Computational Visual Media
https://doi.org/10.1007/s41095-019-0135-2 Vol. 5, No. 1, March 2019, 91–104
Research Article

Recurrent 3D attentional networks for end-to-end active object recognition

Min Liu[1,2], Yifei Shi[1], Lintao Zheng[1], Kai Xu[1] (✉), Hui Huang[3], and Dinesh Manocha[2]

Abstract Active vision is inherently attention-driven: an agent actively selects views to attend in order to rapidly perform a vision task while improving its internal representation of the scene being observed. Inspired by the recent success of attention-based models in 2D vision tasks based on single RGB images, we address multi-view depth-based active object recognition using an attention mechanism, by use of an end-to-end recurrent 3D attentional network. The architecture takes advantage of a recurrent neural network to store and update an internal representation. Our model, trained with 3D shape datasets, is able to iteratively attend the best views targeting an object of interest for recognizing it. To realize 3D view selection, we derive a 3D spatial transformer network. It is differentiable, allowing training with backpropagation, and so achieving much faster convergence than the reinforcement learning employed by most existing attention-based models. Experiments show that our method, with only depth input, achieves state-of-the-art next-best-view performance both in terms of time taken and recognition accuracy.

Keywords active object recognition; recurrent neural network; next-best-view; 3D attention

1 School of Computer, National University of Defense Technology, Changsha 410073, China. E-mail: M. Liu, gfsliumin@gmail.com; Y. Shi, jerrysyf@gmail.com; L. Zheng, lintaozheng1991@gmail.com; K. Xu, kevin.kai.xu@gmail.com (✉).

2 Department of Computer Science and Electrical & Computer Engineering, University of Maryland, College Park, 20742, USA. E-mail: dm@cs.umd.edu.

3 Visual Computing Research Center, Shenzhen University, Shenzhen 518060, China. E-mail: huihuang@szu.edu.cn.

Manuscript received: 2018-12-25; accepted: 2019-01-28

1 Introduction

Active object recognition plays a central role in robot-operated autonomous scene understanding and object manipulation. The problem involves online planning of the views used by a visual sensor on a robot for greatest accuracy and confidence in object recognition. This is also referred to as the next-best-view (NBV) problem for active object recognition. Recently, 3D object recognition has advanced greatly thanks to the rapid development of 3D sensing techniques (e.g., depth cameras) and the proliferation of 3D shape repositories. Our work adopts a 3D geometric data-driven approach, in a setting of 2.5D depth acquisition.

Most existing works on view selection are based on the paradigm of information theoretic view evaluation, e.g., Refs. [1, 2]. For example, from a set of candidates, the view maximizing the mutual information between observations and object classes is selected. Such methods often present two issues. Firstly, to estimate the mutual information, unobserved views must be sampled and the corresponding data must be synthesized from a learned generative model, making view estimation inefficient [3]. Secondly, the object recognition model is typically learned independently of the view planner, although the two are really coupled in an active recognition system [4].

Some works formulate active recognition as a reinforcement learning problem, to learn a viewing policy under various observations. In particular, a few recent works have attempted end-to-end reinforcement learning based on recurrent neural networks [4–6]. Applying a learned policy appears to be much more efficient than sampling from a generative model. However, these models are known to be hard to train, with difficult parameter tuning

and relatively long training time [7]. Moreover, the success of these methods highly depends on the hand-designed reward functions.

The recent development of attention-based deep models has led to significant success in 2D vision tasks based on RGB images [7, 8]. Attention-based models achieve both efficiency and accuracy by focusing the processing only on the most informative parts of the input with respect to a given task. Information gained from different fixations is integrated into an internal representation, to approach the desired goal and guide future attention. Such a mechanism, being both goal-directed and stimulus-driven [9], fits well to the problem setting of active recognition, accommodating object recognition and guided acquisition in a unified optimization framework.

However, the popular formulation of attention models based on recurrent neural networks [7] suffers from the problem of non-differentiable recognition loss over attention locations, making network optimization by backpropagation impossible. To make it learnable, training is often turned into a partially observable Markov decision process (POMDP), which comes back to reinforcement learning. The recently introduced differentiable spatial transformer networks (STNs) can be used to actively predict image locations for 2D object detection and recognition [10]. Motivated by this, we opt to use STN units as our localization networks.

However, extending standard STNs to predict views in 3D space while keeping their differentiability is non-trivial. To facilitate the backpropagation of loss gradient from a 2.5D depth image to 3D viewing parameters, we propose to parameterize the depth value at each pixel (x, y) in a depth image over the parameters of the corresponding view (θ, φ): $d(x, y) = f(\theta, \varphi)$, through a ray casting based depth computation at each pixel. Our attention model provides efficient view planning and robust object recognition, as demonstrated by our experimental evaluations. Our work contains two main technical contributions:

- A 3D attentional architecture that integrates RNN and STN for simultaneous object recognition and next-best-view (NBV) selection.
- A differentiable extension of STN for view selection in 3D space, leading to an end-to-end attentional network which can be trained efficiently.

2 Related work

Active object recognition has a rich literature in robotics, vision, and graphics (see surveys such as Refs. [11, 12]). We provide a brief review of 3D object recognition, especially active methods (categorized into information theoretic and policy learning approaches). We then discuss some recent attempts on end-to-end learning for NBV selection.

2.1 3D object recognition

One of the most popular methods for 3D object recognition is directly deploying deep learning on point sets [13, 14], but one shortcoming of these works is that point features are treated independently. Based on pioneering work, the Attentional ShapeContextNet [15], which connects shape contexts with convolutional neural networks (CNNs), is able to represent local and global shape information, and achieve competitive results on benchmark datasets. Inspired by image classification using CNNs, view-based methods for 3D object recognition have performed best so far. Multi-view CNN [3] renders a 3D shape to gray images from different views, uses a CNN to extract features for each rendered image, and aggregates features from all rendered images with max pooling. A hierarchical view-group-shape architecture is proposed in Ref. [16] aiming to treat each view discriminatively, while all views are treated equally in Ref. [3]. Impressive improvement is gained in Ref. [16], but it still needs to evenly sample several views before testing, which means all views are fixed when testing. This kind of method is not suitable for certain scenarios, such as the robot-operated NBV problem, which always tries to achieve the highest accuracy with as few as possible views.

2.2 Information theoretic approaches

Information theoretic formulation represents a standard approach to active vision problems. The basic idea is to quantify the information gain of each view by measuring mutual information between observations and object classes [1], entropy reduction of object hypotheses [17], or decrease in belief uncertainty about the object that generated the observations [18]. The optimal views are those which are expected to provide the maximal information gain. The estimation of information gain usually involves learning a generative object model (likelihood

or belief state) so that the posterior distribution of object class under different views can be estimated. Different methods have been utilized in estimating information gain, such as Monte Carlo sampling [1], Gaussian process regression [2], and reinforcement learning [19].

2.3 Policy learning approaches

Another line of research seeks to learn viewing policies. The problem is often viewed as a stochastic optimal control one and cast as a partially-observable Markov decision process. In Ref. [20], reinforcement learning is utilized to offline learn an approximate policy that maps a sequence of observations to a discriminative viewpoint. Kurniawati et al. [21] employed a point-based approximate solver to obtain a non-greedy policy offline. In contrast to offline learning, Lauri et al. [22] attempted to apply Monte Carlo tree search (MCTS) to obtain an online active hypothesis testing policy for view selection. Our method learns and compiles viewing policies into the hidden layers of an RNN, leading to a high-capacity view planning model.

