**Abstract**

A longstanding question in computer vision concerns the representation of 3D shapes for recognition: should 3D shapes be represented with descriptors operating on their native 3D formats, such as voxel grid or polygon mesh, or can they be effectively represented with view-based descriptors?

We address this question in the context of learning to recognize 3D shapes from a collection of their rendered views on 2D images.

We first present a standard CNN architecture trained to recognize the shapes’ rendered views independently of each other, and show that a 3D shape can be recognized even from a single view at an accuracy far higher than using state-of-the-art 3D shape descriptors.

Recognition rates further increase when multiple views of the shapes are provided. In addition, we present a novel CNN architecture that combines information from multiple views of a 3D shape into a single and compact shape descriptor offering even better recognition performance.

The same architecture can be applied to accurately recognize human hand-drawn sketches of shapes.

We conclude that a collection of 2D views can be highly informative for 3D shape recognition and is amenable to emerging CNN architectures and their derivatives.

**1. Introduction**

One of the fundamental challenges of computer vision is to draw inferences about the three-dimensional (3D) world from two-dimensional (2D) images.

Since one seldom has access to 3D object models, one must usually learn to recognize

and reason about 3D objects based upon their 2D appearances from various viewpoints. Thus, computer vision researchers have typically developed object recognition algorithms from 2D features of 2D images, and used them to classify new 2D pictures of those objects.

But what if one does have access to 3D models of each object of interest? In this case, one can directly train recognition algorithms on 3D features such as voxel occupancy

or surface curvature.

The possibility of building such classifiers of 3D shapes directly from 3D representations has recently emerged due to the introduction of large 3D shape repositories,

such as 3DWarehouse, TurboSquid, and Shapeways.

For example, when Wu et al. [37] introduced the ModelNet 3D shape database, they presented a classifier for 3D shapes using a deep belief network architecture trained on voxel representations.

While intuitively, it seems logical to build 3D shape classifiers directly from 3D models, in this paper we present a seemingly counterintuitive result – that by building classifiers

of 3D shapes from 2D image renderings of those shapes,

we can actually dramatically outperform the classifiers built directly on the 3D representations.

In particular, a convolutional neural network (CNN) trained on a fixed set of rendered views of a 3D shape and only provided with a single view at test time increases category recognition accuracy by a remarkable 8% (77%!85%) over the best models [37] trained on 3D representations.

With more views provided at test time, its performance further increases

One reason for this result is the relative efficiency of the 2D versus the 3D representations. In particular, while a full resolution 3D representation contains all of the information about an object, in order to use a voxel-based representation in a deep network that can be trained with available samples and in a reasonable amount of time, it would appear that the resolution needs to be significantly reduced. For example, 3D ShapeNets use a coarse representation of shape, a 30\_30\_30 grid of binary voxels. In contrast a single projection of the 3D model of the same input size corresponds to an image of 164\_164 pixels, or slightly smaller if multiple projections are used. Indeed, there is an inherent trade-off between increasing the amount of explicit depth information (3D models) and increasing spatial resolution (projected 2D models).

Another advantage of using 2D representations is that we can leverage (i) advances in image descriptors [22, 26] and (ii) massive image databases (such as ImageNet [9]) to pre-train our CNN architectures.

Because images are ubiquitous and large labeled datasets are abundant,

we can learn a good deal about generic features for 2D image categorization and then fine-tune to specifics about 3D model projections.

While it is possible that some day as much 3D training data will be available, for the time being this is a significant advantage of our representation.

