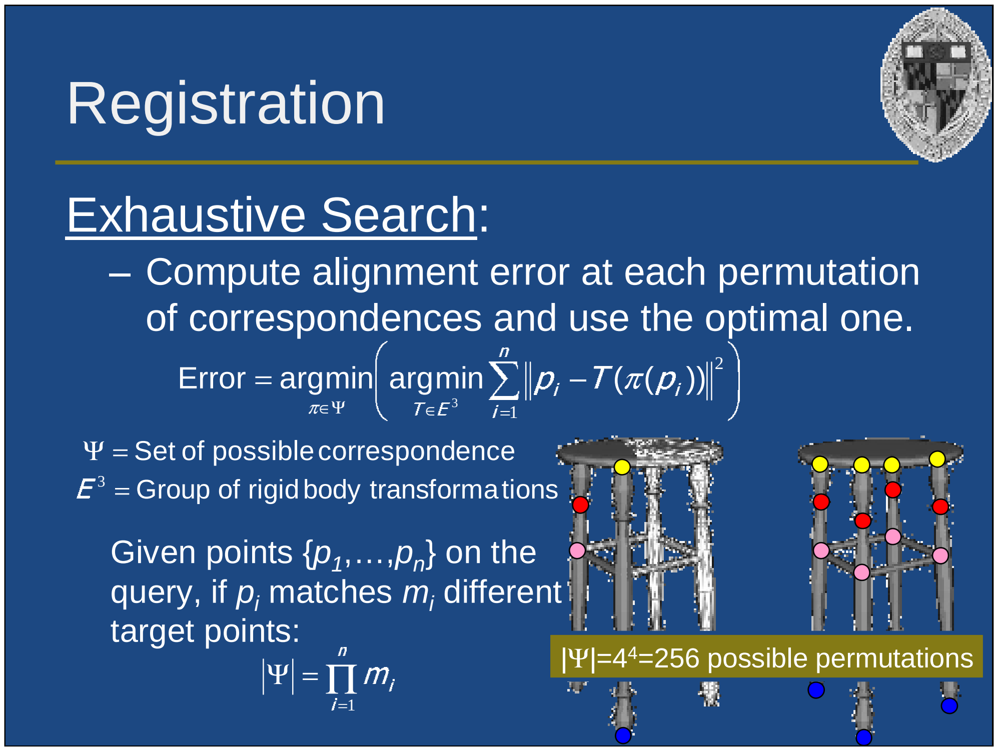
$\Psi$ = Set of possible correspondence

$E^3$ = Group of rigid body transformations



Of course, it is theoretically possible to exhaustively search over the space of all possible correspondences for the set of correspondences that define the optimal alignment.

However, in practice this type of approach becomes intractable.



## Registration

**Exhaustive Search:**

- Compute alignment error at each permutation of correspondences and use the optimal one.

$$\text{Error} = \underset{\pi\in\Psi}{\operatorname{argmin}}\left(\underset{T\in E^3}{\operatorname{argmin}}\sum_{i=1}^{n}\left\|p_i - T(\pi(p_i))\right\|^2\right)$$

$\Psi$ = Set of possible correspondence

$E^3$ = Group of rigid body transformations

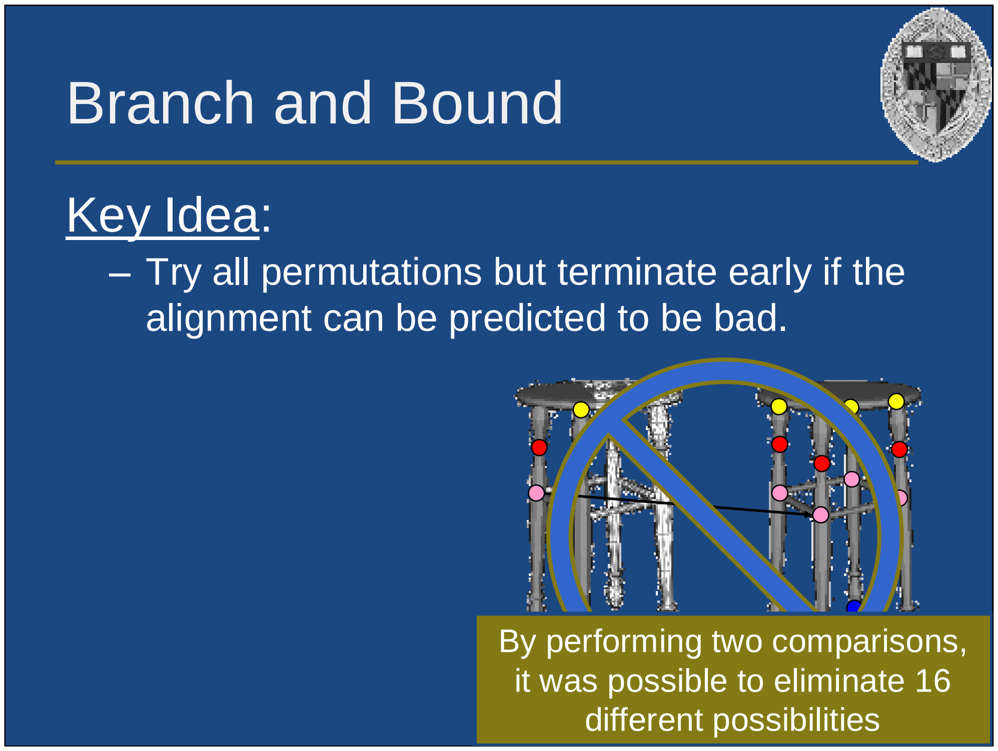
Given points $\{p_1,\ldots,p_n\}$ on the query, if $p_i$ matches $m_i$ different target points:

$$|\Psi| = \prod_{i=1}^{n} m_i$$

$|\Psi|=4^4=256$ possible permutations

Of course, it is theoretically possible to exhaustively search over the space of all possible correspondences for the set of correspondences that define the optimal alignment.

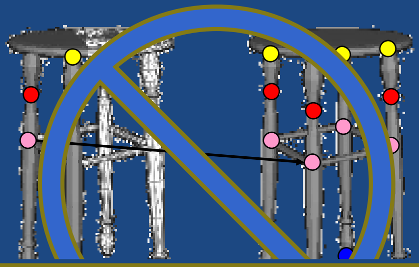
However, in practice this type of approach becomes intractable.



## Branch and Bound

Key Idea:

– Try all permutations but terminate early if the alignment can be predicted to be bad.



By performing two comparisons, it was possible to eliminate 16 different possibilities

One way to try and address this problem is using a “branch and bound” technique.

The idea behind this type of approach is to find the optimal choice of correspondences by descending the decision tree, incrementally making choices about which pairs of points should be in correspondence, and terminating early if the current set of correspondences can be predicted to give a bad alignment.