2.4 End-to-end learning approaches

The recent rapid development of deep learning models has aroused the interest of end-to-end learning of active vision policies [23]. Malmir et al. [24] used deep Q-learning to find the optimal policy for view selection from raw images. Our method shares similarities with the recent work of Wu et al. [3] in taking 2.5D depth images as input and 3D shapes as training data. They adopted a volumetric representation of 3D shapes and trained a convolutional deep belief network (CDBN) to model the joint distribution over volume occupancy and shape category. By sampling the distribution, shape completion can be performed based on observed depth images, over which virtual scanning is conducted to estimate the information gain of a view. Unlike their method, our attention model is trained offline, so no online sampling is required, making it efficient for online active recognition. The works of Jayaraman and Grauman [4], Xu et al. [5], and Chen et al. [6] are the most closely related to ours. Compared to MV-RNN [5], VERAM [6] explicitly integrates view confidence and view location constraints into the reward function, and deploys strategies for view enhancement. In these methods, the recognition and control modules are jointly optimized based on reinforcement learning. We employ spatial transformer units [10] as our locator networks to obtain a fully differentiable network.

2.5 Spatial transformer networks

A spatial transformer is a differentiable sampling-based network, which gives neural networks the ability to actively transform the input data, without any spatial supervision in training. Generally, a spatial transformer is composed of a localization network and a generator. An STN achieves spatial attention by first passing the input image into a localization network which regresses the transformation, and then generating a transformed image using the generator. The transformed image is deemed to be easier to recognize or classify, and thus more suitable for the desired task.

An STN is a good fit to our problem setting. Due to its differentiability, it enables end-to-end training with backpropagation, making the network easier to learn. It is relatively straightforward to employ for object localization in the image of a given view. However, when using it to predict views in 3D, we face the problem of non-differentiable pixel depth values over viewing parameters, which is addressed by our work.

3 Approach

We first provide an architectural overview of our recurrent attentional model, followed by detailed description of the individual parts. We also provide details of the loss function, model training, and inference.

3.1 Architecture overview

Figure 1 shows the architecture of our recurrent attentional model. The main body of the model is a recurrent neural network (RNN) for modeling the sequential dependencies between consecutive views. In our setting, the 3D shape is located at the center of a sphere with fixed radius (see Fig. 2), and at each time step, the model first takes a view (parameterized in the local spherical coordinate system) as input, generates depth images using ray casting and extracts features from the depth images. Then the model amalgamates information of past views, makes a prediction of the categorical label, and produces an

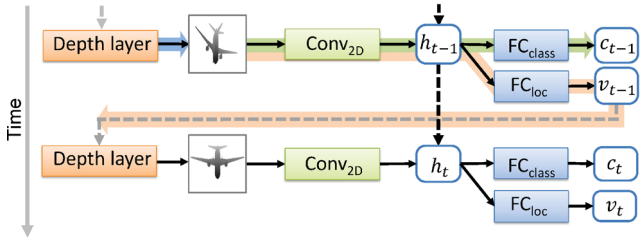

Fig. 1 Our recurrent attentional model. The dashed arrows indicate information flow across time steps while the solid ones represent that within a time step. The bold arrows underline the data flows of the three subnetworks, i.e., depth layer (DL, blue), 3D spatial transformer networks (3D-STN, orange), and shape classifier (SC, green).

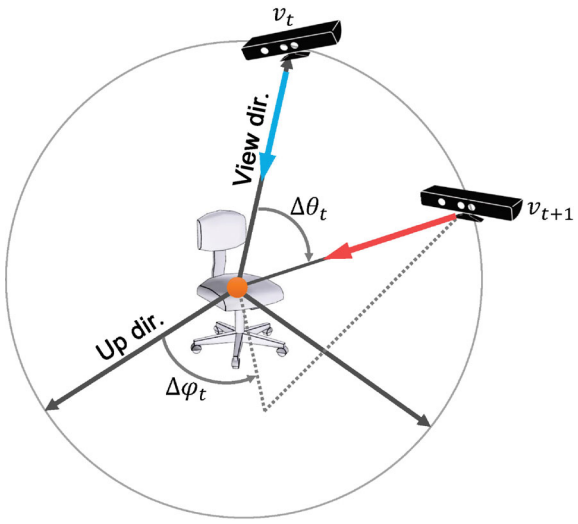

Fig. 2 The viewing sphere around the object being recognized. The view at the next time step, v_{t+1}, is parameterized in the local spherical coordinate system based on the current view v_t.

update to the current view for future observation. To do so, we embed three parts into the RNN: a depth layer (DL) for generating depth images of objects, a 3D spatial transformer network (3D-STN) for regressing the update to the current view for the shape, and a shape classifier (SC) for depth-based object recognition.

Our approach works as follows. Given a current view of an object, it first uses a ray casting algorithm to generate a depth image, which is fed into a stack of convolutional layers (Conv_{2D}) for feature extraction. The extracted features are aggregated with those extracted from previous views, with the help of the RNN hidden layers. The aggregated features are then used for classification and predicting the update to the current view, with the fully connected layers FC_{class} and FC_{loc}, respectively. With the predicted update to the current view, a next-best-view ($v_{t+1} = v_t + \Delta v_t$) can be obtained. Using v_{t+1}, a new depth image can be generated, which serves as the input at the next time step.

As shown in Fig. 1, DL is a single layer subnetwork. 3D-STN encompasses the convolutional layers Conv_{2D} and the fully connected layers FC_{loc}. SC is composed of Conv_{2D} and FC_{class}, which is a standard convolutional neural network (CNN) classifier. Moreover, the convolutional layers are shared by the SC and the 3D-STN.

In order to make our attentional network learnable, we require the generated depth image to be differentiable with respect to the viewing parameters. A basic assumption behind the parameterization of depth values in terms of viewing parameters and the intersection point is that the intersection point does not change when the view change is small, so the first-order derivative with respect to viewing parameters can be approximated by keeping the point fixed. Details are provided in Section 3.2.2.

3.2 Depth layer for depth image generation

The depth layer (DL) is a critical part of our model, both for depth image generation and loss backpropagation. With loss backpropagation, this layer allows us to build up an end-to-end trained deep neural network. During forward propagation, we use ray casting to generate depth images, and record every hit point position on the surface of shapes into a table, which is used in backpropagation. During backpropagation, we fill the gap between depth images loss gradient and camera views loss gradient.