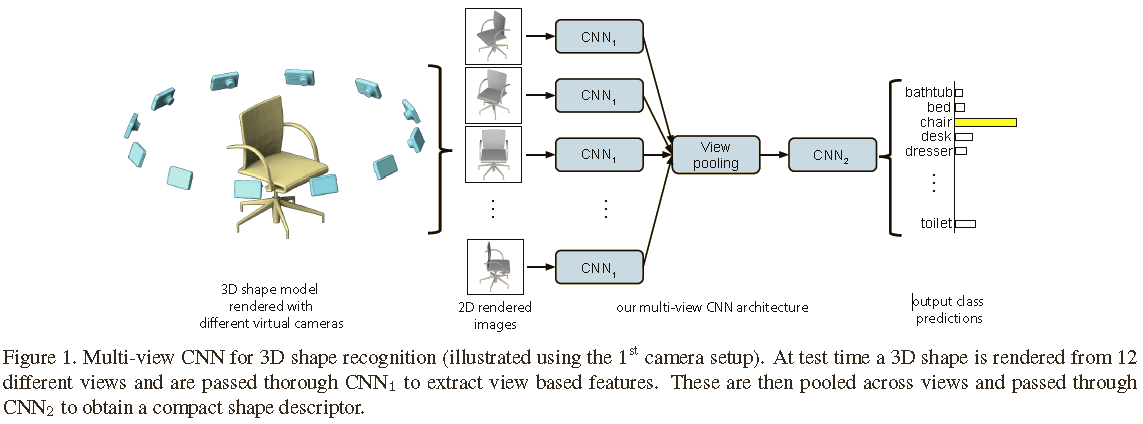


Figure 1. Multi-view CNN for 3D shape recognition (illustrated using the 1st camera setup). At test time a 3D shape is rendered from 12

different views and are passed thorough CNN1 to extract view based features. These are then pooled across views and passed through

CNN2 to obtain a compact shape descriptor.

Although the simple strategy of classifying views independently works remarkably well (Sect. 3.2), we present new ideas for how to “compile” the information in multiple 2D views of an object into a compact object descriptor using a new architecture called multi-view CNN (Fig. 1 and Sect. 3.3).

This descriptor is at least as informative for classification (and for retrieval is slightly more informative) than the full collection of view-based descriptors of the object.

Moreover it facilitates efficient retrieval using either a similar 3D object or a simple hand-drawn sketch, without resorting to slower methods that are based on pairwise comparisons of image descriptors. We present state-of-the-art results on 3D object classification, 3D object retrieval using 3D objects, and 3D object retrieval using sketches (Sect. 4).

Our multi-view CNN is related to “jittering” where transformed copies of the data are added during training to learn invariances to transformations such as rotation or translation. In the context of 3D recognition the views can be seen as jittered copies.

The multi-view CNN learns to combine the views instead of averaging, and thus can use the

more informative views of the object for prediction while ignoring others.

Our experiments show that this improves performance (Sect. 4.1) and also lets us visualize informative views of the object by back-propagating the gradients of the network to the views (Fig. 3).

Even on traditional image classification tasks multi-view CNN can be a better alternative to jittering. For example, on the sketch recognition benchmark [11] a multi-view CNN trained on jittered copies performs better than a standard CNN trained with the same jittered copies (Sect. 4.2).

Pre-trained CNN models, data, and the complete source code to reproduce the results in the paper are available at

<http://vis-www.cs.umass.edu/mvcnn>.

**2. Related Work**

Our method is related to prior work on shape descriptors for 3D objects and image-based CNNs. Next we discuss representative work in these areas.

Shape descriptors. A large corpus of shape descriptors has been developed for drawing inferences about 3D objects in both the computer vision and graphics literature.

Shape descriptors can be classified into two broad categories:

3D shape descriptors that directly work on the native 3D representations of objects,

such as polygon meshes, voxel-based discretizations,

point clouds, or implicit surfaces, and viewbased descriptors that describe the shape of a 3D object by “how it looks” in a collection of 2D projections.

With the exception of the recent work of Wu et al. [37] which learns shape descriptors from the voxel-based representation of an object through 3D convolutional nets, previous 3D shape descriptors were largely “hand-designed” according to a particular geometric property of the shape surface or volume.

For example, shapes can be represented with histograms or bag-of-features models constructed out of surface normals and curvatures [15], distances, angles, triangle areas or tetrahedra volumes gathered at sampled surface points [25], properties of spherical functions defined in volumetric grids [16], local shape diameters measured at densely sampled surface points [4], heat kernel signatures on polygon meshes [2, 19], or extensions of the SIFT and SURF feature descriptors to 3D voxel grids [17.[

Developing classifiers and other supervised machine learning algorithms on top of such 3D shape descriptors poses a number of challenges.

First, the size of organized databases with annotated 3D models is rather limited compared to image datasets,

e.g., ModelNet contains about 150K shapes (its 40 category benchmark contains about 4K shapes). In contrast, the ImageNet database already includes tens of millions of annotated images. Second, 3D shape descriptors tend to

be very high-dimensional, making classifiers prone to overfitting due to the so-called ‘curse of dimensionality’.

On the other hand view-based descriptors have a number of desirable properties: they are relatively low-dimensional, efficient to evaluate, and robust to 3D shape representation artifacts, such as holes, imperfect polygon mesh tesselations, noisy surfaces. The rendered shape views can also be directly compared with other 2D images, silhouettes or even hand-drawn sketches.