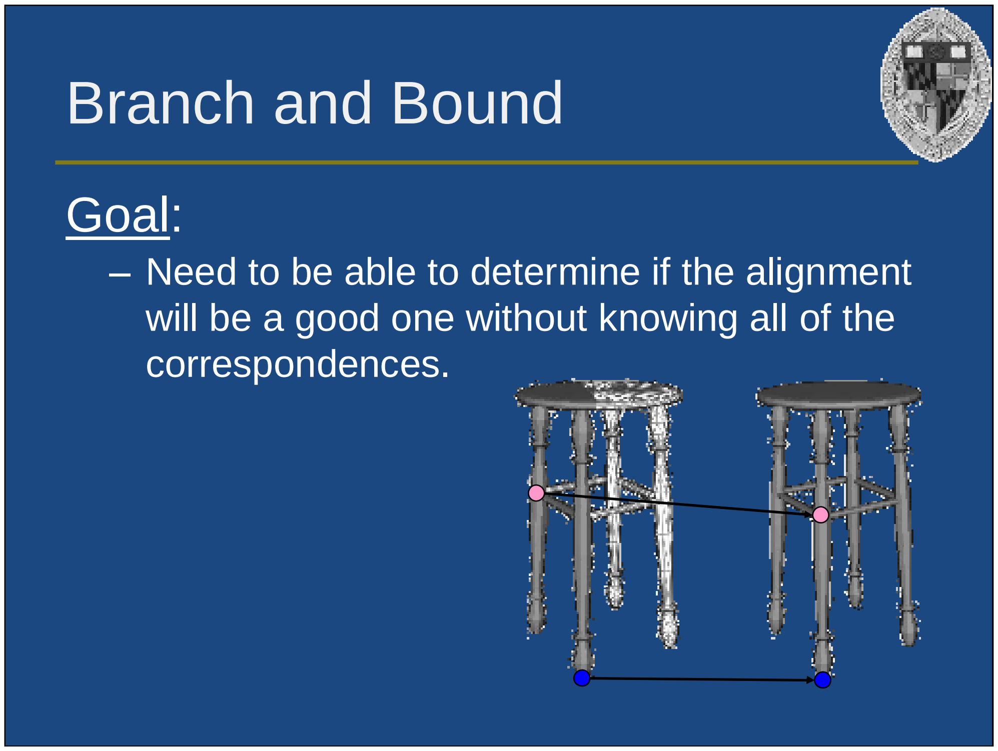
As an example consider the image on the bottom right.

If at the first step we associate the blue feature point in the partial scan to the corresponding feature point on the front leg of the whole model,

And then we associate the pink feature point in the partial scan to the corresponding feature point on the front leg of the whole model,

We can predict that any alignment derived from these decisions must be bad since points on different legs of the partial scan would end up on the same leg of the whole model.

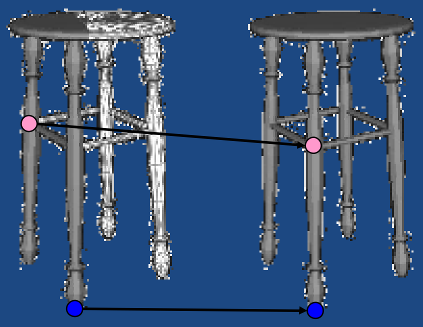
Thus, by terminating early, we can avoid performing 16 unnecessary alignments.



## Branch and Bound

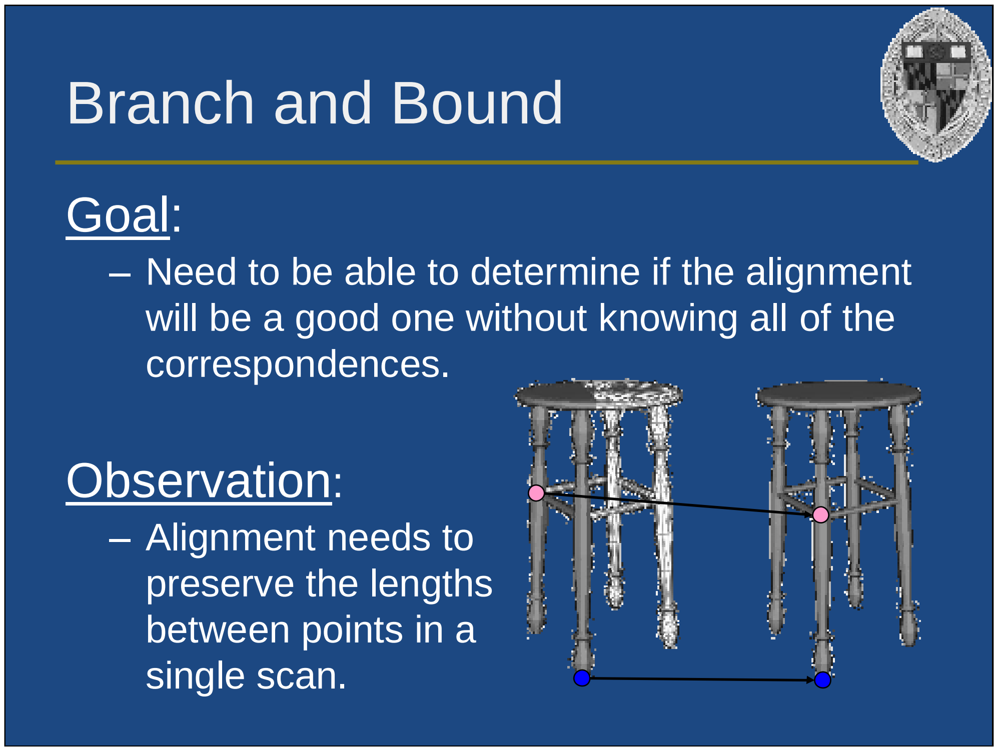
Goal:

– Need to be able to determine if the alignment will be a good one without knowing all of the correspondences.



In order to be able to use this type of “branch and bound” approach, we need to be able to predict if a subset of correspondences will give rise to a large alignment error.

The difficulty in this is that we cannot tell what the final alignment will be, knowing only a subset of the correspondences.



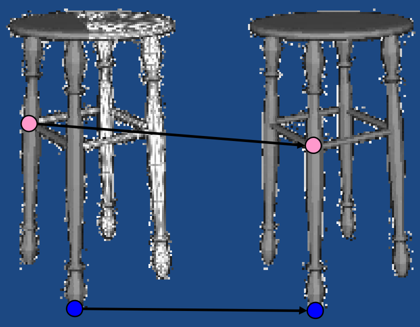
## Branch and Bound

Goal:

– Need to be able to determine if the alignment will be a good one without knowing all of the correspondences.