3.2.1 Ray casting

Ray casting is used to solve the general problem of determining the hit points of a shape intersected by a ray. As illustrated in Fig. 3, a view is represented by (R, θ_t, φ_t), the radial distance to the shape center, polar, and azimuthal angle, respectively. In this spherical coordinate system, R is a constant, and the view direction points to the shape's center, which is also the origin of the spherical coordinate system. (R, θ_t, φ_t) can be easily transformed into Cartesian coordinates (X_v, Y_v, Z_v) by using Eq. (1), where (θ_t, φ_t) can be obtained by Eq. (2).

$$\begin{cases} X_v = R \sin\theta_t \cos\varphi_t \\ Y_v = R \sin\theta_t \sin\varphi_t \\ Z_v = R \cos\theta_t \end{cases} \tag{1}$$

$$\begin{cases} \theta_t = \theta_{t-1} + \Delta\theta_{t-1} \\ \varphi_t = \varphi_{t-1} + \Delta\varphi_{t-1} \end{cases} \tag{2}$$

As shown in Fig. 3, a projection plane lies between the viewpoint v_t and the shape. m is the intersection point of the viewing line and the projection plane. For ray casting, we need two points (or one point together with a direction). The position of v_t can be directly computed by Eq. (1). However, finding position of m is not straightforward. To make things simpler, we use a specific setting as follows: when $\theta_t = 0$, the position of v_t is $(0, 0, R)$, the up direction of the camera is set parallel to the X-axis of spherical coordinate system, and the origin of pixel coordinates is at $(0, 0, R - f)$, where f is the focal length of the camera. To go from projection coordinates to pixel coordinates, the camera intrinsic matrix should be applied. If we have a position (u_m, v_m) in pixel coordinates, the projection coordinates (x_m, y_m) can be calculated by

$$\begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{d}x & 0 & -u_0\mathrm{d}x \\ 0 & \mathrm{d}y & -v_0\mathrm{d}y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_m \\ v_m \\ 1 \end{bmatrix} \tag{3}$$

where $\mathrm{d}x, \mathrm{d}y$ is a single pixel length along the x- and y-axes of projection coordinates, u_0, v_0 represents the principal point, ideally at the center of the depth images, and the skew coefficient between x- and y-axes is set to 0. Since we have a predefined up direction for the camera, we can compute the world coordinate position of each pixel in the projection plane. When v_t is located in (R, θ_t, φ_t), we just apply two rotation matrices to get the new Cartesian coordinate position of each pixel, the first rotation being θ_t around the Y-axis, and the second being φ_t around the Z-axis.

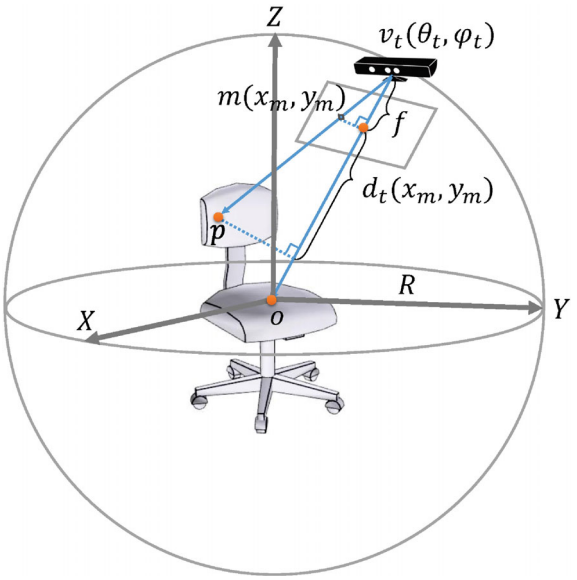

Fig. 3 Given a view $v_t = (\theta_t, \varphi_t)$, the depth value of a pixel $d_t(x_m, y_m)$ in the corresponding depth image is parameterized with repsect to the view parameters and the position of the intersection point, p, computed by ray casting.

For each pixel in the projection plane, we can form a ray from v_t to m, and the extension of this ray will hit or miss the shape surface. Using a ray casting algorithm, we can get the position (X_p, Y_p, Z_p) of the hit point (p) on the shape surface (if hit) and the hit distance (D_{vp}) between the start point (v_t) and hit point (p). As shown in Fig. 3, if a ray hits the shape, the depth value of the related pixel is represented by $d_t(x_m, y_m)$. The distance between v_t and p is calculated by Eq. (4). Using similar triangles gives Eq. (5). We use a table to record all hit points and their related pixels.

$$D_{vp} = \sqrt{(X_v - X_p)^2 + (Y_v - Y_p)^2 + (Z_v - Z_p)^2} \tag{4}$$

$$d_t(x_m, y_m) = \frac{f}{\sqrt{x_m^2 + y_m^2 + f^2}} D_{vp} - f \tag{5}$$

3.2.2 *Backpropagation*

Backpropagation through the depth layer computes loss gradients of input $(\Delta\theta_{t-1}, \Delta\varphi_{t-1})$, given loss gradients of output (depth image). During backpropagation, each pixel will be given a $\partial\text{loss}/\partial d_t$. We obtain the depth value d_t, and the hit point position (X_p, Y_p, Z_p) (which can be directly retrieved from the recorded table), so $\partial\text{loss}/\partial\Delta\theta_{t-1}$ and $\partial\text{loss}/\partial\Delta\varphi_{t-1}$ in Eq. (7) can be calculated along with Eqs. (1), (2), (4), (5), and (6).

$$\begin{cases} \dfrac{\partial d_t}{\partial \Delta\theta_{t-1}} = \dfrac{\partial d_t}{\partial D_{vp}} \times \dfrac{\partial D_{vp}}{\partial X_v} \times \dfrac{\partial X_v}{\partial \theta_t} \times \dfrac{\partial \theta_t}{\partial \Delta\theta_{t-1}} \\ \qquad + \dfrac{\partial d_t}{\partial D_{vp}} \times \dfrac{\partial D_{vp}}{\partial Y_v} \times \dfrac{\partial Y_v}{\partial \theta_t} \times \dfrac{\partial \theta_t}{\partial \Delta\theta_{t-1}} \\ \qquad + \dfrac{\partial d_t}{\partial D_{vp}} \times \dfrac{\partial D_{vp}}{\partial Z_v} \times \dfrac{\partial Z_v}{\partial \theta_t} \times \dfrac{\partial \theta_t}{\partial \Delta\theta_{t-1}} \\ \dfrac{\partial d_t}{\partial \Delta\varphi_{t-1}} = \dfrac{\partial d_t}{\partial D_{vp}} \times \dfrac{\partial D_{vp}}{\partial X_v} \times \dfrac{\partial X_v}{\partial \varphi_t} \times \dfrac{\partial \varphi_t}{\partial \Delta\varphi_{t-1}} \\ \qquad + \dfrac{\partial d_t}{\partial D_{vp}} \times \dfrac{\partial D_{vp}}{\partial Y_v} \times \dfrac{\partial Y_v}{\partial \varphi_t} \times \dfrac{\partial \varphi_t}{\partial \Delta\varphi_{t-1}} \\ \qquad + \dfrac{\partial d_t}{\partial D_{vp}} \times \dfrac{\partial D_{vp}}{\partial Z_v} \times \dfrac{\partial Z_v}{\partial \varphi_t} \times \dfrac{\partial \varphi_t}{\partial \Delta\varphi_{t-1}} \end{cases} \tag{6}$$

$$\begin{cases} \dfrac{\partial \text{loss}}{\partial \Delta\theta_{t-1}} = \dfrac{\partial \text{loss}}{\partial d_t} \times \dfrac{\partial d_t}{\partial \Delta\theta_{t-1}} \\ \dfrac{\partial \text{loss}}{\partial \Delta\varphi_{t-1}} = \dfrac{\partial \text{loss}}{\partial d_t} \times \dfrac{\partial d_t}{\partial \Delta\varphi_{t-1}} \end{cases} \tag{7}$$

Each pixel of the depth image will be given a $\partial\text{loss}/\partial\Delta\theta_{t-1}$ and $\partial\text{loss}/\partial\Delta\varphi_{t-1}$. We just average all $\partial\text{loss}/\partial\Delta\theta_{t-1}$ and $\partial\text{loss}/\partial\Delta\varphi_{t-1}$ to get the final loss gradients of $(\Delta\theta_{t-1}, \Delta\varphi_{t-1})$.