An early example of a view-based approach is the work by Murase and Nayar that recognizes objects by matching their appearance in parametric eigenspaces formed by large sets of 2D renderings of 3D models under varying poses and illuminations.

Another example, which is particularly popular in computer graphics setups, is the LightField descriptor that extracts a set of geometric and Fourier descriptors from object silhouettes rendered from several different viewpoints.

Alternatively, the silhouette of an object can be decomposed into parts and then represented by a directed acyclic graph (shock graph).

Cyr and Kimia defined similarity metrics based on curve matching and grouped similar views, called aspect graphs of 3D models . Eitz et al. compared human sketches with line drawings of 3D models produced from several different views based on local Gabor filters, while Schneider et al. proposed using Fisher vectors on SIFT features for representing human sketches of shapes. These descriptors are largely “hand-engineered” and some do not generalize well across different domains.

**Convolutional neural networks.** Our work is also related to recent advances in image recognition using CNNs.

In particular CNNs trained on the large datasets such as ImageNet have been shown to learn general purpose image descriptors for a number of vision tasks such as object detection, scene recognition, texture recognition and finegrained classification.

We show that these deep architectures can be adapted to specific domains including shaded illustrations of 3D objects, line drawings, and human sketches to produce descriptors that have superior performance compared to other view-based or 3D shape descriptors in a variety of setups.

Furthermore, they are compact and efficient to compute. There has been existing work on recognizing 3D objects with CNNs [21] using two concatenated views (binocular images) as input.

Our network instead learns a shape representation that aggregates information from any number of input views without any specific ordering, and always outputs a compact shape descriptor of the same size.

Furthermore, we leverage both image and shape datasets to train our network.

Although there is significant work on 3D and 2D shape descriptors, and estimating informative views of the objects (or, aspect graphs), there is relatively little work on learning

to combine the view-based descriptors for 3D shape recognition.

Most methods resort to simple strategies such as performing exhaustive pairwise comparisons of descriptors extracted from different views of each shape, or concatenating descriptors from ordered, consistent views.

In contrast our multi-view CNN architecture learns to recognize 3D shapes from views of the shapes using image-based CNNs but in the context of other views via a view-pooling layer. As a result, information from multiple views is effectively accumulated into a single, compact shape descriptor.

**3. Method**

As discussed above, our focus in this paper is on developing view-based descriptors for 3D shapes that are trainable, produce informative representations for recognition and retrieval tasks, and are efficient to compute.

Our view-based representations start from multiple views of a 3D shape,

generated by a rendering engine. A simple way to use multiple views is to generate a 2D image descriptor per each view, and then use the individual descriptors directly for recognition tasks based on some voting or alignment scheme.

For example, a na¨ıve approach would be to average the individual descriptors,

treating all the views as equally important.

Alternatively, if the views are rendered in a reproducible order, one could also concatenate the 2D descriptors of all the views. Unfortunately, aligning a 3D shape to a canonical orientation is hard and sometimes ill-defined. In contrast to the above simple approaches, an aggregated representation combining features from multiple views is more desirable since it yields a single, compact descriptor representing the 3D shape.

Our approach is to learn to combine information from multiple views using a unified CNN architecture that includes a view-pooling layer (Fig. 1).

All the parameters of our CNN architecture are learned discriminatively to produce a single compact descriptor for the 3D shape.

Compared to exhaustive pairwise comparisons between singleview representations of 3D shapes, our resulting descriptors can be directly used to compare 3D shapes leading to significantly higher computational efficiency.

**3.1. Input: A Multiview Representation**

3D models in online databases are typically stored as polygon meshes, which are collections of points connected with edges forming faces. To generate rendered views of polygon meshes, we use the Phong reflection model.

The mesh polygons are rendered under a perspective projection and the pixel color is determined by interpolating the reflected intensity of the polygon vertices.

Shapes are uniformly scaled to fit into the viewing volume.

To create a multi-view shape representation, we need to setup viewpoints (virtual cameras) for rendering each mesh. We experimented with two camera setups.

For the 1st camera setup, we assume that the input shapes are upright oriented

along a consistent axis (e.g., z-axis).

Most models in modern online repositories, such as the 3D Warehouse, satisfy this requirement, and some previous recognition methods also follow the same assumption [37]. In this case, we create 12 rendered views by placing 12 virtual camerasaround the mesh every 30 degrees (see Fig. 1).