Observation:

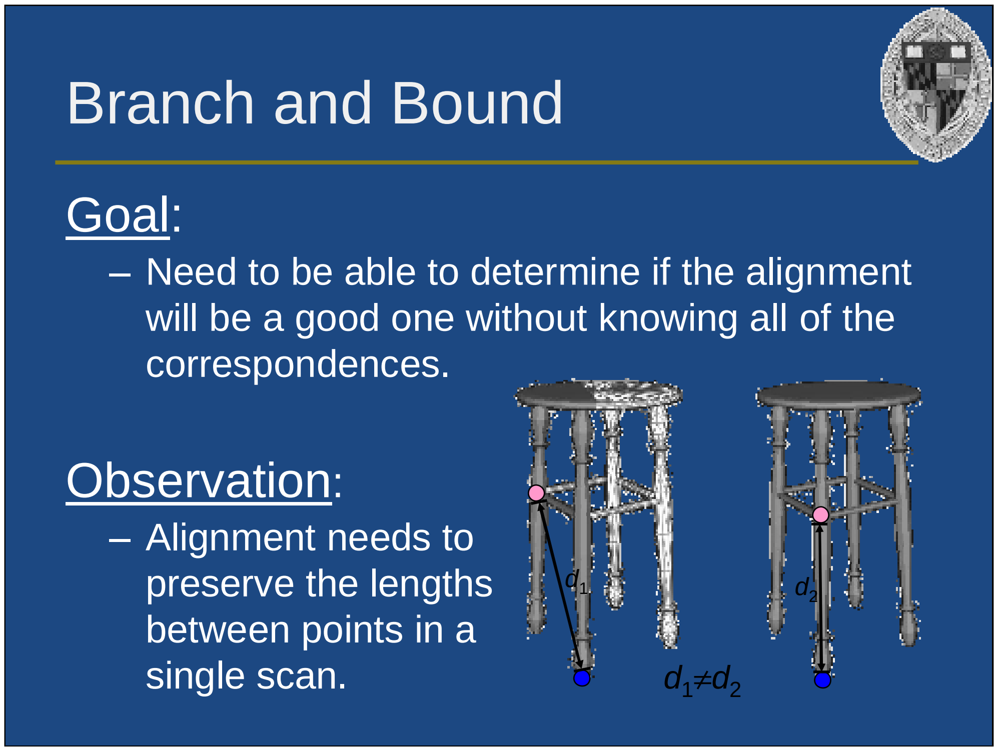
– Alignment needs to preserve the lengths between points in a single scan.



A solution to this problem proposed by Gelfand et al. is based on the observation that the set of inter-feature distances between features within a scan need to be preserved across correspondences.

Since these distances depend only on the correspondences and not on the aligning registration, this provides a method for evaluating the quality of registration knowing only a subset of the correspondences.

As an example, consider the case shown in the image in the bottom right when we try to map points from different legs to corresponding points on the same leg.



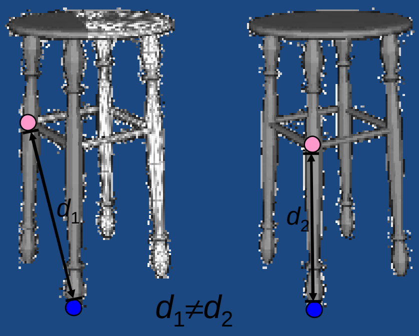
## Branch and Bound

Goal:

– Need to be able to determine if the alignment will be a good one without knowing all of the correspondences.

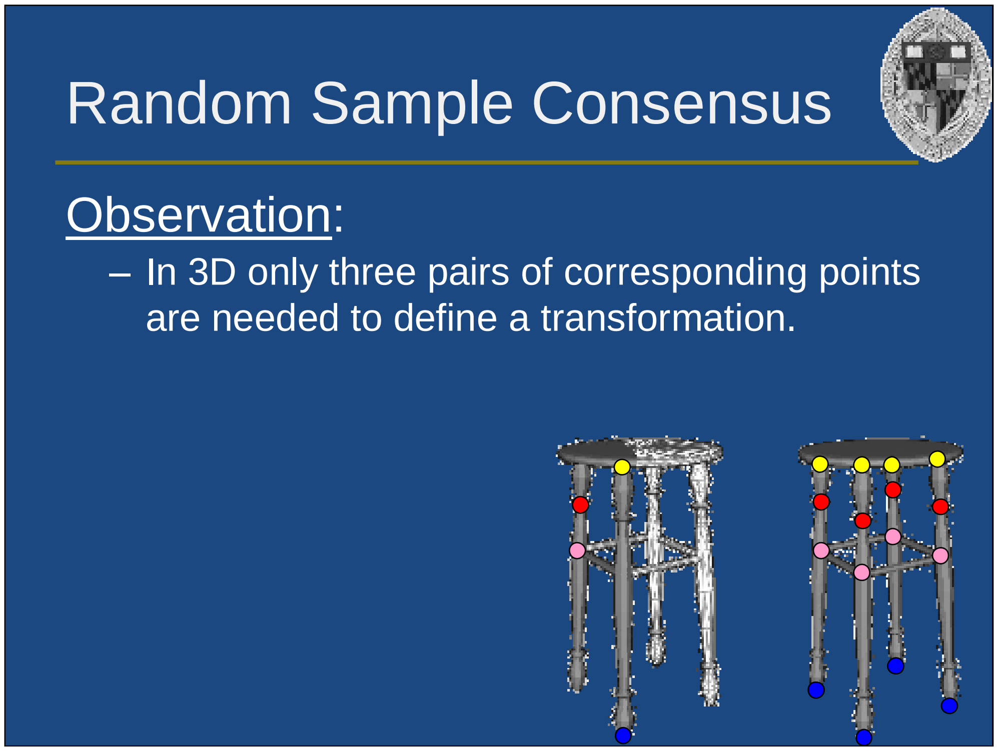
Observation:

– Alignment needs to preserve the lengths between points in a single scan.



We can use the fact that the distance between the blue and pink feature points on the partial scan has to be equal to the distance between the corresponding blue and pink feature points on the whole model.

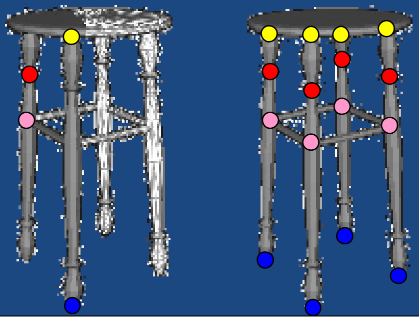
Since these distance are not equal, we know that regardless of the aligning transformation the registration will be bad, and we can terminate early.



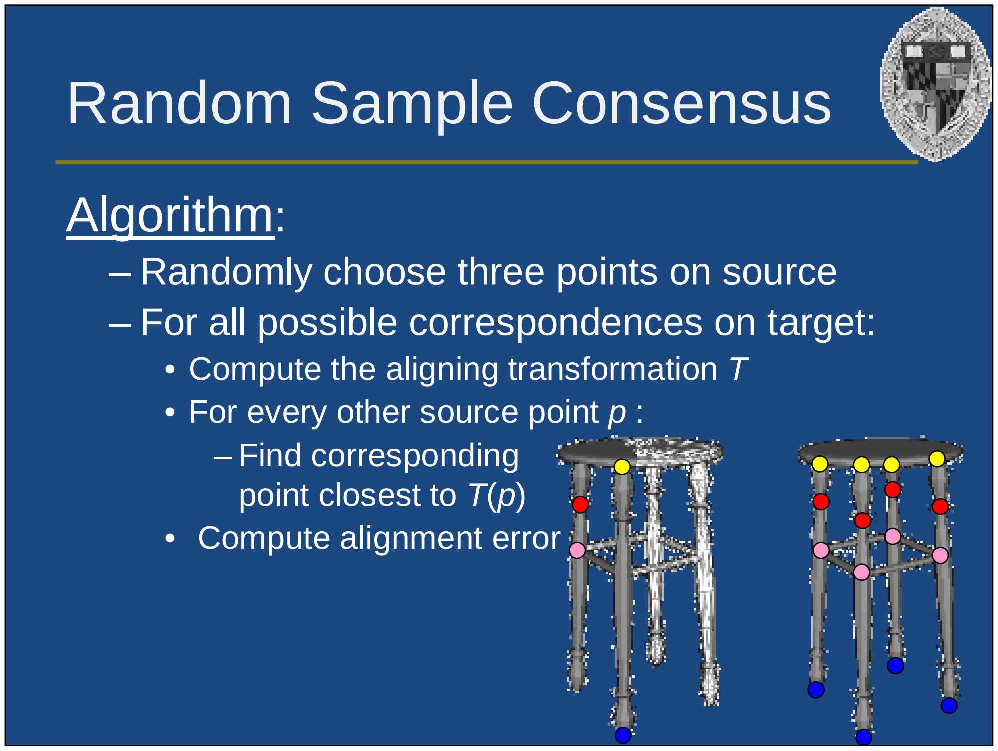
## Random Sample Consensus

Observation:

– In 3D only three pairs of corresponding points are needed to define a transformation.