3.3 3D-STN for view selection

Given an input depth image, the goal of 3D-STN is to extract image features, and regress the 3D viewing parameters of the update to the current view $(\Delta\theta_t, \Delta\varphi_t)$ with respect to the recognition task. The 3D-STN comprises two subnetworks: Conv_{2D} and FC_{loc}. During forward passes, Conv_{2D} takes the depth image d_t as input and extracts features. Using the features, FC_{loc} outputs the update to the current view. The viewing parameter is parameterized in the local spherical coordinate system based on the current view v_t (θ_t, φ_t) (see Fig. 2), and represented as a tuple $(\Delta\theta_t, \Delta\varphi_t)$. Note that the viewing parameters do not include radius (R), since R is set to be a constant.

Specifically, the convolutional network Conv_{2D} first extracts features from the depth image output by depth layer:

$$e_t = f^{\text{2D}}_{\text{Conv}}(d_t, W^{\text{2D}}_{\text{Conv}}) \tag{8}$$

where $W^{\text{2D}}_{\text{Conv}}$ are the weights of Conv_{2D}. These features are amalgamated with those of past views stored in the RNN hidden layer h_{t-1}:

$$h_t = g(W_{\text{ih}} e_t + W_{\text{hh}} h_{t-1}) \tag{9}$$

where $g(\cdot)$ is a nonlinear activation function. W_{ih} and W_{hh} are weights for input-to-hidden and hidden-to-hidden connections, respectively. The aggregated features in h_t are then used to regress the update to the current view:

$$(\Delta\theta_t, \Delta\varphi_t) = f^{\text{loc}}_{\text{FC}}(h_t, W^{\text{loc}}_{\text{FC}}) \tag{10}$$

where $W^{\text{loc}}_{\text{FC}}$ are the weights of FC_{loc}. The viewing parameters are then used by the depth layer to generate a new depth image with the object during training:

$$d_{t+1} = f_{\text{DL}}(\theta_t + \Delta\theta_t, \varphi_t + \Delta\varphi_t) \tag{11}$$

3.4 Shape classifier for object recognition

The depth image output by the depth layer at each time step is passed into a shape classifier (SC, Conv_{2D} + FC_{class}) for class label prediction. Note that the classification is also based on the aggregated features of both current and past views:

$$c_t = f^{\text{class}}_{\text{FC}}(h_t, W^{\text{class}}_{\text{FC}}) \tag{12}$$

where $W^{\text{class}}_{\text{FC}}$ are the weights of FC_{class}.

3.5 Loss function

We employ cross-entropy loss to train our model. Cross-entropy loss measures the performance of a classification model whose output is a probability value between 0 and 1. Our loss function is

$$L = -\sum_{c=1}^{k} y_{o,c} \log(p_{o,c}) \tag{13}$$

where k is the number of classes, y is a binary indicator indicating whether class label c is the correct classification for current observation o, and p is the predicted probability that observation o is of class c.

3.6 Training and inference

To make the training more efficient, we decompose our training into two parts and tune their parameters separately: (1) pre-training of the SC; (2) joint training of the 3D-STN, SC, and RNN. The first part is trained by virtually recognizing 3D shapes in the training dataset, using generated depth images. We hope the Conv_{2D} of SC have a good ability to extract features for depth images. For each shape, we randomly select dozens of views to generate depth images, and feed them to a pre-trained CNN classifier (we use AlexNet [25]), as shown in Fig. 4, in order to train it for an image classification task.

To train 3D-STN, we evenly select 50 views as initial views, and from each initial view, we start virtual recognition for an episode of 10 time steps. In each step, the network takes images as input, and outputs both class label and the update to the current view. The network is trained by the cross entropy loss in classification. The training leverages parameters of the pre-trained SC, and tunes the parameters of the 3D-STN, RNN, and SC simultaneously, using backpropagation through time

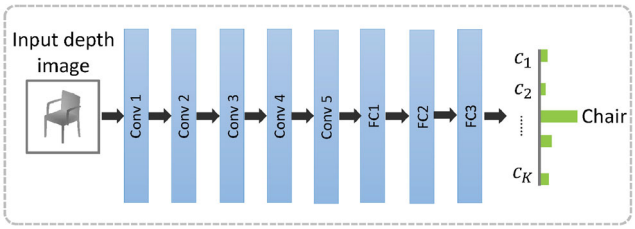

Fig. 4 Architecture of our shape classifier, which is borrowed from AlexNet [25]. It takes depth images as input, and outputs classification results.

(BPTT) [26]. The number of initial views is a trade-off between network performance and computation density. With more initial views to explore, networks achieve better performance, but the training time is greatly increased. We use 50 initial views so that our networks can achieve good performance in an affordable training time.

At inference time, given an object, a depth image is generated using a ray casting algorithm with a random initial view from our selected 50 views. The generated depth image is then passed into our attentional model. The latter firstly extracts features from the depth image, which are then used both for object classification (SC) and next-best-view prediction (3D-STN). Our 3D-STN automatically regresses the update to the current view. Given the current view and the view update, our camera moves to the next-best-view, and generates a new depth image. RNN hidden layers help to aggregate current depth image features with those from the past views. This is repeated until termination conditions are met. We set two termination conditions for inference: (i) the classification uncertainty, measured by the Shannon entropy of the classification probability, is less than a threshold (0.1), or (ii) the maximum number (10) of time steps has been reached.

4 Experimental setup

4.1 3D shape datasets

Our method has been evaluated using three large-scale 3D shape datasets: ModelNet10 [27], ModelNet40 [3], and ShapeNetCore55 [28]. ModelNet10 and ModelNet40 are two subsets of the Princeton ModelNet which contains 127,915 3D shapes categorized into 660 classes. ModelNet10 contains 10 categories with a total of 4899 3D shapes; these shapes are split into a training set and a test set. The training set contains 3991 shapes, while the test set contains 908 shapes. ModelNet40 contains 12,311 3D shapes in 40 classes; they are also split into a training set (9843 3D shapes) and a test set (2468 3D shapes). ShapeNetCore55 is a richly-annotated, large-scale dataset of 3D shapes, which has 55 common object categories with about 51,300 unique 3D shapes. For ShapeNetCore55, we split all shapes of each class into training (80%) and testing (20%) subsets.

Before training, the center of each 3D shape is placed at the origin of the spherical coordinate system, and the shape is scaled to the range $[-0.5, 0.5]$. The radius of the spherical coordinate system is set to 1. During training, a 3D shape needs to be rendered into a 227×227 depth image from any given viewpoint, using the ray casting algorithm (implemented in parallel on the GPU). All depth values are normalized to $[0, 255]$. If a ray does not hit the shape, the related depth value is set to 255.