The cameras are elevated 30 degrees from the ground plane, pointing towards the centroid of the mesh. The centroid is calculated as the weighted average of the mesh face centers, where the weights are the face areas.

For the 2nd camera setup, we do not make use of the assumption about consistent upright orientation of shapes. In this case, we render from several more viewpoints since we do not know beforehand which ones yield good representative views of the object.

The renderings are generated by placing 20 virtual cameras at the 20 vertices of an icosahedron enclosing the shape.

All cameras point towards the centroid of the mesh. Then we generate 4 rendered views from each camera, using 0, 90, 180, 270 degrees rotation along the axis passing through the camera and the object centroid, yielding total 80 views.

We note that using different shading coefficients or illumination models did not affect our output descriptors due to the invariance of the learned filters to illumination changes, as also observed in image-based CNNs.

Adding more or different viewpoints is trivial, however, we found that the above camera setups were already enough to achieve high performance. Finally, rendering each mesh from all the viewpoints takes no more than ten milliseconds on modern graphics hardware.

**3.2. Recognition with Multiview Representations**

We claim that our multi-view representation contains rich information about 3D shapes and can be applied to various types of tasks. In the first setting, we make use of existing 2D image features directly and produce a descriptor for each view.

This is the most straightforward approach to utilize the multi-view representation.

However, it results in multiple 2D image descriptors per 3D shape, one per view, which need to be integrated somehow for recognition tasks.

**Image descriptors.** We consider two types of image descriptors for each 2D view: a state-of-the-art “hand-crafted” image descriptor based on Fisher vectors with multiscale SIFT, as well as CNN activation features.

The Fisher vector image descriptor is implemented using VLFeat.

For each image multi-scale SIFT descriptors are extracted densely.

These are then projected to 80 dimensions with PCA, followed by Fisher vector pooling with a Gaussian mixture model with 64 components, square-root and `2 normalization.

For our CNN features we use the VGG-M network mixture model with 64 components, square-root and `2 normalization.

For our CNN features we use the VGG-M network from which consists of mainly five convolutional layers conv1…5 followed by three fully connected layers fc6…8 and a softmax classification layer.

The penultimate layer fc7 (after ReLU non-linearity, 4096-dimensional) is used as image descriptor. The network is pre-trained on ImageNet images from 1k categories, and then fine-tuned on all 2D views of the 3D shapes in training set.

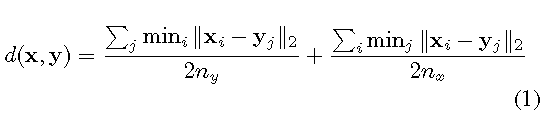
As we show in our experiments, fine-tuning improves performance significantly.

Both Fisher vectors and CNN features yield very good performance in classification and retrieval compared with popular 3D shape descriptors (e.g., SPH, LFD) as well as 3D ShapeNets.

**Classification**. We train one-vs-rest linear SVMs (each view is treated as a separate training sample) to classify shapes using their image features.

At test time, we simply sum up the SVM decision values over all 12 views and return the class with the highest sum. Alternative approaches, e.g., averaging image descriptors, lead to worse accuracy.

**Retrieval**. A distance or similarity measure is required for retrieval tasks. For shape x with nx image descriptors and shape y with ny image descriptors, the distance between them is defined in Eq. 1. Note that the distance between two 2D images is defined as the `2 distance between their feature vectors, 



To interpret this definition, we can first define the distance between a 2D image xi and a 3D shape y as d(xi; y) = minj kxi 􀀀 yjk2. Then given all nx distances between x’s 2D projections and y, the distance between these two shapes is computed by simple averaging. In Eq. 1, this idea is applied in both directions to ensure symmetry.

We investigated alternative distance measures, such as

minimun distance among all nx \_ ny image pairs and the distance between average image descriptors, but they all led to inferior performance.

**3.3. Multiview CNN: Learning to Aggregate Views**

Although having multiple separate descriptors for each 3D shape can be successful for classification and retrieval compared to existing 3D descriptors, it can be inconvenient and inefficient in many cases. For example, in Eq. 1, we need to compute all nx\_ny pairwise distances between images in order to compute distance between two 3D shapes.

Simply averaging or concatenating the image descriptors leads to inferior performance.

In this section, we focus on the problem of learning to aggregate multiple views in order