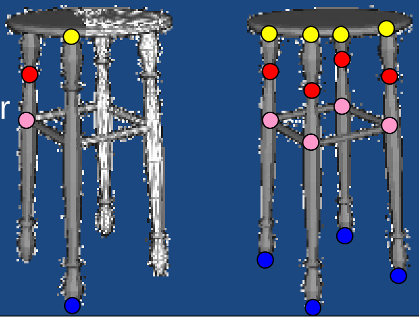
An alternative approach is to use Fischler and Bolles's RANSAC algorithm, leveraging the fact that three corresponding point pairs provide enough information to define a rigid body transformation.



## Random Sample Consensus

Algorithm:

- Randomly choose three points on source
- For all possible correspondences on target:
  - Compute the aligning transformation *T*
  - For every other source point *p* :
    - Find corresponding point closest to *T*(*p*)
  - Compute alignment error



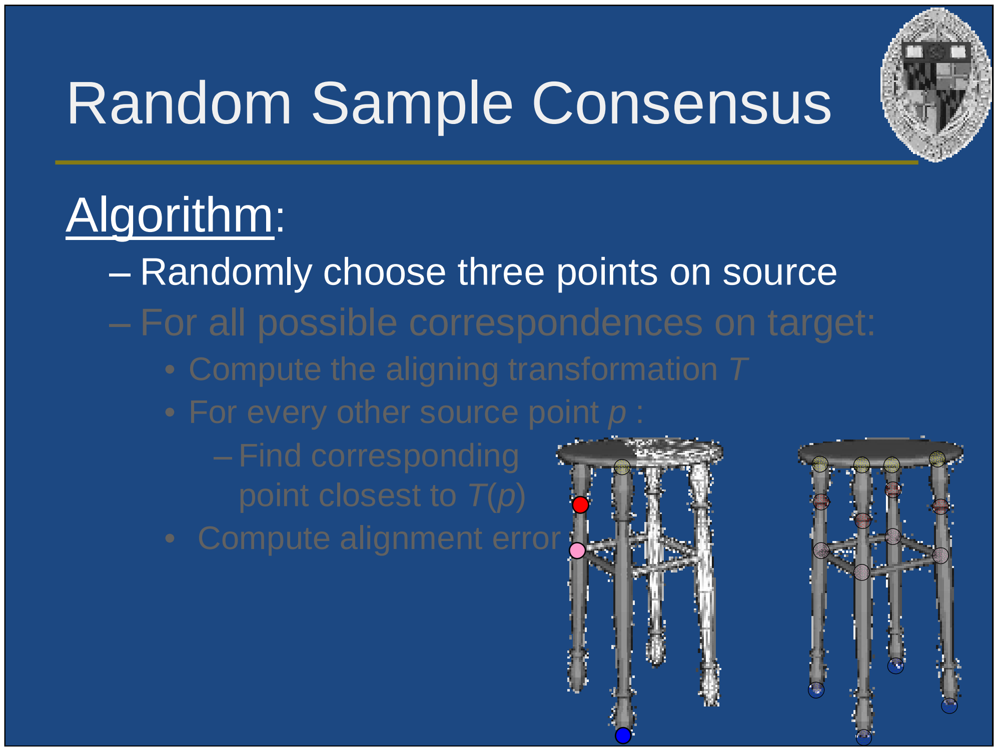
The idea behind this approach is to randomly choose three source feature points.

For each possible correspondence defined by this triplet of points, we can define a unique, rigid body transformation T.

Using this transformation, we choose the correspondences for the rest of the source points by choosing the point on the target that would be brought closest by the transformation T.

Then, we compute the alignment error for this set of correspondences.

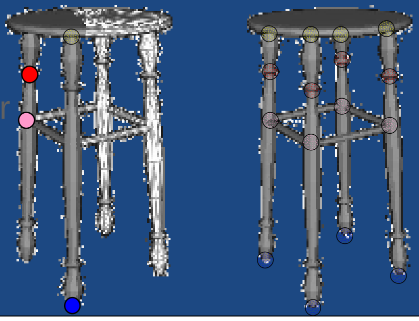
The advantage of this approach is that after choosing the initial source points, the correspondences for the other source points are independent of each other. They are simply chosen so as to minimize the distance from the transformation of the source point. Thus, it transforms the correspondence finding problem from one whose complexity is multiplicative in the number of different potential correspondences to an algorithm whose complexity is additive.



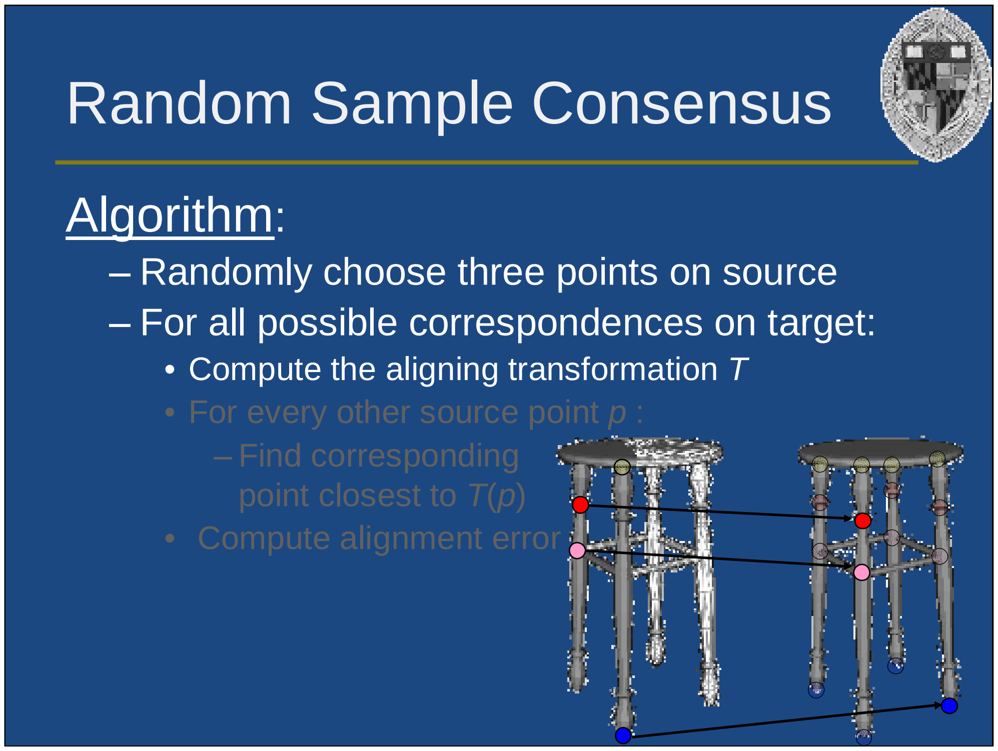
## Random Sample Consensus

Algorithm:

- Randomly choose three points on source
- For all possible correspondences on target:
  - Compute the aligning transformation *T*
  - For every other source point *p* :
    - Find corresponding point closest to *T*(*p*)
  - Compute alignment error



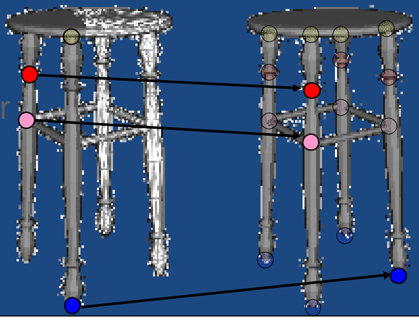
As an example, let us consider the case when the red, pink and blue points are chosen on the source.



## Random Sample Consensus

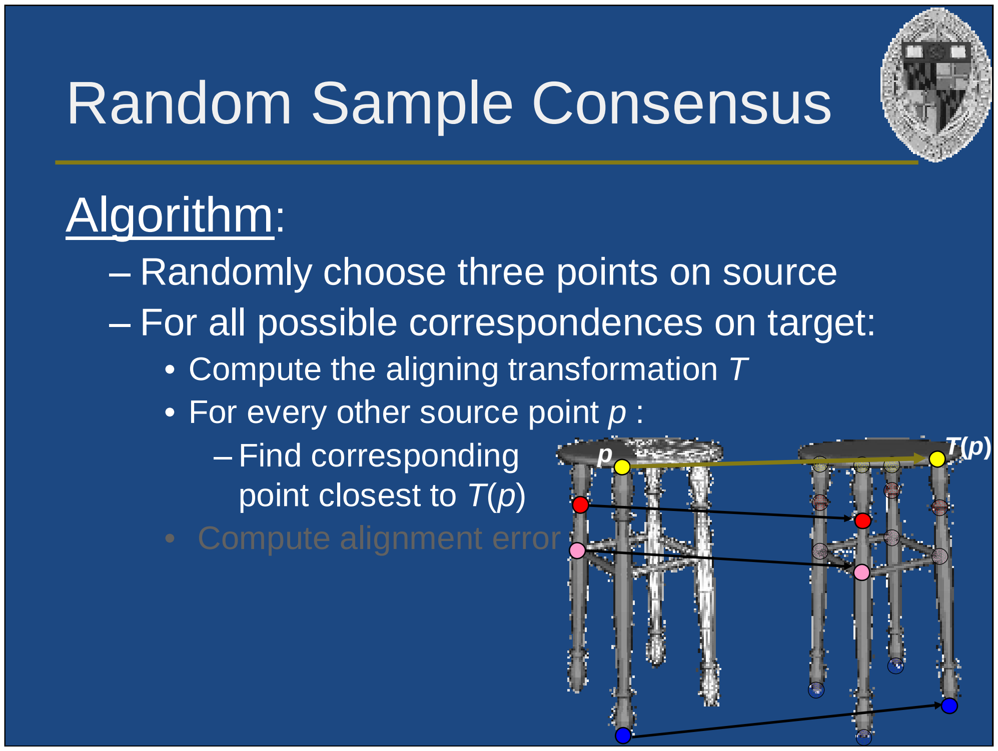
Algorithm:

- Randomly choose three points on source
- For all possible correspondences on target:
  - Compute the aligning transformation *T*
  - For every other source point *p* :
    - Find corresponding point closest to *T*(*p*)
  - Compute alignment error



If we are lucky, then we will choose the correct corresponding points on the target.

In this example, this choice of correspondences defines an aligning rotation about the vertical axis by 90 degrees.

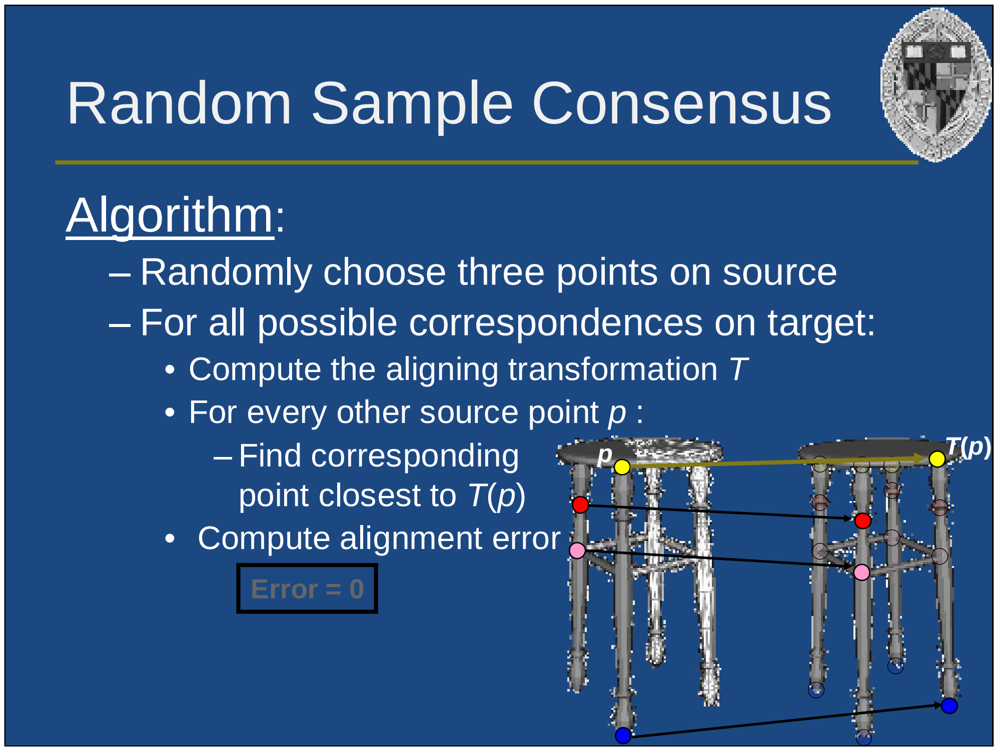


## Random Sample Consensus

Algorithm:

- Randomly choose three points on source
- For all possible correspondences on target:
  - Compute the aligning transformation *T*
  - For every other source point *p* :
    - Find corresponding point closest to *T*(*p*)
  - Compute alignment error

This, in turn, will dictate where the yellow feature point needs to be mapped, since we have to map it to the point on the target closest to the 90 degree rotation of the yellow source point.