4.2 Parameters of subnetworks

We use the ReLU activation functions for all hidden layers, $f(x) = \max(0, x)$, for fast training. The shape classifier has the AlexNet architecture [25] which contains 5 convolutional layers and 3 fully connected layers followed by a soft-max layer (see Fig. 4). The same parameter settings as used in the original paper are used for the various layers. The 3D-STN contains 5 convolutional layers (shared with the shape classifier) and 2 fully connected layers. In summary, there are about $62M$ parameters in total being optimized when training the 3D-STN, RNN, and SC. The maximum number of views for each view sequence is 10, the learning rate of pre-training the SC is set to 0.01, and the learning rate of joint training the 3D-STN, SC, and RNN is set to 0.001.

5 Results and evaluation

5.1 Hidden layer size

The size of the hidden layer is clearly an important hyper-parameter that affects the performance of recurrent neural networks. In this experiment, we evaluated the effect of hidden layer size. We carried out object recognition experiments on ModelNet40 to determine accuracy. Five views were used for each sequence, and the radius (R) remained 1.

The candidate sizes of the hidden layer were 64, 128, 256, 512, and 1024. Results are shown in Table 1, and show that our recurrent neural network achieves best performance with a hidden layer size of 256. All the following experiments were conducted with a fixed hidden layer size of 256.

Table 1 Effect of hidden layer size on accuracy using ModelNet40

Hidden layer size	64	128	256	512	1024
Accuracy (%)	87.1	87.4	**88.3**	87.6	87.8

5.2 Radius

We use a fixed radius of spherical coordinate system in our experiments. However, the radius has a big impact on the recognition performance. If the radius is too large, the 3D shape will be very small in projection on the depth image plane. The aim of this experiment is to find the best radius for our task.

All 3D shapes were scaled to the range $[-0.5, 0.5]$. We conducted a comparison using different radii: 0.5, 1.0, 1.5, 2.0, and 2.5 using ModelNet40 and 5 views. Results are shown in Table 2. Best results are obtained with a radius of 1.0, under the condition that we scale our 3D shape to a range of $[-0.5, 0.5]$. Results show that camera should neither be too far nor too near to the 3D shape. If the camera is too far from the 3D shape, accuracy will drop dramatically. However, if the camera is too near to the 3D shape, the shape projection in the depth image will be clipped, decreasing the recognition accuracy. Moreover, if the projection of the 3D shape spans the full depth image, some details of the shape border will be lost during the convolution operation.

5.3 Next best view

We tackle the next-best-view (NBV) problem as seeking single camera views in order to improve accuracy of 3D shape recognition. In this setting, finding the NBV can be seen as an incremental approach to a sensing strategy for multi-view active object recognition, which always tries to achieve the highest possible accuracy with the smallest number of views. But, with partial observation of a 3D shape, it is impossible to acquire the globally optimal NBV, so we use two criteria for evaluating NBV estimation: recognition accuracy and information gain.

To evaluate the performance of NBV estimation, we compare our attentional method against four alternatives, including a baseline method and three state-of-the-art methods. The baseline method *Rand* selects the next view randomly. The state-of-the-art NBV techniques used for comparison include *3DShapeNets* [3], *MV-RNN* [5], and the active recognition method based on *Pairwise* learning [29]. We trained our model on the training set from ModelNet40 with the task of classification over its 40 classes, and tested on the ModelNet40 test set. For a fair comparison of the quality of the determined NBVs, all methods were evaluated only with depth images. We let each method predict their own NBV sequences, and uniformly used a pre-trained multi-view CNN shape classifier (MV-CNN) [30] for shape classification. Figure 5 plots the recognition rate versus the number of predicted NBVs. Our method achieves the fastest accuracy increase.

Table 2 Effect of radius on accuracy using ModelNet40

Radius	0.5	1.0	1.5	2.0	2.5
Accuracy (%)	81.4	**88.2**	82.6	75.8	69.3

To further evaluate the computed NBVs, we plot in Fig. 6 the information gain for different NBV methods. The information gain is measured by the decrease of Shannon entropy of the classification probability distributions:

$$I_t(p_t, p_{t-1}) = H(p_{t-1}) - H(p_t) \tag{14}$$

where $H(p) = -\sum_k p(y_k) \log p(y_k)$. Compared to the alternatives, the NBV sequences output by our method attain larger information gain in early steps,

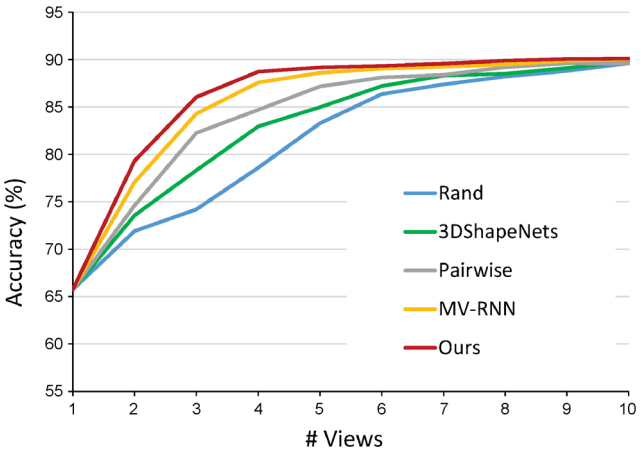

Fig. 5 Recognition accuracy versus the number of views for five methods (*Rand*, *3DShapeNets* [3], *MV-RNN* [5], *Pairwise* [29], and ours).

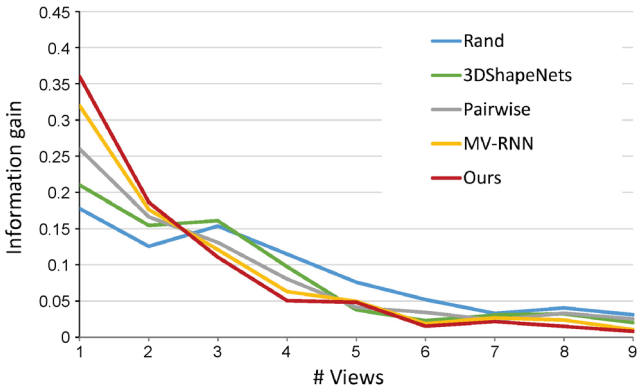

Fig. 6 Information gain for views selected by five methods (*Rand*, *3DShapeNets* [3], *MV-RNN* [5], *Pairwise* [29], and ours).

demonstrating higher efficiency.

5.4 3D shape recognition

To evaluate recognition performance, we carried out a comparison experiment using three datasets: ModelNet10, ModelNet40, and ShapeNetCore55. We compared our method against recent competing methods: *3DShapeNets* [3], *MV-RNN* [5], and *Pairwise* learning [29]. For a fair comparison, all methods were evaluated only with depth images. For *3DShapeNets* and *Pairwise* learning, we selected the next view along the sequence by following a straight path around the viewing sphere from the beginning to the end of the sequence [29].

Table 3 shows the recognition results for a random initial view from our selected 50 views. Our method achieves the best results. We note that *3DShapeNets* [3], *MV-RNN* [5], and our methods are depth-based, while *Pairwise* learning [29] uses both grayscale and depth images. In our experiment, however, we only use depth images for *Pairwise* learning to ensure a fair comparison.