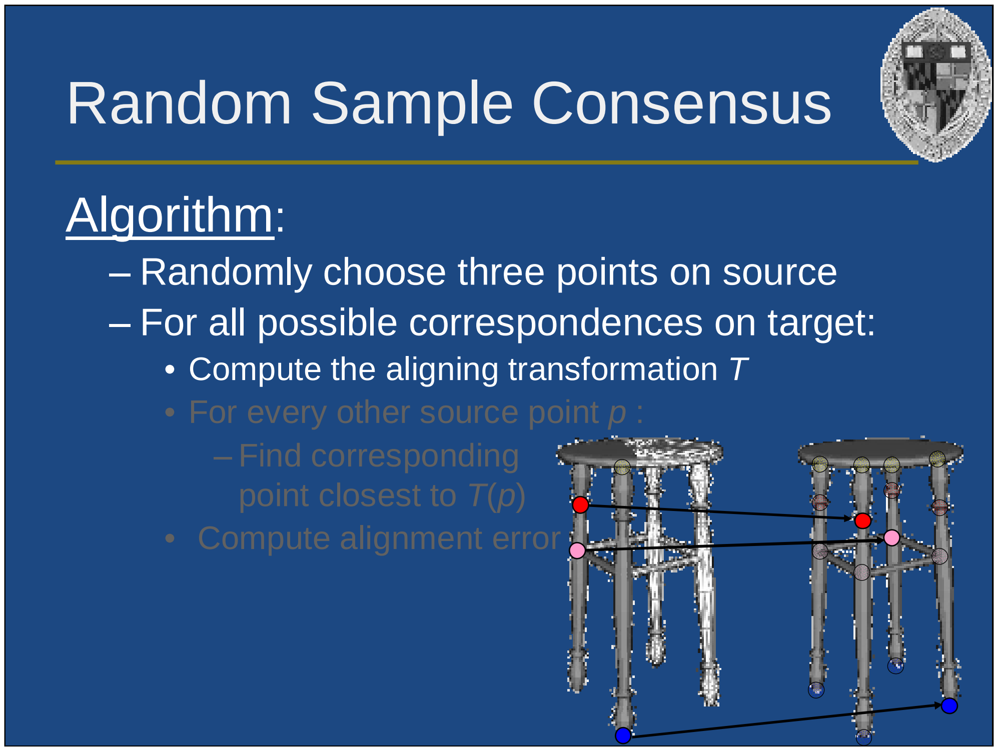
## Random Sample Consensus

Algorithm:

- Randomly choose three points on source
- For all possible correspondences on target:
  - Compute the aligning transformation *T*
  - For every other source point *p* :
    - Find corresponding point closest to *T*(*p*)
  - Compute alignment error

Error = 0

Since all source points are perfectly mapped to target points, we get an alignment error of zero.



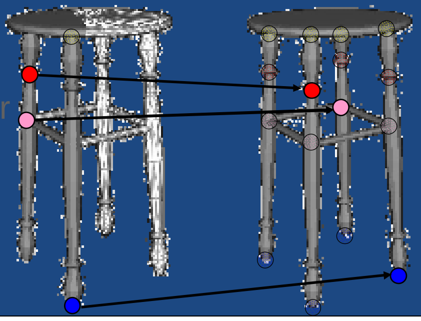
## Random Sample Consensus

Algorithm:

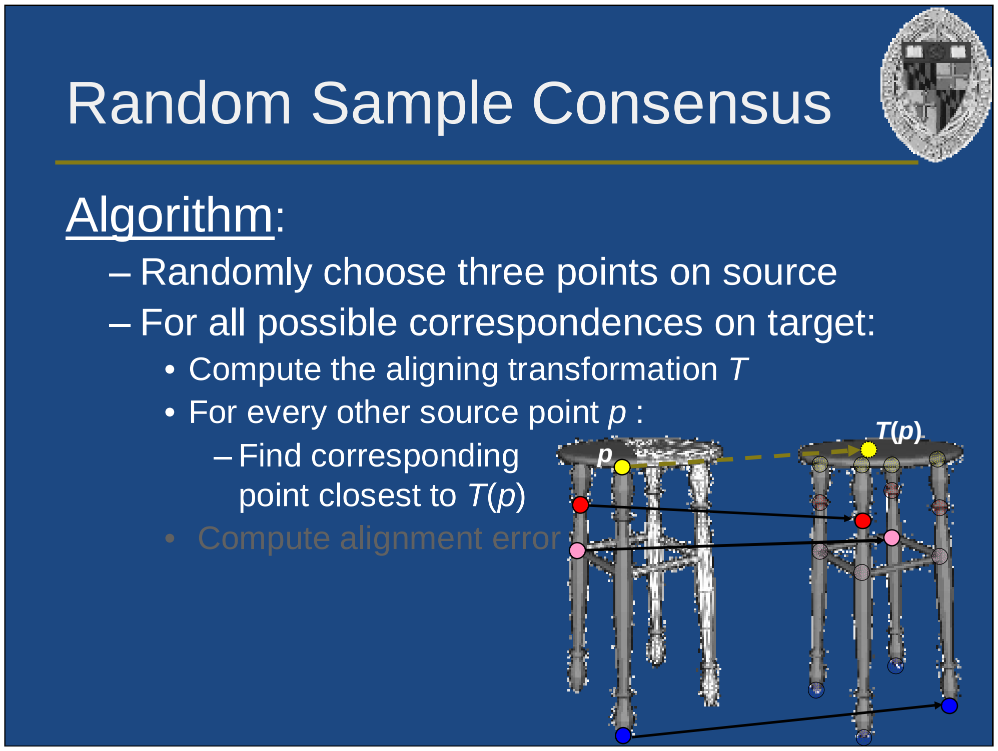
– Randomly choose three points on source

– For all possible correspondences on target:

- Compute the aligning transformation *T*
- For every other source point *p* :
  – Find corresponding point closest to *T*(*p*)
- Compute alignment error



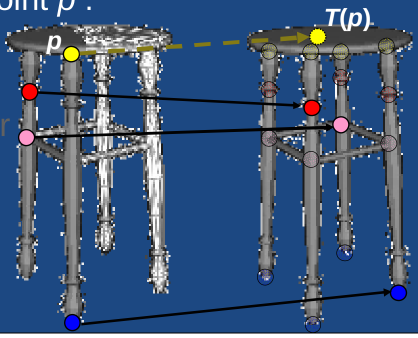
If, instead, we were to map the three source points to some other corresponding target points,



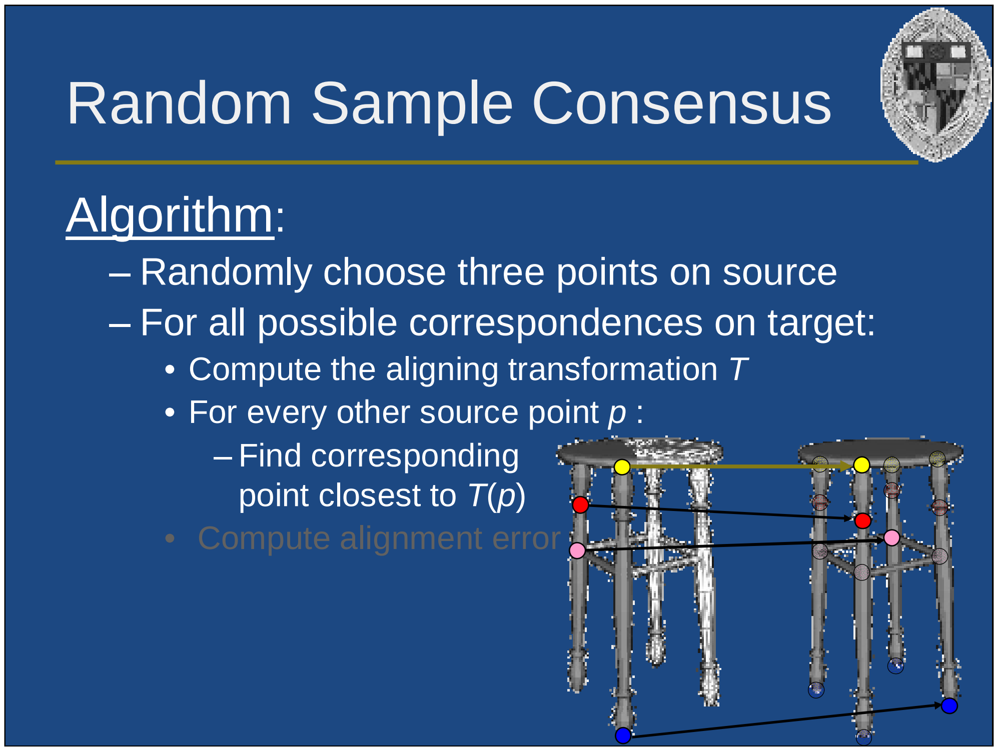
## Random Sample Consensus

Algorithm:

- Randomly choose three points on source
- For all possible correspondences on target:
  - Compute the aligning transformation *T*
  - For every other source point *p* :
    - Find corresponding point closest to *T*(*p*)
  - Compute alignment error



This would define some other transformations that would map the yellow source point to some point that is not a target feature,

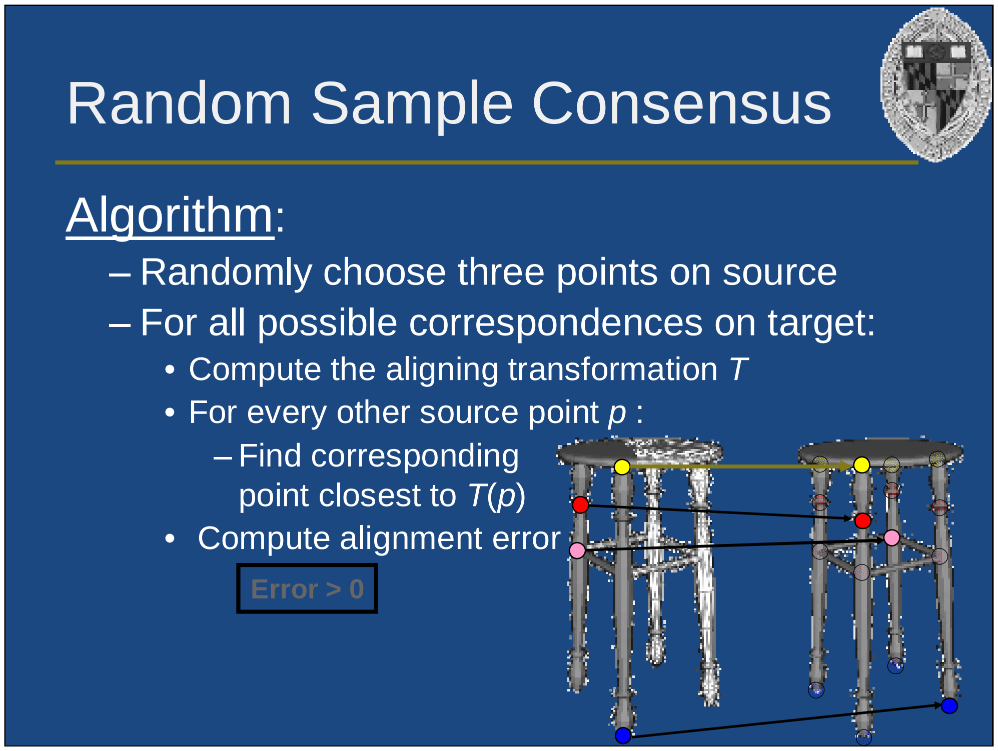


## Random Sample Consensus

Algorithm:

- Randomly choose three points on source
- For all possible correspondences on target:
  - Compute the aligning transformation *T*
  - For every other source point *p* :
    - Find corresponding point closest to *T*(*p*)
  - Compute alignment error

And hence the yellow source feature point would be set to correspond to the incorrect target feature,

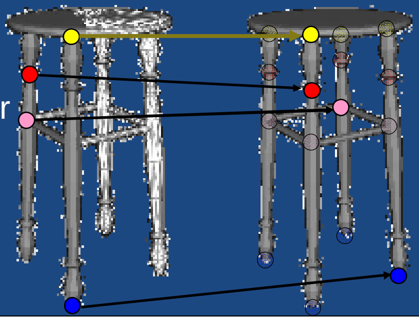


## Random Sample Consensus

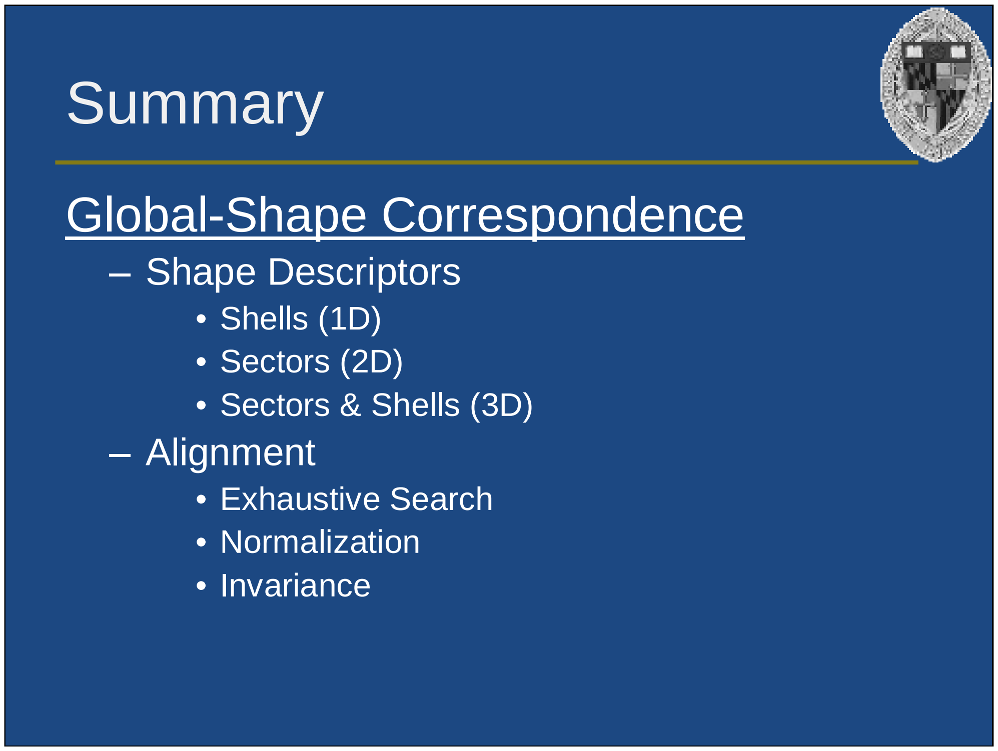
Algorithm:

- Randomly choose three points on source
- For all possible correspondences on target:
  - Compute the aligning transformation *T*
  - For every other source point *p* :
    - Find corresponding point closest to *T*(*p*)
  - Compute alignment error

Error > 0



Resulting in an overall alignment error that is larger than zero.