We also note that VERAM [6] achieves a recognition accuracy of 92.1% on ModelNet40 with 9 views of gray images. They align all shapes and render 144 gray images for each shape with a Phong reflection model. Reinforcement learning is adapted to solve the problem that gradients from observation subnetworks cannot be back propagated to recurrent subnetworks. They integrate view confidence and view location constraints into the reward function. Moreover, three strategies (sign, clamp, and ELU) are deployed to enhance gradients. For the RNN, they deploy long short-term memory (LSTM) units. For a fair comparison, we tested with three experimental settings of VERAM. Firstly, we changed 144 viewpoints to 50 viewpoints. Secondly, we used our ray casting method to generate depth images instead of Phong gray images. Thirdly, we modified the LSTM of RNN to use linear mapping to keep the same setting as our method. Both VERAM and our method use AlexNet and the same radius. We found that this modified VERAM gives a recognition accuracy of 88.7% for ModelNet40 with 9 views, which is inferior to our method (89.8%).

5.5 Timings

Table 4 lists the training and testing time for our method on both ModelNet10 and ModelNet40 datasets. Since the shape classifier is pre-trained outside the joint training networks (3D-STN, RNN, and SC), we report its pre-training time separately.

Table 5 compares the training time of three methods: *3DShapeNets*, *MV-RNN*, and ours, for ModelNet40. The training of *3DShapeNets* involves learning the generative model with CDBN. *MV-RNN* is trained with reinforcement learning. The comparison shows the improved training efficiency of our model compared to the two alternatives. All timings were obtained on a workstation with an Intel Xeon E5-2670 @ 2.30 GHz × 24 with 64 GB RAM and an Nvidia Titan Xp graphics card with 12 GB memory.

5.6 Visualization

To visually investigate the behavior of our model, we visualize in Fig. 7 view sequences produced by *Pairwise* learning [29] and our method. The results

Table 3 Recognition accuracy for four methods and three datasets

Method	Image	ModelNet10				ModelNet40				ShapeNetCore55			
		3 views	6 views	9 views	Average	3 views	6 views	9 views	Average	3 views	6 views	9 views	Average
3DShapeNets	Depth	76.9	81.8	82.5	80.4	71.2	74.8	78.1	74.7	69.3	71.7	72.8	71.3
MV-RNN	Depth	86.4	88.9	89.5	88.2	84.3	86.5	88.6	86.4	71.1	73.8	76.3	73.7
Pairwise	Depth	84.9	87.7	89.2	87.3	82.7	85.1	88.3	85.3	70.8	73.9	76.2	73.6
Ours	Depth	87.7	89.8	91.2	**89.5**	86.1	88.7	89.8	**88.2**	71.6	74.5	76.9	**74.3**

Table 4 Training and testing time for our method

Dataset	Training		Testing
	Joint training	Pre-training SC	
ModelNet10	7.0 hours	2.7 hours	0.1 s
ModelNet40	14.3 hours	5.2 hours	

Table 5 Training and testing time for *3DShapeNets*, *MV-RNN*, and our method

Method	Training	Testing
3DShapeNets	96 hours	240 s
MV-RNN	66 hours	0.1 s
Ours	19.5 hours	0.1 s

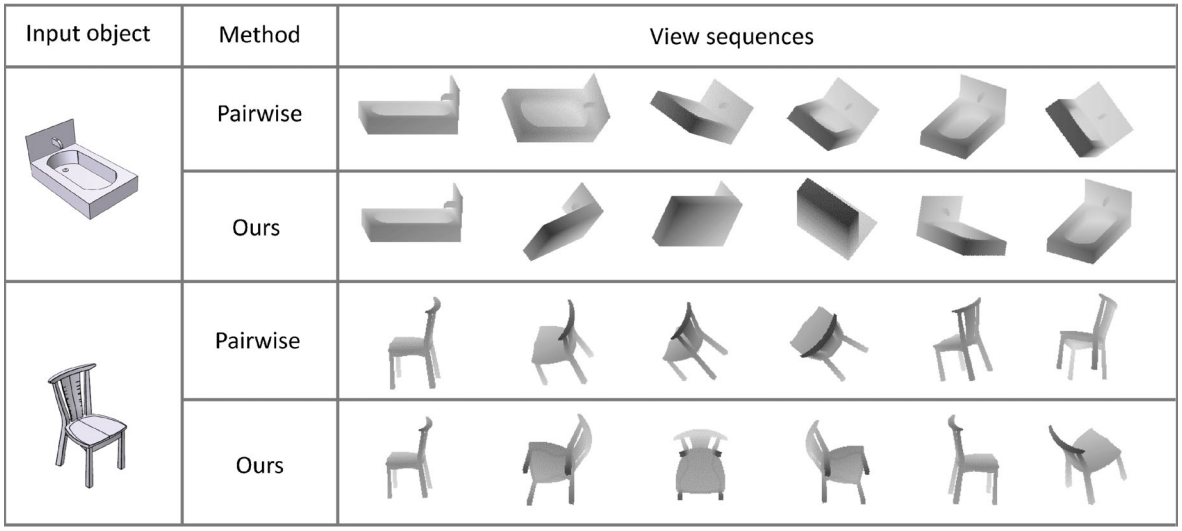

Fig. 7 View sequences produced by *Pairwise* learning [29] and our method for objects from ModelNet40.

demonstrate that our method can correctly recognize the objects with plausibly planned views. We note that the regressed view sequences tend to have better coverage of shapes giving higher recognition rates than *Pairwise* learning. In our method, we start training our model from separate 50 initial views, which means we can start from different initial views when testing. More results of our method are shown in Fig. 8. All input objects are from the test set of ModelNet40.