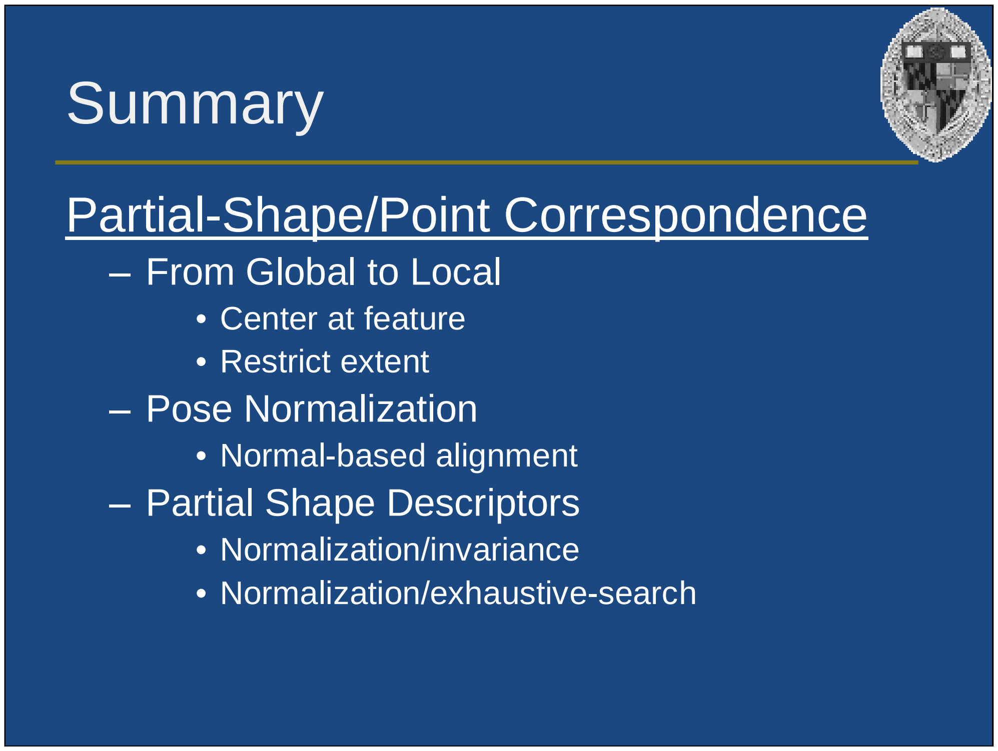


## Summary

**Global-Shape Correspondence**

- Shape Descriptors
  - Shells (1D)
  - Sectors (2D)
  - Sectors & Shells (3D)
- Alignment
  - Exhaustive Search
  - Normalization
  - Invariance

In summary, we began by covering some of the techniques for whole-shape matching, describing several different types of shape descriptors and then moving on to a discussion of the different methods for addressing the alignment problem.



## Summary

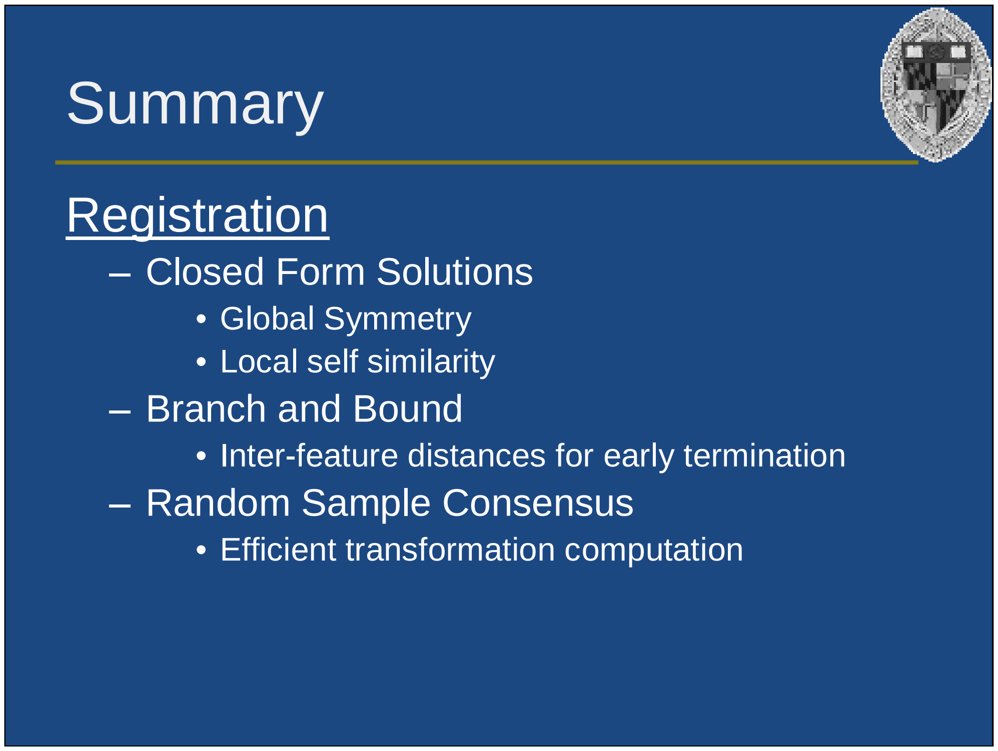
Partial-Shape/Point Correspondence

- From Global to Local
  - Center at feature
  - Restrict extent
- Pose Normalization
  - Normal-based alignment
- Partial Shape Descriptors
  - Normalization/invariance
  - Normalization/exhaustive-search

We showed that global shape descriptors can be used in local shape matching and correspondence detection by translating the descriptors so that they are centered about the feature point and restricting radius of extent.

While the shape descriptors themselves naturally carry over to the partial shape context, the pose normalization did not carry over as well, due to the fact that it uses global shape information to define the canonical frame.

We discussed alternative method for normalization, using local shape information, and considered different methods that use either descriptor invariance or exhaustive search to address the partial-shape matching problem.



## Summary

**Registration**

- Closed Form Solutions
  - Global Symmetry
  - Local self similarity
- Branch and Bound
  - Inter-feature distances for early termination
- Random Sample Consensus
  - Efficient transformation computation

Finally, having discussed methods for establishing potential correspondences we described methods for using the correspondences to define the aligning registration. We found that due to both local and global self-similarity, the correspondences established were not one-to-one and hence closed form solutions could not be directly applied to the scan registration problem.

We discussed two techniques that addressed this problem:

The branch and bound technique of Gelfand et al. which uses the independence of inter-feature distances from aligning registration to define an early termination algorithm for registration, and

An implementation of the RANSAC algorithm that leverages the fact that aligning registrations only need three pairs of corresponding points.