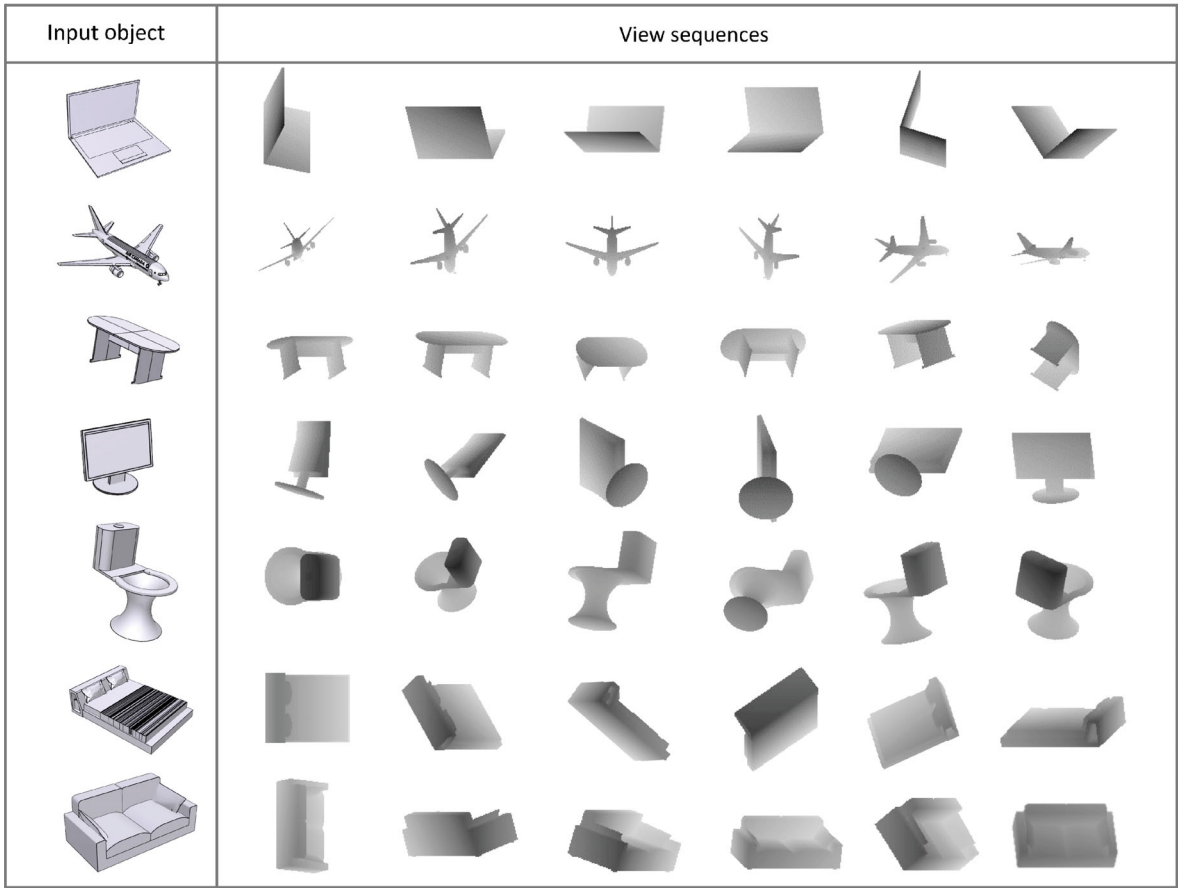

Fig. 8 Visualization of the active recognition process by showing the NBV sequence with depth images for input objects from ModelNet40.

6 Conclusions

6.1 Summary

We have proposed a 3D attentional formulation for the active object recognition problem. This was mainly motivated by the resemblance of mechanisms of human attention and active perception [31], and the significant progress made in utilizing attentional models to address complicated vision tasks such as image captioning [8]. In developing such a model, we utilized RNNs for learning and storing the internal representation of the object being observed, CNNs for performing depth-based recognition, and STNs for selecting the next-best-views. The carefully designed 3D STN makes the whole network differentiable and hence easy to train. Experiments on well-known datasets demonstrate the efficiency and robustness of our active recognition model.

6.2 Limitations

A drawback of learning a policy offline is that physical restrictions during testing are hard to incorporate and when the environment is changed, the computed policy may no longer be useful. This problem can be alleviated by learning from a large amount of training data using a high capacity learning model such as deep neural networks, as we do. Our method does not handle mutual occlusion between objects which is a frequent case in cluttered scenes. One possible solution is to train the recognition network using depth images with synthetic occlusion.

6.3 Future work

In future, we would like to investigate a principled solution for handling object occlusion in real indoor scenes, e.g., by using an STN to help localize those shape parts which are both visible and discriminative, in a similar spirit to Ref. [32]. Another interesting direction is to study multi-agent attention in achieving cooperative vision tasks, such as multi-robot scene reconstruction and understanding. It would be particularly interesting to study the shared and distinct attentional patterns of heterogeneous robots such as mobile robots and drones.

Acknowledgements

We thank the anonymous reviewers for their valuable comments. This work was supported, in part, by National Natural Science Foundation of China (Nos. 61572507, 61622212, and 61532003). Min Liu is supported by the China Scholarship Council.

References

[1] Denzler, J.; Brown, C. M. Information theoretic sensor data selection for active object recognition and state estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 24, No. 2, 145–157, 2002.

[2] Huber, M. F.; Dencker, T.; Roschani, M.; Beyerer, J. Bayesian active object recognition via Gaussian process regression. In: Proceedings of the 15th International Conference on Information Fusion, 1718–1725, 2012.

[3] Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; Xiao, J. 3D ShapeNets: A deep representation for volumetric shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1912–1920, 2015.

[4] Jayaraman, D.; Grauman, K. Look-ahead before you leap: End-to-end active recognition by forecasting the effect of motion. In: *Computer Vision – ECCV 2016. Lecture Notes in Computer Science, Vol. 9909*. Leibe, B.; Matas, J.; Sebe, N.; Welling, M. Eds. Springer Cham, 489–505, 2016.

[5] Xu, K.; Shi, Y.; Zheng, L.; Zhang, J.; Liu, M.; Huang, H.; Su, H.; Cohen-Or, D.; Chen, B. 3D attention-driven depth acquisition for object identifiation. *ACM Transactions on Graphics* Vol. 35, No. 6, Article No. 238, 2016.

[6] Chen, S.; Zheng, L.; Zhang, Y.; Sun, Z.; Xu, K. VERAM: View-enhanced recurrent attention model for 3D shape classification. *IEEE Transactions on Visualization and Computer Graphics* doi: 10.1109/TVCG.2018.2866793, 2018.

[7] Mnih, V.; Heess, N.; Graves, A.; Kavukcuoglu, K. Recurrent models of visual attention. In: Proceedings of the Advances in Neural Information Processing Systems 27, 2204–2212, 2014.

[8] Xu, K.; Ba, J. L.; Kiros, R.; Courville, A.; Salakhutdinov, R.; Zemel, R. S.; Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. In: Proceedings of the 32nd International Conference on Machine Learning, Vol. 37, 2048–2057, 2015.

[9] Corbetta, M.; Shulman, G. L. Control of goal-directed and stimulus-driven attention in the brain. *Nature Reviews Neuroscience* Vol. 3, No. 3, 201–215, 2002.

[10] Jaderberg, M.; Simonyan, K.; Zisserman, A.; Kavukcuoglu, K. Spatial transformer networks. In: Proceedings of the Advances in Neural Information Processing Systems 28, 2017–2025, 2015.

[11] Scott, W. R.; Roth, G.; Rivest, J.-F. View planning for automated three-dimensional object reconstruction and inspection. *ACM Computing Surveys* Vol. 35, No. 1, 64–96, 2003.

[12] Dutta Roy, S.; Chaudhury, S.; Banerjee, S. Active recognition through next view planning: A survey. *Pattern Recognition* Vol. 37, No. 3, 429–446, 2004.

[13] Qi, C. R.; Su, H.; Mo, K.; Guibas, L. J. PointNet: Deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 652–660, 2017.

[14] Qi, C. R.; Yi, L.; Su, H.; Guibas, L. J. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In: Proceedings of the Advances in Neural Information Processing Systems 30, 5099–5108, 2017.

[15] Xie, S.; Liu, S.; Chen, Z.; Tu, Z. Attentional ShapeContextNet for point cloud recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4606–4615, 2018.

[16] Feng, Y.; Zhang, Z.; Zhao, X.; Ji, R.; Gao, Y. GVCNN: Group-view convolutional neural networks for 3D shape recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 264–272, 2018.

[17] Borotschnig, H.; Paletta, L.; Prantl, M.; Pinz, A. Appearance-based active object recognition. *Image and Vision Computing* Vol. 8, No. 9, 715–727, 2000.

[18] Callari, F. G.; Ferrie, F. P. Active object recognition: Looking for differences.*International Journal of Computer Vision* Vol. 43, No. 3, 189–204, 2001.

[19] Arbel, T.; Ferrie, F. P. Entropy-based gaze planning. *Image and Vision Computing* Vol. 19, No. 11, 779–786, 2001.

[20] Paletta, L.; Pinz, A. Active object recognition by view integration and reinforcement learning. *Robotics and Autonomous Systems* Vol. 31, No. 1, 71–86, 2000.

[21] Kurniawati, H.; Hsu, D.; Lee, W. S. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: Proceedings of the Robotics: Science and Systems, Vol. 2008, 2008.

[22] Lauri, M.; Atanasov, N.; Pappas, G.; Ritala, R. Active object recognition via Monte Carlo tree search. In: Proceedings of the Workshop on Beyond Geometric Constraints at the International Conference on Robotics and Automation, 2015.

[23] Levine, S.; Finn, C.; Darrell, T.; Abbeel, P. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research* Vol. 17, No. 39, 1–40, 2016.

[24] Malmir, M.; Sikka, K.; Forster, D.; Movellan, J.; Cottrell, G. W. Deep Q-learning for active recognition of germs: Baseline performance on a standardized dataset for active learning. In: Proceedings of the British Machine Vision Conference, 161–171, 2016.

[25] Krizhevsky, A.; Sutskever, I.; Hinton, G. E. ImageNet classification with deep convolutional neural networks. In: Proceedings of the Advances in Neural Information Processing Systems 25, 1097–1105, 2012.

[26] Mozer, M. C. A focused back-propagation algorithm for temporal pattern recognition. *Complex Systems* Vol. 3, No. 4, 349–381, 1989.

[27] Wu, Z.; Song, S.; Khosla, A.; Tang, X.; Xiao, J. 3D ShapeNets for 2.5D object recognition and next-best-view prediction. *arXiv preprint* arXiv:1406.5670, 2014.

[28] Chang, A. X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; Xiao, J.; Yi, L.; Yu, F. ShapeNet: An information-rich 3D model repository. *arXiv preprint* arXiv:1512.03012, 2015.

[29] Johns, E.; Leutenegger, S.; Davison, A. J. Pairwise decomposition of image sequences for active multi-view recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 3813–3822, 2016.

[30] Su, H.; Maji, S.; Kalogerakis, E.; Learned-Miller, E. Multi-view convolutional neural networks for 3D shape recognition. In: Proceedings of the IEEE International Conference on Computer Vision, 945–953, 2015.

[31] Bajcsy, R. Active perception. *Proceedings of the IEEE* Vol. 76, No. 8, 966–1005, 1988.

[32] Xiao, T.; Xu, Y.; Yang, K.; Zhang, J.; Peng, Y.; Zhang, Z. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 842–850, 2015.

Min Liu is a Ph.D. candidate in the School of Computers, National University of Defense Technology. He received his B.S. degree in geodesy and geomatics from Wuhan University and M.S. degree in computer science from National University of Defense Technology in 2013 and 2016, respectively. He is visiting the University of Maryland at College Park from 2018 to 2020. His research interests mainly include robot manipulation and 3D vision.

Yifei Shi received his B.S. degree in geodesy and geomatics from Wuhan University and M.S. degree in computer science from National University of Defense Technology in 2012 and 2015, respectively. He is pursing a doctorate in computer science at the National University of Defense Technology. His research interests mainly include data-driven scene understanding, RGBD reconstruction, and 3D vision.

Lintao Zheng received his B.S. degree in applied mathematics from Xi'an Jiaotong University and M.S. degree in computer science from the National University of Defense Technology in 2013 and 2016, respectively. He is pursuing a doctorate in computer science at the National University of Defense Technology. His research interests mainly include computer graphics, deep learning, and robot vision.

Kai Xu is an associate professor at the School of Computers, National University of Defense Technology, where he received his Ph.D. degreee in 2011. He conducted visiting research at Simon Fraser University during 2008–2010, and Princeton University during 2017–2018. His research interests include geometry processing and geometric modeling, especially on data-driven approaches to the problems in those directions, as well as 3D-geometry-based computer vision. He has published over 60 research papers, including 21 SIGGRAPH/TOG papers. He has organized two SIGGRAPH Asia courses and one Eurographics STAR tutorial. He is currently serving on the editorial boards of *Computer Graphics Forum*, *Computers & Graphics*, and *The Visual Computer*. He also served as paper co-chair of CAD/Graphics 2017 and ICVRV 2017, as well as a PC member for several prestigious conferences including SIGGRAPH Asia, SGP, PG, GMP, etc. His research work can be found in his personal website: `http://www.kevinkaixu.net`.

Hui Huang is Distinguished Professor, Founding Director of the Visual Computing Research Center, Shenzhen University. She received her Ph.D. degree in applied math from the University of British Columbia in 2008 and another Ph.D. degree in computational math from Wuhan University in 2006. Her research interests are in computer graphics and vision, focusing on geometric modeling, shape analysis, point optimization, image processing, 3D/4D acquisition, and creation. She is currently an Associate Editor-in-Chief of *The Visual Computer* (TVC) and on the editorial boards of *Computers & Graphics* and *Frontiers of Computer Science*. She has served on the program committees of almost all major computer graphics conferences including SIGGRAPH Asia, EG, SGP, PG, 3DV, CGI, GMP, SMI, GI, and CAD/Graphics. She was a CHINAGRAPH 2018 Program Vice-Chair, in addition to SIGGRAPH Asia 2017 Technical Briefs and Posters Co-Chair, SIGGRAPH Asia 2016 Workshops Chair and SIGGRAPH Asia 2014 Community Liaison Chair. She is the recipient of an NSFC Excellent Young Scientist Award, Guangdong Technological Innovation Leading Talent Award, CAS Youth Innovation Promotion Association Excellent Member Award, Guangdong Outstanding Graduate Advisor Award, CAS International Cooperation Award for Young Scientists, and CAS Lujiaxi Young Talent Award. She is also a CCF Distinguished Member and ACM/IEEE Senior Member.

Dinesh Manocha is the Paul Chrisman Iribe Chair in Computer Science & Electrical and Computer Engineering at the University of Maryland at College Park. He is also the Phi Delta Theta/Matthew Mason Distinguished Professor Emeritus of Computer Science at the University of North Carolina Chapel Hill. He has won many awards, including Alfred P. Sloan Research Fellow, NSF Career Award, ONR Young Investigator Award, and the Hettleman Prize for scholarly achievement. His research interests include multi-agent simulation, virtual environments, physically-based modeling, and robotics. His group has developed a number of packages for multi-agent simulation, crowd simulation, and physics-based simulation that have been used by hundreds of thousands of users and licensed to more than 60 commercial vendors. He has published more than 480 papers and supervised more than 35 Ph.D. dissertations. He is an inventor of 9 patents, several of which have been licensed to industry. His work has been covered by the New York Times, NPR, Boston Globe, Washington Post, ZDNet, as well as DARPA Legacy Press Release. He is a Fellow of AAAI, AAAS, ACM, and IEEE and also received the Distinguished Alumni Award from IIT Delhi. See `http://www.cs.umd.edu/dm`.